Prva izdaja

PODATKOVNE STRUKTURE IN ALGORITMI

Zbirka nalog

Andrej Brodnik
Rok Požar

Univerza na Primorskem
Fakulteta za matematiko, naravoslovje in informacijske tehnologije

Izbrana poglavja iz računalništva in informatike:

Podatkovne strukture in algoritmi

Zbirka nalog

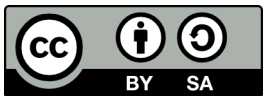
1. izdaja

Andrej Brodnik in Rok Požar

Drugo učno gradivo (študijsko gradivo)
strani: 176
računalništvo in informatika, dodiplomski študij

Koper 2023

Andrej Brodnik in Rok Požar
Podatkovne strukture in algoritmi, Zbirka nalog, 1. izdaja
2.05 drugo učno gradivo (študijsko gradivo)
Koper, 2023
Samozaložba



Kazalo

Slike

Predgovor

> Vendar ni zlato tisto, kar hočem
> tako zelo, kot je zgolj iskanje zlata.
>
> *Yet it isn't the gold that I'm wanting*
> *So much as just finding the gold.*
>
> Robert Service, Prekletstvo Yukona (*The Spell of the Yukon*)

Zvedava bralka in bralec, pred nami je odsev poti iskanja, po kateri hodimo že več kot deset let na Univerzi na Primorskem, na Fakulteti za matematiko, naravoslovje in informacijske tehnologije ter na Univerzi v Ljubljani, na Fakulteti za računalništvo in informatiko. Hodimo skupaj z asistenti in študenti pri predmetih *Podatkovne strukture in algoritmi* oziroma sestrskem predmetu *Algoritmi in podatkovne strukture*.

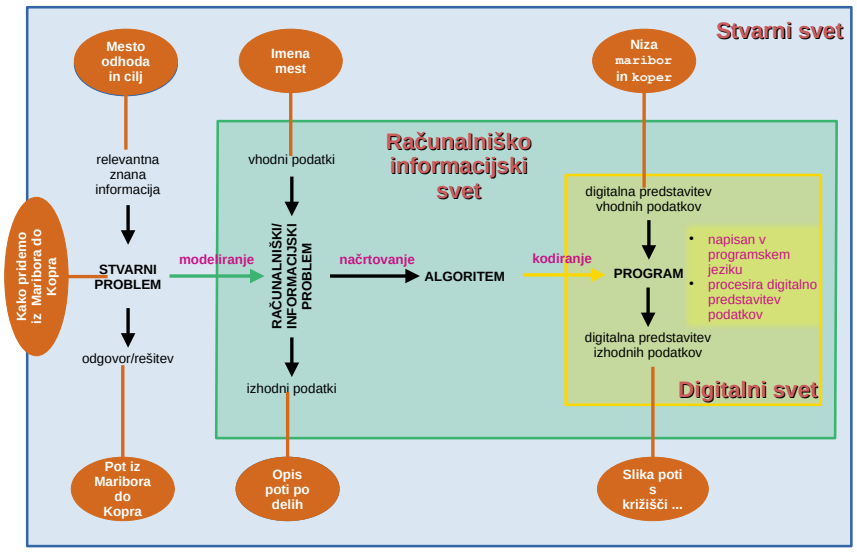
Preddverje

Pot je pri obeh predmetih speljana na podoben način. Nanjo stopimo skozi preddverje osnovnih definicij, ki nas popeljejo v čarobni svet podatkovnih struktur in algoritmov. Po starem latinskem pregovoru *Nomen est omen* nas opremijo z izrazoslovjem in osnovnimi orodji, ki nam omogočajo smiselno premišljevati in razglabljati o posameznih rešitvah. Pri tem si ob rešitvi vedno zastavljamo naslednjih pet vprašanj:

1. Pravilnost: Ali je rešitev pravilna?
2. Časovna zahtevnost: Koliko časa potrebuje, da naračuna rezultat?
3. Prostorska zahtevnost: Koliko pomnilnika potrebuje, da naračuna rezultat?
4. Zahtevnost problema: Ali obstaja boljša rešitev?

5. Inženirska zahtevnost: Kako zahtevna je koda, ki implementira rešitev?

Eden ključnih pojmov, ki nam omogočajo razmišljati o podatkovnih strukturah in algoritmih je abstrakten opis stroja, ki izvaja naše rešitve. Za boljše razumevanje si poglejmo, kako pravzaprav pripravimo računalnik do tega, da nam reši problem. Gre za večstopenjski proces, kot ga tudi popisuje sl. 1. V



Slika 1: Od stvarnega problema do njegove rešitve.

prvem koraku stvarni problem, npr. iskanje najboljše poti med Mariborom in Koprom, modeliramo tako, da dobimo abstraktni računalniški problem, katerega bo reševal naš stroj. Da pa bo stroj lahko to počel, potrebuje program. Malce poenostavljeno povedano je program v programskem jeziku kodiran algoritem, katerega smo začrtali tako, da nam reši računalniški problem.

Pri načrtovanju algoritma potrebujemo abstraktni opis stroja, *model računanja*, za katerega, po eni strani, načrtujemo algoritem in po drugi strani dovolj verno opisuje resničen stroj. Vloga modela računanja je, da abstrahira pogled na stroj, ki izvaja našo rešitev. Na ta način se naša rešitev na nek način poenostavi, saj ni več vpeta v strogo definicijo programskega jezika, ampak jo lahko popišemo s pomočjo (pol)formalne psevdokode. Opisana abstrakcija pa ima dobre in slabe strani. Dobra je brez dvoma opisani odmik od stroge programske kode. Slaba pa je, da lahko na ta način prezremo pomembne podrobnosti. Ker so ta zbirka in tudi predmeta namenjeni

začetnikom, smo se odločili, da damo prednost abstrakciji. Posledično so opisi rešitev v nedoločeni psevdokodi, ki spominja včasih na Python in včasih na Javo.

Dežela podatkovnih struktur

Prva dežela, v katero stopimo iz preddverja, je dežela podatkovnih struktur. Osnovni podatkovni strukturi, ki ju spoznamo sta slovar in vrsta s prednostjo. Pri snovanju podatkovnih struktur vedno pričnemo pri preprostih strukturah, o katerih se sprašujemo, kje so njihove omejitve. Ko razumemo omejitve, jih poskušamo odpraviti in dobiti boljše rešitve.

Tudi naloge v tej zbirki je potrebno tako razumeti in brati. Včasih naloge nimajo enoznačne rešitve, ampak je njihova rešitev odvisna od predpostavk, se pravi zahtev, ki jih postavimo. Tako lahko predpostavka naredi tudi rešitev, ki se zdi na prvi pogled povsem napačna za smiselno.

Od zvedave bralke in bralca se pričakuje predvsem, da razume rešitev in ne toliko, da se nauči vsa vrtenja rdeče-črnih dreves. Zelo malo verjetno je, da bomo še kdaj v življenju kodirali rdeče-črna drevesa. Veliko bolj verjetno pa je, da bomo na osnovi rdeče-črnih drevesa zasnovali novo rešitev. Pravimo, da bomo obstoječo podatkovno strukturo *razširili*.

V deželi podatkovnih struktur bomo srečali niz različnih implementacij slovarja (od povezanega seznama pa do Bloomovega filtra) in vrste s prednostjo (od implicitne dvojiške kopice pa do eksplicitne binomske in Fibonaccijeva kopice). Srečali bomo še številska drevesa, ki so tako zelo uporabna pri bioinformatiki in delu z nizi.

Pred odhodom iz dežele podatkovnih struktur se bomo še spoznali z drevesom, ki nam omogoča učinkovito delo z disjunktnimi množicami.

Dežela algoritmov

Iz dežele podatkovnih struktur nas pot privede v deželo algoritmov. Le-ta je bogato poseljena z množico različnih in zanimivih algoritmov. A nas ne zanimajo toliko sami algoritmi, kot tehnike in metode njihovega snovanja.

Tako na primer spoznamo *požrešno metodo* skozi urejanje z izbiranjem, medtem ko metodo *deli in vladaj* skozi urejanje z zlivanjem in hitro urejanje. Prav slednje nas ponovno zapelje v svet naključnostnih rešitev, katerega smo bili že obiskali tudi v deželi podatkovnih struktur, ko smo si ogledali implementacije slovarja s preskočnim seznamom ali razpršilno funkcijo. Prav ta inventivnost pri uporabi načrtovalskih metod je pogosto tisto, po čemer

spraševanje vprašanja v nalogah.

Na tem mestu še beseda o nalogah. Naloge običajno sestojijo iz treh, včasih štirih vprašanj. Običajno je eno vprašanje lahko in zahteva zgolj mehaniko rešitve, medtem ko drugo že pričakuje kakšno posplošitev. Tretje vprašanje je praviloma zastavljeno tako, da zahteva razumevanje in ustvarjanje nečesa novega.

Vrnimo se na našo pot po deželi algoritmov, ki se nadaljuje z metodo, katera združuje obe prejšnji – požrešno ter deli in vladaj. Gre za metodo dinamičnega programiranja. Naloge so različno težke in tudi ponujajo vprašanja o metodi pomnenja (*memoisation*).

Tako rekoč zadnji koraki po deželi algoritmov nas popeljejo še v čudoviti svet grafov ter kako v njem uporabimo prej omenjene metode načrtovanja algoritmov.

In, ko mislimo, da smo že našli zlato, se nam ponovno skrije v naslednjo deželo ter zato zvedavo pogledamo v deželo nedeterminizma, NP-polnosti in naključnostnega reševanja problemov. Še bomo lahko uživali v našem zvedavem potovanju.

Vodiči potovanja in nadaljnji itinerar

Naloge v pričujoči zbirki so nastajale kot naloge na kolokvijih in pisnih izpitih od leta 2009 naprej. V tem času se je izmenjala vrsta asistentov, ki so občasno pogledali in komentirali naloge ter dali s tem svoj pečat. Zahvalo zaslužijo prizadevni asistenti Matevž Jekovec, Iztok Lapanja, Bojan Klemenc, mag. David Paš, Tine Šukljan in dr. Lan Žagar.

Še tako dobri vodiči so se nekje naučili svojega dela, ali kot sta slikovito dejala Bernard iz Chartresa in sir Isaac Newton, ko sta razlagala izvor svojega uspeha:

> Nanos gigantum humeris insidentes.
>
> Bernardus Carnotensis, 12. stoletje

> If I have seen further it is by standing on the shoulders of Giants.
>
> sir Isaac Newton, 1675

Spisek tistih, ki so nama naklonili svoja ramena je dolg, a vendar naj omeniva vsaj tri, ki so naju vodili skozi deželi podatkovnih struktur in algoritmov – Boštjan Vilfan, Tomaž Pisanski in J. Ian Munro.

Slovenski ljudski pregovor pravi, da *kdor dela, dela tudi napake* in tako so se verjetno tudi nama seveda nehote prikradle v zbirko. Hvaležna bova, če nama boste napake, ki jih najdete, posredovali. Pa naj gre za slovnične, oblikovne ali vsebinske napake.

Na koncu a nikakor nazadnje še posebna in osebna zahvala za neskončno potrpežljivost in podporo pri najinem iskanju velja najinim najbližjim. Zahvala najinima kraljicama Piji in Antoniji ter princesama in princema Heleni in Lauri ter Erazmu in Jakobu.

Draga zvedava bralka in bralec želiva vama veliko zanimivih trenutkov in da vaju bo pot prevzela tako kot je prevzela Roberta Servicea.

Andrej Brodnik
Rok Požar

Medno, Izola, veliki traven 2023

Poglavje 1

Osnove

Ta del nas želi spodbuditi k razmišljanju o načrtovanju in analiziranju algoritmov. Prvi pogoj pri načrtovanju algoritma je, da jasno razumemo problem, ki ga rešujemo. Nato je potrebno izbrati primerno strategijo, na podlagi katere algoritem razvijemo. Sledi formalen dokaz pravilnosti algoritma. To pomeni, da je potrebno pokazati, da algoritem za vsak možen vhodni podatek vrne pravilen rezultat. Pogosto uporabljena metoda pri dokazovanju pravilnosti algoritma sloni na pojmu zančne invariance, oziroma splošneje na principu popolne matematične indukcije.

Naslednji korak je analiza algoritma, pri čemer nas zanima, koliko časa in prostora algoritem potrebuje, da vrne rešitev. Oceno kompleksnosti algoritma izrazimo s funkcijo, ki opisuje časovno oziroma prostorsko zahtevnost algoritma v odvisnosti od velikosti vhodnega podatka. Zato da lahko med seboj primerjamo različne algoritme definiramo družine funkcije, ki jih označujemo z znaki O, Ω, oziroma Θ.

Nenazadnje nas zanima, ali se da algoritem izboljšati. Slednje vprašanje je povezano z oceno zahtevnosti problema. Drugače povedano, zanima nas, vsaj koliko časa/prostora potrebuje katerikoli algoritem v izbranem računskem modelu, da reši problem.

Cilj

Da znamo slediti izvajanju algoritma, dokazati njegovo pravilnost in oceniti njegovo časovno ter prostorsko zahtevnost.

Naloga 1. *Uvod in osnove.* Vprašanja:

1. Zapišite program, ki poišče drugo najmanjše število v polju `A`, in dokažite pravilnost vašega programa.

2. Recimo, da je Peter Zmeda napisal program `Drugi(A)`, ki vrne indeks v polju `A`, kjer se nahaja drugi najmanjši element. Časovna zahtevnost programa `Drugi(A)` je $T_D(n)$, kjer je n dolžina polja `A`. (i) Uporabite program `Drugi()`, da poiščete tretji najmanjši element v polju `A`. Pri tem ne smete iz polja izbrisati nobenega elementa tako, da zmanjšate velikost polja. (ii) Kakšna je časovna zahtevnost vaše rešitve? Utemeljite odgovor.

3. Za naslednje funkcije

$$n^{11},\ \frac{1}{n}+13,\ n\cdot 3^{\sqrt{n}}+n,\ \frac{n^{2,4}}{\log n}$$

zapišite, katerim od naslednjih družin funkcij

$$O(1), O(2^n), \Omega(n\log n), O(e^n), O(n^3), \Omega(n^{\frac{1}{3}}), O(\log\log n)$$

pripadajo.

Namig: Za vsako od funkcij je možno, da pripada večim družinam.

Naloga 2. *Uvod in osnove.* Imamo naslednjo funkcijo:

```
1 int bla(int n){
2   if n == 0 return 0
3   if n == 1 return 1
4   return bla(n-1) + bla(n-2) + 1
5 }
```

Vprašanja:

1. Kaj izpiše funkcija za naslednje vrednosti n: 5, 6, 7, 8, 9, 10 in 11?

2. Zapišite rekurzivno enačbo za časovno zahtevnost funkcije `bla` v odvisnosti od n in jo utemljite.

3. Izračunajte časovno in prostorsko zahtevnost funkcije v obliki $O(\ldots)$.

Namig: Pri dokazu časovne zahtevnosti najprej ocenite zahtevnost in nato uporabite indukcijo. Časovna (!) zahtevnost je večja kot polinomska.

4. [Dodatno] Na predavanjih smo večkrat omenjali, da lahko pospešimo kakšen algoritem, če uporabimo več prostora. Ali lahko to naredite za zgornji primer?

Naloga 3. *Uvod in osnove.* Imamo naslednji program:

```
int array foo(int array a) {
  int n= a.length();
  for i= 1 to n-1
    for j= i+1 to n
      if a[i] > a[j] {
        x= a[i]; a[i]= a[j]; a[j] = x;
      }
  return a;
}
```

Vprašanja:

1. Opisno utemeljite, kaj dela zgornji program.

2. Kakšna je invarianca pred začetkom notranje zanke? Utemeljite odgovor.

3. Kakšna je časovna zahtevnost programa? Dokažite odgovor.

Naloga 4. *Uvod in osnove.* Imamo naslednjo funkcijo:

```
int foo(int x, y)
  if x == 0 return y
  y= 2*y + x % 2
  return foo(x \ 2, y)
}
```

Vprašanja:

1. Sledite izvajanju funkcije za naslednje klice: `foo(5, 0)`, `foo(6, 0)` in `foo(7, 0)`.

2. Kakšna je časovna zahtevnost programa pri klicu `foo(n, 0)`? Utemeljite svoj odgovor.

3. Imamo funkciji $f_1(n) = 12n + 11 \lg n$ in $f_2(n) = 11/n + 12 \lg n$. Katerim izmed naslednjih družin pripada vsaka od funkcij: $O(n)$, $\Omega(n)$, $\Theta(n)$, $O(\log n)$, $\Omega(\log n)$ in $\Theta(\log n)$? Utemeljite svoj odgovor.

Naloga 5. *Uvod in osnove.* Imamo naslednji program:

```
1  foo(int array a) {
2    int n= a.length; // dolzina polja
3    int b= a[n-1];
4    int j= 0;
5    for int i= 0 to n - 2 {
6      if a[i] < b {
7        c= a[i]; a[i]= a[j];  a[j]= c;
8        j= j + 1
9      }
10   }
11   c= a[j]; a[j]= a[n-1]; a[n-1]= c;
12 }
```

Vprašanja:

1. Opisno utemeljite, kaj dela zgornji program.

2. Kakšna je invarianca pred začetkom notranje zanke? Utemeljite odgovor.

3. Kakšna je časovna zahtevnost programa? Dokažite odgovor.

Naloga 6. *Uvod in osnove.* Vprašanja:

1. (i) Napišite funkcijo, ki kot parameter sprejme polje `a[0..n-1]` celih števil in vrne *drugi* največji element. (ii) Kakšna je časovna zahtevnost vaše rešitve? Utemeljite odgovor.

Namig: Več točk boste dobili, če boste natančno izračunali število primerjav in ne samo podali družine funkcij $O()$.

2. Pokažite, v kateri od naslednjih družin funkcij

$$O(n \log n),\ \Omega(n \log n),\ \omega(n \log n),$$

$$O(n),\ \Omega(n),\ \omega(n),$$

$$O(\log n),\ \Omega(\log n)\ \text{ in }\ \omega(\log n)$$

je funkcija, ki popisuje časovno zahtevnost vaše funkcije.

3. Pokažite pravilnost delovanja vaše funkcije.

Naloga 7. *Uvod in osnove.* Vprašanja:

1. (i) Napišite funkcijo, ki kot parameter sprejme polje `a[0..n-1]` celih števil in vrne *število sodih števil* v polju. (ii) Kakšna je časovna zahtevnost vaše rešitve? Utemeljite odgovor. (iii) Ali je rezultat kaj različen, če merite primerjave ali če merite dostope do pomnilnika? Komentirajte rezultat.

Namig: Več točk boste dobili, če boste natančno izračunali število primerjav in ne samo podali družine funkcij $O()$.

2. Pokažite, v kateri od naslednjih družin funkcij

$$O\left(\frac{n}{\log n}\right),\ \Omega\left(\frac{n}{\log n}\right),\ \omega\left(\frac{n}{\log n}\right),$$

$$O(n),\ \Omega(n),\ \omega(n),$$

$$O(\log n),\ \Omega(\log n)\ \text{ in }\ \omega(\log n)$$

je funkcija, ki popisuje časovno zahtevnost vaše funkcije.

3. Pokažite pravilnost delovanja vaše funkcije.

Namig: Zelo verjetno boste uporabili indukcijo in pri tem invarianco zanke.

Naloga 8. *Uvod in osnove.* Peter Zmeda je našel listek z naslednjo kodo:

```
1  int KrNeki(A, k) {
2    for j:= 1 to k {
3      for i:= j to len(A)-1 {
4        if A[i] < A[j-1] {
5          x:= A[j-1]; A[j-1]:= A[i]; A[i]:= x
6        }
7      }
8    }
9    return A[k-1]
10 }
```

Vprašanja:

1. (i) Kakšna je časovna zahtevnost funkcije `KrNeki` v odvisnosti od parametrov $n =$ `len(A)` in `k`? Utemeljite odgovor. (ii) Ali je rezultat kaj različen, če merite primerjave ali če merite dostope do pomnilnika? Komentirajte rezultat.

NAMIG: Več točk boste dobili, če boste natančno izračunali število primerjav in ne samo podali družine funkcij $O()$.

2. (i) Kaj vrne (izračuna) funkcija `KrNeki`? Utemeljite odgovor. (ii) Kakšna je zančna invarianca notranje `for` zanke? (iii) Pokažite, da jo funkcija ohranja.

3. Pokažite, v kateri od naslednjih družin funkcij

$$O\left(\frac{n^2}{\log n}\right),\ \Omega\left(\frac{n^2}{\log n}\right),\ \omega\left(\frac{n^2}{\log n}\right),$$

$$O(n^2),\ \Omega(n^2),\ \omega(n^2),$$

$$O(\log n),\ \Omega(\log n)\ \text{ in }\ \omega(\log n)$$

je funkcija, ki popisuje časovno zahtevnost funkcije `KrNeki`. Pri tem predpostavite, da je $k = n^{\frac{1}{2}}$.

Naloga 9. VPRAŠANJA:

1. Napišite algoritem, ki pri danem celoštevilskem polju `A` dolžine n poišče drugo najmanjše število.

2. Dokažite, da vaš algoritem deluje pravilno.

3. Pokažite, kakšna je njegova prostorska in časovna zahtevnost – prostora, ki ga zasede polje `A`, ne upoštevajte.

4. Ali se dâ najti drugi najmanjši element hitreje? Utemeljite odgovor.

Naloga 10. Imamo naslednje funkcije ($\epsilon < 1$)

$$n,\ \frac{n}{\log n},\ n^{2-\epsilon},\ n^{1-\epsilon}\ \text{ in }\ n\log n \tag{1.1}$$

ter pripadajoče družine funkcij

$$O(n),\ O\left(\frac{n}{\log n}\right),\ O(n^{2-\epsilon}),\ O(n^{1-\epsilon})\ \text{ in }\ O(n\log n). \tag{1.2}$$

VPRAŠANJA:

1. Za vsako družino funkcij iz (1.2) zapišite, katere funkcije iz (1.1) ji pripadajo. Utemeljite odgovor.

Namig: Pomagajte si s tranzitivnostjo.

2. Naš prijatelj Peter Zmeda je na razprodaji kupil izredno učinkovito implementacijo sklada `superSklad`. Implemetacija nudi običajne metode `Push`, `Pop` in `Top` ter metodo `Make`, ki naredi nov sklad. Vse metode imajo časovno zahtevnost $O(1)$.

Edina težava, ki jo Peter sedaj ima, je, da mu njegov šef ni naročil, da kupi sklad, ampak vrsto, ki bo imela operacije `Enqueue`, `Dequeue`, `First` in `Make`.

Kako naj si Peter pomaga s skladom? Utemeljite odgovor.

3. Kakšna je časovna zahtevnost operacij narejene vrste v najslabšem in v amortiziranem smislu? Utemeljite odgovor.

Naloga 11. *Osnove.* Urejanje z mehurčki je eno najstarejših urejanj:

```
1 int[] Uredi (A):
2   for i = len(A)-1..1:
3     for j = 0..i-1:
4       if A[j] >= A[j+1]:
5         tmp = A[j+1]; A[j+1] = A[j]; A[j] = tmp
6   return A
```

V splošnem naj velja $|\mathtt{A}| = n$.
Vprašanja:

1. (i) Kaj točno počne notranja `for` zanka po `j`? (ii) Dokažite pravilnost svoje trditve.

Namig: Pri dokazu uporabite indukcijo in invarianco zanke. Pozor, notranja zanka je odvisn od `i` in ne od n.

2. (i) Koliko *primerjav* najmanj naredi funkcija `Uredi(A)`? Pokažite, v katerem primeru se to zgodi. (ii) In koliko primerjav največ naredi funkcija `Uredi(A)`? V katerem primeru se to zgodi?

3. (i) Ali zgornja koda urejanja z mehurčki stabilno uredi tabelo `A`? (ii) Pokažite svojo trditev. (iii.) Če ne ureja stabilno, ali se jo da spremeniti, da bo urejala stabilno? Utemeljite svojo trditev.

Naloga 12. Naš prijatelj Peter Zmeda je dobil za nalogo, da napiše algoritem, ki pri danem celoštevilskem polju `A` dolžine n poišče drugo najmanjše število:

```
int DrugoNajmanjse (int A[], int n)
```

Vprašanja:

1. Pomagajte Petru in zapišite omenjeni algoritem.

2. Dokažite, da vaš algoritem deluje pravilno.

3. Pokažite, kakšna je njegova prostorska in časovna zahtevnost – prostora, ki ga zasede polje `A`, ne upoštevajte.

4. Ali se dâ najti drugi najmanjši element hitreje? Utemeljite odgovor.

Naloga 13. *Osnove.* Definirajmo funkcijo

```
int FooBar(c, n):
  int p= 2**n
  result= 1
  while (p > 0):
    result= result * c
    p= p-1
  return result
```

Vprašanja:

1. Kakšna je časovna zahtevnost funkcije `FooBar` v odvisnosti od parametra n? Utemeljite odgovor.

2. Ali lahko pospešite funkcijo `FooBar`? Če da, kako in kakšna je tedaj časovna zahtevnost, in če ne, zakaj ne.

3. Dokažite

$$n \ln n = O(n^{3/2})$$

in

$$n \ln n = \Omega(n^{1/2}).$$

Naloga 14. Imamo naslednji algoritem:

```
KajDela(a, n):
  if n == 0 return new par(neskončno, neskončno)
  if n == 1 return new par(a[0], neskončno)
  if n == 2 {
    if a[0] < a[1] return new par(a[0], a[1])
    else           return new par(a[1], a[0])
  } else {
    par= KajDela(a, n-1);
    if a[n-1] < par.prvi
      return new par(a[n-1], par.prvi)
    else if a[n-1] < par.drugi
      return new par (par.prvi, a[n-1])
    else return par
  }
```

Vprašanja:

1. Kaj izračuna algoritem? Utemeljite odgovor.

2. Dokažite z indukcijo, da algoritem v resnici počne to, kar menite, da počne.

3. Kakšna je časovna zahtevnost algoritma? Utemeljite odgovor.

4. Imamo naslednje razrede funkcij: $O(n \log n)$, $O(n \log \log n)$ in $O(n/\log n)$. Poleg tega imamo še funkcije $7n - 1$, $\sqrt{n} + n$ in $n^\pi/\sqrt{n}$. Katerim razredom pripada posamezna funkcija? Utemeljite odgovor.

Naloga 15. Imamo naslednji program za urejanje (urejanje z mehurčki), katerega koda je prepisana iz wikipedije:

```
procedure bubbleSort(A: list of sortable items)
  repeat
    swapped = false
    for i = 1 to length(A) - 1 inclusive do:
      if A[i-1] > A[i] then
        swap(A[i-1], A[i])
        swapped = true
      end if
    end for
  until not swapped
end procedure
```

Vprašanja:

1. Peter Zmeda je slišal, da algoritem deluje v času $O(n^3)$. Ali je to res? Utemeljite odgovor.

Namig: Najprej izračunajte časovno zahtevnost zgornjega algoritma.

2. Peter prav tako ne verjame povsem wikipediji in njenim podatkom. Pokažite, da zgornji algoritem pravilno uredi števila.

Namig: Uporabite indukcijo na dolžino že urejenega dela seznama števil.

Naloga 16. *Osnove.* Peter Zmeda je našel naslednjo kodo, ki predstavlja urejanje z mehurčki (*bubble sort*):

```
1  def bubbleSort(A):
2    n = A.length()
3    repeat
4      zamenjava= false
5      for i=1 to n-1 do:
6        if A[i-1] > A[i] then:
7          swap(A[i-1], A[i])
8          zamenjava= true
9    until not zamenjava
```

Vprašanja:

1. (i) Kakšna je časovna zahtevnost urejanja z mehurčki v najslabšem in kakšna v najboljšem primeru? (ii) Kakšni morajo biti podatki, da nastopi prva, in kakšni, da nastopi druga?

2. Pa koda sploh deluje pravilno? (i) Zapišite zančno invarianco notranje zanke `for`. (ii) Pokažite, da se le-ta ohranja skozi izvajanje programa.

3. (i) Zapišite kodo za urejanje z izbiro (*selection sort*), ki najprej postavi na pravo mesto največji element, nato naslednjega največjega in tako naprej. (ii) Uporabite kodo za urejanje z izbiro tako, da izboljšate čas izvajanja algoritma za mehurčno urejanje.

Naloga 17. Imamo eno-razsežen prostor in v njem točke $p_1, p_2, \ldots, p_n$.
Vprašanja:

1. Napišite algoritem, ki poišče točki p_i in p_j, ki sta najbolj narazen – med njima je največja razdalja. Pokažite pravilnost vašega algoritma.

2. Kakšna je časovna in kakšna prostorska zahtevnost vašega algoritma? Utemeljite odgovor.

3. Recimo, da imamo tokrat n točk v dvo-razsežnem prostoru.[1] (i) Podajte algoritem, ki sedaj poišče najbolj oddaljeni točki. Razdaljo merimo kot evklidsko razdaljo. (ii) Kakšna je časovna zahtevnost vašega algoritma?

Naloga 18. Imamo celoštevilsko polje `A[1..n]`. Peter bi rad v polju našel najdaljše podzaporedje naraščajočih števil. Recimo, v polju

$$[97, 38, 82, 86, 10, 51, 38, 88, 79, 55, 45, 40]$$

je to podzaporeje $[38, 82, 86]$, ki ima očitno dolžino 3.
Vprašanja:

1. (i) Zapišite algoritem, ki v polju dolžine n poišče najdaljše naraščajoče podzaporedje. (ii) Dokažite njegovo pravilnost.

2. (i) Kakšna je časovna zahtevnost vašega algoritma? (ii) Ali bi lahko poiskali podzaporedje hitreje? Utemeljite odgovor.

3. (i) Naslednje funkcije

$$n^{\frac{3}{2}},\ \frac{n}{\log n^2},\ \frac{n}{\log n},\ n^{1-\epsilon}$$

razvrstite po hitrosti naraščanja. (ii) Utemeljite svoj odgovor.

Naloga 19. Imamo naslednji program (prvi element polja naj ima indeks 0):

```
1 public int foobar(int[] a, int b, int c, int d) {
2   if c < 0    return d;
3   if b > a[c] d++;
4   return foobar(a, b, c-1, d)
5 }
```

[1] Vsaka točka p_i je podana s koordinatama (x_i, y_i).

Vprašanja:

1. Recimo, da imamo polje `a` = $[1, 101, 12, 13, 65, 14, 17]$. Kakšno vrednost vrne klic `foobar(a, 20, 4, 7)`? Utemeljite odgovor.

2. Kaj v resnici počne funkcija? Torej, glede na opis, katera je verjetno najbolj pravilna vrednost parametra `d` ob prvem klicu funkcije? Utemeljite odgovor.

3. Kakšna je časovna zahtevnost algoritma? Dokažite svojo trditev.

Naloga 20. *Osnove.* Peter je našel listek z naslednjo psevdokodo:

```
abcd= ['a', 'b', 'c', č,..., 'ž']
množica KajDela(n):
  i= 1; j= 1; k= 0; rez= { }
  while i < n:
    i= 2*i
    if j mod 2 = k:
      rez= Union (rez, abcd[j])
    if (j+1) = size(abcd):
      k= (k+1) mod 2
    j= (j+1) mod size(abcd)
  return rez
```

V zgornji kodi je `abcd` vektor (polje) 25 črk slovenske abecede, ki se prične na indeksu 0. Slednje pomeni, da je `abcd[0]` = `a`, `abcd[1]` = `b` in tako naprej.
Vprašanja:

1. Kaj vrne klic funkcije `KajDela(300)` in kaj `KajDela(200)`?

2. Kakšna je časovna zahtevnost funkcije `KajDela` v odvisnosti od parametra n? Utemeljite odgovor.

3. Kaj v resnici dela funkcija? Utemeljite odgovor.

Naloga 21. Naslednji program, katerega koda je prepisana iz wikipedije, v urejenem polju `myA` poišče element `myX`:

```
public int myFunction(int[] myA, int myX) {
  int low = 0;
  int high = myA.length - 1;
  int mid;

  while (myA[low] <= myX && myA[high] >= myX) {
    mid = low +
          ((myX - myA[low]) * (high - low)) /
          (myA[high] - myA[low]);
    if (myA[mid] < myX)      low = mid + 1;
    else if (myA[mid] > myX) high = mid - 1;
    else                          return mid;
  }
  if (myA[low] == myX) return low;
  else return -1;
}
```

Vprašanja:

1. Recimo, da imamo polje `myA` $= [1, 12, 13, 14, 17, 65, 101]$ in najprej element `myX` $= 14$ ter nato `myX` $= 100$. Za obe vrednosti simulirajte izvajanje programa in preštejte število primerjanj ključa z elementi polja, pri čemer primerjanji na koncu `while` zanke štejte kot eno (`if`, `else if`, `else`).

2. Program je presenetljivo podoben iskanju z razpolavljanjem. Kje je razlika in zakaj menite je ta razlika? Pojasnite, kako točno deluje.

3. Kakšna, menite, je njegova časovna zahtevnost? Utemeljite odgovor. Natančnejši kot bo rezultat, več točk boste dobili.

Naloga 22. *Osnove, statične in dinamične podatkovne strukture.* Peter Zmeda je dobil tabelo `A` celih števil, ki se indeksirajo od 0 naprej. Dovoljeno je branje števila `A[i]`, kjer $0 \leq i <$ `length(A)`$= n$. Peter mora poiskati najmanjše število v tabeli. Našel je listek:

```
integer KrNeki(x, y) { ?? }
integer Najmanjši(A) {
  n= length(A)
  if n = 1 return {A[n-1]}
  else {return KrNeki(Najmanjši(A[0..n-2]), A[n-1])}
}
```

Zapis v kodi `A[i..j]` naredi novo tabelo, v kateri so elementi stare tabele od vključno i do vključno j. Žal se je na listku pobrisala implementacija funkcije `KrNeki`.

VPRAŠANJA:

1. (i) Pomagajte Petru in dopišite funkcijo `KrNeki`, da bo algoritem res poiskal najmanjši element v tabeli. (ii) Dokažite pravilnost funkcije `Najmanjši` z vašo novo funkcijo `KrNeki`.

2. (i) Kakšna je časovna zahtevnost funkcije `Najmanjši` v odvisnosti od parametra n? Utemeljite odgovor. (ii) Bi lahko s tehniko pomnenja časovno zahtevnost izboljšali? Utemeljite odgovor.

3. (i) Najprej razširite podatkovno strukturo (poleg tabele morda dodajte še kaj), da bo večkratni zaporedni klic funkcije `Najmanjši` čim bolj učinkovit. (ii) Dodajmo še operacijo `Spremeni(A, i, v)`, ki spremeni element `A[i]= v`. Ponovno razširite podatkovno strukturo, da bosta klica učinkovita.[2]

---

[2]Seveda lahko večkrat zapored kličemo eno od operacij in v tem primeru želimo biti čim učinkovitejši.

Poglavje 2

Podatkovne strukture

Organizacija podatkov je ena prvih in najpomembnejših nalog, s katero se razvijalec programske opreme sreča. Način, kako organiziramo podatke v pomnilniku računalnika, skupaj z naborom operaciji, ki jih izvajajmo nad temi podatki, definira podatkovno strukturo.

Pravilno zasnovana podatkovna struktura zmanjša skupno časovno/prostorsko zahtevnost rešitve problema ter poveča njeno razširljivost. Ena podatkovna struktura lahko recimo podpira hitro izvedbo za nekatere operacije in počasno za druge, medtem ko lahko druga podatkovna struktura izvede ravno nasprotno. Preprost primer takšnega usklajevanja je odločitev, ali naj bodo podatki v pomnilniku urejeni ali ne. Vzdrževanje urejenosti podatkov lahko izboljša čas iskanja na račun počasnejšega vstavljanja, medtem ko lahko ohranitev neurejenosti podatkov omogoči hitro vstavljanje na račun počasnejšega iskanja.

V splošnem je zato potrebno skrbno izbrati podatkovno strukturo na podlagi tega, katere operacije se bodo v rešitvi najbolj pogosto uporabljale. V tem poglavju obravnavamo podatkovne strukture slovar, vrsta s prednostjo, številsko drevo in disjunktne množice.

Cilj

Da razumemo različne implementacije obravnavanih podatkovnih struktur in jih znamo razširiti za podporo dodatne funkcionalnosti.

2.1 Slovar

Množice so temeljne tako za računalništvo kot za matematiko. Za razliko od matematičnih množic, ki so nespreminjajoče, lahko množice, nad katerimi

izvajamo operacije, rastejo, se sčasoma skrčijo ali kako drugače spremenijo. Takšne množice imenujemo dinamične.

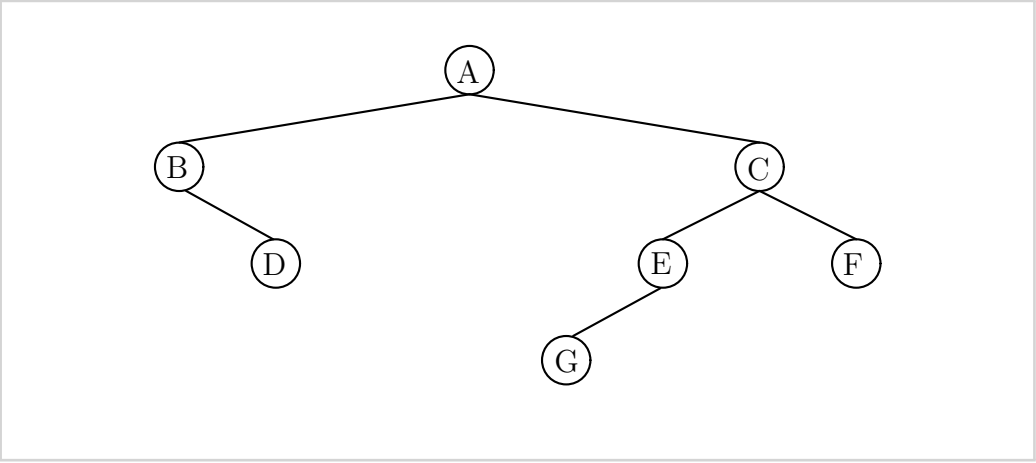
Nad množico lahko izvajamo različne vrste operacij. Dinamični množici, ki podpira vstavljanje elementa, brisanje elementa in poizvedovanje po elementu, oziroma po njegovem ključu, rečemo slovar.

V poglavju se srečamo z nizom izvedb slovarja. Le-te so odvisne od narave ključa, od zahteve po omejenem času odgovora in od zahteve po pravilnosti odgovora. Narava ključa je opredeljena s tem, ali so ključi elementi linearno urejene množice ali ne. Dalje, omejenost časa odgovora je odvisna, ali uporabimo naključnostni ali deterministični pristop. Kar zadeva pravilnosti odgovora obstajajo primeri uporabe, ko smo zadovoljni, če je odgovor občasno napačen.

Cilj

Da se seznanimo z različnimi konkretnimi implementacijami slovarja kot so: povezan seznam, iskalno drevo (dvojiško iskalno drevo, AVL-drevo, B-drevo, rdeče-črno drevo), razpršilna tabela, Bloomov filter in preskočni seznam.

Naloga 23. Imamo drevo, kot ga prikazuje sl. 2.1.



Slika 2.1: Primer dvojiškega drevesa.

Vprašanja:

1. Izpišite zaporedje vozlišč, ki jih dobimo pri vmesnem obhodu (*inorder*) drevesa na sl. 2.1.

2. Drevo lahko rekonstruiramo, če imamo na voljo dva obhoda. Naj bo `v[1..n]` zaporedje vozlišč, ki jih dobimo pri vmesnem obhodu in `p[1..n]` zaporedje vozlišč, ki jih dobimo pri prvem (*preorder*) obhodu. Zapišite algoritem, ki iz zaporedji `v[...]` in `p[...]` rekonsturira drevo. Psevdokoda bo povsem dovolj. Na primer:

- Vozlišče `p[1]` je koren drevesa in vozlišča ... so v levem ter ... v desnem poddrevesu.
- Potem je koren levega ...

3. Utemeljite, zakaj zapis samo enega obhoda ni dovolj za rekonstrukcijo drevesa.

Naloga 24. *Osnove.* Pri pregledovanju internetnega prometa se pogosto sprašujemo, ali je paket z nekega IP naslova že prišel ali se je pojavil prvič. Pri tem se moramo zavedati, da paketi prihajajo zelo pogosto in mora biti naša rešitev predvsem zelo hitra, ne prevelika in po možnosti čim natančnejša (včasih se lahko zmoti).
Vprašanja:

1. Zamislite si rešitev vključno s podatkovno strukturo in jo opišite čim podrobneje.

Namig: Vaša rešitev mora zagotavljati zahteve iz opisa naloge. Oziroma, v njenem opisu morate utemeljiti, da jih zagotavlja in kako, sicer ne boste dobili vseh točk. Za vse točke tudi ocenite časovno in prostorsko zahtevnost vaše rešitve.

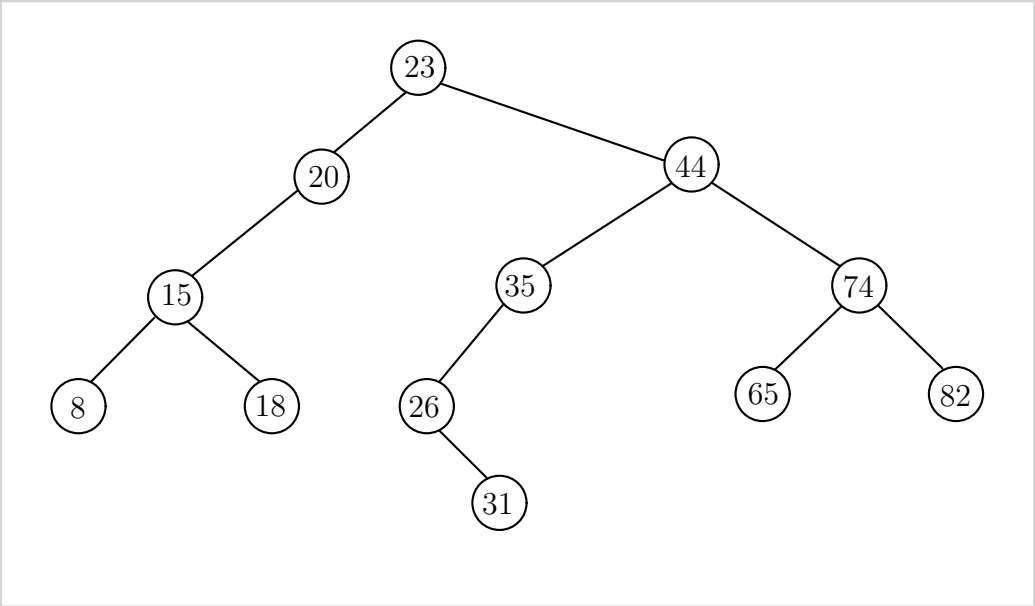
2. Vašo rešitev bo Peter Zmeda uporabil pri svojem IDS sistemu (*Intrusion Detection System*). Da jo bo lahko uporabil, mora rešitev podpirati funkciji:

```
1 void Initi(n);
2 bool AlreadySeen(unsigned int IPaddress);
```

Parameter n pri inicializaciji pove, koliko največ različnih IP naslovov lahko pričakujemo, medtem ko naj pri poizvedbi o IP naslovu funkcija vrne `true`, če je paket že kdaj prišel (ali vsaj meni, da je prišel) oziroma `false`, če ni (oziroma meni, da ni).

3. V resnici so IDS sistemi malce zapletenejši. Tako funkcija `AlreadySeen` vrne `true` samo, če je bil paket viden v zadnjih t sekundah. Kako boste nadgradili vašo rešitev, da bo pravilno odgovarjala ob spremenjeni zahtevi?

Naloga 25. Na sl. 2.2 imamo primer iskalnega dvojiškega drevesa. Vsako in



Slika 2.2: Iskalno drevo.

tudi iskalno drevo lahko izpišemo na različne načine. Recimo, pri vmesnem obhodu drevesa na sl. 2.2 dobimo izpis 3, 15, 18, 20, 23, 26, 31, 35, 44, 65, 74, 82.

Vprašanja:

1. (i) Izpišite drevo s sl. 2.2 po plasteh. (ii) Napišite algoritem, ki poljubno dvojiško drevo izpiše po plasteh.

2. Naš prijatelj Peter Zmeda je pred obratno nalogo. Ima izpis dvojiškega drevesa po plasteh in bi rad rekonstruiral izvorno drevo, za katerega pa niti ne ve, ali je bilo iskalno. (i) Ali lahko vedno rekonstruira drevo? (ii) Utemeljite odgovor.[1]

Namig: Če menite da lahko, zapišite algoritem in pokažite njegovo pravilnost ter če ne, predstavite konkreten primer, ko se to ne da.

3. Se vaš odgovor na prejšnje vprašanje kaj spremeni, če Peter vé, da je drevo iskalno? Utemeljite odgovor.

[1]Podobne probleme rešujemo pri postopku prenosa (`marshaling`). Ne nazadnje, to izvaja entitetni par na predstavitveni plasti ISO-OSI modela.

Naloga 26. *Obiskovanje dreves.* Naš prijatelj Peter Zmeda je slišal, da imamo različne obhode po drevesu: a) premega, kjer najprej obiščemo vozlišče in nato poddrevesa; b) vmesnega, kjer obiščemo vozlišče med obiskoma poddreves[2]; in c) obratnega, kjer najprej obiščemo vsa poddrevesa ter na koncu samo vozlišče.
VPRAŠANJA:

1. Peter že nekaj časa opazuje definicije in se sprašuje, ali lahko pri obhodu dvojiškega drevesa dobimo enak vmesni in obratni obhod. (i) Narišite dvojiško drevo z desetimi vozlišči, ki ima enak vmesni in obratni obhod. (ii) Sedaj se Peter sprašuje, koliko največ vozlišč ima dvojiško drevo, ki ima enak premi in obratni obhod. Najdite odgovor in ga utemeljite.

2. Poleg opisanih obhodov je Peter našel še definicijo obhoda po plasteh: najprej koren, nato njegovi otroci, pa vnuki in tako naprej. (i) Za vaše drevo iz prve polovice prvega vprašanja te naloge izpišite vozlišča po plasteh. (ii) Kaj opazite? (iii) Zapišite algoritem, ki bo izpisal k-tiško drevo po plasteh.

3. (i) Kakšna je časovna zahtevnost vašega algoritma? Utemeljite odgovor. (ii) Opišite dvojiško drevo, ki ima enake vmesni obhod, premi obhod in obhod po plasteh. Utemeljite svoj odgovor.

Naloga 27. Dvojiško drevo z enim samim vozliščem je višine 1.
VPRAŠANJA:

1. Koliko največ in koliko najmanj vozlišč ima dvojiško drevo višine 2016? Utemeljite odgovora.

2. Zapišite funkcijo `Visina(t)`, ki sprejme kot parameter drevo `t` in vrne njegovo višino.

3. V Petrovem podjetju *ButaSoft* se je pred kratkim zaposlil tudi mladi Arnold. Peter nekako ne zaupa povsem Arnoldovi implementaciji AVL dreves in se je odločil napisati program, ki preveri, ali je drevo `t` v resnici AVL drevo. Pomagajte mu in napišite funkcijo `JeAVL(t)`, ki vzame kot parameter urejeno dvojiško drevo `t` in vrne TRUE, če je drevo AVL (uravnoteženost!) in FALSE sicer.

[2] V primeru k-tiških dreves koren obiščemo celo večkrat.

Naloga 28. *Drevesa.* Na predavanjih smo spoznali tri vrste obhodov dreves. V tej nalogi bomo imeli opravka samo z dvojiškimi drevesi.
VPRAŠANJA:

1. (i) Ali obstaja dvojiško drevo, pri katerem ob vmesnem (*in-order*) obhodu dobimo zaporedje števil

$$5, 8, 66, 31, 27, 75, 29, 62, 36?$$

Utemeljite odgovor. (ii) Ali obstaja dvojiško drevo, pri katerem ob obratnem (*post-order*) obhodu dobimo zgornje zaporedje števil? Utemeljite odgovor.

2. (i) Ali je možno, da pri premem (*pre-order*) obhodu dveh različnih dvojiških dreves dobimo enaki zaporedji števil? Utemeljite odgovor. (ii) Recimo, da imamo izpisa premega in vmesnega obhoda drevesa. Zapišite algoritem, ki na podlagi obhodov rekonstruira drevo.

3. Kakšna je časovna zahtevnost vašega algoritma? Utemeljite odgovor.

Naloga 29. Poldinamične podatkovne strukture so strukture, ki poleg poizvedovanj omogočajo samo vstavljanja in ne brisanj. VPRAŠANJA:

1. Definirajte podatkovno strukturo poldinamični slovar.

NAMIG: Zapišite funkciji/metodi, ki ga sestavljata.

2. Opišite učinkovito implementacijo poldinamičnega slovarja. Pri opisu utemeljite (analizirajte) učinkovitost delovanja.

NAMIG: Poudarek je na *učinkovitosti*, ki upošteva lastnost poldinamičnosti. Če boste samo uporabili neko implementacijo slovarja boste dobili pol točk ali manj. Upoštevajte tako časovno kot prostorsko zahtevnost.

3. Sedaj pa obratno. Recimo, da imate učinkovito implementacijo poldinamičnega slovarja. Kako bi z njegovo pomočjo realizirali celoten slovar? Ocenite časovno in prostorsko zahtevnost kot funkcijo časovne in prostorske zahtevnosti poldinamičnega slovarja.

NAMIG: Tukaj ne smete predpostaviti, da lahko kakorkoli popravljate kodo poldinamičnega slovarja.

Naloga 30. *Slovar.* V Butalah se zelo radi igrajo naslednjo spominsko igrico, v kateri sodelujeta $n + 2$ igralca. Vsak od prvih n igralcev po vrsti pove naključno številko x_i. Naslednji igralec, $(n + 1)$-vi, pove naključno številko q in zadnji igralec odgovori DA, če je kateri od $x_i = q$, oziroma NE. Naj bo $n = 5$ in $x_1 = 39$, $x_2 = 12$, $x_3 = 27$, $x_4 = 90$ in $x_5 = 25$. Če je $q = 9$, potem je odgovor NE, če pa je $q = 39$, pa je odgovor DA. VPRAŠANJA:

1. Zadnji igralec mora uporabiti nekakšno podatkovno strukturo, da bo porabil skupno čim manj časa za beleženje x_i in odgovarjanje na vprašanje q. (i) Katere operacije mora podpirati struktura? (ii) Predlagajte čim bolj učinkovito podatkovno strukturo. Utemeljite odgovor. (iii) Kakšna je časovna zahtevnost posameznih operacij in kakšna je časovna zahtevnost vseh operacij skupaj? Utemeljite odgovor.

2. Ali bi vašo podatkovno strukturo kaj spremenili, če veste, da velja $x_i \in [1, 2, \ldots, M]$? Utemeljite odgovor.

NAMIG: Pri utemeljitvi predvsem razmislite o (iii) v podvprašanju (1).

3. V Butalah so igrico malce spremenili in sicer sedaj odgovor na vprašanje q ni več DA oziroma NE, ampak vrednost prvega večjega števila od q med vsemi x_i. Če takšnega števila ni, je odgovor -1. V našem primeru je sedaj pri $q = 9$ odgovor 12 in pri $q = 39$ je odgovor 90. (i) Kakšno podatkovno strukturo predlagate tokrat? Utemeljite odgovor. (ii) Kakšna je tokrat časovna zahtevnost posameznih operacij in kakšna je časovna zahtevnost vseh operacij skupaj? Utemeljite odgovor.

Naloga 31. *Statistika.* Ena ključnih operacij za izdelavo statistike je štetje velikosti vzorca. V tej nalogi boste predlagali podatkovne strukture, ki nam bodo omogočale dovolj dobro štetje velikosti vzorca in morale bodo delovati učinkovito.
VPRAŠANJA:

1. V Sloveniji imamo 11 registrskih območij (MS, MB, CE, NM, KK, SG, LJ, KR, PO, GO in KP). Na cesti bi radi šteli, koliko vozil iz posameznega registrskega območja se je peljalo mimo našega števca. (i) Predlagajte ustrezno podatkovno strukturo za štetje. (ii) Utemeljite odgovor. (iii) Kakšna

je prostorska zahtevnost vaše strukture? (iv) Kakšna je časovna zahtevnost operacij `Pristej(oznaka)`, ki poveča števec za oznako `oznaka` za 1, in `Koliko(oznaka)`, ki vrne trenutno vrednost števca za oznako `oznaka`.

2. Tokrat bomo šteli promet na požarni pregradi. Bolj natančno, radi bi vedeli, če je do požarne pregrade prišel kakšen paket z določenega IP naslova. (i) Ponovno predlagajte ustrezno podatkovno strukturo za štetje. (ii) Utemeljite odgovor. (iii) Kakšna je prostorska zahtevnost vaše strukture? (iv) Kakšna je časovna zahtevnost operacij `Prišel(ip)`, ki v strukturi zabeleži, da je prišel paket z danega `ip` naslova, in `JePrisel(oznaka)`, ki vrne, ali je prišel paket z danega `ip` naslova.

3. (i) Nadgradite podatkovno strukturo iz prejšnjega vprašanja tako, da bo tudi štela število paketov, ki pridejo z določenega `ip` naslova. (ii) Kakšna je prostorska zahtevnost vaše strukture? (iii) Kakšna je časovna zahtevnost operacij `Pristej(ip)` in `Stevilo(ip)`?

Naloga 32. *Osnove.* Definirajmo funkcijo `vsebovan(p, t)`, ki ugotovi, če so vsi znaki niza `p` vsebovani v nizu `t` v enakem vrstnem redu. Poleg tega definirajmo še funkcijo `niPresek(p, t)`, ki vrne znake, ki niso hkrati v `p` in `t`. Pri obeh funkcijah naj še velja $|p| = m$ in $|t| = n$. Recimo, da imamo polji $\mathtt{t} = [\mathtt{p}, \mathtt{o}, \mathtt{s}, \mathtt{t}, \mathtt{a}, \mathtt{j}, \mathtt{a}]$ in $\mathtt{p} = [\mathtt{s}, \mathtt{a}, \mathtt{t}]$, potem `vsebovan(`$p$`, `$t$`)` vrne FALSE in `niPresek(`$p$`, `$t$`)` vrne $\{\mathtt{p}, \mathtt{o}, \mathtt{j}\}$.

VPRAŠANJA:

1. (i) Zapišite funkcijo `vsebovan(p, t)` za poljubna p in t. (ii) Kakšna je časovna zahtevnost vašega algoritma? Utemeljite odgovor.

2. (i) Zapišite še funkcijo `niPresek(p, t)` za poljubna p in t. (ii) Kakšna je časovna zahtevnost tega vašega algoritma? Utemeljite odgovor.

NAMIG: Upoštevajte, da so črke nizov p in t lahko poljubna števila.

3. Sedaj predpostavimo, da so črke v obeh nizih iz končne množice $\Sigma = \{a_1, a_2, \ldots, a_s\}$, kjer $s \ll n, m$. (i) Kako pa sedaj izgleda vaša funkcija `niPresek(p, t)`? (ii) Kakšna je časovna in prostorska vaše nove funkcije?

Naloga 33. *Grafi.* Odnose med ljudmi lahko modeliramo s pomočjo grafa. V Butalah so posebej ponosni, da gojijo dva odnosa in sicer: *poznam* in *najboljši prijatelj* (oziroma na kratko *prijatelj*). Odnos A poznam B pomeni, da

oseba A pozna osebo B, ni pa nujno res obratno. Po drugi strani pa odnos A prijatelj B pomeni, da je B najboljši prijatelj A ter A in B se seveda vzajemno poznata. Seveda, nekdo ima lahko samo enega najboljšega prijatelja in slednji mora obstajati.

V Butalah so ponudili svojim občanom novo storitev *ButBum* (*Butalski Album*), v katero vsak Butalec vpisuje, kdo je njegov najboljši prijatelj.
VPRAŠANJA:

1. Za izvedbo ButBum storitve so se dogovorili s Petrom Zmedo. Od njega pričakujejo podporo za operaciji `DodajPrijatelja(A, B)` in `Prijatelj(A)`. Prva operacija `DodajPrijatelja(A, B)` doda najboljšega prijatelja B osebe A. Pri tem morata tako A kot B obstajati. Poleg tega, če je oseba A že imela najboljšega prijatelja C, le-ta to ni več. Druga operacija `Prijatelj(A)` vrne najboljšega prijatelja osebe A. Če oseba A ne obstaja, ali če obstaja in nima najboljšega prijatelja, vrne NULL. (i) Predlagajte podatkovno strukturo, ki jo naj Peter uporabi za implementacijo in podajte implementacijo obeh operacij. Utemeljite pravilnost vaše odločitve in kode. (ii) Kakšna je časovna zahtevnost vaših operacij? Utemeljite odgovor. (iii) Recimo, da bi odnos *prijatelj* modelirali z grafom. Kakšen je ta graf? Utemeljite odgovor.

NAMIG: Odločitev običajno utemeljujemo s tem, da je učinkovita.

2. Stvar zaključena. Skoraj. Butalci so namreč prišli na idejo, da bi razširili ButBum z dodatno storitvijo in sicer `Povabi(A, besedilo)`, ki pošlje povabilo na rojstnodnevno zabavo osebe A z besedilom `besedilo`. Ker so Butalci družabni ljudje, naj bi povabilo prejel najboljši prijatelj osebe A, nato najboljši prijatelj tega najboljšega prijatelja in tako naprej. (i) Implementirajte novo operacijo. (ii) Kakšna je njena časovna zahtevnost? Utemeljite odgovor.

3. Uspešno podjetništvo je neskončen vir novih idej. Sedaj bi radi pri ButBum dodali še tretjo storitev in sicer podporo za odnos *pozna*. (i) Predlagajte podatkovno strukturo za podporo nove storitve in utemljite odgovor. (ii) Ali lahko pri ButBum za zagon nove storitve kako uporabijo podatke, ki jih že imajo, s tem, da ponujajo storitev *najboljši prijatelj*? Utemeljite odgovor.

NAMIG: Natančnejši kot bo vaš odgovor, več točk boste dobili.

Naloga 34.
VPRAŠANJA:

1. Na predavanjih smo spoznali dva opisa obnašanja funkcij: Ω in O. Recimo, da velja: $f(n) = O(n \log n)$. Katera/katere od naslednjih izjav *ni/niso* pravilna/pravilne:

$$f(n) = \Omega(n), \quad f(n) = \Omega(n \log n), \quad f(n) = \Omega(n^2)?$$

Utemeljite odgovor.

2. Ena od implementacij slovarja, ki smo jo omenili, je bil neurejen seznam. Opišite primer, ko je ta implementacija učinkovitejša od dvojiškega uravnoteženega iskalnega drevesa.

Naloga 35. *Slovar.* Slovar nastopa v različnih oblikah, vendar moramo obliko izbrati namenu primerno.
Vprašanja:

1. Prva oblika slovarja so drevesa. (i) Koliko največ in koliko najmanj vozlišč je v B-drevesu višine 5 ($b = 4$, število naslednikov)? Utemeljite odgovor.

2. V opisano B-drevo po vrsti vstavite elemente od 1 do 15. (i) Izrišite drevo po vstavljenih prvih petih, desetih in po vseh elementih.

3. (i) Opišite (skicirajte) B-drevo z $b = 4$, ko vanj vstavite elemente od 1 do n. (ii) Koliko vozlišč ima takšno drevo? Utemeljite odgovor.

Naloga 36. Pri preskočnem seznamu, ki smo ga spoznali na predavanjih, velja, da so odseki seznama določene višine dolgi en element – vedno je skupaj samo en ($= 2-1$) element določene višine. Na primer, v preskočnem seznamu z 18 elementi so višine elementov:

$$1, 2, 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2, 1, 5, 1, 2.$$

Takšnemu preskočnemu seznamu lahko rečemo dvojiški preskočni seznam. Podobno lahko definiramo trojiški preskočni seznam, kjer so seznami določene višine dolgi dva ($= 3-1$) elementa. Na primer, višine elementov v preskočnem seznamu z 18 elementi so:

$$1, 1, 2, 1, 1, 2, 1, 1, 3, 1, 1, 2, 1, 1, 2, 1, 1, 3.$$

Vprašanja:

1. Opišite primer problema, ko je primerneje uporabiti preskočni seznam (i) kot AVL drevo, oziroma (ii) kot razpršilno tabelo.

2. Narišite trojiški preskočni seznam z naslednjimi elementi:

74, 16, 37, 27, 17, 82, 6, 43, 50, 67, 13, 40, 87, 98, 49, 71, 39, 9, 38, 14, 53, 91, 64, 94, 21.

Namig: Naloga je *zelo* lahko, le nekaj dela je potrebno vložiti.

3. Napišite podprogram, ki vstavi element x v trojiški preskočni seznam.

Naloga 37. *Slovar in pričakovani čas.* Recimo, da imamo naslednje ključe: 24, 1, 15, 10, 14, 23, 18, 19, 6, 5 in 21. Ključe boste vstavili v razpršilno tabelo in prešteli število vseh dostopov v tabelo, ki jih boste morali pri tem opraviti. To številko potem delite s številom vstavljenih elementov in boste dobili povprečno število dostopov.
Vprašanja:

1. Najprej vstavite ključe v razpršilno tabelo velikosti $m = 15$, pri čemer je tabela na zažetku prazna. Kot razpršilno funkcijo uporabite funkcijo $h(k) = k \bmod m$. V primeru sovpadanja uporabiti odprto naslavljanje z linerano funkcijo. Prikažite (i) posamezne korake pri vstavljanju elementov, (ii) podajte skupno in povprečno število dostopov do podatkovne strukture, (iii) največje in najmanjše število dostopov pri vstavljanju enega elementa ter (iv) velikost celotne podatkovne strukture.

2. Nato vstavite ključe v razpršilno tabelo velikosti $m = 7$, pri čemer je tabela na začetku prav tako prazna. Kot razpršilno funkcijo uporabite ponovno funkcijo $h(k) = k \bmod m$ (m je sedaj drugačen od prejšnjega vprašanja). V primeru sovpadanja uporabiti veriženje z neurejenim seznamom. Ponovno prikažite (i) posamezne korake pri vstavljanju elementov, (ii) podajte skupno in povprečno število dostopov do podatkovne strukture, (iii) največje in najmanjše število dostopov pri vstavljanju enega elementa ter (iv) velikost celotne podatkovne strukture.

3. Primerjajte in komentirajte rezultate obeh vprašanj.

Naloga 38. *Slovar in zagotovljeni čas.* Vprašanja:

1. Tokrat bomo naredili premi obhod dvojiškega iskalnega drevesa. Kakšne vrednosti dobimo kot rezultat obhoda?

2. Peter Zmeda je slišal, da lahko iz urejenega polja dolžine n zelo preprosto v linearnem času naredi uravnoteženo dvojiško iskalno drevo, žal pa je pozabil podrobnosti. Napišite psevdokodo algoritma, ki kot vhodni podatek dobi polje `a[0..n-1]` in v linearnem času ustvari uravnoteženo dvojiško iskalno drevo.

Namig: Lotite se dela rekurzivno: najprej koren, nato levo poddrevo ter na koncu še desno poddrevo.

3. [dodatno] Ali se vam iz odgovorov na zgornji vprašanji porodi kakšna zamisel?

Naloga 39. Vprašanja:

1. Narišite najmanjše AVL drevo višine 5 – to je AVL drevo višine 5, ki ima najmanj elementov.

2. V AVL drevo na sl. 2.3 vstavite element s ključem 5. Narišite drevo po vstavljanju in utemeljite korake.

3. Če vstavljamo ključe v naraščajočem vrstnem redu v AVL drevo, bo višina drevesa monotono naraščala. Dokažite, ali je izjava pravilna ali ne.

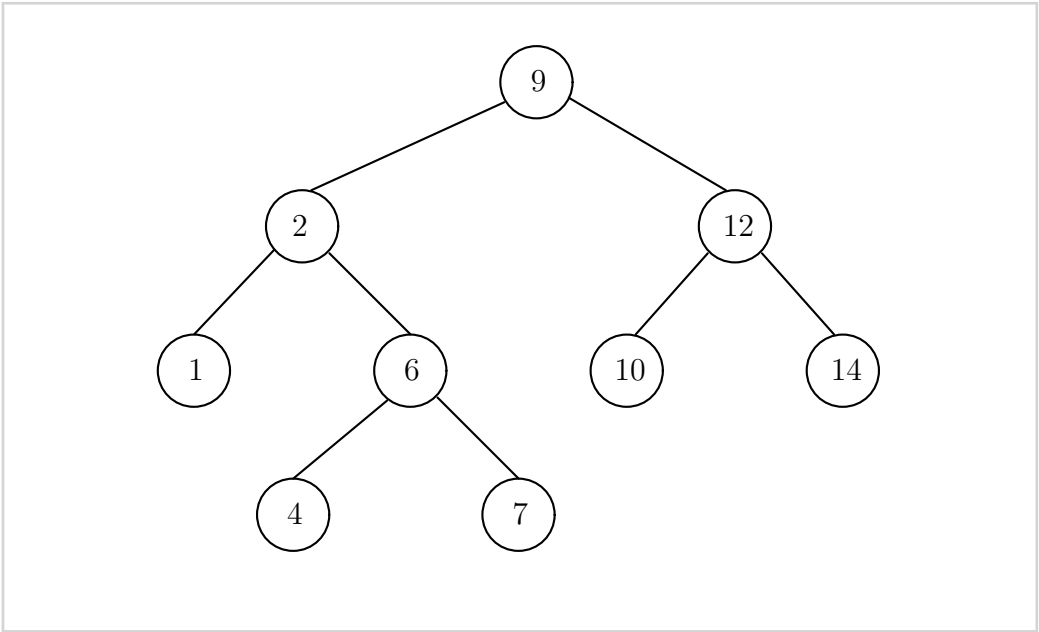
Naloga 40.
Vprašanja:

1. Koliko elementov najmanj je v rdeče-črnem drevesu višine[3] 5? Utemeljite odgovor.

2. Ali je lahko drevo na sliki sl. 2.3 rdeče-črno drevo? Utemeljite odgovor.

3. V 2-3-4 (TTF) drevo po vrsti vstavljajte elemente 1, 2, 3, ... (i) Kako se drevo spreminja? Kaj opazite? (ii) Recimo, da smo v drevo vstavili po vrsti elemente od 1 do n. Kako izgleda drevo? Kako visoko je (ne v O zapisu)?

[3] Drevo s samo enim elementom ima višino 0.



Slika 2.3: Primer drevesa.

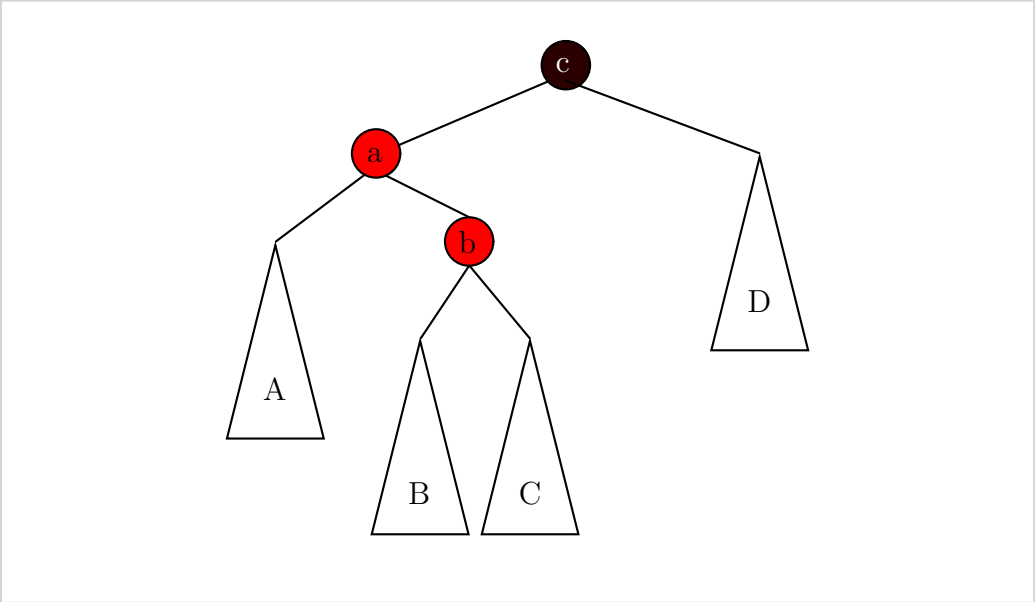
Naloga 41. *Slovar.* Recimo, da imamo množico $\{1, 2, \ldots, n\}$.
Vprašanja:

1. (i) Najprej naj bo $n = 7$ in vstavite po vrsti elemente od 1 do 7 v AVL drevo. Izrišite drevo po vsakem vstavljanju. (ii) Opišite, kako izgleda na koncu drevo, če vstavite elemente od 1 do poljubno velikega n.

2. (i) Narišite AVL drevo višine 5 (če imamo samo en element, je takšno drevo višine 1), ki ima najmanj elementov. (ii) Opišite, kakšno naj bo zaporedje operacij, ki ustvari takšno drevo.

3. Na predavanjih smo govorili o brisanju elementov iz AVL drevesa in profesor je narobe trdil, da je pri brisanju potrebno samo eno popravljanje (vrtenje). (i) Narišite drevo in zapišite, katero brisanje v njem zaheva več popravljanj. (ii) Koliko popravljanj je lahko potrebnih v drevesu višine h? Utemljite odgovor, se pravi, narišite drevo, v katerem so potrebna ta popravljanja.

Naloga 42. V tem vprašanju bomo obravnavali rdeče-črno drevo. Na sl. 2.4 je konfiguracija rdeče-črnega drevesa po tem, ko smo v poddrevo B vstavili



Slika 2.4: Rdeče-črno drevo po vstavljanju v poddrevesu B.

element ter sta vozlišči a in b rdeči ter vozlišče c črno. Poleg tega so koreni poddreves A, B, C in D črni ter tudi črne višine poddreves enake.[4]
Vprašanja:

1. Kako, če je potrebno, se naj spremeni oblika drevesa na sl. 2.4? Narišite in utemeljite odgovor.

2. Zapišite psevdokodo, ki opravi omenjeno preoblikovanje.

3. B-drevo. Naš prijatelj Peter Zmeda je pregledoval kodo za brisanje iz B drevesa in zazdelo se mu je, da bi lahko bila učinkovitejša, če bi dovolil, da je v vsakem vozlišču namesto vsaj $\frac{b}{2}$ elementov $\frac{b}{3}$ elementov. Komentirajte njegovo idejo. Pri tem se osredotočite na zapletenost kode, velikost zasedenega pomnilnika in učinkovitost operacij.

Naloga 43. *2-3-4 drevesa in razpršilne tabele.* Recimo, da imamo naslednje ključe: J E S E N S K I R O K.
Vprašanja:

1. Pokažite, kako se zgornji ključi vstavijo v na začetku prazno 2-3-4 drevo. Po vsakem koraku izrišite sliko strukture – na koncu boste imeli v strukturi po dva E, S in K.

[4]Črna višina poddrevesa je število črnih vozlišč od korena do poljubnega lista.

2. Isto zaporedje ključev vstavite v razpršilno tabelo, pri čemer je tabela na začetku prazna in velika 9 elementov. Na začetku uporabite kot razpršilno funkcijo $h(k) = (k * p) \bmod m$, kjer je $p = 11$ in m velikost tabele. V primeru sovpadanja uporabite veriženje. Kot vrednosti ključev k vzemite ASCII vrednosti zgornjih črk – A = 65, B = 66,... Ponovno, na koncu boste imeli v strukturi po dva E, S in K.

3. Ali lahko katero od uporabljenih struktur uporabimo tudi za stabilno (!!) urejanje (sortiranje)? Utemeljite odgovor.

Naloga 44.
VPRAŠANJA:

1. Koliko elementov najmanj je v AVL drevesu višine[5] 6? Utemeljite odgovor.

2. Ali je lahko drevo na sliki sl. 2.3 AVL drevo? Utemeljite odgovor.

3. Imamo n elementov, ki jih najprej vstavimo v AVL drevo in nato še v rdeče-črno drevo. Katero od dreves je lahko višje? Odgovor utemeljite.

Naloga 45. VPRAŠANJA:

1. Narišite najvišje in najnižje AVL drevo s 15 vozlišči.

2. Kakovost rekurzivne podatkovne strukture je definirana z najdaljšo potjo od začetka do konca strukture – na primer, od korena drevesa do njegovega najbolj oddaljenega lista ali od začetka seznama do njegovega zadnjega elementa.

Opišite primer operacij nad preskočnim seznamom, ko je ta pot še posebej dolga. Utemeljite odgovor.

3. Recimo, da imamo besedilo $t = a_1a_2a_3 \dots a_n$, kjer so a_i iz končne abecede Σ. Peter Zmeda bi rad za besedilo t naračunal Parikhove vrednosti, kar pomeni, da bi rad za vsako črko a iz Σ preštel število njenih pojavitev v besedilu t. Recimo, v besedilu ABRAKADABRA se A pojavi štirikrat in tako naprej.

Očitno gre za slovar vseh črk, kjer je črka ključ in število njenih pojavitev podatek. Predlagajte učinkovito izvedbo slovarja in začrtajte algoritem, ki izračuna zahtevane vrednosti.

---

[5]Drevo s samo enim elementom ima višino 0.

2.2 Vrste s prednostjo

Vrsta s prednostjo je podatkovna struktura za vzdrževanje elementov, izbranih iz linearno urejene univerzalne množice, in podpira osnovne operacije vstavljanje elementa, iskanje najmanjšega elementa in brisanje najmanjšega elementa.

Vrsta s prednostjo se uporablja v rešitvah, kjer je potreben učinkovit dostop do najmanjšega elementa, kot so na primer rešitve, ki uporabljajo požrešni pristop. Tipičen primer njene uporabe je Dijkstrov algoritem za iskanje najkrajših poti v uteženem grafu ali Primov algoritem za računanje najcenejšega vpetega drevesa v uteženem grafu.

Zanimiva je tudi uporaba vrste s prednostjo pri izvedbi urejanja z izbiro, kjer v vsaki iteraciji poiščemo naslednji najmanjši element. Prav vrsta s prednostjo omogoča njegovo učinkovito iskanje in nas privede do urejanja elementov s kopico.

Ker vrsta s prednostjo ne podpira splošnega iskanja, so njene izvedbe v splošnem časovno in prostorsko učinkovitejše od izvedb slovarja.

Cilj

Da razumemo učinkovite izvedbe vrste s prednostjo, kot so dvojiška, binomska in Fibonaccijeva kopica.

Naloga 46. Peter Zmeda je slišal, da obstajajo različne vrste kopic kot izvedbe vrst s prednostjo. Tako je slišal, da obstajata Fibonaccijeva in binomska kopica.
Vprašanja:

1. Nad Fibonaccijevo kopico po vrsti naredite naslednje operacije in sproti izrisujte podatkovno strukturo (`I` pomeni vstavi, `M` minimum in `DM` zbriši najmanjši element):

`I` 17, `I` 3, `I` 5, `I` 1, `M`, `I` 10, `DM`, `I` 4, `DM`, `DM`.

2. Iste operacije izvedite še nad binomsko kopice ter ponovno sproti izrisujte izgled strukture.

3. Ali obstaja primer, ko je Fibonaccijeva kopica primernejša od binomske kopice in ali obstaja primer, kjer je binomska kopica primernejša od Fibonacijeve? Oba primera utemeljite.

Naloga 47. *Slovarji tako in drugače.* Ena od oblik slovarja, ki smo jo srečali, je tudi Bloomov filter, ki pa ima samo operaciji `Insert(elt)` in `Find(elt)`. Bloomov filter uporablja bitni vektor, kamor shranjuje vse podatke. Za razliko od običajnega slovarja lahko tukaj večkrat vstavimo isti element. Vprašanja:

1. Tudi naš prijatelj Peter je izvedel za Bloomov filter in se je lotil njegove implementacije. Zataknilo se mu je že pri izbiri razpršilne funkcije. Po tehtnem premisleku in prebiranju dokumentacije je ugotovil, da je najbolje, če naredi razprševanje na enak način kot pri *Cuckoo* razprševanju, ki velja za enega najboljših. Komentirajte njegovo odločitev.

2. Če pri Bloomovem filtru nadomestimo bitni vektor z vektorjem števcev, dobimo števni Bloomov filter. Psevdokoda vstavljanja potem izgleda takole:

```
1 Insert(elt):
2   for i= 1...k:
3     bf[h(i, elt)]++
```

V psevdokodi je `bf[]` vektor števcev in imamo `k` razpršilnih funkcij. Za izračun i-te funkcije nad `elt` uporabimo klic funkcije `h(i, elt)`. Sedaj imamo nad Bloomovim filtrom namesto operacije `Find(elt)`, ki poišče element `elt`, funkcijo `Count(elt)`, ki vrne, kolikokrat je bil element `elt` vstavljen v Bloomov filter. Zapišite psevdokodo funkcije `Count(elt)`, utemeljite njeno pravilnost ter analizirajte njeno časovno in prostorsko zahtevnost.

3. Pri binomski vrsti s prednostjo smo omenjali, da za učinkovito izvedbo operacij `DecreaseKey(&elt, d)` in `Delete(&elt)` potrebujemo neposredno referenco na element v strukturi (zgoraj označeno z `&`). Kaj pa če referenc nimamo in moramo implementirati omenjeni funkciji tako, da dobimo kot parameter sam element `elt`? Predlagajte in utemeljite rešitev.

Namig: Za reševanje tega vprašanja *ne* potrebujemo Bloomovega filtra.

Naloga 48. Min-kopica, ki smo jo srečali na predavanjih, je podatkovna struktura, pri kateri veljajo naslednje lastnosti: (i) najmanjši element je v korenu; (ii) vsi listi so na isti globini ali največ na dveh; (iii) listi so levo poravnani. Min-kopica je primer implementacije vrste s prednostjo. Včasih želimo hitro dostopati ne samo do najmanjšega, ampak tudi do največjega elementa. Zato definiramo minmax-kopico. Pri minmax-kopici veljajo naslednja pravila:

1. plasti v minmax-kopici štejemo od vrha navzdol in jih razdelimo na lihe (koren, njegovi vnuki in tako naprej) ter na sode (otroci korena in pravnuki korena ter tako naprej);

2. v lihih plasteh velja, da je v vozlišču najmanjši element podkopice, v sodih plasteh pa, da je v korenu največji element kopice;

3. vsi listi so na isti globini ali največ na dveh; in

4. listi so levo poravnani.

Recimo, da imamo v minmax-kopici naslednje elemente: 1, 3, 4, 11, 12, 13, 16, 17, 24, 25, 25, 27, 30 in 77. Potem je v korenu element 1, ker je najmanjši, v enem od njegovih otrok je element 77, ker je največji, ter v drugem, recimo, 24. Vprašanja:

1. Razvrstite zgornje elemente v minmax-kopico.

2. Zapišite funkciji `Min(pq)` in `Max(pq)`, ki vrneta največji in najmanjši element v kopici `pq`. Za vsako izmed funkcij utemeljite pravilnost njenega delovanja in analizirajte njeno časovno zahtevnost.

3. Zapišite funkcijo `DelMax(pq)`, ki iz kopice `pq` izbriše največji element. Utemeljite njeno pravilnost in analizirajte njeno časovno zahtevnost.

Namig: Morda je preprostejša funkcija `DelMin(pq)`? Če se vam zdi, zapišite to funkcijo ter utemeljite njegovo pravilnost in analizirajte njeno časovno zahtevnost. Dobili boste sicer nekaj manj točk.

Naloga 49. Pri urejanju s kopico uporabljamo naslednjo metodo za izgradnjo kopice nad poljem `a[0..n-1]`:

```
1 function Skopici(a, n)
2   koren= (n-2) div 2
3   while koren >= 0 do
4     Potopi(a, koren, n-1)
5     koren= koren-1
```

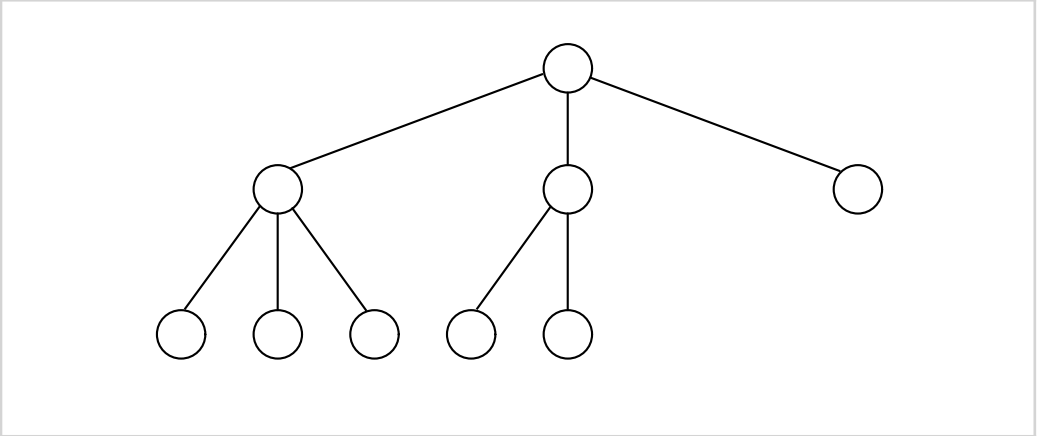
Ta koda uporablja implicitno shranjeno kopico elementov, ki jo zgradi neposredno v polju `a`.
Vprašanja:

1. V zgornji kodi manjka funkcija `Potopi`, ki v polju `a[0..n-1]` potopi element na mestu `koren` v podkopici, katere koren je na indeksu `koren`. Zapišite to funkcijo.

2. Če ima funkcija `Potopi(a,i,n)` časovno zahtevnost $O(\log(n-i))$, kakšna je časovna zahtevnost funkcije `Skopici`? Utemeljite odgovor.

3. Recimo, da imamo funkcijo `Potopi` narejeno. Zapišite vse funkcije, ki jih mora implementirati vrsta s prednostjo.

Naloga 50. *Vrste s prednostjo.* V splošnem so kopice k-tiške, medtem ko smo na predavanjih spoznali dvojiške kopice. Primer trojiške kopice je na sl. 2.5. Naj bo to min kopica, kar pomeni, da je v korenu katerekoli podkopice



Slika 2.5: Trojiška kopica.

najmanjši element te podkopice.
Vprašanja:

1. (i) V kopico na sl. 2.5 vstavite vrednosti

$$5, 8, 66, 31, 27, 75, 29, 62, 36.$$

Narišite končno kopico. (ii) Nad dobljeno kopico izvedite operacijo `DelMin` in zopet narišite končno kopico. Operacijo izvedite s potapljanjem vrzeli in nato dvigovanjem elementa po potrebi. (iii) Nad kopico iz (i.) izvedite ponovno operacijo `DelMin` in spet narišite končno kopico. Tokrat operacijo izvedite tako, da element iz lista prestavite na vrh kopice in ga potopite, kolikor je pač potrebno.

2. (i) Natančno koliko primerjav ste izvedli v prvem vprašanju pri brisanju (ii.) in koliko pri brisanju (iii.)? (ii) Če imamo v trojiški kopici n elementov, koliko primerjav je potrebnih pri brisanju s potapljanjem vrzeli? Utemeljite odgovor.

NAMIG: Upoštevajte, da se vrzel najprej potopi do dna in nato element dvigne.

3. Doslej smo se ukvarjali s trojiškimi kopicami, a odslej se bomo ukvarjali s k-tiškimi kopicami za nek poljuben k. (i) Koliko primerjav je potrebnih pri brisanju s potapljanjem vrzeli v najslabšem primeru? Utemeljite odgovor, ki ima parametra n in k. (ii) Kateri k je optimalen? Utemeljite odgovor.

Naloga 51. *Vrste s prednostjo.*
VPRAŠANJA:

1. Katere operacije definirajo osnovno obliko vrste s prednostjo? Zapišite jih in opišite, kaj počno.

2. Ali lahko implementiramo vrsto s prednostjo s preskočnim seznamom? Utemeljite svoj odgovor: če da, kakšne so *prednosti* in kakšne *slabosti*; in če ne, zakaj ne.

3. Na predavanjih smo spoznali dvojiško kopico, ki sestoji iz korena in dveh podkopic. Definirajmo eniško kopico, ki sestoji iz korena in ene podkopice. Kakšna je časovna zahtevnost vstavljanja v eniško kopico? Utemeljite odgovor.

Naloga 52. Recimo, da imamo n elementov, shranjenih v naslednjih podatkovnih strukturah: i.) dvojiška (binarna) kopica, ii.) razpršena tabela, iii.) rdeče-črno drevo. Peter se je odločil, da bo omenjene strukture uporabil za reševanje problema z naslednjimi operacijami: i.) `S.Insert(x)`, ki v strukturo vstavi nov element x; in ii.) `S.Find()`, ki vrne tri trenutno najmanjše elemente v strukturi.
VPRAŠANJA:

1. Za vsako od naštetih podatkovnih struktur zapišite primer, kjer prekaša drugi implementaciji.

2. Začnite s prazno dvojiško kopico in vanjo po vrsti vstavite naslednje elemente:

$$12, 19, 3, 6, -1, 2, 20, 11.$$

Narišite kopico po vsakem vstavljanju.

3. Imate predlog za kakšno še učinkovitejšo rešitev? Utemeljite odgovor.

Naloga 53. Obstaja cela vrsta izvedb vrste s prednostjo. Značilnost vseh je, da lahko dostopamo do najmanjšega elementa v času $O(1)$. V tej nalogi bomo imeli opravka z vrsto s prednostjo, v kateri je $n > 2018$ elementov.
Vprašanja:

1. Recimo, da imamo izvedbo z dvojiško (binarno) kopico. V njej iščemo $k = 4$ najmanjši element. Kje vse se lahko nahaja? Utemeljite odgovor.

2. Kaj pa, če imamo binomsko kopico. Kje vse se lahko nahaj sedaj $k = 4$ najmanjši element? Ponovno utemeljite odgovor.

3. Vračamo se k dvojiški kopici. V prvem vprašanju očitno ni bilo potrebno preiskati celotne kopice, ko smo iskali $k = 4$ najmanjši element. Koliko mora biti *vsaj* k, kot funkcija n, da bomo morali preiskati celotno kopico? Utemeljite odgovor.

Naloga 54. *Vrste s prednostjo.* Vedno je struktura *min* struktura. Imejmo naslednje zaporedje operacij:

```
I1, I2, I3, I4, I5, I6, I7, M, I8, DM, I9, DM, I10,
```

kjer `M` pomeni vračanje najmanjšega elementa, `DM` brisanje najmanjšega elementa in `Ix` vstavljanje elementa `x` v vrsto s prednostjo.
Vprašanja:

1. (i) Imejmo prazno dvojiško kopico in nad njo izvedite zgornje operacije. Narišite strukturo po vsaki operaciji. (ii) Sedaj imejmo še prazno leno binomsko kopico in nad njo izvedite zgornje operacije. Ponovno po vsaki operaciji narišite strukturo.

2. Recimo, da imamo v vrsti s prednostjo elemente od 1 do n. (i) Kje vse se lahko nahaja element n v dvojiški kopici? Utemeljite odgovor. (ii) Kje vse se lahko nahaja element n v (ne-leni) binomski kopici? Utemeljite odgovor.

3. Sedaj imamo dvojiško kopico, ki vsebuje elemente z vrednostmi od 1 do $n = 2^k - 1$. Koliko največ je lahko vrednost otroka korena celotne kopice? Utemeljite odgovor.

2.3 Razširjanje podatkovnih struktur

Pri načrtovanju algoritmov se često uporablja postopek razširjanja osnovne podatkovne strukture za podporo dodatne funkcionalnosti. Pogosto zadostuje, da osnovno podatkovno strukturo razširimo s shranjevanjem dodatnih informacij. Pri tem je potrebno biti pozoren, saj moramo dodane informacije posodabljati in vzdrževati z osnovnimi operacijami na podatkovni strukturi.

Tipičen primer je računanje ranga elementa, to je, njegovega mesta v linearno urejeni množici. Za izvedbo ranga lahko, na primer, razširimo katerokoli iskalno drevo tako, da za vsako vozlišče dodatno hranimo informacijo o velikosti poddrevesa, katerega koren je to vozlišče.

CILJ

Da znamo problem rešiti tako, da nadgradimo/razširimo že znano podatkovno strukturo.

Naloga 55. *Kako težki so?* Otroci so se na dvorišču igrali naslednjo igro ugibanja. Ugibali so skupno težo določene podmnožice otrok, in sicer je ta podmnožica definirana na podlagi višine otrok. Tako je bila poizvedba lahko: „Kolikšna je skupna teža vseh otrok, ki so višji od 112 cm in nižji od 127 cm?“ Recimo, da to poizvedbo poimenujemo `Teza(112, 127)`.

Recimo, da imamo naslednjo množico otrok s težami (prva številka predstavlja višino in druga težo): (120, 112), (128, 78), (153, 108), (98, 130), (117, 116), (98, 102), (149, 122), (129, 113), (116, 102) in (114, 103).
VPRAŠANJA:

1. Pri opisani množici otrok odgovorite na naslednje poizvedbe:

`Teza(108, 141)`, `Teza(121, 151)`, `Teza(97, 114)`, `Teza(149, 124)` in `Teza(124, 123)`.

2. Recimo, da je v množici n otrok. Opišite podatkovno strukturo, ki časovno kar se da učinkovito odgovarja na poizvebo `Teza()`. Utemeljite njeno pravilnost ter njeno prostorsko in časovno učinkovitost.

3. Množica otrok na dvorišču seveda ni stalna, ker jih mame kličejo bodisi na kosilo bodisi na večerjo, ali pa se pridružujejo skupini. Dodelajte podatkovno strukturo, da bo podpirala še operaciji

`Odsel(visina, teza)` in `Prisel(visina, teza)`.

Kakšna je časovna zahtevnost vseh treh vaših operacij? Kakšna je prostorska zahtevnost podatkovne strukture? Utemeljite odgovor.

Naloga 56. Imamo naslednja števila

$$59, 29, 93, 40, 40, 21, 82, 37, 36, 82, 15, 38, 6, 8, 5, 86, 68, \tag{2.1}$$

ki so shranjena v polju `S` pričenši od indeksa 0 naprej. Poleg tega definirajmo `Max(S, i, j)`, ki vrne največji element v polju `S` med vključno indeksoma i in j. Recimo, `Max(S, 7, 12)` vrne 82.
VPRAŠANJA:

1. (i) Predlagajte implementacijo operacije `Max(S,i,j)` za poljubna i in j, ki ima časovno zahtevnost $O(1)$. Pred tem lahko izvedete *poljubno predprocesiranje*. (ii) Kakšna je prostorska zahtevnost vaše rešitve? Utemeljite odgovor.

2. Osnovna operacija nad poljem je dostop do i. elementa – na primer, `S[4]` pomeni peti element v polju `S`. Običajno je polje implementirano kot implicitna podatkovna struktura. Lahko pa jo implementiramo tudi kako drugače. Recimo, da je implementirana kot preskočni seznam – seveda elementi niso urejeni po velikosti v takšnem seznamu. Na primer, implementacija polja `S` števil v vrstici (2.1) ima kot prvi element 59, nato 29 in tako naprej. (i) Opišite postopek, kako dostopati do i. elementa v preskočnem seznamu. (ii) Kakšna je časovna zahtevnost vaše operacije?

3. Nabor operacij razširimo z operacijo `Insert(S, i, e)`, ki na i. mesto vstavi element e in vse preostale elemente premakne za enega naprej. Recimo, če nad našimi podatki (2.1) izvedemo operacijo `Insert(S, 1, 77)`, potem bo na 0. mestu 59, na prvem bo vstavljena 77, na drugem 29, in tako naprej do 25. mesta, kjer je 46. (i) Opišite učinkovito implementacijo podatkovne strukture in operacij `Max()` in `Insert()`. (ii) Ali lahko izvedemo operacijo `Max()` v času $o(\log n)$, kjer je n število elementov v polju? Utemeljite odgovor.

Naloga 57. Tokrat so Petra poklicali organizatorji njujorškega maratona, ki imajo posebno željo. Želijo namreč postaviti spletno stran, preko katere bi lahko pregledovali trenutni vrstni red tekačev. Od Petra želijo, da njegova rešitev podpira čim učinkoviteje naslednje funkcije:

- `RaceStart()` – tekma se je pričela;
- `RaceEnded()` – tekma se je zaključila;
- `CurrentTime(who, time)` – ki osebi `who` popravi trenutni čas[6]; in
- `Place(who)` – ki vrne trenutno mesto tekmovalca `who`.

Spletna stran mora seveda delovati tudi še po zaključku tekme.
Vprašanja:

1. Za izračun ranga in za izbiro smo uporabili razširjene podatkovne strukture. Opišite, kako se spremenijo vrednosti v vozliščih pri enojnem in pri dvojnem vrtenju AVL drevesa. Narišite strukturo pred vstavljanjem, po vstavljanju pred vrtenjem in še po vrtenju.

2. Opišite podatkovno strukturo, ki bo rešila zgoraj opisani problem, in podajte opis ali psevdokodo posamezne operacije. Učinkovitejša bo vaša rešitev, več točk boste dobili.

3. Kakšna je časovna zahtevnost posamezne operacije? Utemeljite odgoovor.

Naloga 58. Definirajmo najprej osnovni, statični problem, kjer imamo polje a celih števila $a_j, j = 0, \ldots, n-1$. Nad poljem a definiramo predponske vsote $s_k = \sum_{j=0}^{k} a_j$ ter funkcijo `Sestej(k)`, ki vrne s_k. Poleg tega definirajmo razširjeni, dinamični problem, kjer imamo še operaciji vstavljanja in brisanja elementa a_k. Tako funkcija `Vstavi(x, k)` na k-to mesto vstavi vrednost x, medtem ko vse elemente za njim premakne za eno mesto naprej: $a_k = x$, a_{k+1} postane prejšnji a_k in tako naprej ter dolžina polja se poveča za ena. Po drugi strani `Izloci(k)` izloči a_k ter vse kasnejše elemente v polju a prestavi za eno mesto nazaj ter posledično zmanjša dolžino polja za 1.
Vprašanja:

[6]Lahko predpostavite, da sta `who` in `time` celi števili – štartna številka in čas v sekundah.

1. Predlagajte in opišite podatkovno strukturo, ki bo učinkovito podpirala vse tri operacije, ter ocenite njihove časovne zahtevnosti. Pri opisu je smiselno, da si pomagate s sliko.

2. Peter Zmeda je pred časom našel rešitev statičnega problema, ki za vse $i = 0, 1, \ldots, n-1$ izračuna s_i, in jo spravil v predal. Ali si lahko z njo pomaga pri izvedbi funkcije `Sestej(k)`? Utemeljite odgovor in za učinkovitejšo rešitev dobite več točk.

NAMIG: Očitno se vrednost `Sestej(k)` spremeni, če pred a_k vstavimo nek element.

3. Kakšna je časovna zahtevnost vaše rešitve? Utemeljite odgovor.

Naloga 59. Definirajmo najprej osnovni, statični problem, kjer imamo polje a celih števila $a_j, j = 0, \ldots, n-1$. Nad poljem a definiramo predponske vsote $s_k = \sum_{j=0}^{k} a_j$ ter funkcijo `Sestej(k)`, ki vrne s_k. Poleg tega definirajmo razširjeni, dinamični problem, kjer imamo še operaciji vstavljanja in brisanja elementa a_k. Tako funkcija `Vstavi(x, k)` na k-to mesto vstavi vrednost x, medtem ko vse elemente za njim premakne za eno mesto naprej: $a_k = x$, a_{k+1} postane prejšnji a_k in tako naprej ter dolžina polja se poveča za ena. Po drugi strani `Izloci(k)` izloči a_k ter vse kasnejše elemente v polju a prestavi za eno mesto nazaj ter posledično zmanjša dolžino polja za 1.
VPRAŠANJA:

1. Naj `V x k` pomeni klic funkcije `Vstavi(x, k)`, `I k` pomeni klic funkcije `Izloci(k)` in `S k` klic funkcije `Sestej(k)`. Če začnemo s praznim poljem a in izvedemo naslednje operacije `V 7 0 V 5 0 V 12 0 S 1`, dobimo kot odgovor 17, saj je polje a po vseh treh vstavljanjih `12 5 7`. Kaj vrne naslednje zaporedje operacij

`V 7 0 V 5 0 V 12 0 S 1 V 20 2 I 0 S 2 V 15 2 S 3`,

ki prične s praznim poljem a?

2. Predlagajte in opišite podatkovno strukturo, ki bo učinkovito podpirala vse tri operacije, ter ocenite njihove časovne zahtevnosti. Pri opisu je smiselno, da si pomagate s sliko.

3. Vrnimo se k osnovnemu, statičnemu problemu. Recimo, da imamo vse s_i naračunane. Ali lahko na podlagi le-teh najdemo podzaporedje števil $a_l, \ldots, a_r$, katerih vsota je 0? Začrtajte algoritem in utemeljite njegovo pravilnost ter časovno zahtevnost.

Naloga 60. Imamo naslednja števila

$$59, 29, 93, 40, 40, 21, 82, 37, 36, 82, 15, 38, 6, 8, 5, 86, 68, \tag{2.2}$$

ki so shranjena v polju (*array*) `stev` in sicer tako, da je `stev[0]` enak 59, `stev[1]` je 29 in tako naprej do `stev[24]`, ki je 46. V nalogi si bomo ogledali operacijo `Max(i, j)`, ki vrne največji element v polju `stev` med vključno indeksoma i in j. Recimo, `Max(1,4)` vrne 93.
Vprašanja:

1. (i) Predlagajte učinkovito implementacijo operacije `Max`, kjer *ne bomo izvedli nobenega predprocesiranja*. (ii) Kakšna je časovna in prostorska zahtevnost vaše rešitve? Utemeljite odgovor.

2. (i) Tokrat predlagajte učinkovito implementacijo operacije `Max`, vendar *lahko izvedemo predprocesiranja*. (ii) Kakšna je časovna in prostorska zahtevnost vaše rešitve? Utemeljite odgovor.

3. Nabor operacij razširimo z operacijo `Insert(i, e)`, ki na i. mesto vstavi element e in vse preostale elemente premakne za enega naprej. Recimo, če nad našimi podatki (2.2) izvedemo operacijo `Insert(1, 77)`, potem bo na 0. mestu 59, na prvem bo vstavljena 77, na drugem 29, in tako naprej do 25. mesta, kjer je 46. Opišite učinkovito implementacijo podatkovne strukture in operacij `Max` in `Insert`.

Namig: Polje morda ne bo več predstavljeno kot polje[7], ampak indeksi bodo še vedno šteli elemente po vrsti.

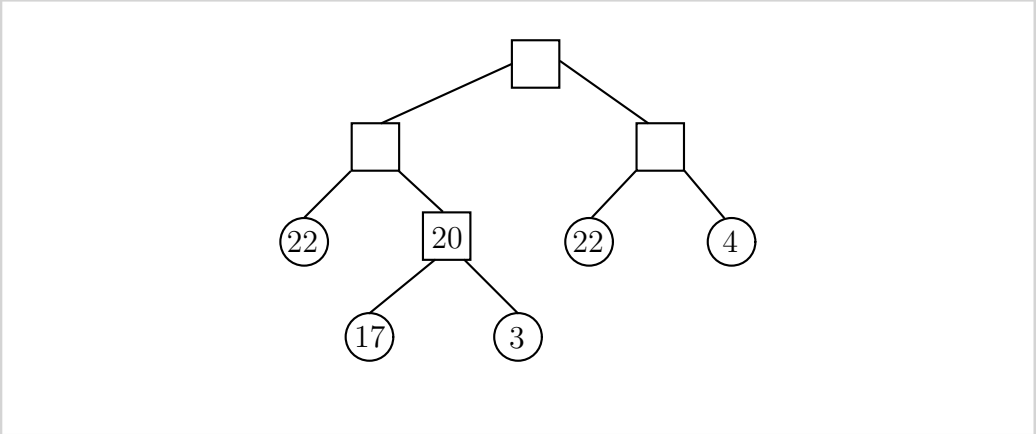
Naloga 61. Imamo drevo z vrednostmi v listih (krogi na sl. 2.6). Poleg tega definiramo pravilo, da je vrednost v notranjih vozliščih (kvadrati na sl. 2.6) enaka vsoti vrednosti v obeh otrokih.
Vprašanja:

1. Na sl. 2.6 dopolnite vrednosti v vseh vozliščih.

2. Zapišite rekurzivni algoritem, ki bo naračunal vrednosti v notranjih vozliščih in pokažite njegovo pravilnost.

3. Ocenite časovno in prostorsko zahtevnost vašega algoritma in odgovor utemeljite.

[7] Morda kot drevo?



Slika 2.6: Drevo z vrednostmi v listih.

Naloga 62. Imamo drevo z vrednostmi v listih (krogi na sl. 2.6). Poleg tega definiramo pravilo, da je vrednost v notranjih vozliščih (kvadrati na sl. 2.6) enaka vsoti vrednosti v obeh otrokih.
Vprašanja:

1. Predpostavimo, da so v listih samo nenegativne vrednosti. (i) Formalno dokažite, da je v korenu kateregakoli poddrevesa največja vrednost v poddrevesu. (ii) Pokažite, da zgornja trditev ne velja, če so lahko v listih negativne vrednosti.

2. Ostajamo pri pozitivnih vrednostih v listih. Peter Zmeda je dobil nalogo, da napiše funkcijo `IzpisiDo(t, value)`, ki ima kot parametra drevo `t`, kot smo ga definirali, in vrednost `value`. Funkcija naj izpiše vozlišča drevesa v padajočem vrstnem redu po njihovih vrednostih in to do vključno vrednosti `value` – vozlišč z vrednostjo manjšo od `value` naj ne izpiše. Napišite to funkcijo.

3. Kakšna je časovna zahtevnost vaše rešitve?

Naloga 63. Imamo polje `A[1..n]` celih števil. Implementirati želimo funkcijo `Sodih(i,j)`, ki vrne število sodih števil v polju med vključno indeksoma `i` in `j`. Za polje

$$[97, 38, 82, 86, 10, 51, 38, 88, 79, 55, 45, 40] \tag{2.3}$$

klic `Sodih(1, 3)` vrne vrednost 2, saj sta števili 38 in 82 sodi, medtem ko klic `Sodih(-100, +100)` vrne vrednost 7.
Vprašanja:

1. (i) Zapišite algoritem `Sodih`, ki deluje nad globalno spremenljivko `A` in pri tem ne uporablja predprocesiranja. (ii) Kakšna je njegova časovna zahtevnost?

2. Tokrat je dovoljeno predprocesiranje, kar dovoljuje izgradnjo dodatne podatkovne strukture. (i) Opišite dodatno podatkovno strukturo, ki bo omogočala poizvedbe v času $O(1)$. (ii) Kakšna je časovna zahtevnost predprocesiranja? Poleg tega utemeljite časovno zahtevnost $O(1)$ za poizvedbe. (iii) Kakšna je prostorska zahtevnost vaše dodatne podatkovne strukture?

3. Sedaj problem razširimo tako, da poleg funkcije `Sodih` dodamo še funkciji `Vstavi(i, x)` in `Izloci(i)`. Prva v polje `A` na mesto `i` vstavi element `x`. Na primer, iz polja (2.3) po klicu `Vstavi(3, 26)` dobimo polje

$$[97, 38, 82, 26, 86, 10, 51, 38, 88, 79, 55, 45, 40],$$

in če sedaj pokličemo še funkcijo `Izloci(6)`, dobimo

$$[97, 38, 82, 26, 86, 51, 38, 88, 79, 55, 45, 40].$$

(i) Opišite podatkovno strukturo in vse tri funkcije nad njo. (ii) Kakšna je prostorska zahtevnost vaše podatkovne strukture? (iii) Kakšna je časovna zahtevnost vaše rešitve (vse tri operacije)?

Namig: Učinkovitejša kot bo vaša rešitev, več točk boste prejeli.

Naloga 64. V Butalah so se odločili prebarvati ograjo okoli vaške cerkve. Označili so, kje se ograja začne in nato razpisali natečaj, da naj vsak Butalec pove, od katerega metra do katerega metra naj se ograja pobarva. Tako je Luka Kratkohlačnica predlagal, da se pobarva med 4. in 8. metrom, Gregor Copatka med 7. in 12. metrom, Benda Cigan pa med 22. in 33. metrom. Tako sta nastala dva pasova barve med metroma 4. in 12. metrom ter med 22. in 33. metrom.
Vprašanja:

1. Na koncu je butalska srenja predlagala naslednje pasove za barvanje: 68 in 76, 55 in 52, 17 in 79, 36 in 56, 47 in 38, 29 in 18, 20 in 70, 7 in 5, 55 in

49, 83 in 26, 88 in 10, 66 in 19, 5 in 2, 50 in 82, 80 in 82, 75 in 39, 10 in 9, 85 in 15, 39 in 12. Kakšna je skupna dolžina pobarvane ograje?

2. Zapišite algoritem, ki bo najprej prebral število n, ki predstavlja število prebivalcev, in nato n parov števil ter na koncu izračunal, kakšna je dolžina pobarvane ograje.

3. Kakšna je časovna in kakšna prostorska zahtevnost vašega algoritma? Utemeljite odgovor.

Naloga 65. *Drevesa.*
Vprašanja:

1. Imamo B-drevo višine 5 in $b = 7$. (i) Koliko največ elementov je v takšnem drevesu in (ii) koliko najmanj? Utemeljite oba odgovora. Upoštevajte, da ima drevo z enim samim vozliščem višino 1.

2. Na predavanjih smo srečali problem ranga in izbire (poleg vstavljanja in brisanja). Časovno zahtevnost smo opredelili s številom primerjav, ki jih je naredil naš algoritem. Tokrat bomo šteli število dostopov in sicer na enkrat (v konstantnem času) lahko preberemo 512B. Podatki, ki jih vstavljamo, so 32-bitna števila in tudi reference (kazalci, naslovi) so 32 bitne vrednosti. (i) Predlagajte učinkovito rešitev za opisani problem. (ii) Ocenite časovno zahtevnost vaše rešitve in utemeljite odgovor.

Namig: Rešitev tega vprašanja je verjetno malce daljša, saj morate obdelati *vse* operacije in biti natančni pri opisih ter definicijah podatkovnih struktur.

3. Ali se da vašo rešitev iz prejšnjega vprašanja poenostaviti in/ali pospešiti, če sta edini operaciji rang in izbira? Utemeljite odgovor.

Naloga 66. *Slovar.* Poznamo več implementacij slovarja, vse pa podpirajo osnovne operacije `insert(x)`, `delete(x)` in `find(x)`.
Vprašanja:

1. Recimo, da želi Peter Zmeda dodati operacijo `range(x, y)`, ki vrne seznam vseh elementov z v slovarju, za katere velja $x \leq z \leq y$. Pomagajte mu in napišite implementacijo funkcije `range` za izvedbo slovarja z AVL drevesom, s preskočnim seznamom in z razpršilno tabelo z odprtim naslavljanjem.

NAMIG: Natančnejša kot bo vaša implementacije, več točk boste dobili – npr. pazite na uporabljene podatkovne strukture ipd.

2. Za vsako od treh implementacij funkcije `range` iz prvega vprašanja ocenite časovno zahtevnost in utemeljite odgovore.

3. Recimo, da dodatno zahtevamo, da naj bo vrnjen seznam elementov urejen po velikosti. (i) Se časovne zahtevnosti implementacij funkcij kaj spremene? Utemeljite odgovor. (ii) Za katero implementacijo slovarja bi se odločili, če bi jo morali razširiti s funkcijo `range`? Utemeljite odgovor.

Naloga 67. *Razširjeni slovar.* V Butalah je družbeni položaj osebe odvisen od številke njegove noge – večja kot je noga, pomembnejša je oseba. Tako osebe predstavimo kot pare celih števil: $e = (\text{rojstvo}, \text{velikost})$, kjer prvo število predstavlja datum rojstva osebe in drugo njegovo velikost noge. Datum rojstva je ključ. V splošnem imamo množico elementov $\{e_1, e_2, \ldots e_n\}$.
VPRAŠANJA:

1. Naj bo $n = 7$ in naj bodo elementi: $(40.606, 41)$, $(81.083, 36)$, $(55.452, 43)$, $(85.624, 35)$, $(8.799, 38)$, $(26.128, 42)$, $(27.232, 39)$. (i) Elemente po vrsti vstavite v preskočni seznam in upoštevajte, da ste dobili naslednje naključne vrednosti (0 je grb in 1 je cifra):

```
1 0 1 1 0 0 1 0 0 0 1 0 0 0 0 1 0 1 0 0 1.
```

Izrišite seznam z vstavljenimi elementi.

2. (i) Kdaj je bila rojena najpomembnejša oseba med osebami, rojenimi med 25.000 in 50.000? (ii) Opišite, kako ste našli odgovor na to vprašanje. (iii) Opišite, kako bi razširili podatkovno strukturo preskočnega seznama, da boste učinkovito odgovarjali na poizvedbo: `Max(d1, d2)`, ki vrne datum rojstva osebe, ki ima največjo nogo in je rojena med datumoma `d1` in `d2`.

NAMIG: Učinkovitejša rešitev prinaša več točk.

3. (i) Kakšna je časovna zahtevnost vaše rešitve? (ii) Opišite, kako deluje sedaj operacija `Insert((d, s))`, kjer `(d, s)` predstavlja rojstni dan osebe in številko njegove noge.

Naloga 68. *Slovar – osnove.* Recimo, da imamo naslednja števila, ki so urejena po velikosti: 1, 4, 19, 19, 19, 24, 39, 40, 42, 46, 50, 53, 56, 58, 59, 61, 66, 69, 74, 77, 79 in 90.
Vprašanja:

1. Narišite najplitvejše urejeno binarno drevo, ki vsebuje vse zgornje elemente.

2. (i) Napišite algoritem, ki iz n *različnih* urejenih števil v polju `A` zgradi najplitvejše urejeno dvojiško drevo. Utemeljite, da je vaše drevo res najplitvejše. (ii) Kakšna je časovna zahtevnost vašega algoritma? Utemeljite odgovor.

3. Predlagajte podatkovno strukturo (ali strukture), ki bo porabila najmanj časa za naslednje zaporedje operacij: (i) najprej vstavimo n elementov (`Insert`); in (ii) nato naredimo n^2 poizvedb (`Find`). Koliko časa porabi vaša struktura (ali strukture) za vse operacije? Utemeljite odgovor.

Namig: Boljša kot bo vaša rešitev, več točk boste dobili.

Naloga 69. Imamo množico $S = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ parov in naslednjo poizvedbo: `Min(l, d)`, ki med elementi množice S poišče tisti element (a, b), ki ima najmanjšo (drugo) vrednost b, pri čemer velja $l \leq a \leq d$. Recimo, da so v S pari (120, 112), (128, 78), (153, 108), (98, 130), (117, 116), (98, 102), (149, 122), (129, 113), (116, 102) in (114, 103). Potem klic `Min(116, 120)` vrne 102.
Vprašanja:

1. Pri dani S odgovorite na naslednje poizvedbe:

> `Min(108, 141)`, `Min(121, 151)`, `Min(97, 114)`, `Min(149, 124)` in `Min(124, 123)`.

2. (i) Opišite podatkovno strukturo za hranjenje S, ki bo omogočala učinkovito odgovarjanje na poizvedbo `Min`. Množica S vsebuje n parov. (ii) Utemeljite njeno pravilnost ter njeno prostorsko in časovno učinkovitost.

3. (i) Opišite podatkovno strukturo za hranjenje S, če dovolimo še dodajanje in izločanje elementov iz S. (ii) Kakšna je časovna zahtevnost vseh treh vaših operacij (poizvedba, vstavljanje in izločanje)? (iii) Kakšna je prostorska zahtevnost podatkovne strukture? Utemeljite odgovor.

Naloga 70. *Slovar tako in drugače.*
VPRAŠANJA:

1. Peter se je na predavanjih učil, da se pri preskočnem seznamu uporablja kovanec, da dobimo višino novovstavljenega elementa. Žal knjižnica, ki jo ima na razpolago, ne nudi funkcije `Kovanec()`, ki bi vračala vrednosti `grb` ali `cifra`, ampak ponuja funkcijo `Kocka()`, ki vrača vrednosti od 1 do 6. Nekaj časa je premišljeval ter se nato odločil, da kos kode:

```
visina= 0
while Kovanec() == grb:
  visina++
```

spremeni v

```
visina= 0
while Kocka() == 1:
  visina++
```

Komentirajte njegovo odločitev. Bo preskočni seznam sploh še uporaben? Utemeljite odgovor.

2. Tokrat ima Peter osnovno strukturo slovarja, v katerem se kot ključ uporablja ime in priimek osebe. Kot podatek pa je njen spol. (i) Predlagajte podatkovno strukturo, ki bo poleg vstavljanja, brisanja in iskanja nudila še funkciji `KolikoMoskih(o1, o2)` oziroma `KolikoZensk(o1, o2)`, ki vrneta, koliko moških, odnosno žensk, je po abecedi med osebama o_1 in o_2. Na primer, med `Peter Zmeda` in `Mojca Hitra`. (ii) Zapišite psevdokodo implementacij funkcij ter utemeljite njuno pravilnost.

3. Kakšna je časovna in prostorska zahtevnost vaše rešitve? Utemeljite odgovor.

Naloga 71. Peter zmeda je našel listek papirja z naslednjimi števili: 20, 68, 41, 86, 70, 84, 75, 16, 55, 37, 5, 27, 61, 50, 98, 29, 44, 7, 90 in 47. Nad števili bi želeli učinkovito izvesti naslednji operaciji:

`Min(v1, v2)`: ki poišče najmanjše število med števili, ki so v območju `v1` ... `v2`. Na primer, `Min(-100, +100)` vrne 5 kakor tudi `Min(5, +100)`, medtem ko `Min(6, +100)` vrne 7.

`Mediana(v1, v2):` ki vrne mediano med števili, ki so v območju `v1` ... `v2`. Na primer, `Mediana(-100, +100)` vrne 44 ali 47 (odvisno od definicije). Klic `Mediana(75, 100)` pa vrne 86.

Vprašanja:

1. Zapišite, kaj vrnejo naslednji klici funkcij:

`Min(63, 71)`, `Min(32, 49)`, `Min(10, 11)`, `Min(68, 78)`, `Min(3, 43)` in `Mediana(4, 21)`, `Mediana(25, 98)`, `Mediana(58, 87)`, `Mediana(46, 60)`, `Mediana(15, 87)`.

2. (i) Predlagajte, kako naj organizira (shrani) podatke z listka, da bo učinkovito izvajal operaciji. Zapišite algoritma za operaciji nad vašo podatkovno strukturo. (ii) Kakšna je časovna in prostorska zahtevnost vaše rešitve? Utemeljite odgovor. (iii) Se da boljše? Utemeljite odgovor.

3. Spremenite vašo podatkovno strukturo tako, da bo omogočala tudi vstavljanje in brisanje elementov. Opišite algoritme operacij sedaj.

Naloga 72. *Slovar – nadgradnja.* Peter Zmeda dela domačo nalogo in je dobil naslednja števila v naključnem vrstnem redu: 69, 74, 58, 4, 61, 90, 19, 46, 1, 19, 40, 59, 39, 56, 19, 50, 42 in 53. Ker bo števila po vrsti vstavljal v preskočni seznam, potrebuje še naključne vrednosti 0 in 1:

1 0 1 0 1 0 0 1 0 1 0 1 1 0 1 0 1 0 0 0 1 0 1 0 0 0 0 1 0 1 0 0 0 1
0 1 1 1 0 0 0 0 0 1 1.

Vprašanja:

1. Kako izgleda Petrov preskočni seznam po vstavljanju števil? Upoštevajte, da je višina elementa število zaporednih enic plus 1: konkretno, prvi vstavljeni element, ki je 69, bo imel višino 2, saj eni enici sledi ničla (podčrtani prva 1 in 0 zgoraj). Pri vstavljanju naslednjega elementa, ki je 74, uporabite naslenjo enico in ničlo ter tako naprej.

2. Definirajmo nad slovarjem funkcijo `Left(x)`, ki vrne največji element, ki je v slovarju in je manjši od elementa `x` – vrne levega soseda elementa `x`. Peter mora implementirati novo funkcijo. Pomagajte mu.

(i) Opišite, kako naj deluje nova funkcija[8]. (ii) Kakšna je časovna zahtevnost vašega algoritma? Utemeljite odgovor. (iii) Bi lahko nadgradili (spremenili) preskočni seznam, da bi pohitrili iskanje levega elementa? Utemeljite odgovor.

3. Petrov učitelj je neizmeren vir idej, ki jih mora nato Peter implementirati. Tokrat se je spomnil, da bi implementacijo slovarja s preskočnim seznamom nadomestil z razpršilno funkcijo, ker je nekje prebral, da naj bi bila slednja učinkovitejša. (i) Kako naj sedaj Peter implementira funkcijo `Left(x)`? (ii) Kakšna je časovna zahtevnost implementacije? Utemeljite odgovor. (iii) Kaj pa sedaj, se da kaj narediti, da bi bila implementacija hitrejša? Utemeljite odgovor.

Naloga 73. *Preskočni seznami in razpršilne tabele.* Recimo, da imamo naslednje ključe: T R E T J I R O K.
Vprašanja:

1. Pokažite, kako se vstavijo v na začetku prazen preskočni seznam, pri čemer so naključne vrednosti višin, ki jih uporabljamo: 1, 3, 1, 1, 2, 3, 1, 4 in 1. Po vsakem koraku izrišite sliko strukture.

2. Isto zaporedje ključev vstavite v razpršilno tabelo, pri čemer je tabela na začetku prazna in velika 6 elementov. Na začetku uporabite kot razpršilno funkcijo $h(k) = (k * p) \bmod m$, kjer je $p = 11$ in m velikost tabele. V primeru sovpadanja uporabite veriženje. Kot vrednosti ključev k vzemite ASCII vrednosti zgornjih črk – A = 65, B = 66,...

3. Ali lahko katero od uporabljenih struktur uporabimo tudi za urejanje (sortiranje)? Utemeljite odgovor.

4. Ali lahko katero izmed struktur uporabimo za operacijo `rang`, ki vrne, kateri po vrsti je element po velikosti? Utemeljite odgovor.

Naloga 74. Imamo naslednje elemente

$$17, 2, 4, 5 \text{ in } 12 \tag{2.4}$$

[8]Psevdokoda ali resničen algoritem prinaša več točk.

ter generator naključnih bitov, ki generira bite

$$1100001001101000. \tag{2.5}$$

Vprašanja:

1. Vstavite elemente iz (2.4): (i.) v preskočni seznam, pri čemer uporabite bite iz (2.5); in (ii.) v rdeče črno drevo.

V obeh primerih narišite strukturo po vsakem vstavljanju.

2. Kakšna je časovna zahtevnst vstavljanja v preskočni seznam in kakšna v rdeče-črno drevo?

Namig: Bodite pazljivi za kakšno vrsto časa/zahtevnosti gre.

3. Peter Zmeda bi želel dodati slovarju metodo `Left(x)`, ki vrne največji element v slovarju, ki je *manjši* od `x` – vrne levega soseda `x`. Kako naj to metodo implementira v rdeče-črnem drevesu in kako v preskočnem seznamu?

Kakšni sta časovni zahtevnosti implementacij? Utemeljite odgovor.

Namig: Morda lahko predlagate kakšno nadgradnjo posamezne podatkovne strukture, da dobite učinkovitejšo rešitev?

Naloga 75. Peter Zmeda je nekje na spletu našel prevedeno kodo za funkcijo `Zmelji(k)`, ki vzame celo število k in vrne rezultat, ki je v intervalu $\{0, 1, \ldots, m-1\}$. Poleg prevedene kode je še zagotovilo, da funkcija zagotavlja skoraj idealno razprševanje.

Vprašanja:

1. Peter želi s funkcijo `Zmelji` narediti podatkovno strukturo slovar na polju `a[0..n-1]`. Predpostavimo, da velja $n = m$. Težava, ki jo mora razrešiti, je sovpadanje. Zapišite funkciji za vstavljanje in iskanje v podatkovno strukturo slovar z uporabo funkcije `Zmelji` ter pri tem uporabite odprto naslavljanje z linearnim pregledovanjem.

2. Peter je kmalu ugotovil, da njegova podatkovna struktura večino časa zaseda preveč prostora, saj je polje `a[0..n-1]` precej prazno. Zato se je odločil, da bo n zmanjšal (torej $n < m$). Ali lahko še vedno učinkovito uporablja funkcijo `Zmelji`? Utemeljite odgovor. Učinkovito pomeni, da bo razpršenost elementov po tabeli še vedno naključna in da računanje naslova, kjer naj bi se v polju element nahajal, ni prepočasno.

3. Recimo, da uporabimo uravnoteženo dvojiško drevo za implementacijo slovarja `S`. (i) Razširite vozlišča tako, da boste lahko v $O(1)$ implementirali

`S.Min()`, `S.Max()`, `S.Predhodnik(x)` in `S.Naslednik(x)`. (ii) Pokažite, kako boste vzdrževali razširitev v drevesu pri vstavljanjih in brisanjih.

Namig: Pri prvem delu vprašanja morate podati *opis* podatkovne strukture (lahko si pomagate s sliko) in *psevdokodo funkcij*, pri čemer lahko predpostavite, da ste element x že našli. Pri drugem delu vprašanja lahko v odgovoru uporabite poljubno uravnoteženo drevo kot na primer AVL ali rdeče-črno.

Naloga 76. *Slovar.* Tokrat bomo imeli opravka s slovarjem, v katerega lahko vstavimo isti element večkrat. Tako imamo naslednje elemente: 3, 5, 10, 7, 10, 5, 2 in 10.
Vprašanja:

1. (i) Vstavite elemente v 2-3-4 drevo in drevo izrišite. (ii) Pretvorite drevo v rdeče-črno drevo ter ga ponovno izrišite.

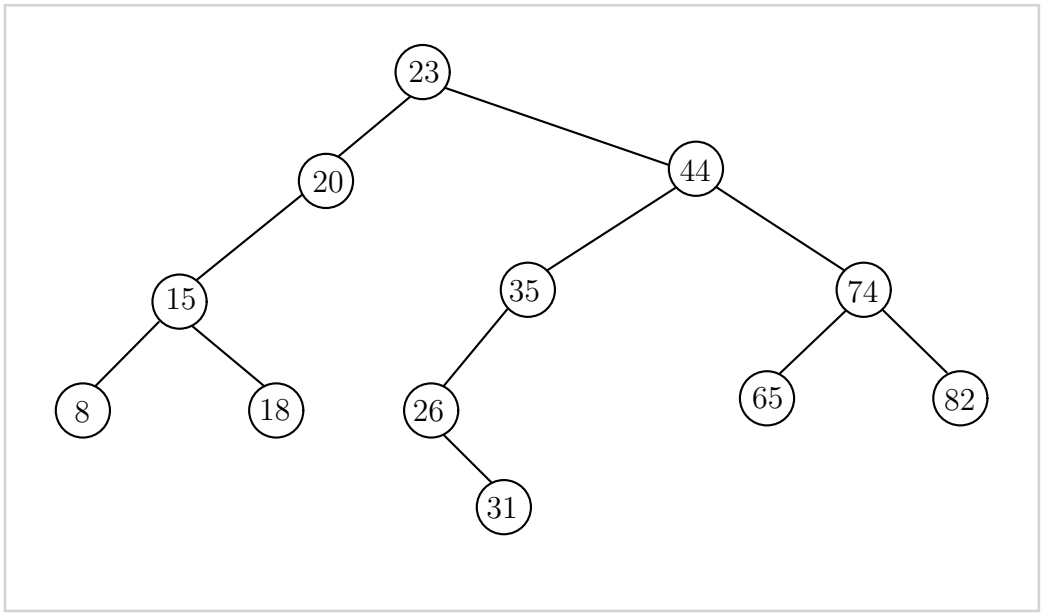
2. Imamo običajno iskalno dvojiško drevo. (i) Zapišite funkcijo `Insert(x)`, ki vstavi element `x` v drevo tudi v primeru, če že obstaja v njem. (ii) Zapišite še funkcijo `Find(k)`, ki vrne vse elemente iz drevesa, katerih ključ je `k`. (iii) Kakšna je časovna zahtevnost vaše rešitve? Utemeljite odgovor.

3. Pri delu z velikimi količinami podatkov je običajen način to, da najprej veliko elementov vstavimo v slovar in nato naredimo poizvedbo po določenem elementu. Naj bo število vstavljenih elementov n, po katerih pride m poizvedb. Predlagajte rešitev, ki bo zagotavljala čim manjšo skupno zahtevnost v najslabšem primeru vseh $n + m$ operacij. Utemeljite svoj odgovor.

Namig: Za opis rešitve opišite podatkovno strukturo (ali strukture) in operacije nad njo (ali njimi).

Naloga 77. Peter Zmeda je dobil listek, na katerem je bilo narisano drevo na sl. 2.7.
Vprašanja:

1. (i) Ali je drevo AVL drevo? Utemeljite odgovor. (ii) Ali ga lahko pobarvate tako, da postane rdeče-črno drevo? Utemeljite odgovor. (iii) Drevo razširite tako, da v vsako vozlišče dodate podatek o največjem in najmanjšem elementu v poddrevesu tega vozlišča.



Slika 2.7: Dvojiško drevo.

2. Definirajmo funkcijo `Range(a,b)`, ki vrne vse elemente v drevesu, ki ležijo na intervalu $[a, b]$, se pravi vključno z a in b. (i) Kaj vrne klic funkcije `Range(18,45)` na drevesu s sl. 2.7? (ii) Razširjeno drevo naj ima naslednjo definicijo vozlišča:

```
vozlišče:
  int koren
  int min, max
  poddrevo levo, desno
```

Na tako definiranem drevesu zapišite (psevdo)kodo funkcije `Range(a,b)`.

3. Kakšna je prostorska in kakšna časovna zahtevnost vaše funkcije, če je v drevesu n elementov in k izmed njih leži na intervalu $[a, b]$? Utemeljite odgovor.

Naloga 78. *Razširjeni slovar.* V NBB (*Narodna banka Butale*) vsaka stranka dobi ob vpisu identifikacijski številko, ki je enolično naravno število. Ob tem velja, da ima najstarejša stranka najnižjo vrednost in zadnja, ki se je vpisala, največjo. Banka nato za vsako stranko vodi evidenco, koliko butalskih tolarjev (BTT) ima na računu. Z drugimi besedami, NBB vodi

slovar elementov `(id, v)`, kar pomeni, da ima stranka `id` na računu `v` BTT. Trenutno ima NBB n strank. Peter Zmeda je avtor programske opreme in je implementiral slovar z uporabo AVL drevesa. Direktor verjame, da imajo starejše stranke več sredstev na računu in zato želi, da Peter doda funkcijo `Vsota(id)`, ki vrne vsoto sredstev na računih strank, kateri identifikacijska številka je manjša ali enaka `id`.
VPRAŠANJA:

1. Naj bo $n = 7$ in so v banki naslednje stranke: $(74, 7344)$, $(22, 4631)$, $(8, 6888)$, $(54, 8181)$, $(84, 9356)$, $(31, 8500)$, $(97, 6169)$, kjer je prva številka identifikacijska številka stranke in druga znesek, ki ga ima le-ta na računu. (i) Elemente po vrsti vstavite v AVL drevo in na koncu izrišite drevo z vstavljenimi elementi.

2. (i) Kaj vrne klic funkcije `Vsota(74)`? (ii) Opišite, kako ste našli odgovor na to vprašanje. (iii) Opišite, kako bi razširili AVL drevo, da boste učinkovito odgovarjali na poizvedbo `Vsota(id)`. (iv) V drevesu, ki ste ga dobili v podvprašanju 1(i) , sedaj tako razširite vozlišča.

NAMIG: Učinkovitejša rešitev prinaša več točk.

3. (i) Kakšna je časovna zahtevnost vaše rešitve? Utemeljite odgovor. (ii) Iz drevesa, ki ste ga dobili v podvprašanju 1(i) in ste ga razširili v podvprašanju 2(iv), izbrišite element z `id` $= 22$.

Naloga 79. *Uravnotežena drevesa.*
VPRAŠANJA:

1. Pri AVL drevesih se uporablja za ravnoteženje enojno in dvojno vrtenje. (i) Narišite primer enojnega vrtenja. V sliko vključite višine poddreves in pokažite, kako se spreminjajo. (ii) Narišite primer dvojnega vrtenja. V sliko vključite višine poddreves in pokažite, kako se spreminjajo.

2. Ravnoteženja se dogajajo pri vstavljanju in brisanju v AVL drevesa. (i) Koliko ravnoteženj največ se lahko dogodi pri vstavljanju? Utemeljite odgovor. (ii) Koliko ravnoteženj največ se lahko dogodi pri brisanju elementa? Utemeljite odgovor.

3. Naš prijatelj Peter Zmeda je nekje slišal, da lahko podatkovno strukturo slovar nadgradi, da bo podpirala tudi poizvedbo `Num(a, b)`, ki vrne število elementov v slovarju, za katere velja, da ležijo na intervalu (a, b). Ne ve pa, če to lahko naredi učinkovito za vsako implementacijo slovarja. (i) Naštejte tri implementacije slovarja, ki zahtevajo $O(n)$ časovno zahtevnost za opisano operacijo, kjer je n število elementov v slovarju. (ii) Utemeljite svoj odgovor.

Naloga 80.
VPRAŠANJA:

1. Na predavanjih smo spoznali abstraktni podatkovni strukturi slovarja in vrste s prednostjo. V katerih operacijah se razlikujeta? Opišite te operacije.

2. Peter Zmeda je svoj slovar implementiral z uporabo preskočnega seznama. Slovar bi rad preoblikoval v podatkovno strukturo vrste s prednostjo, pri čemer želi ohraniti kot implementacijo vrsto s prednostjo. Opišite implementacijo vseh operacij, ki jih mora na novo implementirati – zapišite psevdokodo.

3. Včasih človek res ne razume Petra. Tokrat je uspešno implementiral vrsto s prednostjo in sicer z uporabo binomskih dreves. Sedaj bi rad na osnovi te implementacije dodal operacije, ki jih potrebuje, da bo dobil slovar. Ponovno opišite implementacijo manjkajočih operacij – zapišite psevdokodo.

4. Recimo, da veste, da boste morali nad istimi podatki implementirati hkrati slovar in vrsto s prednostjo. Za katero osnovno podatkovno strukturo (seznam, preskočni seznam,...) bi se odločili? Utemeljite odgovor.

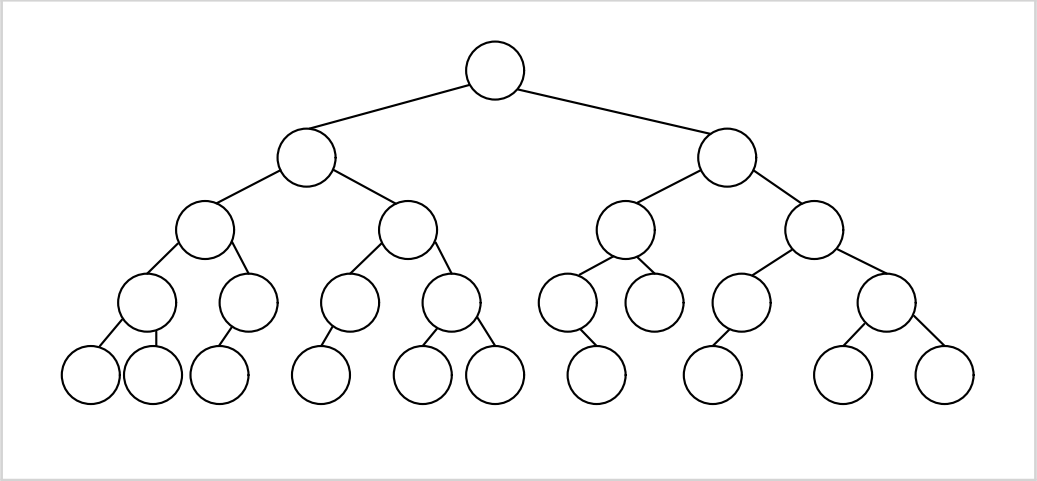
Naloga 81. Na predavanjih smo spoznali več različnih implementacij vrst s prednostjo. V tej nalogi si bomo pogledali več izrecnih (*eksplicitnih*) predstavitev vrst s prednostjo. Pri vseh bomo uporabili naslednja števila/ključe

$$\begin{aligned} &59, 29, 93, 40, 40, 21, 82, 37, 36, 82, 15, 38, 6, 8, 5, 86, 68, 26, \\ &35, 88, 57, 63, 87, 11, 46. \end{aligned} \tag{2.6}$$

Potrebovali pa bomo še rekurzivno podatkovno strukturo na sliki sl. 2.8.
VPRAŠANJA:

1. (i) Utemeljite, zakaj je podatkovna struktura na sliki sl. 2.8 lahko dvojiška (binarna) kopica. (ii) V strukturo razvrstite elemente (2.6) tako, da bodo predstavljali dvojiško kopico. (iii) Utemeljite, ali obstaja več možnih razporeditev.

2. (i) Iz zgoraj omenjenih elementov tvorite še binomsko kopico (ne drevo) in jo narišite. (ii) V svojo kopico vstavite element 49.



Slika 2.8: Rekurzivna struktura.

3. Recimo, da imamo vrsto s prednostjo implementirano kot binomsko kopico. (i) Razširite implementacijo tako, da bo podpirala tudi poizvedbo `Max()`, ki vrne največji element. (ii) Podprite še funkcijo `Min(i)`, ki vrne i. najmanjši element in je `i` največ 3.

Namig: Zaradi spremembe podatkove strukture bo potrebno verjetno popraviti tudi implementacijo ostalih operacij (`Merge` in `DelMin`).

2.4 Številska drevesa

Osnovni namen številskega drevesa je shranjevanje nabora nizov (besed, ključev). Številsko drevo je iskalno drevo, ki nam omogoča uporabo posameznih črk določenega ključa za iskanje, pri čemer so ključi shranjeni v listih.

Velikost številskega drevesa lahko zmanjšamo, če vozlišča, ki imajo samo enega naslednika, izpustimo. S tem zagotovimo, da ima vsako notranje vozlišče vsaj dva otroka. Da omogočimo učinkovito iskanje elementov, v vozlišča dodamo podatek o številu izpuščenih vozlišč. Dobljeno drevo imenujemo Patricijino drevo.

Če pri opisani različici številskega drevesa govorimo o stiskanju po poti od korena do lista, lahko izvedemo tudi stiskanje po plasteh. V tem primeru več plasti v drevesu stisnemo v eno polje in na ta način spuščanje po plasteh drevesa nadomestimo z enim samim vpogledom v polje.

CILJ

Da razumemo prednosti in slabosti stiskanja v številskem drevesu ter njihovo široko uporabo (npr. v bioinformatiki kot priponska drevesa).

Naloga 82. Pri Petru Zmedi so v službi dobili nove pametne telefone, ki bi jih želeli nadgraditi tako, da bi znali sami dopolnjevati imena iz imenika, ko bi jih uporabnik vpisoval. Na primer, naj bosta v imeniku dve imeni, ki se pričneta na črko M: MATEJ in MARKO. Ko bi uporabnik vpisal M, se bi samodejno izpisalo MA(2), kjer številka 2 pomeni, da sta v imeniku dve imeni, ki se pričneta na MA. Če nato vpiše še T, se bi izpisalo MATEJ.
VPRAŠANJA:

1. (i) Opišite podatkovno strukturo, ki naj jo uporabi Peter, da bo učinkovito rešil nalogo. (ii) Opišite postopek vstavljanja v podatkovno strukturo.

2. [DODATNO] Ker ima Peter zelo veliko prijateljev, se včasih ne more natančno spomniti njihovih imen, ampak se spomni samo zaporedja črk sredi imena – na primer pri MATEJ-u se spomni samo ATE. Imate predlog podatkovne strukture za rešitev tega problema?

3. Petrov predstojnik se je ravno vrnil z izpopolnjevanja, kjer je izvedel, da so razpršilne tabele zelo učinkovita podatkovna strutkura za hranjenje podatkov, ter je zato zahteval od Petra, da jih uporabi v svoji rešitvi (navkljub predlogu iz prvega vprašanja). Kako naj Peter uporabi razpršilno tabelo za vseeno še relativno učinkovito rešitev?

4. Predlagana aplikacija na telefonu je bila Petru všeč, le motilo ga je, da se je izpisala samo številka. Zato se je odločil, da aplikacijo nadgradi tako da, če se uporabnik dotakne številke (v zgornjem primeru MA(2) številke 2), se mu v meniju ponudita obe imeni ter bi lahko izbral želeno ime. Seveda je imen lahko zelo veliko, a s tem se Peter še ne ukvarja. Predlagajte nadgradnjo vaše podatkovne strukture iz prvega vprašanja, da bo učinkovito pomagala Petru nadgraditi aplikacijo.

Naloga 83. *Številska drevesa.*
VPRAŠANJA:

1. Najprej (i) zapišite psevdokodo za vstavljanje v Patricijino drevo in (ii) vstavite v Patricijino drevo z abecedo $\Sigma = \{0, 1\}$ naslednje ključe

$$0010, 100, 010, 100011, 111001, 0011, 000010$$

ter narišiti drevo po vsakem vstavljanju.

2. Včasih imamo opravka z besedili, katerih črke so iz poljubno velike abecede – doslej je bila naša abeceda vedno končna. Kako bi v tem primeru učinkovito izvedli posamezno vozlišče?

3. Recimo, da imamo besedilo $t = a_1a_2a_3 \ldots a_n$, iz katerega lahko tvorimo n predpon $p_1 = a_1$, $p_2 = a_2a_1$, $p_3 = a_3a_2a_1$, $\ldots$, $p_i = a_i \ldots a_2a_1$, $\ldots$, $p_n = a_na_{n-1}a_{n-2} \ldots a_1$. Sedaj vse predpone p_i vstavimo v številsko drevo. Ali nam takšno drevo pomaga pri učinkovitem iskanju vzorca $v = v_1v_2 \ldots v_m$ v besedilu t? Kako? Utemeljite odgovor.

Naloga 84. *Številska drevesa.* Imamo dvojiško abecedo $\Sigma = \{0, 1\}$ in naš prijatelj Peter Zmeda je dobil naslednje elemente:

(`100010`, P), (`011011`, o), (`10011`, d), (`011`, a), (`11010`, t),
(`01`, e) in (`010`, k).

Vprašanja:

1. Pomagajte Petru in narišite številsko drevo, v katerega so vstavljeni zgornji elementi in je stisnjeno po poteh (PATRICIA).

2. Peter je nekje slišal, da lahko številska drevesa stiskaš tudi po plasteh. V drevesu iz prvega vprašanja stisnite prvi dve plasti, prve tri plasti, prve štiri plasti in prvih pet plasti. (i) Za vsako od stiskanj izračunajte razmerje med številom praznih elementov v dobljenem polju in dolžino polja. (ii) Za katero od stiskanj se naj odloči Peter in zakaj?

3. Petrov šef se je domislil, da bi opisane ključe uporabil za šifriranje sporočil. Na primer, koda `010011011` predstavlja sporočilo `ko`. (i) Kako lahko uporabi Peter številsko drevo (PATRICIA) za dešifriranje sporočil?

Namig: Opišite postopek tako, da zapišete zanko, ki bere črke z vhoda (šifrirano sporočilo), uporablja številsko drevo (kako?) in sproti izpisuje dešifrirano sporočilo.

(ii) Utemeljite, ali so ali niso zgornje šifre uporabne za šifriranje sporočil, ki vsebujejo samo črke `P`, `o`, `d`, `a`, `t`, `e`, `k`? (iii) Kako bi razširili drevo, da bi se pravilno odzvali, ko preberemo šifro, ki ne predstavlja nobene zgoraj definirane črke.

Naloga 85. *Številska drevesa.* Imamo dvojiško abecedo $\Sigma = \{0, 1\}$ in naš prijatelj Peter Zmeda je dobil naslednje elemente:

> (`01010011`, P), (`00000111`, o), (`00100001`, d), (`01010001`, a), (`11101100`, t), (`10010101`, e) in (`01001010`, k).

VPRAŠANJA:

1. Pomagajte Petru in narišite številsko drevo, v katerega so vstavljeni zgornji elementi in je stisnjeno po poteh (PATRICIA).

2. Peter je nekje slišal, da lahko številska drevesa stiskaš tudi po plasteh. V drevesu iz prvega vprašanja stisnite prvi dve plasti, prve tri plasti, prve štiri plasti in prvih pet plasti. (i) Za vsako od stiskanj izračunajte razmerje med številom praznih elementov v dobljenem polju in dolžino polja. (ii) Za katero od stiskanj se naj odloči Peter in zakaj?

3. Petrov šef je doslej vedno govoril, da naj bo v podatkovni strukturi največ en primerek posameznega ključa. Sedaj je spremenil mnenje in pravi, da je lahko v drevesu poljubno mnogo elementov z enakimi ključi. (i) Opišite, kako naj izgleda v tem primeru številsko drevo in utemeljite pravilnost vaše odločitve. (ii) Na podlagi opisa vstavite v drevo iz prvega vprašanja še element (`00100001`, X). (iii) Kaj pa če vemo, da v drevesu nikoli ne bosta več kot dva elementa z enakima ključema? Kako se naj vaša rešitev spremeni, da bo učinkovitejša? Utemeljite odgovor.

Naloga 86. *Številska drevesa.* Peter Zmeda je dobil niz $T =$ `1000100` nad abecedo $\Sigma = \{0, 1\}$. V T želi poizvedovati po poljubnem podnizu. VPRAŠANJA:

1. (i) Zgradite iz T priponsko drevo. (ii) Zgradite še priponsko polje iz T.

2. (i) Koliko podnizov v T se prične na niz $q =$ `00`? (ii) Razširite priponsko drevo, ki ste ga zgradili v podvprašanju 1(i) tako, da boste lahko učinkovito odgovarjali na poizvedbo `HowMany(T, q)`, ki vrne število podnizov v T, ki se začno na podniz q. (iii) Kakšna je časovna zahtevnost poizvedbe? Utemeljite odgovor.

NAMIG: Pri določitve časovne zahtevnosti lahko uporabljate količine $n = |T|$, $m = |q|$ in $\sigma = |\Sigma|$.

3. Vrnimo se k priponskemu drevesu iz podvprašanja 1(i). (i) Spremenite ga tako, da bo zgrajeno na besedilu $T' = \texttt{10001001} = T \cdot 1$, kjer je operator „·“ stik ali konkatenacija. (ii) Opišite čim učinkovitejši algoritem, ki bo priponsko drevo za poljubno besedilo T popravil tako, da bo novo drevo narejeno za besedilo $T \cdot w$, kjer je w neka beseda iz abecede Σ.

Naloga 87. Usmerjevalnik lahko definiramo kot napravo, ki ima k priključkov[9]. IP paket pride na enega od priključkov, usmerjevalnik pa se mora odločiti, na kateri priključek mora prispeli paket preposlati. Podatek, na kateri priključek preposlati paket, usmerjevalnik hrani v podatkovni strukturi *usmerjevalna tabela*. Primer majhne usmerjevalne tabele je:

```
1 Destination       Gateway        Flags     Netif  Expire
2 default           192.168.2.1    UGS       em0
3 127.0.0.1         link#3         UH        lo0
4 192.168.2.155/25  link#1         U         em0
5 192.168.2.101     link#1         UHS       lo0
6 192.168.2.0/23    link#2         U         igb0
7 192.168.126.1     link#2         UHS       lo0
```

Usmerjevalno tabelo lahko implementiramo s številskim drevesom nad abecedo $\Sigma = \{0, 1\}$, pri čemer so ključi naslovi mrež, podatki pa priključki (vmesniki, `NetIf` v zgornji tabeli), kjer je priključena določena mreža. Na primer, v zgornji tabeli imamo ključa[10]

```
192.168.2.155/25 = 11000000 10101000 00000010 1.......
192.168.2.0/23 = 11000000 10101000 0000001.  ........
```

in podatka `em0` oziroma `igb0`. Ko pride paket, ga usmerjevalnik pošlje na tisti priključek, katerega ključ se najbolj ujema z njegovim IP naslovom. Na primer, če pride paket z naslovom 192.168.2.140, ga usmerjevalnik prepošlje na priključek `em0`, saj se bolje ujema s prvim ključem, in če pride za naslov 192.168.2.14, ga prepošlje na naslov `igb0`.

Vprašanja:

1. Iz ključev in podatkov[11]

[9] k je lahko vse od majhnega števila kot je 4 pa tja do več deset.

[10] Presledki so zgolj zaradi preglednosti, v resnici gre (v prvem primeru) za niz 25 ničel in enic, saj je 25 dolžina maske.

[11] Za lažje delo so ključi največ 8 bitni, medtem ko so podatki enočrkovni.

(136/1, A), (138/8, B), (16/7, C), (162/8, Č), (92/5, D),
(32/4, E), (91/2, F), (10/6, G), (82/3, H), (47/3, J)

zgradite številsko drevo, ki je stisnjeno po poti (PATRICIA).

2. Razmislite, ali bi bilo smiselno stisniti drevo še po plasteh. Utemeljite odgovor.

3. Recimo, da pride paket z naslovom d. Zapišite algoritem, ki bo na podlagi usmerjevalne tabele, implementirane kot številsko drevo, vrnil vmesnik, na katerega naj se paket prepošlje.

Naloga 88. *Številska drevesa.*
VPRAŠANJA:

1. Za Patricijino drevo (PATRICIA) velja, da so poti stisnjene po višini, kar pomeni, da ima vsako vozlišče vedno dva naslednika, če je abeceda $\Sigma = \{0, 1\}$. Dokažite, da je v Patricijinem drevesu, v katerem je n elementov, natančno $n - 1$ notranjih vozlišč.

2. Imamo Patricijino drevo, kjer vstavljamo elemente vedno pri listih[12]. V takšno drevo po vrsti vstavite naslednje elemente in po vsakem vstavljanju narišite drevo: `01010011`, `0000101` in `10101`.

NAMIG: Pri risanju ne pozabite, da imajo notranja vozlišča dodaten podatek. Kateri že?

3. Recimo, da imamo sedaj trojiško abecedo $\Sigma = \{0, 1, X\}$ in n elementov. Koliko je največja višina Patricijinega drevesa in koliko najmanjša? Utemeljite odgovor.

Naloga 89. *Številska drevesa.* Peter Zmeda je našel listek z naslednjim besedilom

$$t = GACCGAGTAGAG. \tag{2.7}$$

VPRAŠANJA:

1. (i) Iz besedila v (2.7) zgradite priponsko drevo. (ii) Kateri vzorec dolžine 2 se največkrat ponovi? Kateri vzorec dolžine 3 se največkrat pojavi?

[12]Indeksi črk v besedi niso nujno padajoči od korena.

2. (i) Opišite podatkovno strukturo, ki bo omogočala ugotoviti, kateri vzorec dolžine k v besedilu dolžne n se največkrat ponovi. (ii) Opišite postopek, kako ugotovimo, kateri vzorec dolžine k v besedilu dolžne n se največkrat ponovi. (iii) Kakšna je prostorska zahtevnost vaše strukture? Utemeljite odgovor.

Namig: Podatkovna struktura bo zasnovana na številskem drevesu in primerno razširjena.

3. Kakšna je časovna zahtevnost vašega postopka? Utemeljite odgovor.

Namig: Vaš odgovor lahko predpostavi, da imate priponsko drevo že zgrajeno. Časovna zahtevnost naj vključuje tako n kot k.

Naloga 90. Imamo besedo $w =$ `ANANAS`.
Vprašanja:

1. Zgradite priponsko drevo (*suffix tree*) iz w in pri tem označite konec besede z znakom \$.

2. Naredite BWT iz besede w.

3. Recimo, da imamo priponsko drevo nad besedo dolžine n. (i) Opišite algoritem, ki v drevesu poišče najdaljši ponavljajoči niz iz w. (ii) Kakšna je časovna zahtevnost vašega algoritma? (iii) Ali lahko algoritem pospešimo, če že pred gradnjo drevesa vemo, da bomo iskali najdaljše ponavljajoče nize? Utemeljite odgovor.

Naloga 91. *Številska drevesa.* Peter Zmeda je v bioinformatiki, zato je njegova abeceda $\Sigma = \{\text{A}, \text{C}, \text{G}, \text{T}\}$ iz štirih črk. Za obdelavo podatkov bo uporabil Patricijino drevo, kjer je notranje vozlišče definirano kot (psevdokoda):

```
node:
  int skipValue;
  node[Σ] subTree;
```

list kot (psevdokoda):

```
leaf:
  string elt
```

in je tip `string` dejansko polje (tabela) črk z indeksi od 0 do `len-1`.
VPRAŠANJA:

1. Narišite Patricijino drevo glede na zgornjo definicijo, v katerem so elementi: ACGCTCC, ACCGGC, TCGC, TCGAG, CCCC, CCCG in GCGCGC.

2. Naj bo `T` Patricijino drevo z vozlišči, kot so opisana zgoraj. Kar se da natančno zapišite funkcijo `T.Find(string elt)`, ki vrne `true`, če je `elt` v drevesu in sicer `false`.

3. Kar se da natančno zapišite še funkcijo `T.Insert(string elt)`.

Naloga 92. *Številska drevesa.* Peter Zmeda se ukvarja tudi z bioinformatiko, kjer je njegova abeceda $\Sigma = \{\text{A}, \text{C}, \text{G}, \text{T}\}$ iz štirih črk. Pogosto uporabljena podatkovna struktura v bioinformatiki je priponsko drevo, ki je v resnici številsko drevo, v katerega vstavimo vse pripone nekega niza. Na primer, niz $s =$ TTATGTA ima osem pripon, kjer je prva sam niz s, naslednja TATGTA, nato ATGTA in vse tako do A ter praznega niza.
VPRAŠANJA:

1. (i) Zapišite številsko drevo, ki bo vsebovalo vse pripone zgoraj zapisanega niza. Pri tem shranite v liste indekse, kje se pripona prične v besedilu. Na primer, `atgta` se prične na 3. mestu. (ii) Spremenite ga v Patricijino drevo – temu drevesu rečemo *priponsko drevo*.

2. (i) Zapišite algoritem, ki bo zgradil priponsko drevo iz danega niza dolžine n. (ii) Kakšna je njegova časovna zahtevnost? Utemeljite odgovor. (iii) Kakšna, menite, je časovna zahtevnost najhitrejšega možnega algoritma? Utemeljite odgovor.

NAMIG: Razmislite, koliko vozlišč ima priponsko drevo in kako to vpliva na časovno zahtevnost.

3. Priponska drevesa omogočajo hitro iskanje vzorca dolžine m v nizu in to v času $O(m)$. (i) Opišite postopek iskanja. (ii) Sedaj bomo razširili funkcijo iskanja na `find(p1, p2, k)`, ki v besedilu poišče vzorec, ki se začne s `p1` in zaključi s `p2`, celotna dolžina pa je k. Na primer, na ta način lahko iščemo vzorec TAT.TA (na mestih pik so poljubne črke) s klicem funkcije `find(TAT, TA, 6)`. V nizu s se iskani vzorec nahaja na drugem mestu. Predlagajte učinkovito podatkovno strukturo in algoritem za takšno iskanje ter utemeljite svoj odgovor.

NAMIG: Samo priponsko drevo ne bo zadoščalo. Pomislite, kako uporabiti informacijo, shranjeno v listih priponskega drevesa. Je pa odgovor presentljivo kratek.

Naloga 93. Recimo, da imamo naslednje besedilo: `11011110`. Iz njega naredimo naslednje elemente (k, v): $(\texttt{11011110}, 1)$, $(\texttt{1011110}, 2)$, $(\texttt{011110}, 3)$, $(\texttt{11110}, 4)$, $(\texttt{1110}, 5)$, $(\texttt{110}, 6)$, $(\texttt{10}, 7)$ in $(\texttt{0}, 8)$, kjer je k ključ in v vrednost.
VPRAŠANJA:

1. (i) Zapisane elemente vstavite v številsko drevo (*trie*) in nato (ii) drevo stisnite po poteh (PATRICIA).

2. Sedaj bomo številsko drevo stisnili še po plasteh. Recimo, da dovolimo količnik stiskanja $\alpha = 1/3$. Slednje pomeni, da je lahko največ α del elementov v polju enaku `null`. Vzemite osnovno drevo iz prejšnjega podvprašanja in ga stisnite še po plasteh upoštevaje α. Pri tem utemeljite konstrukcijo.

NAMIG: Na primer, zakaj stisnjenih i plasti in ne $i - 1$ oziroma ne $i + 1$.

3. Pokažite, kako lahko z vašo rešitvijo implementirate funkcijo `Position` iz prejšnjega vprašanja.

Naloga 94. *Drevesa in disjunktne množice.* Na predavanjih smo omenjali premi, vmesni in obratni obhod drevesa.
VPRAŠANJA:

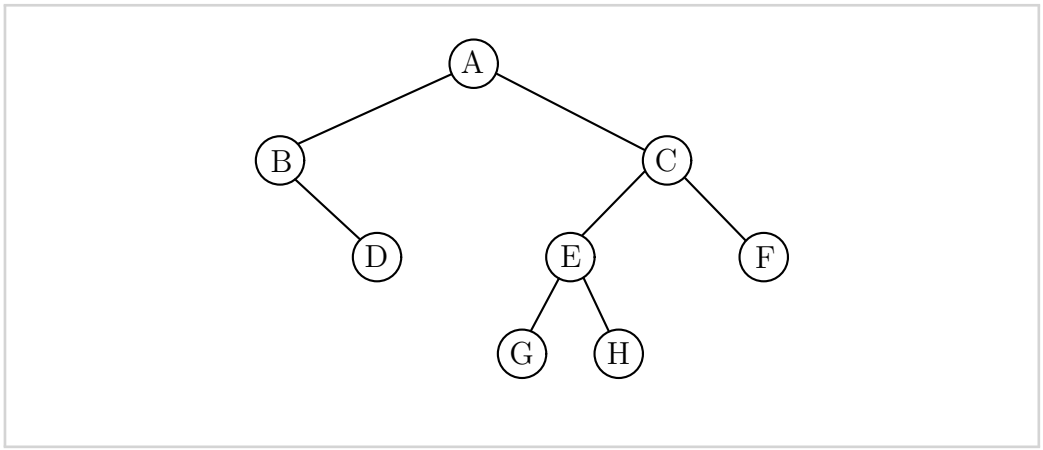
1. Na sliki sl. 2.9 imamo neko binarno drevo. Izpišite vozlišča ob obratnem (*postorder*) obhodu. Pokažite, da takšen izpis, ki ga tvorimo ob obratnem obhodu, ne določa enolično topologije (oblike) drevesa.

NAMIG: Poiščite še kakšno drevo, ki tvori enako zaporedje vozlišč ob obratnem obhodu.

2. Kateri od omenjenih obhodov ima smisel pri številskih drevesih (*tries*)? Utemeljite odgovor.

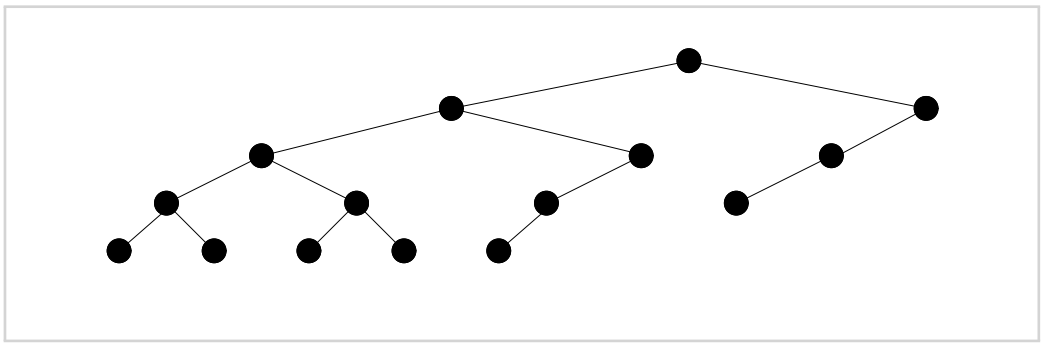
3. V prvem vprašanju smo omenili, da samo iz obratnega obhoda ne moremo rekonstruirati oblike drevesa. Se pa lahko rekonstruira oblika drevesa iz izpisa premega in obratnega obhoda. Pokažite to trditev oziroma predstavite algoritem.

NAMIG: Poskusite najprej z drevesom s sl. 2.9.



Slika 2.9: Primer dvojiškega (ne-iskalnega) drevesa.

Naloga 95. Levo kanonično dvojiško drevo je drevo, ki je levo poravnano. To pomeni, če vozlišča v neki plasti pregledujemo z leve proti desni, velja: najprej ima prvih $p-1$ vozlišč 2 naslednika; nato ima p-to vozlišče šteto z leve enega ali nobenega naslednika; ter preostala vozlišča nimajo naslednikov. Vozliščem brez naslednikov rečemo *listi*. Drevo na sl. 2.10 ima na plasti 4 z



Slika 2.10: Primer levega kanoničnega drevesa.

leve dve vozlišči z dvema naslednikoma, nato eno z enim naslednikom ($p = 3$) ter na koncu dve vozlišči brez naslednikov.

Vprašanja:

1. Naj bo drevo s sl. 2.10 številsko drevo nad abecedo $\{0, 1\}$ in naj bodo v njegovih listih posamezni elementi. Zapišite ključe vseh elementov.

2. [Dodatno] Ali kaj opazite, ko primerjate poljubne elemente? Utemeljite odgovor.

3. Stisnite drevo najprej po plasteh z $\alpha = 0,90$ in nato še po poteh (PATRICIA).

4. Vrnimo se k drevesu na sl. 2.10, ki ga bomo sedaj shranili v *implicitni podatkovni strukturi*. Opišite, kako takšno drevo shranimo v implicitno podatkovno strukturo. Za polovico točk zapišite, kako bi implicitno shranili drevo s sl. 2.10.

2.5 Disjunktne množice

V tem razdelku se ukvarjamo z elementi iz neke končne množice. Ti elementi so porazdeljeni po množicah tako, da je vsak element v natančno eni množici. Potemtakem velja, da je presek dveh množic prazen ali z drugimi besedami, množici sta disjunktni.

Nad množicami definiramo operacije tvorjenja eno-elementne množice, unije dveh množic in iskanja množice, ki ji dani element pripada. Najpreprostejša podatkovna struktura, ki podpira omenjene operacije, je verjetno povezan seznam. Neučinkovitost povezanega seznama nadgradimo z uporabo drevesne strukture.

CILJ

Da razumemo različne izvedbe podatkovnih struktur za podporo operacij nad disjunktnimi množicami. Pri tem se zavedamo, da nepotrebna urejenost strukture vodi v njeno učinkovitost.

Naloga 96. Peter Zmeda se je odločil za malce drugačno rešitev problema disjunktnih množic. Za množico n elementov je definiral polje oznak `id[0..n-1]`, v katerem je za vsak element i zapisano ime predstavnika množice `id[i]`, kateri pripada i. Recimo, v primeru polja, kjer je $n = 10$,

i	0	1	2	3	4	5	6	7	8	9
`id[i]`	1	1	2	2	6	2	6	9	9	9

pripada element 3 množici, katere predstavnik je 2, in prav tako elementa 2 ter 5. Element 4 pa po drugi strani pripada množici, katere predstavnik je 6.
VPRAŠANJA:

1. Napišite funkcijo `MakeSet()`, ki na začetku ustvari vse množice, v katerih je seveda po en element.

2. Napišite funkcijo `SameSet(p,q)`, ki vrne TRUE, če sta p in q v isti množici, in FALSE sicer.

3. S protiprimerom (i) pokažite, da spodnja funkcija za unijo ne deluje pravilno in (ii) jo popravite.

```
1 void Union(p, q) {
2   if SameSet(p, q) return;
3   for i= 0 to n-1
4     if id[i] == id[p] id[i] = id[q];
5 }
```

Naloga 97. Butale imajo precej razvito gospodarstvo in tako imajo vrsto podjetij. Po končanem šolanju se Butalci zaposlijo v različnih podjetjih in so precej ponosni na to, v katerem podjetju delajo. Podjetja pa se občasno prevzemajo, da optimizirajo stroške in povečajo svojo konkurenčnost.
VPRAŠANJA:

1. Katera od podatkovnih struktur, ki smo jih obravnavali na predavanjih, bi bila uporabna osnova za rešitev naloge? Utemeljite odgovor.

2. Podrobneje opišite podatkovno strukturo, ki bi pomagala Butalcem, da bi lahko učinkovito odgovarjali na poizvedbe, v katerem podjetju so zaposleni.

NAMIG: V resnici morate opisati, kako učinkovito implementirati operacije: `SeZaposli(X, Y)`, ki pomeni, da se je oseba `Y` zaposlila v podjetju `X`; `Prevzame(X, Y)`, ki pomeni, da je podjetje `X` prevzelo podjetje `Y` in je podjetje `Y` prenehalo obstajati; in `ZaposlenV(X)`, ki vrne ime podjetja, v katerem je zaposlena oseba `X`.

3. Včasih želijo Butalci tudi izvedeti, kdo vse dela v podjetju `X`. Opišite učinkovito izvedbo takšne poizvedbe in utemeljite, kako učinkovita (časovno in prostorsko) je ter zakaj.

Naloga 98. Recimo, da imamo univerzalno množico $\{1, 2, \ldots, 16\}$ in vsak njen element predstavlja ločeno samostojno množico.
VPRAŠANJA:

1. Nad množico množic simulirajte operacije:

`Union(1, 7)`, `Union(2, 9)`, `Union(3, 9)`, `Union(15, 10)`, `Union(16, 1)`, `Union(12, 12)` in `Union(13, 7)`.

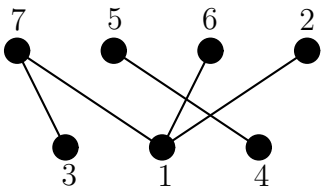
Zapišite množice, ki ste jih dobili na koncu.

2. In spet Petrov učitelj. Tokrat želi, da mu Peter implementira podatkovno strukturo, ki bo podpirala poleg operacij `Union()` in `Find()`, ki smo ju spoznali na predavanjih pri disjunktnih množicah, še operacijo `List(x)`, ki vrne vse elemente, ki pripadajo množici, katere pripadnik je tudi `x`. (i) Opišite novo podatkovno strukturo. (ii) Kakšna je časovna zahtevnost vsake od treh funkcij? Utemeljite odgovor.

3. Pri disjunktnih množicah velja, da nek element pripada natančno eni množici. Recimo, da tokrat vsak element pripada največ dvema množicama. (i) Opišite ustrezno podatkovno strukturo, ki bo podpirala učinkoviti operaciji `Union()` in `Find()`. (ii) Opišite obe operaciji ter za vsako operacijo posebej utemeljite njeno pravilnost in analizirajte njeno časovno in prostorsko zahtevnost. (iii) Ali opazite kakšen vsebinski in konceptualni problem definicije tega problema ter kako bi se ga lotili?

Namig: Kaj pravzaprav vrne `Find()`?

Naloga 99. *Disjunktne množice.* Graf sestoji iz množice vozlišč V in množice povezav E. Del grafa (podgraf) predstavlja komponento $K(V_K, E_K)$, če lahko iz poljubnega vozlišča komponente $v \in V_K$ po povezavah komponenete pridemo do vsakega drugega vozlišča komponente. Na primer, na sl. 2.11 imamo komponenti K in L z vozlišči $V_K = \{4, 5\}$ in $V_L = \{1, 2, 3, 6, 7\}$.



Slika 2.11: Graf z dvema komponentama.

Vprašanja:

1. Peter Zmeda je pridno hodil na predavanja in si je posebej dobro zapomnil poglavje o delu z disjunktnimi množicami (`union()` in `find()`). Pomagajte

mu sestaviti algoritem, ki bo uporabil omenjeni operaciji, da bo preštel število komponent v grafu.

NAMIG: Poleg samih struktur za delo z disjunktnimi množicami bo verjetno potrebno uporabiti še kakšno preprosto strukturo.

2. Kakšna je časovna in kakšna prostorska zahtevnost vašega algoritma? Utemeljite odgovor.

3. Petrov šef je zadovoljen s Petrovim delom, vendar potrebuje še podatek, katera vozlišča sestavljajo določeno komponento. Uporabite algoritem iz prvega dela naloge, ki ga razširite tako, da bo poleg števila komponent za vsako komponento izpisal tudi vozlišča, ki jo sestavljajo.

NAMIG: Ponovno bo potrebno poleg samih struktur za delo z disjunktnimi množicami uporabiti še kakšno dodatno strukturo.

Naloga 100. *Disjunktne množice.* V butalski občini imajo naselja $c_1, c_2, \ldots, c_n$. Ker je občina precej velika, se je butalski župan odločil, da bodo občino razdelili na okraje. S tem so definirali relacijo R s predpisom

$$R(c_i, c_j) = \begin{cases} 1, & \text{če sta } c_i \text{ in } c_j \text{ v istem okraju;} \\ 0, & \text{sicer.} \end{cases}$$

VPRAŠANJA:

1. Zapišite algoritem, ki bo ustvaril po en slovar za vsak okraj, pri čemer bodo v posameznem slovarju tista naselja, ki so v istem okraju.

NAMIG: Predpostavite, da sta podatkovni strukturi *slovar* in *disjunktne množice* že implementirani, tako da samo smiselno uporabite pravilne funkcije. Natančnejša in hitrejša bo vaša rešitev več točk bo prinesla.

2. (i) Katero implementacijo slovarja predlagate in zakaj? (ii) Kakšna je časovna zahtevnost vašega algoritma? Utemeljite odgovor.

NAMIG: Morda zaradi učinkovitosti rešitve uporabite določeno implementacijo slovarja, ki dovoljuje poleg običajnih treh operacij še učinkovito izvedbo kakšne druge operacije.

3. Utemeljite pravilnost vaše rešitve.

Naloga 101. Pri disjunktnih množicah imamo operaciji združevanja (`Union`) dveh množic in iskanje množice, kateri pripada določen element (`Find`). Imamo seveda še operacijo, ki iz elementa naredi množico (`MakeSet`). Učinkovita izvedba iskanja množice, kateri pripada element, ob iskanju predstavnika popravlja tudi informacijo pri elementih strukture.
Vprašanja:

1. Zaradi učinkovitosti je pomembno, kateri element pri združevanju postane predstavnik nove množice. Na predavanjih smo omenili način, da je predstavnik večje od združevanih množic tudi predstavnik združene množice. (i) Definirajte ustrezno podatkovno strukturo, ki bo omogočala učinkovito izvedbo operacij. (ii) Zapišite kodo operacij in pri tem uporabite definirano podatkovno strukturo. Natančnejša in učinkovitejša bo koda, več točk boste dobili.

2. Imejmo elemente $1, 2, \ldots, 9$. (i) Zapišite zaporedje operacij združevanja in iskanja, ki bo pri zadnjem iskanju zahtevalo največ dostopov do podatkovne strukture. (ii) Preštejte število dostopov in utemeljite, da je to res najslabši primer.

3. Peter Zmeda je opazil, da je v resnici največje število dostopov do podatkovne strukture odvisno od razdalje najbolj oddaljenega elementa od korena. Tako je prišel do ideje, da bi spremenil kriterij, kateri element naj bo predstavnik združene množice pri združevanju. Predstavnik združene množice naj postane predstavnik tiste množice, ki je pred združevanjem imel bolj oddaljen element od korena. (i) Definirajte podatkovno strukturo, ki bo omogočala učinkovito implementacijo operacij. (ii) Zapišite implementacijo operacij.

Namig: Če bodo težave pri implementaciji, zapišite, kje jih imate in zakaj.

Naloga 102. Imamo množico $\{1, 2, \ldots, 10\}$, iz katerih tvorimo 10 disjunktnih množic s po enim elementom, pri čemer uporabimo učinkovito podatkovno strukturo s predavanj.
Vprašanja:

1. Nad tako tvorjenimi množicami izvedemo naslednje operacije:

```
F 5, U 3 5, U 1 5, U 7 1, F 5
```

kjer `F x` vrne ime množice, kateri pripada element `x` in `U x y` naredi unijo množic, katerim pripadata elementa `x` in `y`.

Za vsako od zgornjih operacij narišite, kako se spreminja podatkovna struktura ter preštejte in zapišite število opravljenih primerjav ob vsaki operaciji.

2. Zapišite psevdokodo operacije (funkcije) `F`.

NAMIG: Najprej zapišite definicijo podatkovne strukture, ker sicer bo psevdokoda velika zmešnjava.

3. Recimo, da imamo opravka z množico $\{1, 2, \ldots, n\}$, iz katerih ponovno naredimo n disjunktnih nepraznih množic. Nad temi množicami izvedemo n operacij `F` in n operacij `U` v poljubnem vrstnem redu. Koliko primerjanj vsega skupaj bomo izvedli po zaključku vseh $2n$ operacij? Odgovor utemeljite.

NAMIG: Za skoraj vse točke ne pričakujem povsem točnega odgovora. Razmišljajte o najboljšem in o najslabšem primeru.

Naloga 103. Peter Zmeda in njegova prijateljica Špela se igrata naslednjo igrico:

1. na začetku imamo n lučk, ki so označene od 1 do n;
2. Peter k krat poveže po dve lučki;
3. Špela se z žico dotakne ene od lučk in potem vse lučke, ki jih je Peter povezal z dotaknjeno lučko, zasvetijo.

VPRAŠANJA:

1. Naj bo $n = 10$ in Peter nato poveže pare: $(5, 7)$, $(1, 8)$, $(2, 9)$, $(3, 8)$, $(4, 7)$ in $(7, 1)$. Špela se z žico dotakne lučke 4. Katere od naslednjih lučk svetijo: 2, 5, 8, 9 in 10?

2. Pri velikih n postane igrica precej težko izvedljiva, zaradi česar sta se Špela in Peter odločila napisati računalniški program, ki bo simuliral igrico. Program in posledično podatkovna struktura mora nudite naslednje ukaze:

- `Povezi(x, y)`, ki poveže lučki `x` in `y`;
- `Prizgi(x)`, s katerim se dotaknemo lučke `x` in
- `Sveti(x)`, ki vrne DA ali NE odvisno od tega, ali lučka `x` sveti.

Opišite, kako naj izgleda podatkovna struktura, ki podpira opisano igirco, ter kako so implementirani posamezni ukazi.

NAMIG: Morda lahko uporabite kakšno znano podatkovno strukturo.

3. Opišite, kako bi shranili popolno k-tiško drevo kot implicitno podatkovno strukturo. Z drugimi besedami, kako bi shranili omenjeno drevo v polju. Pri tem morate opisati, kako od vozlišča, shranjenega na indeksu `v[i]`, pridemo do:

- vozlišča, ki predstavlja njegovega j-tega naslednika ($1 \leq j \leq k$); in
- kako do njegovega starša.

NAMIG: Polje naj se prične na indeksu 1. Koren drevesa je `v[1]`, prvi naslednik na `v[2]`, drugi na `v[3]` in tako naprej. Pri kopici (*heap*) je uporabljena podobna preslikava, le da je tam $k = 2$.

Naloga 104. Tokrat so klicali Petra z univerze. Na univerzi so sprejeli nova pravila, kdo je lahko član komisije za zagovor zaključnega dela. Da lažje razumemo pravila, najprej definirajmo odnosa ,,soavtor“ in ,,posredni soavtor“. Recimo, da sta Janez in Peter avtorja članka. Potem je Janez Petrov soavtor in obratno. Odnos posredni soavtor je definiran tako, da je vsak soavtor avtorja tudi njegov posredni soavtor in vsak soavtor posrednega soavtorja je prav tako posredni soavtor prvega avtorja. Univerza sedaj zahteva, da član komisije ne sme biti posredni soavtor ne kandidata niti ne njegovega mentorja.

Peter mora narediti aplikacijo, ki prebere vse članke, ki naj se upoštevajo, ter z njih razbere avtorje. Nato uporabnika povpraša po kandidatu ter njegovemu mentorju. Od tu sledijo poizvebe z imeni morebitnih članov komisije ter odgovori, ali ustrezajo zahtevam ali ne.

VPRAŠANJA:

1. Predlagajte in opišite podatkovno strukturo, ki naj jo Peter uporabi v svoji aplikaciji.

2. Na predavanjih smo si pogledali razpršilne tabele. Sovpadanja smo reševali s pomočjo veriženja ali z naslavljanjem. Obstajajo še drugi načini, en izmed njih je tako imenovani *Cuckoo Hashing*. Opišite delovanje tega razprševanja.

3. (i) Napišite psevdokodo programa, ki v polju `a` števil dolžine `a.length` poišče najmanjše število. (ii) Koliko primerjav je potrebnih, da najdete najmanjše število? (iii) Koliko primerjav je najmanj potrebnih, da najdete največje število? Dokažite svojo izjavo.

Naloga 105. Tokrat so klicali Petra z univerze. Na univerzi so sprejeli nova pravila, kdo je lahko član komisije za zagovor zaključnega dela. Da lažje razumemo pravila, najprej definirajmo odnosa ,,soavtor“ in ,,posredni soavtor“. Recimo, da sta Janez in Peter avtorja članka. Potem je Janez Petrov soavtor in obratno. Odnos posredni soavtor je definiran tako, da je vsak soavtor avtorja tudi njegov posredni soavtor in vsak soavtor posrednega soavtorja je prav tako posredni soavtor prvega avtorja. Univerza sedaj zahteva, da član komisije ne sme biti posredni soavtor ne kandidata niti ne njegovega mentorja.

Peter mora narediti aplikacijo, ki prebere vse članke, ki naj se upoštevajo, ter z njih razbere avtorje. Nato uporabnika povpraša po kandidatu ter njegovemu mentorju. Od tu sledijo poizvebe z imeni morebitnih članov komisije ter odgovori, ali ustrezajo zahtevam ali ne.

Vprašanja:

1. Predlagajte in opišite učinkovito podatkovno strukturo, ki naj jo Peter uporabi v svoji aplikaciji.

2. Na univerzi so ugotovili, da baza člankov ni dokončna in lahko članke dodajamo. Ali lahko še vedno uporabljate enako podatkovno strukturo kot v prvem vprašanju? Predlagajte učinkovito rešitev ter utemeljite odgovor.

3. Kakšna je časovna in prostorska zahtevnost obeh vaših rešitev? Utemeljite odgovor. Upoštevajte tri vrste operacij: predprocesiranje, poizvedba in v drugem primeru tudi dodajanje.

Naloga 106. *Paroma disjunktne množice.* Peter je tokrat dobil univerzalno množico $U = \{1, 2, \ldots, n\}$ in učinkovite implementacije funkcij `Makeset()`, `Union()` in `Find()`, kot smo jih opisali na predavanjih. V pomoč, amortizirana časovna zahtevnost posamezne funkcije je bila $O(\log^* n)$.[13]

[13] V knjigi je omenjena amortizirana časovna zahtevnost posamezne operacije $O(\alpha(m, n))$, kjer je m število operacij. Ta funkcija v resnici raste še počasneje kot $\log^* n$.

Vprašanja:

1. Naj bo $n = 6$ in imejmo naslednji niz klicev funkcij:

`Makeset(5)`, `Makeset(1)`, `Makeset(2)`, `Makeset(4)`, `Makeset(6)`, `Makeset(3)`, `Find(5)`, `Union(5,1)`, `Union(2,3)`, `Find(5)`, `Union(1,2)`, `Union(3,4)`, `Union(6,6)`, `Find(5)` in `Find(1)`.

(i) Narišite podatkovno strukturo po vsakem klicu funkcije. (ii) Za vsakega od klicev opišite, katere primerjave ste naredili in koliko jih je bilo.

2. Amortizirana časovna zahtevnost niza m klicev funkcij je definirana kot skupna poraba časa za vse klice in nato deljena s številom klicev funkcij m. Seveda, slednje pomeni, da je lahko eden od klicev zelo „drag“, drugi pa bistveno cenejši. (i) Opišite, kako izgleda podatkovna struktura, pri kateri pride do tega „dragega“ klica. Utemeljite odgovor. (ii) Kako izgleda zaporedje klicev funkcij, ki privedejo do podatkovne strukture v prejšnjem delu vprašanja? Utemeljite odgovor.

3. Spet Petrov šef. Tokrat želi poleg implementacije običajnih funkcij še funkcijo `Members(x)`, ki vrne seznam članov množice, kateri pripada element `x`. Časovna zahtevnost vaše rešitve naj bo čim boljša in vse točke dobite, če bo sorazmerna $O(k + \log^* n)$, kjer je k število vrnjenih elementov.

Naloga 107. Imamo naslednjo igro. Opravka imamo z dominami, ki se lahko med seboj zlepijo. Vsaka domina ima svoj id: 1, 2, 3, 4, 5, 6. Na vsaki domini je poleg id-ja še vrednost domine, ki je na začetku kar enaka id-ju. Kot rečeno, lahko domine med seboj spajamo in ko domini spojimo, dobimo večjo domino, katere vrednost je enaka zmnožku vrednosti domin, če sta bili vrednosti obeh domin lihi, oziroma v nasprotnem primeru vsoti obeh vrednosti. Na primer, če zlepimo domini 1 in 3, ima nova domina vrednost $3 = 1 \cdot 3$, medtem ko, če zlepimo domini 5 in 4, ima nova domina vrednost $9 = 5 + 4$. Če sedaj zlepimo obe zlepljeni domini, se pravi (1, 3) in (5, 4) dobimo veliko domino (1, 3, 4, 5) z vrednostjo $27 = 3 \cdot 9$. Lepljenje domin definiramo z operacijo $L(d_1, d_2)$, kjer sta d_1 in d_2 id-ja kateregakoli delčka domin, ki jih želimo zlepiti. Zgoraj opisano lepljenje tako opišemo z ukazi:

$$L(1,3), L(5,4), L(1,5)$$

ali pa tudi z

$$L(1,3), L(5,4), L(1,4)$$

oziroma

$$L(1,3), L(5,4), L(3,5)$$

ali pa celo

$$L(1,3), L(5,4), L(3,4).$$

Vprašanja:

1. Recimo, da imamo domine z id-ji od 1 do 8 in naslednje operacije:

$$L(1,2), L(3,4), L(5,6), L(7,8), L(2,3), L(6,7), L(1,8).$$

Kakšna je vrednost končne (super)domine?

Namig: Opravljajte vse operacije po vrsti in si izpisujte vrednosti.

2. V zgornjem primeru smo imeli $n = 2^k$ ($k = 3$) domin. Poleg tega smo jih lepili vedno paroma po indeksih. Izračunajte vrednost domine po opisanem lepljenju za poljuben k.

3. Opišite podatkovno strukturo, ki vam bo omogočala, da za poljuben n domin hitro opravljate operacijo lepljenja. Pri tem morate ugotoviti seveda vrednosti obeh lepljenih domin ter nato izračunati ter zabeležiti vrednost zlepljene domine.

Namig: Na predavanjih smo lepljenje imenovali malce drugače, iskanje vrednosti domine pa kako že? Odgovor na to vprašanje ni daljši od petih vrstic.

Naloga 108. Imamo naslednjo igro. Opravka imamo z dominami, ki se lahko med seboj zlepijo. Vsaka domina ima svoj id: 1, 2, 3, 4, 5, 6. Na vsaki domini je poleg id-ja še vrednost domine, ki je na začetku kar enaka id-ju. Kot rečeno, lahko domine med seboj spajamo in ko domini spojimo, dobimo večjo domino, katere vrednost je enaka manjši od vrednosti spojenih domin. Na primer, če zlepimo domini 1 in 3, ima nova domina vrednost $1 = \min(1,3)$, medtem ko, če zlepimo domini 5 in 4, ima nova domina vrednost 4. Če sedaj zlepimo obe zlepljeni domini, se pravi (1, 3) in (5, 4) dobimo veliko domino (1, 3, 4, 5) z vrednostjo 1. Lepljenje domin definiramo z operacijo $L(d_1, d_2)$, kjer sta d_1 in d_2 id-ja kateregakoli delčka domin, ki jih želimo zlepiti. Zgoraj opisano lepljenje tako opišemo z ukazi:

$$L(1,3), L(5,4), L(1,5)$$

ali pa tudi z

$$L(1,3), L(5,4), L(1,4)$$

oziroma

$$L(1,3), L(5,4), L(3,5)$$

ali pa celo

$$L(1,3), L(5,4), L(3,4).$$

Poleg tega imamo na voljo še operacijo $V(x)$, ki vrne vrednost super-domine, kateri pripada v tem trenutki domina z id-jem x.
Vprašanja:

1. Recimo, da imamo domine z id-ji od 1 do 8 in naslednje operacije:

$$L(1,2), L(3,4), V(1), V(3), V(7),$$
$$L(5,6), L(7,8), L(2,3), V(1), V(3), V(7),$$
$$L(6,7), L(1,8), V(1), V(3), V(7).$$

Kakšne so vrednosti, ki jih vrnejo posamezne $V()$ operacije?

Namig: Opravljajte vse operacije po vrsti in si izpisujte vrednosti.

2. V zgornjem primeru smo imeli $n = 2^k$ $(k = 3)$ domin. Poleg tega smo jih lepili vedno paroma po indeksih. Izračunajte vrednost domine po opisanem lepljenju za poljuben k. Utemeljite odgovor.

3. Opišite podatkovno strukturo, ki vam bo omogočala, da za poljubno število domin n hitro opravljate operacijo lepljenja in preverjanja vrednosti.

Naloga 109. Butalska uprava želi posodobiti svoj informacijski sistem. Tako je s pomočjo Petra Zmede pripravila javni razpis in vanj vnesla zahtevo, da mora sistem podpirati naslednje funkcije:

```
ZaposliV(podjetje, oseba)
  // oseba se zaposli v podjetju
Prevzemi(podjetje1, podjetje2)
  // podjetje1 prevzame podjetje2
  // vključno z njihovimi zaposlenimi
ZaposlenV(oseba)
  // rezultat poizvedbe je, v katerem podjetju
  // je zaposlena oseba
```

Vprašanja:

1. (i) Kakšno podatkovno strukturo bi uporabili? Utemeljite odgovor. (ii) Zapišite psevdokodo vaše rešitve – implementacijo vseh zgornjih funkcij.

NAMIG: Seveda lahko uporabite kakšno znano podatkovno strukturo, ki je ni potrebno potem implementirati. Na primer, slovar, vrsto s prednostjo ali še kaj tretjega.

2. (i) Komentirajte zgornji nabor funkcionalnosti. (ii) Se vam zdi smiselen? (iii) Bi kaj dodali ali odvzeli? Utemeljite odgovor.

3. Podjetje ima seveda svoje ime (morda tudi naslov), zato imamo zahtevo po še eni funkciji:

```
1  PodjetjeIme(podjetje) -> ime
```

(i) Kako bi razširili podatkovno strukturo, da bi učinkovito podprli še to funkcionalnost? (ii) Ustrezno nadgradite funkcije, ki ste jih implementirali na začetku naloge?

Poglavje 3

Algoritmi

Drugi del zbirke se osredotoča na algoritme, oziroma na tehnike in metode za njihovo načrtovanje. Naloge v zbirki vključujejo med drugim požrešno metodo, deli in vladaj, dinamično programiranje. Zaključujemo še s temo o P in NP.

3.1 Urejanje

Urejanje je eden izmed najbolj osnovnih problemov pri študiju algoritmov. V tem razdelku se posvetimo klasičnim algoritmom za reševanje problema urejanja danega končnega zaporedja elementov. Algoritmi kot so urejanje z mehurčki, urejanje z vstavljanjem, urejanje z zlivanjem, hitro urejanje in urejanje s kopico vsi temeljijo na primerjanju elementov med seboj. Z uporabo odločitvenega drevesa lahko pokažemo, da v primerjalnem modelu ne moremo urediti n števil z manj kot $\Omega(n \log n)$ primerjavami. Tako sta recimo urejanje s kopico in urejanje z zlivanjem optimalna algoritma v tem modelu računanja.

Vernejše modeliranje računalnika prinaša model računanja RAM, ki omogoča urejanje v času $o(n \log n)$. V tem modelu računanja se srečamo, ob dodatnem upoštevanju končne univerzalne množice, z urejanjem s štetjem ter metodama korenskega urejanja in urejanja z vedri.

Cilj

Da razumemo implementacije različnih klasičnih algoritmov za urejanje in znamo izmed njih izbrati najboljšega za določeno aplikacijo. Poleg tega pri implementaciji razumemo uporabo požrešne metode in metode deli in vladaj.

Naloga 110. *Urejanje.* Marsikdaj nam vedenje o podatkih lahko pomaga, da lahko uporabimo ali načrtamo učinkovitejši algoritem.
Vprašanja:

1. Recimo, da moramo urediti 1.000.000 števil, pri čemer vemo, da imamo po 1.000 kopij števil med 1 in 1.000. Predlagajte časovno učinkovit postopek ter ocenite in utemeljite njegovo učinkovitost.

2. Ponovno imamo 1.000.000 števil in ponovno jih je po 1.000 kopij posameznih števil, za katera pa sedaj vemo samo to, da so 32 bitna. Predlagajte tokrat časovno učinkovit postopek ter ocenite in utemeljite njegovo učinkovitost.

3. Recimo, da imamo enak primer kot v prejšnjem vprašanju, le da so sedaj 1.024 bitna. Kakšen bi bil pa sedaj časovno učinkovit postopek? Ocenite in utemeljite njegovo učinkovitost.

Naloga 111. Urejanje (*sorting*) je ena najpogostejših operacij, ki jih počnemo z računalniki. Ena od oblik urejanja je urejanje z zlivanjem. Recimo, da sta $S_1[1..n_1]$ in $S_2[1..n_2]$ urejeni zaporedji števil.
Vprašanja:

1. Zapišite algoritem, ki zlije S_1 in S_2 v zaporedje $S[1..(n_1 + n_2)]$.

2. Pokažite, da je v najslabšem primeru potrebna vsaj $2(n_1 + n_2) - 1$ primerjava za zlivanje zaporedij S_1 in S_2.

3. Sedaj predpostavimo, da sta zaporedji S_1 in S_2 shranjeni v istem polju $A[1..(n_1+n_2)]$, kjer so na prvih n_1 mestih elementi zaporedja S_1 in na zadnjih n_2 mestih elementi zaporedja S_2. Z drugimi besedami, na mestih $A[1..n_1]$ so elementi zaporedja S_1 in na mestih $A[n_1+1..(n_1+n_2)]$ so elementi zaporedja S_2.

Zapišite algoritem, ki bo zlil zaporedji in shranil rezultat v polje A ter pri tem porabil čim manj dodatnega prostora razen samega polja A.[1]

Naloga 112. Imamo naslednja števila

$$59, 29, 93, 40, 40, 21, 82, 37, 36, 82, 15, 38, 6, 8, 5, 86, 68,$$

[1] Takšnemu zlivanju rečemo zlivanje v istem prostoru ali v angleščini: *in-place merging.*

ki so shranjena v polju (*array*) `stev[0..24]` – v splošnem dolžine n. Eno od urejanj, ki smo jih obravnavali, je *hitro urejanje*, ki sestoji iz korakov: razdeli in uredi vsakega od delov.
Vprašanja:

1. Korak *razdeli* opišemo s psevdokodo:

```
Naključno izberi števila a, b in c iz polja stev.
Izračunaj mediano m med števili a, b in c.
Razdeli stev tako,
 da so v njem najprej elementi, ki so manjši od m,
 nato elementi, ki so enaki m,
 ter nato še elementi, ki so večji od m.
```

(i) Naj bodo `a`, `b` in `c` na indeksih 21, 7 in 12 v `stev[0..24]`. Razdelite polje `stev[0..24]`. (ii) Zapišite kodo za zgornjo psevdokodo. (iii) Kakšna je njena časovna zahtevnost v primerjalnem modelu? Utemeljite odgovor.

2. (i) Za vašo kodo iz prejšnjega vprašanja podajte še oceno časovne zahtevnosti v dostopovnem modelu. (ii) V računalnikih dejansko dostop do pomnilnika ne traja enako časa za poljubno mesto. Procesor namreč, ko prebere eno lokacijo, prebere še nekaj dodatnih in vse skupaj spravi v predpomnilnik: recimo, ko prebere lokacijo d, prebere v predpomnilnik mesta od $\lfloor d/k \rfloor\, k$ do $\lceil d/k \rceil\, k - 1$ (oboje vključno). Pri tem je k neka konstanta, ki je odvisna od procesorja. Branje iz predpomnilnika je nato brezplačno. Ocenite časovno zahtevnost vaše kode iz prejšnjega vprašanja v dostopovnem modelu s predpomnilnikom.[2] Utemeljite odgovor.

3. Peter Zmeda je ujel zlato ribico in ribica mu bo izpolnila eno željo. Petrova velika želja je, da bi v hitrem urejanju razdeljevanje naredil v času $\gamma = O(1)$. (i) Kakšna je časovna zahtevnost hitrega urejanja ob izpolnjeni želji? Utemeljite odgovor. (ii) Komentirajte željo in njeno izpolnjevanje.

Naloga 113. Če imamo v računalniku poljubno besedilo, porabimo za shranjevanje posameznih črk besedila od 8-16 bitov – v vsakem primeru za vsako črko enako število bitov. Vendar se črke ne pojavljajo enako pogosto. Kaj bi se zgodilo, če bi za kodiranje različnih črk uporabili različno število bitov?

Poglejmo si primer:

[2] V odgovoru bo k parameter.

Imamo besedilo: `aaabacbba`.

Imamo 3 črke. Za vsako črko potrebujemo po 2 bita (recimo: `a`=`00`, `b`=`01`, `c`=`10`), kar pomeni, da za zakodiranje tega besedila potrebujemo $9 \cdot 2 = 18$ bitov: `000000010010010100`.

Če pa `a` zapišemo kot bitni kodni niz `1`, `b` kot bitni kodni niz `01`, in `c` kot bitni kodni niz `00`, potem je koda besedila `1110110001011` in potrebujemo $5 \cdot 1 + 3 \cdot 2 + 1 \cdot 2 = 13$ bitov!

Vprašanja:

1. Recimo, da je koda črke `x` kodni bitni niz w_x in koda črke `y` kodni bitni niz w_y. Kakšno razmerje *mora* veljati med kodnima nizoma w_x in w_y, da bo kodo $w_x w_y$ možno dekodirati kot `xy`?

2. Imamo abecedo $\Sigma = \{a_1, a_2, \ldots, a_n\}$ in besedilo $t[1..m]$, ki je sestavljeno iz črk naše abecede ($n << m$). (i) Zapišite algoritem, ki izračuna pogostnosti ν_i pojavljanja posameznih črk a_i v besedilu t. (ii) Kakšna je časovna in prostorska zahtevnost vašega algoritma?

3. Za abecedo Σ imamo izmerjene pogostnosti pojavljanja ν_i posameznih črk a_i v besedilu. (i) Opišite, kaj mora veljati za dolžine kodnih nizov w_i posameznih črk a_i; *ali* za več točk (ii) zapišite algoritem, kako izračunati kodne bitne nize za posamezne znake.

Naloga 114. Ena od osnovnih operacij pri hitrem urejanju je razdeli (*partition*). Wikipedia ponuja naslednjo kodo zanj:

```
algorithm Partition(A, lo, hi)
    pivot := A[hi]
    i := lo
    for j := lo to hi - 1 do
        if A[j] <= pivot then
            swap A[i] with A[j]
            i := i + 1
    swap A[i] with A[hi]
    return i
```

Vprašanja:

1. Kaj dobimo, če izvedemo `Partition(S, 3, 7)`, kjer je `S` definiran kot polje nad števili $59, 29, 93, 40, 40, 21, 82, 37, 36, 82, 15, 38, 6, 8, 5, 86, 68$?

2. Če v hitrem urejanju uporabimo opisano kodo, urejanje ni stabilno. Zakaj?

3. Naredite ga stabilnega. Kakšna je časovna zahtevnost vaše rešitve? Utemeljite odgovor.

Naloga 115. Če imamo v računalniku poljubno besedilo, za shranjevanje posameznih črk besedila porabimo od 8-16 bitov – v vsakem primeru za vsako črko enako število bitov. Vendar se črke ne pojavaljajo enako pogosto. Kaj bi se zgodilo, če bi za kodiranje različnih črk uporabili različno število bitov?

Poglejmo si primer:

> Imamo besedilo: `aaabacbba`
>
> Imamo 3 črke. Za vsako črko potrebujemo po 2 bita (recimo: `a`=`00`, `b`=`01`, `c`=`10`), kar pomeni da za zakodiranje tega besedila potrebujemo $9 \cdot 2 = 18$ bitov: `000000010010010100` .
>
> Če pa `a` zapišemo kot bitni kodni niz `1`, `b` kot bitni kodni niz `01`, in `c` kot bitni kodni niz `00`, potem je koda besedila `1110110001011` in potrebujemo $5 \cdot 1 + 3 \cdot 2 + 1 \cdot 2 = 13$ bitov!

Vprašanja:

1. Imamo besedilo $T[1..n]$, ki sestoji iz črk abecede $\Sigma = \{a_1, a_2, \ldots, a_s\}$ ($s << m$). (i) Zapišite psevdokodo algoritma, ki izračuna pogostnosti ν_i pojavljanja posameznih črk a_i v besedilu T. (ii) Kakšna je časovna in prostorska zahtevnost vašega algoritma?

2. Za abecedo Σ imamo izmerjene pogostnosti pojavljanja ν_i posameznih črk a_i v besedilu. Opišite, kakšno pravilo mora veljati za dolžine kodnih nizov w_i posameznih črk a_i.

3. Zapišite algoritem, kako izračunati kodne bitne nize za posamezne črke abecede Σ upoštevaje vaše pravilo.

Naloga 116. Peter Zmeda je našel naslednji košček programske kode:

```
1  void krneki (a[] int; b: int)
2    c= 0;
3    while c < b-1 do
4      d= c+1;
5      while d < b do
6        if a[c] > a[d] then
7          x= a[c]; a[c]= a[d]; a[d]= x
8        endif
9        d= d+1;
10     enddo;
11     c= c+1;
12   enddo;
13 endfunction;
```

Vprašanja:

1. Peter se ne upa uporabiti funkcije `krneki`, ker ne ve, kaj počne. Kaj počne v resnici funkcija?

Namig: Pomagate si lahko s sledenjem izvajanja kode za majhne primere.

2. Po katerih štirih stvareh se sprašujemo ob vsakem algoritmu (funkciji)? Kakšni so vaši odgovori ob zgornji kodi na ta vprašanja? Odgovore utemeljite.

3. Funkcija `krneki` je zapisana nerekurzivno. Zapišite jo v rekurzivni obliki.

Naloga 117. Imamo cela števila $a_j, j = 0, 1, \ldots, n-1$. Nad njimi definiramo predponske vsote s_i s predpisom $s_i = \sum_{j=0}^{i} a_j$.
Vprašanja:

1. Napišite algoritem, ki za vse $i = 0, 1, \ldots, n-1$ izračuna s_i, in dokažite njegovo pravilnost.

2. Kakšna je časovna in kakšna prostorska zahtevnost vašega algoritma ter ali se dâ izračunati s hitreje? Utemeljite odgovor.

3. Recimo, da imamo vse s_i naračunane. Ali lahko na podlagi le-teh najdemo podzaporedje števil $a_l, \ldots, a_r$, katerih vsota je 0? Začrtajte algoritem in utemeljite njegovo pravilnost ter časovno zahtevnost.

Naloga 118. Peter Zmeda se je znašel pred naslednjo nalogo. Vsako število x lahko zapišemo kot produkt praštevil

$$x = p_1^{e_1} p_2^{e_2} \cdots p_i^{e_i} \cdots p_k^{e_k}. \tag{3.1}$$

Na primer $48 = 2^4 3^1$. Očitno ima vsako število x vsaj dva faktorja, kar se zgodi, ko je x praštevilo in sta faktorja 1 in sam x. Faktorje lahko združimo v dva podprodukta x_1 in x_2, kjer $x = x_1 \cdot x_2$. Peter mora sedaj poiskati za dani x takšna x_1 in x_2, da bo njuna razlika najmanjša – x_1 in x_2 imenujmo *težišče*. Recimo, pri $x = 48$ imamo naslednje možne pare (x_1, x_2): $(1, 48)$, $(2, 24)$, $(3, 16)$, $(4, 12)$, $(6, 8)$, $(8, 6)$, $(12, 4)$, $(16, 3)$, $(24, 2)$ in $(48, 1)$. Para, ki ju Peter išče, sta $(6, 8)$ in $(8, 6)$.
Vprašanja:

1. Poiščite težišče za naslednja števila: 24, 96 in 2016.

2. Napišite algoritem, ki, pri danem naboru faktorjev in eksponentov (p_i, e_i) (prim. enakost (3.1)) za x, poišče težišče števila x. Utemeljite pravilnost vašega algoritma.

3. Kakšna je časovna in kakšna prostorska zahtevnost vašega algoritma? Utemeljite odgovor.

Namig: Učinkovitejša kot bo vaša rešitev, več točk boste dobili. Nekaj točk dobite tudi za eksponentno rešitev v vsoti eksponentov e_i iz enakosti (3.1).

Naloga 119. *Urejanje.* Ko govorimo o problemu urejanja, ga običajno obravnavamo v najsplošnejši obliki, kjer so števila povsem naključna, kakor tudi njihov vrstni red. Toda Peter Zmeda dela v velikem podatkovnem centru, kjer obdelujejo podatke enkrat tedensko. Tedaj jih uredijo po velikosti, medtem ko med tednom stranke svoje podatke poljubno spreminjajo.
Vprašanja:

1. Recimo, da je število zapisov v podatkovni bazi n in število sprememb podatkov med tednom m. Na primer, naj bodo urejeni podatki na začetku tedna naslednji

$$6 \quad 7 \quad 18 \quad 28 \quad 31 \quad 39 \quad 44 \quad 52 \quad 68 \quad 92$$

in konec tedna naslednji

$$6 \quad 8 \quad 18 \quad 28 \quad 9 \quad 39 \quad 44 \quad 72 \quad 68 \quad 92.$$

V tem primeru je $n = 10$ in $m = 3$. Recimo, da pri danem n veste, da je $m = O(1)$. Opišite algoritem, ki učinkovito ponovno uredi števila v tem primeru. Kakšna je časovna zahtevnost vašega algoritma. Utemeljite odgovor.

2. In algoritem, če sta n in m povsem splošna? Pri tem upoštevajte, da na začetku ne veste, katere vrednosti so spremenjene, samo, da jih je m. Utemeljite odgovor tudi tokrat in analizirajte časovno zahtevnost vaše rešitve.

NAMIG: Komentirajte rešitev v primerjavi z ono pod prejšnjim vprašanjem.

3. Se bi vaša rešitev kaj spremenila, če bi si med tednom beležili, katere vrednosti so se spremenile? Utemeljite vaš odgovor.

Naloga 120. Imamo polje števil $a_j, j = 0, \ldots, n-1$, kjer je a_j neko celo število. Nad števili definiramo predponsko vsoto s_i na sodih indeksih, kjer je s_i vsota vseh števil na sodih indeksih od a_0 do a_i – očitno, če je i sodo število, potem je a_i del vsote sicer pa ne. Podobno definiramo še predponsko vsoto l_i števil na lihih indeksih.
VPRAŠANJA:

1. Napišite algoritem, ki izračuna s_i in l_i za vse $i = 0, 1, \ldots, n-1$, in dokažite njegovo pravilnost.

2. Kakšna je časovna in kakšna prostorska zahtevnost vašega algoritma ter, ali se dâ izračunati s hitreje? Utemeljite odgovor.

3. Recimo, da imamo vse s_i naračunane. Ali lahko na podlagi le-teh najdemo podzaporedje števil $a_l, \ldots, a_r$, katerih vsota je 0? Začrtajte algoritem in utemeljite njegovo pravilnost ter časovno zahtevnost.

Naloga 121. Peter Zmeda je od prijateljice Špele dobil knjižnico, ki vsebuje funkcijo `Positions`, katera se obnaša na sledeč način

$$\texttt{Positions}(T, p_1) \texttt{ -> } R_1 = \{x_1, x_2, \ldots, x_{r_1}\},$$
$$\texttt{Positions}(T, p_2) \texttt{ -> } R_2 = \{y_1, y_2, \ldots, y_{r_2}\},$$

kjer so x_i in y_i cela števila. Peter je preiskusil delovanje omenjene funkcije in dobil rezultat

$$R_1 = \{5, 22, 8, 77, 3\}, \quad R_2 = \{97, 4, 14, 41\}. \tag{3.2}$$

Vprašanja:

1. Peter ima sedaj nalogo, da v zaporedjih R_1 in R_2 poišče tisti par števil (x_i, y_j), katerih razlika je najmanjša. (i) Na primeru iz (3.2) poiščite ta par. (ii) Napišite algoritem, ki v splošnem poišče ta par. (iii) Ocenite časovno zahtevnost vašega algoritma in utemeljite svoj odgovor.

2. Peter je dobil novo nalogo. Tokrat ima poleg zaporedij R_1 in R_2 še vrednost k in mora poiskati *vse* tiste pare (x_i, y_j), za katere velja $|x_i - y_j| < k$. (i) Poiščite vse pare iz primera (3.2) za $k = 25$. (ii) Napišite algoritem, ki v splošnem poišče vse pare za dani k. (iii) Ocenite časovno zahtevnost vašega algoritma in utemeljite svoj odgovor.

3. Recimo, da vemo $x_i, y_j \in \{1, 2, \ldots, n\}$. Ali bi vaša algoritma lahko delovala kaj hitreje? Utemeljite odgovor.

Naloga 122. *Uvod in osnove.* Imamo naslednji algoritem za urejanje:

```
1  Sort(A) {
2    for(j= 0; j < n-1; j++) {
3      int iMin = j;
4      for(i= j+1; i < n; i++) {
5        if (A[i] < a[iMin]) iMin= i;
6      }
7      if(iMin != j) {
8        tmp= A[j]; A[j]= A[iMin]; A[iMin]= tmp
9      }
10   }
11 }
```

Vprašanja:

1. Recimo, da za vrstico i program porabi c_i časa. Zapišite zahtevnost programa kot funkcijo $c_i, i = 1, 2, \ldots, 11$.

2. Kakšna je časovna zahtevnost programa? Utemeljite odgovor.

3. Primerjajmo zgornji program za urejanje s programom `InsertionSort`, ki smo ga obravnavali na predavanjih. Katerega izmed njiju bi uporabili? Utemeljite svoj odgovor.

3.2 Rang in izbira

Ko imamo množico n elementov, se za nek element pogosto sprašujemo, kateri po velikosti je v tej množici, ali obratno, kateri element je i-ti najmanjši po velikosti. Opisani funkciji poznamo pod imenoma rang in izbira. Posebni element, ki ga iščemo, je srednji element po velikosti in ga imenujemo mediana.

Če je izvedba funkcije rang preprosta, saj samo v linearnem času prištejemo število elementov, ki so manjši, je izbira trši oreh. Naivna metoda bi verjetno najprej uredila elemente v času $O(n \log n)$ in nato vrnila odgovor v konstantnem času. Izkaže se, da v resnici lahko funkcijo izbira z metodo deli in vladaj izvedemo prav tako v linearnem času.

Cilj

Da razumemo, znamo začrtati in uporabiti učinkovite izvedbe funkcij rang in izbira.

Naloga 123. Ta naloga je podobna nalogi izpred nekaj let, a je zgolj podobna. Tokrat Peter rešuje problem, kjer uporabnik dinamično dodaja v podatkovno strukturo nove in nove elemente ter občasno še poizveduje po drugem tercilu, to je vrednost, od katere jih je tretjina večjih. Primer zaporedja operacij, ki jih uporabnik izvaja je (`I` pomeni vstavi element in `T` vrni drugi tercil):

`I` 17, `I` 3, `I` 5, `I` 1, `T`, `I` 10, `I` 4, `T`, `I` 11,...

Vprašanja:

1. Kaj vrne prvi `T` v zgornjem zaporedju in kaj drugi?

2. Predlagajte Petru učinkovito rešitev. Utemeljite svoj odgovor in podajte časovno zahtevnost vaše rešitve.

Namig: Odgovor na to vprašanje ni dolg samo dve vrstici. Podrobnejša oziroma boljša kot bo vaša rešitev, več točk boste dobili.

3. (i) Recimo, da dodamo še ukaz `S`, ki pomeni, da odslej ne bomo več dodajali elementov. Se vaša rešitev kaj spremeni? (ii) Se kaj spremeni, če poleg `S` dodamo še ukaz `M`(i), ki vrne element na i-tem mestu.

Namig: Ponovno odgovor na to vprašanje ni dolg samo dve vrstici in podrobnejši oziroma boljši bo, več točk boste dobili.

Naloga 124. Peter Zmeda je napisal naslednjo funkcijo za iskanje k-tega števila v polju `a[0..n-1]`:

```
function Kti(a, n, k)
  for i= n-1 downto 0 do
    c[i]= 0
    for j= 0 to i do
      if a[j] >= a[i] c[i]++;
  for i= 0 to n-1 do
    if c[i] == k
      println("k-ti po velikosti je a[i] na mestu i");
```

Za sedaj predpostavimo, da so vsi elementi polja `a` med seboj različni.
Vprašanja:

1. Kakšna je časovna in prostorska zahtevnost Petrove funkcije? Utemeljite odgovor.

2. Žal ima Petrova koda napako. Poiščite jo, opišite kje in zakaj se zgodi ter jo popravite.

3. Sedaj predpostavimo, da so lahko v polju enaki elementi. Slednje ima za posledico, da je več elementov k-tih po velikost. Na primer, v polju $(12, 4, 6, 3, 4)$ sta na drugem mestu tako druga kot zadnja štirica. Kako bi popravili Petrov program, da bi deloval pravilno tudi v tem primeru? Utemeljite odgovor – pokažite pravilnost delovanja vaše funkcije.

Naloga 125. *Vrste s prednostjo in rang elementa.*
Vprašanja:

1. Danes res ni Petrov srečni dan – ali pač? Včeraj je v Butalah na bolšjaku našel prekrasno implementacijo vrste s prednostjo `KjutiPQ`, ki ima običajne metode `Make`, `Insert`, `Min` in `DelMin` ter jo nemudoma kupil.

Danes mu je šef ob prihodu v službo dal novo nalogo. V arhivu so našli neko datoteko števil, ki jih želijo urediti po velikosti. Toda glej ga zlomka – urediti jih želijo od največjega števila do najmanjšega. Na prvi pogled si Peter ne bo mogel dosti pomagati s `KjutiPQ` – ali pač? Pomagajte Petru. Na koncu ocenite časovno in prostorsko zahtevnost vaše rešitve. Utemejite odgovor.

2. Med drugim smo na predavanjih srečali binomsko in Fibonaccijevo kopico. Slednja izgleda smiselnejša in boljša rešitev implementacije vrste s

prednostjo od prve. Najdite vsaj en primer, kjer je uporaba binomske kopice ustreznejša od Fibonaccijeve.

NAMIG: Katere lastnosti nima Fibonaccijeva kopica in jo ima binomska?

3. Na predavanjih smo spoznali tudi linearni algoritem za iskanje k-tega elementa med neurejenimi elementi. Prvi korak je bila razdelitev elementov na peterke. Recimo, da razdelimo elemente namesto na peterke na sedmerke. Ali algoritem še vedno deluje in kakšna je njegova časovna zahtevnost? Utemeljite odgovor.

NAMIG: Morda izgleda naloga zahtevna, a ni. Samo ponoviti je potrebno analizo s predavanj.

Naloga 126. *Iskanje k-tega elementa.*
VPRAŠANJA:

1. Peter Zmeda je našel prevedeno knjižnico `mediana`. Ugotovil je, da vsebuje samo funkcijo `mediana(A)`, ki vrne srednji element polja `A`. (i) Kako lahko uporabi to funkcijo, da najde k-ti element za poljuben $1 \leq k \leq n$, kjer je n število elementov v tabeli `A`? (ii) Naj bo $f(n)$ časovna zahtevnost funkcije `mediana(A)`. Kakšna je časovna zahtevnost vaše rešitve? Utemeljite odgovor.

2. Recimo, da za $f(n)$ iz prejšnjega vprašanja velja $f(n) = o(n)$. Ali to vpliva na čas urejanja polja `A`? Utemeljite odgovor.

3. Na predavanjih smo opisali postopek iskanja k-tega elementa, pri katerem razdelimo elemente v peterke. Dokažite, da postopek deluje, če razdelimo elemente na sedmerke.

3.3 Dinamično programiranje

Dinamično programiranje se običajno uporablja za reševanja optimizacijskih problemov, pri katerih se pogosto pojavijo podproblemi iste oblike. Ključna tehnika, ki pospeši reševanje problema, je pomnjenje (memoizacija), ki rešitve vsakega takega podproblema pomni in v primeru, da se podproblem ponovi, vrne shranjeno rešitev.

CILJ

Da znamo za rešitev optimizacijskega problema začrtati rekurzivno rešitev. Nato, da znamo za rekurzivno začrtano rešitev uporabiti tehniko dinamičnega programiranja skupaj s pomnjenjem. S slednjo lahko določene algoritme, ki imajo sicer eksponentno časovno zahtevnost, spremenimo v polinomske algoritme.

Naloga 127. Imamo naslednjo definicijo števila C_n

$$C_n = n + \frac{1}{n} \sum_{1<i\leq n} (C_{i-1} + C_{n-1}) \tag{3.3}$$

in $C_0 = 1$. V tej nalogi učinkovitejša rešitev dobi več točk.
VPRAŠANJA:

1. Izračunajte C_5 iz enačbe (3.3) in pri tem pokažite postopek računanja.

2. (i) Zapišite funkcijo (algoritem), ki izračuna C_n iz enačbe (3.3) z uporabo dinamičnega programiranja, pri čemer uporabite rekurzijo. (ii) Kakšna je časovna in prostorska zahtevnost vaše rešitve? Utemeljite odgovor.

3. (i) Sedaj pa zapišite funkcijo (algoritem), ki prav tako izračuna C_n iz enačbe (3.3) z uporabo dinamičnega programiranja, vendar gradi rešitev od spodaj navzgor. (ii) Kakšna je časovna in prostorska zahtevnost vaše rešitve? Utemeljite odgovor.

Naloga 128. Definirajmo A_n kot

$$A_n = 1 + A_{\lceil \frac{n}{2} \rceil} + A_{\lfloor \frac{n}{2} \rfloor}, \tag{3.4}$$

kjer še velja $A_0 = 0$ in $A_1 = 1$.
VPRAŠANJA:

1. Izračunajte A_9 iz enačbe (3.4) in pri tem pokažite postopek računanja.

2. (i) Zapišite funkcijo (algoritem), ki izračuna A_n iz enačbe (3.4) z uporabo dinamičnega programiranja, pri čemer uporabite rekurzijo. (ii) Kakšna je časovna in prostorska zahtevnost vaše rešitve? Utemeljite odgovor.

3. (i) Sedaj zapišite funkcijo (algoritem), ki prav tako izračuna A_n iz enačbe (3.4) z uporabo dinamičnega programiranja, vendar gradi rešitev od spodaj navzgor. (ii) Kakšna je časovna in prostorska zahtevnost vaše rešitve? Utemeljite odgovor.

Naloga 129. *Dinamično programiranje.* Recimo, da imamo naslednja zapisa DNK:

$$S_1 = \text{ACCGGTCGAGTGCGGAAGCCGGCCGAA in}$$
$$S_2 = \text{GTCGTTCGGAATGCCGTTGCTCTGTAAA.}$$

Ena od razdalj med nizoma je število različnih mest, na katerih se črki v obeh nizih razlikujeta in jo imenujemo *Hammingova razdalja*. Na primer, $d_H(\text{AC}, \text{GCGTT}) = 1 + 3 = 4$, kjer 1 pride iz razlike med A in G ter 3, ker je drugi niz za 3 črke daljši.

Podobnost med S_1 in S_2 lahko definiramo še drugače. Naj bo podzaporedje niza S_1 zaporedje črk, ki jih izberemo iz S_1 z leve proti desni, vendar lahko pri tem kakšno črko spustimo. Primeri podzaporedij niza S_1 so: (i) ACCGGTCGAGTGCGGAAGCCGGCCGAA – sam niz S_1; (ii) ACCGG – začetne črke niza S_1; (iii) GTCGT – 4., 6., 7., 8. in 11. črka v S_1 in podobno naprej. Podobno lahko oblikujemo podzaporedja za S_2. Poleg tega definiramo še skupno podzaporedje, ki hkrati nastopa v S_1 in v S_2. Od zgornjih podzaporedij sta drugi dve podzaporedji skupni. Mero podobnosti med S_1 in S_2 definiramo kot *dolžino najdaljšega skupnega podzaporedja* $d_S()$.
VPRAŠANJA:

1. Zapišite algoritem, ki za niza S_a in S_b izračuna $d_H(S_a, S_b)$.

2. Zapišite algoritem, ki za niza S_a in S_b izračuna $d_S(S_a, S_b)$.

3. Kakšna je časovna in kakšna prostorska zahtevnost vaših algoritmov ter ali se da narediti boljši algoritem? Utemeljite odgovor.

Naloga 130. *Dinamično programiranje.* Recimo, da imamo naslednja zapisa DNK:

$$S_1 = \text{ACCGGTCGAGTGCGGAAGCCGGCCGAA in}$$
$$S_2 = \text{GTCGTTCGGAATGCCGTTGCTCTGTAAA.}$$

Obstaja vrsta različnih mer podobnosti med molekulami DNK. Na primer, ena od razdalj je število različnih mest, na katerih se črki v obeh nizih razlikujeta, in jo imenujemo *Hammingova razdalja*. Drugo razdaljo smo omenili na vajah in se imenuje *Levenshteinova razdalja*. Tokrat pa bomo spoznali razadaljo, ki jo predstavlja *najdaljše skupno podzaporedje*. Skupno podzaporedje je prav tako zaporedje zaporedje črk kot DNK, vendar ima lastnost, da se njegove črke pojavijo v enakem zaporedju v obeh molekulah DNK. Recimo, da imamo zaporedje ACGT. To je skupno podzaporedje v S_1 in S_2, saj se pojavi na mestih 1, 2, 5 in 6 (in še kje) v S_1 ter na mestih 10, 15, 16 in 17 (in še kje) v S_2. Ni pa najdaljše, saj je ACGTT tudi skupno podzaporedje, ki pa je daljše.
VPRAŠANJA:

1. Poiščite najdaljše skupno podzaporedje v S_1 in S_2 (daljše kot bo vaše podzaporedje, več točk boste dobili).

2. Pri tej nalogi imamo opravka z rešitvijo z dinamičnim programiranjem. Slednje pomeni, da bomo iskali maksimum preko različnih možnosti. Zapišite formulo, ki jo boste maksimirali. Očitno bo formula rekurzivna.

NAMIG: Razmislite sledeče: recimo, da je prvi niz (npr. S_1) dolg n znakov in drugi (npr. S_2) m znakov. Upoštevajte:

- Kaj pomeni, če sta zadnja znaka obeh nizov enaka? Kako bo z rekurzijo? Kako se bo krajšal prvi in kako drugi niz?
- Kaj pomeni, če sta zadnja znaka obeh nizov različna? Kako bo z rekurzijo sedaj? Kako se bo krajašal prvi in kako drugi niz? Katere so vse možnosti krajšanja, ki jih moramo pregledati, da najdemo najboljšo?

3. Zapišite algoritem, ki poišče dolžino najdaljšega skupnega podzaporedja. Če uporabljate rekurzijo (je lažje), uporabite tehniko pomnjenja. Ocenite časovno zahtevnost vaše rešitve.

Naloga 131. Peter Zmeda ima za domačo nalogo sedem problemov, pri čemer lahko za vsakega od problemov dobi različno število točk. Poleg tega tudi ocenjuje, da potrebuje za reševanje različnih problemov različno količino časa:

točke	7	9	5	12	14	6	12
čas (v h)	3	4	2	6	7	3	5

Za izdelavo domače naloge ima na voljo 15 ur časa. Očitno ima na voljo premalo časa, da reši vse probleme, in tako se mora odločiti, katere naj rešuje. Opravka imamo z optimizacijskim problemom.
VPRAŠANJA:

1. Najprej predpostavimo, da lahko dobi delne točke. To pomeni, da, če reši pol prvega problema, porabi 1,5h in dobi 3,5 točke. (i) Predlagajte, katere probleme in kako naj jih rešuje, da bo dobil čim več točk. (ii) Utemeljite pravilnost vaše rešitve.

2. Sedaj predpostavimo, da Peter dobi točke samo, če v celoti reši problem (načelo vse ali nič). (i) Katere probleme iz zgornje tabele naj sedaj reši? (ii) Utemeljite pravilnost rešitve tokrat.

3. Kaj pa če imamo n problemov ter iščemo optimalno rešitev, ko moramo probleme v celoti rešiti? Zapišite algoritem.

Naloga 132. Število C_n je definirano z rekurzijo

$$C_n = n + \frac{1}{n} \sum_{1 \leq i \leq n} (C_{i-1} + C_{n-1}). \tag{3.5}$$

V tej nalogi učinkovitejša rešitev dobi več točk.
VPRAŠANJA:

1. Izračunajte C_5 iz enačbe (3.5) in pri tem pokažite postopek računanja.

2. (i) Zapišite funkcijo (algoritem), ki izračuna C_n iz enačbe (3.5) z uporabo dinamičnega programiranja, pri čemer uporabite rekurzijo. (ii) Kakšna je časovna in prostorska zahtevnost vaše rešitve? Utemeljite odgovor.

3. (i) Sedaj pa zapišite funkcijo (algoritem), ki prav tako izračuna C_n iz enačbe (3.5) z uporabo dinamičnega programiranja, vendar gradi rešitev od spodaj navzgor. (ii) Kakšna je časovna in prostorska zahtevnost vaše rešitve? Utemeljite odgovor.

Naloga 133. Recimo, da imamo dva številčna niza

$$t_1 = \texttt{05937299952727},$$
$$t_2 = \texttt{9952727369263848},$$

kjer je abeceda $\Sigma = \{\texttt{0}, \texttt{1}, \texttt{2}, \texttt{3}, \texttt{4}, \texttt{5}, \texttt{6}, \texttt{7}, \texttt{8}, \texttt{9}\}$. Definirajmo p kot podniz besedila t, če nastopa kot njegov del. Na primer `999` je podniz t_1, kakor tudi podniz `27`.

VPRAŠANJA:

1. Najdaljši skupni podniz nizov u in v je podniz, ki je v obeh nizih in ne obstaja noben daljši podniz, ki bi bil tudi v obeh nizih. Poiščite najdaljši skupni podniz nizov t_1 in t_2.

2. (i) Zapišite algoritem, ki bo poiskal najdaljši skupni podniz besedil u in v, kjer je $|u| = n_u$ in $|v| = n_v$. (ii) Kakšna je časovna zahtevnost vašega algoritma? Utemeljite pravilnost odgovora. (iii) Ali menite, da je mogoče najti najdaljši podniz hitreje? Utemeljite odgovor.

3. Sedaj pa problem obrnimo. Še vedno imamo niza u in v, kjer je $|u| = n_u$ in $|v| = n_v$, vendar dobimo od Petra podniz p_{uv}, za katerega trdi, da je najdaljši skupni podniz. (i) Zapišite algoritem, ki preveri pravilnost trditve. (ii) Ocenite in utemeljite časovno zahtevnost vašega algoritma. (iii) Ali se da narediti bolje?

Naloga 134. *Dinamično programiranje.* Cefizelj je uspel priti v butalsko zakladnico. V njej je našel sedem različnih dragocenosti

$$(8, 9), (7, 20), (9, 14), (5, 10), (12, 27), (6, 12), (4, 15), \tag{3.6}$$

kjer prva številka v paru pomeni *težo* dragocenosti v kg in druga njeno *vrednost*.

VPRAŠANJA:

1. Na srečo lahko Cefizelj odnese na enkrat le 25 kg. (i) Katere dragocenosti iz seznama (3.6) naj pobere, da bo plen največji? (ii) Utemeljite, da je plen v resnici največji.

2. Očitno gre za optimizacijski problem, ki ga v resnici lahko rešimo z dinamičnim programiranjem. V splošnem imamo n parov (w_i, v_i) in ne samo sedem kot v seznamu (3.6) ter W skupno dovoljeno težo pobranih dragocenosti. (i) Zapišite optimizacijsko formulo, ki vam bo omogočila zapisati rekurzivno rešitev.

NAMIG: Na predavanjih smo zapisali naslednjo formulo za optimalni vrstni red množenj matrik

$$c(1, n) = \begin{cases} 0, & \text{če } n = 0 \text{ ali } n = 1; \\ \min_{i=1\ldots n-1} c(1, i) + c(i + 1, n - i) + d_0 d_i d_n, & \text{sicer.} \end{cases}$$

(ii) Zapišite rekurzivno rešitev vašega problema z uporabo pomnenja, ki pove, kolikšna je največja vrednost odnešenih dragocenosti. (iii) Kakšni sta prostorska in časovna zahtevnost vaše rešitve? Utemeljite odgovor.

3. (i) Rešitev razširite tako, da boste tudi vrnili, katere dragocenosti je potrebno odnesti, da bo vrednost največja. (ii) Kakšni sta prostorska in časovna zahtevnost vaše rešitve? Utemeljite odgovor[3].

Naloga 135. *Dinamično programiranje.* Imamo operacijo

```
Razcepi(a, k) -> (n1, n2),
```

ki črkovni niz `a` dolžine n = `a.len` razcepi na dva niza, kjer je prvi niz `n1` kopija niza `a` od 1. do (vključno) k. znaka, drugi niz `n2` pa preostanek izvornega niza `a`, kjer je $1 \leq k < n$. Cena te operacije je `a.len` – dolžina niza `a`.

Recimo, da je niz `a` enak

```
PRVI IZPITNI ROK,
```

ki ga lahko na dva načina razcepimo na nize ,,`PRVI `", ,,`IZPITNI `" in ,,`ROK`" (pozor presledki) s pomočjo zaporedja uporabe operacije `Razcepi`.
Vprašanja:

1. (i) Zapišite obe zaporedji operacij, ki razcepita izvorni niz na zahtevane besede in (ii) izračunajte ceno obeh razcepov.

2. V splošnem imamo niz a dolžine n in p indeksov $c_1, c_2, \ldots, c_p$ ($c_i < c_j$, če $i < j$), pri katerih razcepimo izvorni niz. Zapišite formulo za dinamični program, ki poišče najcenejši način razcepa niza.

3. (i) Zapišite psevdokodo dinamičnega programa, kjer boste uporabili tehniko pomnenja. (ii) Ocenite časovno zahtevnost svoje rešitve in utemeljite rezultat.

---

[3]Če pri 2 niste uspeli napisati algoritma, rešite to vprašanje za problem optimalnega vrstnega reda množenja matrik.

Naloga 136. Recimo, da imamo matrike naslednjih razsežnosti: 30×15, 15×10, 10×25 in 25×20.
Vprašanja:

1. Izračunajte optimalni vrstni red množenja zgornjih matrik, če štejemo samo množenja in ne seštevanj. Prikažite izračun.

2. Pri množenju dveh matrik uporabljamo operaciji seštevanja in množenja dveh števil, zapisanih v plavajoči vejici. V prvem vprašanju smo povsem zanemarili operacije seštevanja. Naš prijatelj Peter Zmeda pa je sestavil procesor, v katerem množenje dveh števil v plavajoči vejici porabi 135ns in seštevanje 12ns. (i) Zapišite optimizacijsko formulo dinamičnega programiranja za Petrov procesor. (ii) Zapišite funkcijo, ki vrne predvideni najmanjši čas množenja n matrik.

3. Najprej, (i) popravite zgornjo funkcijo tako, da bo vrnila tudi optimalno zaporedje množenj matrik; in nato (ii) naračunajte optimalni vrstni red za množenje matrik iz uvoda.

Naloga 137. Binomske koeficiente $\binom{n}{k}$ lahko računamo na različne načine. Eden od načinov je z uporabo enačbe

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}, \tag{3.7}$$

pri čemer vemo, da velja $\binom{n}{1} = n$ in $\binom{n}{0} = 1$.
Vprašanja:

1. Z uporabo enačbe (3.7) izračunajte vrednosti $\binom{5}{4}$, $\binom{5}{3}$, $\binom{6}{4}$ in $\binom{6}{3}$.

2. Zapišite algoritem, ki bo izračunal $\binom{n}{k}$ za poljuben $n \geq k \geq 0$. Za učinkovitejšo rešitev boste dobili več točk.

3. Kakšna je prostorska in časovna zahtevnost vašega algoritma? Utemeljite odgovor.

Naloga 138. Peter Zmeda ima dobrega prijatelja Miho, ki prodaja električne kable. Običajno dobijo v prodajalno kable dolžine n metrov, ki jih potem prodajajo po kosih. Vendar imajo zanimivega šefa, ki je določil ceno prodanih kablov glede na dolžino, pri čemer se cene spreminjajo po naslednji tabeli

dolžina (m)	1	2	3	4	5	6	7	8
cena (EUR)	1	5	8	9	10	17	17	20

Vprašanja:

1. Koliko največ lahko Miha iztrži pri razrezu in prodaji kabla dolžine 8m? Utemeljite odgovor.

2. Zapišite algoritem, ki naračuna razrez pri kablu dolžine n metrov in prinese največji dobiček. Utemeljite pravilnost vašega algoritma.

3. Kakšna je časovna in kakšna prostorska zahtevnost vašega algoritma? Utemeljite odgovor.

Naloga 139. V Butalah so čudni tiči in, ker so čudni, imajo tudi čudne vrednosti kovancev. Tako imajo kovance za 31, za 17, za 10 in za 1 BUT. Na solomatu (da, to je avtomat, kjer kupuješ butalsko sol in njene izdelke) je običajno potrebno vrniti drobiž, če stranka plača več, kot je cena izdelka. Nalogo, kakšne kovance naj solomat vrne, so zaupali Petru Zmedi in mu pri tem zabičali, da naj njegov algoritem vrne najmanjše število kovancev. Peter je zasnoval naslednji algoritem:

```
1  Vrni(znesek):
2    while znesek >= 31:
3      vrni kovanec za 31 BUT
4      znesek= znesek-31
5    while znesek >= 17:
6      vrni kovanec za 17 BUT
7      znesek= znesek-17
8    while znesek >= 10:
9      vrni kovanec za 10 BUT
10     znesek= znesek-10
11   while znesek >= 1:
12     vrni kovanec za 1 BUT
13     znesek= znesek-1
```

Vprašanja:

1. (i) Kakšno tehniko je uporabil Peter pri načrtovanju svojega algoritma? Utemeljite odgovor. (ii) Izpišite, koliko kovancev in katere vrne Petrov algoritem pri zneskih 30 BUT, 50 BUT in 70 BUT.

2. Žal Petrov algoritem ne vrne najmanjšega števila kovancev pri vsakem znesku. (i) Poiščite najmanjši znesek, pri katerem Petrov algoritem ne deluje pravilno. (ii) Kako bi opisali zneske, pri katerih Petrov algoritem odpove?

3. (i) Sestavite algoritem, ki vedno naračuna najmanjše število kovancev, ki jih moramo vrniti. (ii) Utemeljite pravilnost vašega algoritma. (iii) Kakšna je časovna in kakšna prostorska zahtevnost vašega algoritma?

Naloga 140. Pri obdelavi besedila pogosto le-to razdelimo na dva ali več delov. Recimo, če imamo besedilo $t =$ `dober dan` in ga razdelimo na dva dela z ukazom `(l, d)= razdeli(t, 3)`, dobimo dva niza in sicer $l =$ `dob` in $d =$ `er dan`. Z drugimi besedami, besedilo t smo razdelili po tretji črki. Cena takšne delitve je dolžina izvornega besedila, kar je v našem primeru 9. Seveda lahko tako l kot d delimo naprej.
Vprašanja:

1. Recimo, da imamo besedilo t dolžine 23 črk, ki ga razdelimo na 4 dele in sicer prvi je dolg 3, drugi 4, tretji 10 in četrti preostalih 6 črk. Koliko najmanj stane takšna razdelitev? Utemeljite odgovor.

Namig: Podajte zaporedje rezanja izvornega besedila t in pri tem upoštevajte ceno vsakega razreza.

2. Zapišite formulo za izračun optimalnega razreza besedila t dolžine n, kjer imamo m točk razreza, podanih v polju R.

Namig: V primeru iz prvega vprašanja je $R = [3, 7, 17]$.

3. Zapišite še algoritem za izračun optimalnega razreza ter ocenite njegovo časovno zahtevnost.

Naloga 141. Da, v Butalah imajo tudi žago *Žagbut*. Na žagi režejo deske na različne dolžine, ki jih merijo v komolcih. En komolec (kml) je približno 44 cm. Cene desk na trgu dosegajo različne cene, kot je razvidno iz naslednje preglednice (BUT so butalski tolarji):

dolžina (kml)	1	2	3	4	5	6	7	8	9
cena (BUT)	2	4	7	9	10	11	12	19	22

Vprašanja:

1. Osnovna dolžina desk, ki jih proizvajajo, je 7 kml. Direktor Luka Kratkohlačnica bi rad optimiral dobiček in sedaj ne ve, ali naj prodaja deske v tej dolžini ali naj jih razreže na krajše deske. (i) Izračunajte optimalne dolžine razrezov in iztrženo ceno.

2. Butalci so nekaj časa uspešno poslovali, nato pa ugotovili, da rezanje nekaj stane. Pomagajte Luki in sestavite algoritem, ki bo upošteval, da en rez stane 2 BUT. (i) Zapišite cenovno funkcijo. (ii) Zapišite psevdokodo algoritma, ki bo upošteval stroške razreza in izračunal optimalne dolžine ter prihodek po prodaji desk.

3. Tudi ta rešitev ni bila pravilna, saj je med stroški razreza potrebno upoštevati ne samo sámo žaganje, ampak tudi vpenjanje. Slednje pa je odvisno od dolžine. Tako cena enega reza stane a/b BUT, kjer sta a in b dolžini dobljenih kosov in velja $a \leq b$. Recimo, da prerežemo 10 komolčno desko na 3 in 7 komolčni deski, slednje stane 3/7 BUT. (i) Zapišite še sedaj cenovno funkcijo, ki bo upoštevala stroške razreza. (ii) Zapišite še algoritem, ki izračuna optimalni razrez.

Naloga 142. *Dinamično programiranje.* Peter Zmeda se je vsemu navkljub lotil reševanja problema optimalnega množenja matrik s požrešno metodo. Njegova metoda je, da seznam matrik $M_1, M_2, \ldots, M_i, M_{i+1}, \ldots, M_n$ razdeli pri tisti matriki M_i, ki ima največje število stolpcev. Z drugimi besedami, najprej bi zmnožil matrike od M_1 do M_i, nato matrike M_{i+1} do M_n ter dobljeni matriki med seboj:

$$(M_1 \cdot M_2 \cdots M_i) \cdot (M_{i+1} \cdots M_n)\,.$$

Vprašanja:

1. Najdite protiprimer, ko Petrov pristop ne da optimalnega vrstnega reda množenja matrik.

2. Cefizelj je velik nepridiprav. Tokrat se je odločil, da bo na tržišče dal „izboljšan“ program za iskanje najboljšega vrstnega reda množenja matrik. Njegova izboljšava seveda odraža njegovo naravo in bo izboljšava v resnici vrnila najslabši vrstni red množenja matrik – se pravi tistega, ki zahteva največ skalarnih produktov. Napišite takšen program.

Namig: Naloga je *zelo* lahka, saj lahko za osnovo vzamemo kodo s predavanj, ki potrebuje samo nekaj manjših popravkov.

3. Floyd-Warshalov algoritem za iskanje najkrajših poti med vsemi pari vozlišč v grafu je tudi primer dinamičnega programiranja. (i) Zapišite algoritem in utemeljite ali vaša inačica uporablja pristop od zgoraj navzdol s pomnjenjem ali od spodaj navzgor. (ii) Kje v vašem algoritmu nastopa ,,dinamičnost“ – iskanje optimalne rešitve na osnovi znanih optimalnih rešitev različnih podproblemov?

Naloga 143. No, tudi to se je zgodilo. Peter Zmeda se je brezmejno zaljubil v Alenčico, ki pa mu njegove pozornosti ne vrača. Zato se je odločil, da njeno pozornost pritegne s tem, da se nauči nekaj novih spretnosti, s katerimi jo bo očaral. Kot dober informatik se je lotil zadeve silno sistematično. Najprej je naredil seznam spretnosti, za katere misli, da se jih lahko nauči, in za vsako spretnost ocenil, koliko časa bi mu vzela, da se je nauči. Poleg tega je pri Alenčini sestri Cvetoslavi poizvedel, koliko Alenčica ceni posamezno spretnost. Vse skupaj je spravil v spodnjo preglednico:

spretnost	*čas učenja*	*vrednost*
igranje orglic	12	11
igranje kitare	20	15
ples	8	15
mešanje koktejlov	6	9
astronomija	15	10
igranje košarke	5	7
kuhanje hrenovk	2	6
dokaz Einstein-Pitagorovega izreka	5	6

Stolpec *vrednost* podaja, koliko Alenčica ceni posamezno spretnost, medtem ko stolpec *čas učenja* podaja čas v dnevih, ki ga Peter potrebuje, da se nauči določene spretnosti.

Vprašanja:

1. Najprej predpostavimo, da se Peter določene spretnosti lahko nauči tudi delno. Z drugimi besedami, če se bo učil igranje košarke samo tri dni, se je bo naučil zgolj 60% in bo Alenčica to cenila ne z vrednostjo 7, ampak samo z vrednostjo 4,2. Peter ima do novega snidenja z Alenčico natančno 24 dni časa. Katere spretnosti in koliko se jih naj nauči, da bo naredil na Alenčico najboljši možen vtis? Utemeljite pravilnost svojega odgovora.

2. Jojmene! Ko je Peter malce premislil vse skupaj, je ugotovil, da tole delno učenje spretnosti ne deluje: spretnosti se mora naučiti povsem, sicer

ne bo učinka. Koliko največ vtisa lahko Peter naredi na Alenčico tokrat, če je do njunega naslednjega snidenja še vedno natančno 24 dni? Utemeljite odgovor.

3. V splošnem imamo n spretnosti, kjer spretnost i vzame t_i časa, da se je Peter nauči in naredi vtis v_i na Alenčico. Zapišite algoritem, ki poišče nabor spretnosti, ki se jih Peter lahko nauči v času T, da bo naredil najboljši vtis na Alenčico.

Naloga 144. *Dinamično programiranje in rekurzivne podatkovne strukture.* Imamo običajno dvojiško iskalno drevo definirano kot:

```
1 public class Drevo {
2   int   koren;
3   Drevo levo, desno
4 }
```

Recimo, da imamo naslednja zapisa DNK:

$$S_1 = \text{ACCGGTCGAGTGCGGAAGCCGGCCGAA},$$
$$S_2 = \text{GTCGTTCGGAATGCCGTTGCTCTGTAAA}.$$

Podobnost med S_1 in S_2 lahko definiramo na različne načine. Na predavanjih smo spoznali mero Levenshteinove razdalje. Tokrat definirajmo razdaljo drugače. Naj bo podzaporedje niza S_1 zaporedje črk, ki jih izberemo iz S_1 z leve proti desni, vendar lahko pri tem kakšno črko spustimo. Primeri podzaporedij niza S_1 so: ACCGGTCGAGTGCGGAAGCCGGCCGAA – sam niz S_1; ACCGG – začetne črke niza S_1; ali GTCGT – 4., 6., 7., 8. in 11. črka v S_1. Podobno lahko oblikujemo podzaporedja za S_2.

Poleg tega definiramo še skupno podzaporedje, ki hkrati nastopa v S_1 in v S_2. Od zgornjih podzaporedij sta drugi dve podzaporedji skupni. Mero podobnosti med S_1 in S_2 definiramo kot dolžino najdaljšega skupnega podzaporedja.

Vprašanja:

1. Na predavanjih smo srečali različne izpise dvojiških dreves: vmesni (*inorder*), premi (*preorder*) in obratni (*postorder*). Tokrat definiramo plastni obhod, ki izpiše elemente drevesa po plasteh: najprej koren; nato oba otroka korena; sledijo vnuki korena; in tako naprej.

Napišite algoritem, ki na ta način izpiše elemente drevesa. Utemeljite pravilnost njegovega delovanja.

Namig: Verjetno je smiselneje najprej zapisati idejo algoritma, ki služi tudi kot utemeljitev, in šele nato psevdokodo.

2. Zapišite algoritem, ki pri danih zaporedjih X in Y ugotovi, ali je X podzaporedje Y ali obratno. (i) Najprej razložite, kako deluje algoritem in nato (ii) zapišite njegovo psevdokodo. (iii) Kakšno tehniko sestavljanja algoritmov ste uporabili in zakaj?

3. Naj bosta S_1 in S_2 dve zaporedji. Napišite algoritem, ki izračuna dolžino njunega najdaljšega skupnega podzaporedja.

Namig: Tukaj uporabite tehniko dinamičnega programiranja. Če boste pravilno sestavili rekurzivno enačbo, boste dobili večino točk.

Naloga 145. *Dinamično programiranje in rekurzivne podatkovne strukture.* Na predavanjih smo večkrat srečali Fibonaccijeva števila, ki so bila definirana kot $F_n = F_{n-1} + F_{n-2}$, kjer je $F_0 = F_1 = 1$. Tokrat definirajmo zaporedje števil malce drugače in sicer kot

$$P_n = \left\lfloor \frac{n}{2} \right\rfloor + P_{\lfloor \frac{n}{2} \rfloor} + P_{\lceil \frac{n}{2} \rceil}, \tag{3.8}$$

kjer je $P_0 = P_1 = 0$.
Vprašanja:

1. Koliko je P_{10}, kjer je P_n definiran v enačbi (3.8)? Prikažite izračun!

2. Zapišite algoritem, ki izračuna P_n za poljuben n, kjer je n parameter. Kakšna je časovna in kakšna prostorska zahtevnost vašega algoritma?

3. Imamo običajno dvojiško iskalno drevo definirano kot:

```
1 public class Drevo {
2   Elt   koren;
3   Drevo levo, desno
4 }
```

Napišite algoritem, ki bo izpisal elemente v vozliščih drevesa na sledeči način:

$$\mathit{koren}\colon n,$$

kjer je *koren* vrednost elementa in n število elementov v drevesu, ki so manjši od njega.

Namig: Uporabite vmesni (*inorder*) obhod z enim ali več dodatnimi parametri.

Naloga 146. *Kateri so najtežji?* Ponovno smo pri igri iz naloge 55, le da tokrat težo pri vsakem otroku zmanjšamo za 111 enot in imamo sedaj pare: $(120, 1)$, $(128, -33)$, $(153, -3)$, $(98, 19), \ldots, (114, -8)$.
Vprašanja:

1. Vsota tako spremenjenih tež med katerima višinama je največja? Opišite postopek, kako ste to izračunali.

2. Recimo, da imamo n otrok in da so vse njihove višine različne. Kakšna je časovna in prostorska zahtevnost vašega postopka iz prvega vprašanja? Ocenite zahtevnosti za posamezne korake (faze), če seveda vaš postopek sestoji iz njih.

3. Če primerjamo teži dveh otrok, njuno razliko zapišemo kot

- A, če je prvi več kot 20 enot lažji od drugega;
- B, če je teža prvega 20 ali manj enot manjša od teže drugega;
- C, če je teža drugega enaka teži prvega ali 20 ali manj enot manjša od teže drugega; ali
- D, če je teža drugega otroka več kot 20 enot manjša od teže prvega otroka.

Če se vrnemo k našemu primeru iz prejšnje naloge, lahko zapišemo množico kot DAACCACCB, če primerjamo paroma zaporedne otroke iz množice. Takšnemu zapisu rečemo karakteristika množice. Recimo, da imamo podani karakteristiki C_1 in C_2 dveh množic. Potem njuno *podobnost* določa zaporedje črk, ki se pojavlja v obeh karakteristikah v enakem vrstnem redu, vendar so lahko vmes še druge črke. Naj bo karakteristika C_1 enaka D**A**A**C**C**A**C**CB**, karkateristika C_2 pa DD**AC**CBBC**A**B**CB**CC. Potem je primer podobnosti zaporedje črk ACACB (črke podobnosti so označene odebeljeno v obeh karakteristikah). Razdaljo med karakteristikama definiramo kot dolžino najdaljše podobnosti.

(i) Kakšno tehniko programiranja bi uporabili, da bi našli najdaljšo podobnost? (ii) Skicirajte algoritem.

3.4 Algoritmi na grafih

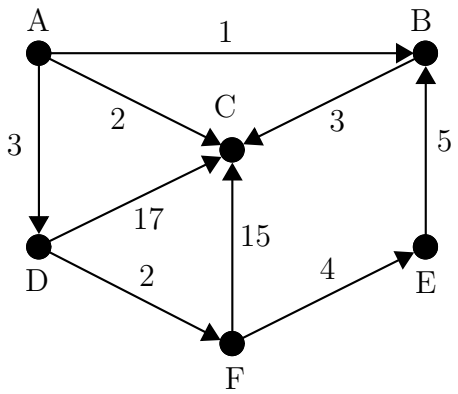
Ogromno zanimivih in povsem praktičnih problemov lahko modeliramo z grafi. V tem razdelku obravnavamo nekaj izmed pomembnejših algoritmov na grafih. Prvi je pregledovanje grafa, ki ga lahko izvedemo v širino ali globino. Pregledovanje v širino v neuteženem grafu izračuna najkrajše poti iz danega vozlišča. Drugo obliko praviloma uporabljamo pri iskanje cikla ali povezanih komponent v grafu, predstavlja pa tudi jedro algoritma za topološko urejanje v usmerjenem grafu.

Poleg preprostih pregledovanj v razdelku srečamo še algoritme za izračun najkrajših poti v uteženem grafu in algoritma za iskanje najcenejšega vpetega drevesa v uteženem grafu. Pri načrtovanju uporabljamo požrešno metodo in metodo dinamičnega programiranja.

Cilj

Da razumemo in znamo implementirati zgoraj omenjene algoritme ter biti sposobni razpoznati probleme, kjer jih lahko uporabimo. Nadalje, da znamo pri načrtovanju algoritmov uporabiti različne metode od požrešne pa do dinamičnega programiranja.

Naloga 147. V tej nalogi se bomo ukvarjali z iskanjem najkrajše poti med enim vozliščem ter vsemi ostalimi vozlišči. Kot primer si oglejmo graf na sl. 3.1.



Slika 3.1: Primer grafa.

Vprašanja:

1. S pomočjo Dijkstrovega algoritma v grafu na sl. 3.1 poiščite najkrajše poti od vozlišča A do vseh ostalih vozlišč. Rešitev naj nazorno prikazuje, kako se spreminjajo vrednosti v podatkovnih strukturah.

2. Naš nesrečni Peter Zmeda si je zapomnil, da Dijkstrov algoritem ne deluje pravilno, če so v grafu povezave z negativno vrednostjo. Dokažite to trditev.

Namig: Naredite graf, za katerega Dijkstrov algoritem ne deluje pravilno.

3. Peter je prišel na preprosto idejo, da bi v grafu $G(V, E)$ ($|V| = n, |E| = m$):

1. poiskal povezavo, ki ima najbolj negativno utež w_m;
2. utežem vseh povezav prištel $-w_m$, da dobi nov graf $G'(V, E')$, kjer imajo vse povezave nenegativno utež;
3. na grafu G' pognal Dijkstrov algoritem in

dobil najkrajše povezave od enega vozlišča do vseh ostalih vozlišč v izvornem grafu G. (i) Analizirajte, kakšna je časovna zahtevnost Petrovega algoritma; in (ii) s protiprimerom pokažite, da ne deluje pravilno.

Naloga 148. *Grafi.* Na predavanjih smo spoznali problem najkrajše poti od enega izvora do vseh ostalih vozlišč. Za rešitev problema smo spoznali Dijkstrov algoritem, vendar smo opozorili, da ne deluje vedno. V nalogi predpostavimo, da imamo graf $G(V, E)$, kjer je $|V| = n$ in $|E| = m$.
Vprašanja:

1. (i) V katerem primeru algoritem ne deluje? (ii) Odgovor utemeljite. (iii) Kateri algoritem uporabimo tedaj in zakaj slednji deluje?

Namig: Bodite pozorni na natančnost vašega odgovora.

2. Kakšna je časovna zahtevnost Dijkstrovega in kakšna onega drugega algoritma? Utemeljite odgovor.

3. Vračamo se k Dijkstrovemu algoritmu. Recimo, da tokrat vemo, da so uteži na povezavah grafa iz končne množice $\{1, 2, \ldots, W\}$ celih števil. Kako lahko izkoristimo to dejstvo? Utemeljite odgovor.

Naloga 149. Imamo utežen graf $G(V, E)$. V grafu G definiramo premer kot najdaljšo najkrajšo razdaljo med katerimakoli vozliščema.
Vprašanja:

1. Opišite algoritem, ki poišče premer grafa. Ali ima lahko takšen algoritem boljšo časovno zahtevnost kot algoritem za iskanje najkrajše razdalje med poljubnima vozliščema grafa? Utemeljite odgovor.

2. Recimo, da imamo p procesorjev. Kako lahko pospešimo iskanje premera grafa? Opišite postopek, utemeljite njegovo pravilnost in ocenite časovno zahtevnost.

Namig: Če je vzporedno iskanje premera prezahtevno, opišite postopek, kako vzporedno poiščete najkrajše razdalje med poljubnima paroma vozlišč. Opišite postopek, utemeljite njegovo pravilnost in ocenite časovno zahtevnost. Dobili boste sicer nekaj manj točk.

3. Recimo, da so v grafu G vse uteži enake. Kako sedaj izgleda vaš algoritem za iskanje premera grafa? Kakšna je njegova časovna zahtevnost? Utemeljite odgovor.

Naloga 150. *Algoritmi na grafih.* Butale so res čudna dežela. Še bolj čudni so pa župani in njihove odločitve. Zadnja je posebej zanimiva, namreč župan Francot Turkavidel je sprejel odlok, da morajo biti vse ceste v Butalah dolge natančno eno butalsko miljo. Peter Zmeda je takoj začutil poslovno priložnost in pričel ponujati spletno storitev, ki je izračunala najkrajšo pot med dvema krajema v butalski deželi. Število krajev v butalski deželi naj bo n in število cest m.
Vprašanja:

1. (i) Opišite *podatkovno* strukturo, ki naj jo Peter uporabi za učinkovito implementacijo svoje storitve. (ii) Zapišite algoritem, ki naračuna razdaljo med krajema a in b. (iii) Kakšna je časovna zahtevnost vašega algoritma? Utemeljite odgovor.

2. Peter se je odločil, da enkrat za vselej naračuna najkrajše poti med vsemi pari krajev v butalski deželi. (i) Zapišite Floyd-Warshalov algoritem za iskanje najkrajših poti med poljubnima krajema vendar tako, da bo možno rekunstruirati pot. (ii) Ali lahko Peter to naračuna hitreje kot z uporabo Floyd-Warshalovega algoritma? Utemeljite odgovor.

3. Nov župan in novi odloki. Francotov sin Gregor Brezhlačnice je malce popravil očetov odlok in sicer tako, da je dovolil, da so lahko ceste sedaj

dolge eno ali dve butalski milji. (i) Kaj naj naredi Peter, da bo lahko še vedno uporabil rešitev iz prvega vprašanja? Utemeljite odgovor. (ii) Koliko se spremeni časovna zahtevnost rešitve? Utemeljite odgovor.

Naloga 151.
VPRAŠANJA:

1. Imamo neutežen in neusmerjen graf $G(V, E)$ ter vozlišče s. Zapišite algoritem, ki bo poiskal v grafu G najbolj oddaljeno vozlišče od vozlišča s. Utemeljite pravilnost algoritma.

NAMIG: Najbolj oddaljeno vozlišče je tisto, do katerega je najkrajša pot najdaljša.

2. Tokrat naj bo graf $G_1(V, E, w)$ še vedno neusmerjen, so pa njegove povezave utežene ter še vedno imamo začetno vozlišče s. Zapišite sedaj algoritem, ki bo poiskal najbolj oddaljeno vozlišče od vozlišča s.

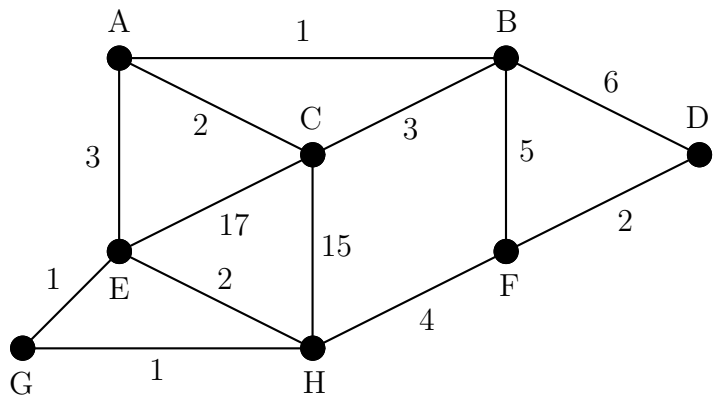
3. Kakšna je časovna zahtevnost obeh vaših algoritmov? Odgovor utemeljite. Za vse točke podajte natančno vrednost vodilnega koeficienta in ne samo O notacijo.

NAMIG: Upoštevajte, da je graf definiran tako z množico vozlišč V kot z množico povezav E, ki sta lahko zelo različni po velikosti.

Naloga 152. Tramvajsko omrežje v Butalah ima obliko grafa na sl. 3.2, oziroma v splošnem grafa $G(V, E)$, kjer je $|V| = n$ in $|E| = m$. Številke nad povezavami predstavljajo dolžine prog, medtem ko so vozlišča postajališča.
VPRAŠANJA:

1. Pavluša Očalasti je vsak ponedeljek moral obhoditi vse tramvajske tire kot so označeni na sl. 3.2. (i) Pomagajte mu in zapišite vrstni red postaj, skozi katere naj gre. (ii) Koliko je skupna dolžina prehojene Pavlušine poti in ali bi lahko bila krajša? Utemeljite odgovor.

2. Postajališče C je na glavnem trgu v Butalah pred mestno hišo, kjer domuje župan Vrban Podvrbosmuk. Vrban je naročil Petru Zmedi, da naj mu naračuna najkrajše poti od njegove mestne hiše do vseh ostalih postajališč. Naračunajte jih še vi. Pri tem pokažite postopek izračuna.



Slika 3.2: Tramvajsko omrežje v Butalah.

3. Pavluši gre tole sprehajanje po tirih presneto dobro od rok. Tako bi rad naredil algoritem, ki bi v poljubnem mestu izračunal vrstni red obiskov postajališč tako, da bi prehodil vse tire. Napišite ta algoritem, pokažite njegovo pravilnost in ocenite njegovo časovno zahtevnost.

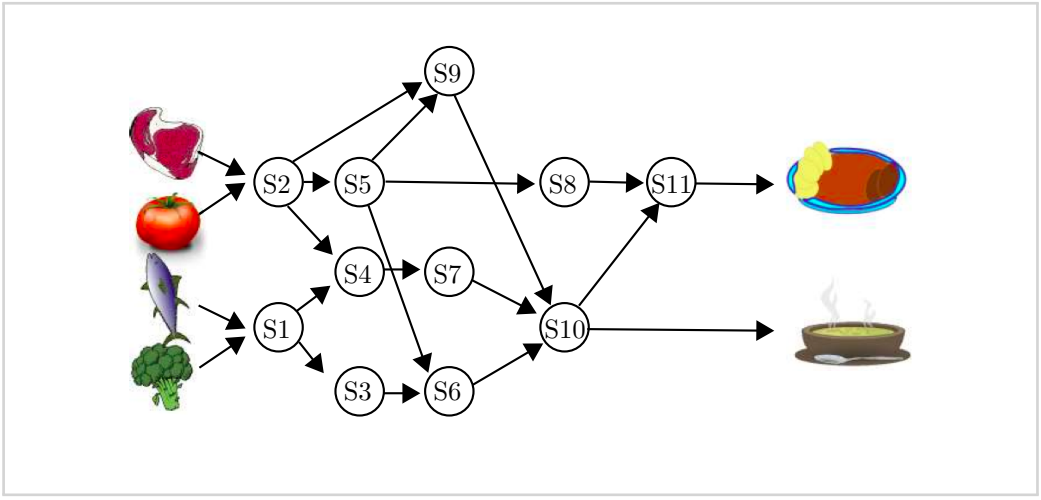
Naloga 153. Na osnovnošolskem tekmovanju *Bober* je bila naslednja naloga:

> Pri bobrih kuhanje ni tako preprosto kot pri ljudeh. Mama Vanda pripravlja dve jedi iz štirih sestavin – mesa, paradižnika, ribe in brokolija. Ribe in brokoli zmeša in kuha pet minut (S1). Prav tako pet minut kuha paradižnik in meso (S2). Paradižnik in meso razdeli na tri dele; prva dva ločeno kuha še pet minut (S5 in S9), drugega pa pomeša s polovico zmesi brokolija in rib, ter spet kuha pet minut (S4). Celoten postopek kaže sl. 3.3, kjer vsak krogec predstavlja pet minut kuhanja.

Vprašanja:

1. Koliko časa potrebuje mama Vanda, da skuha kosilo, če ima (i) neomejeno število loncev; (ii) samo dva lonca. Utemeljite odgovora!

2. V splošnem lahko vsak recept za pripravo jedi, oziroma kakršenkoli postopek za izvedbo nekega dela, ki sestoji iz opravil, modeliramo z grafom $G(V, E)$. (i) Opišite, kako bi skonstruirali v splošnem graf, s katerim bi opisali postopek sestavljen iz opravil.



Slika 3.3: Kuhanje mame Vande.

NAMIG: Opišite, kako izgledata množici V in E.

Recimo, da imamo usmerjen graf $G(V, E)$, ki opisuje izvajanje posameznih opravil postopka. (ii.) Ali graf G vedno opisuje postopek, ki je izvedljiv? Utemeljite odgovor in, če ne, zapišite algoritem, s katerim bi preverili izvedljivost postopka.

3. Zapišite algoritem, ki za dani usmerjeni graf $G(V, E)$, ki opisuje izvedljiv postopek, izračuna najkrajši čas izvedbe.

4. [DODATNO] Razmislite in opišite algoritem za izračun najkrajšega časa, če se lahko izvaja hkrati samo k opravil.

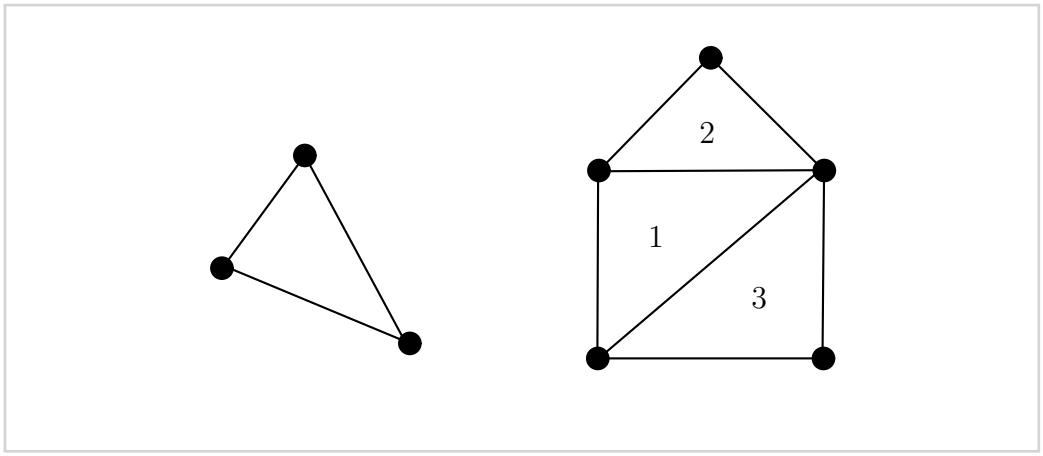
Naloga 154. *Grafi so drevesa in drevesa so grafi.* Peter Zmeda predava podatkovne strukture na Univerzi v Butalah. Študentom je dal za nalogo napisat različne algoritme na grafih: iskanje najcenejšega vpetega drevesa, iskanje najcenejših povezav iz enega vozlišča do ostalih in še nekaj drugih. Pravilnost delovanja študentskih rešitev je preverjal na primerih grafov. Seveda je tudi sam rešil nalogo, da bi imel pravilne rešitve. V nalogi imamo graf $G(V, E, w)$, kjer je $|V| = n$, $|E| = m$ in funkcija $w(u, v)$ predstavlja utež na povezavi $(u, v) \in E$.
VPRAŠANJA:

1. (i) Zapišite algoritem, ki vrne najkrajše poti $G_n(V_n, E_n)$ od vozlišča $s \in V$ do vseh ostalih vozlišč. (ii) Kako veliki sta množici V_n in E_n? Utemeljite

odgovor. (iii) Ali je možno, da je Peter naračunal drugačen graf $G_n(V_n, E_n)$ kot študent Luka Kratkohlačnica in sta oba pravilna? Utemeljite odgovor. Če ne, zakaj ne, če da, opišite primer grafa $G(V, E)$, ko se to lahko zgodi.

2. (i) Zapišite algoritem, ki vrne najcenejše vpeto drevo $G_d(V_d, E_d)$. (ii) Ali je pomembno, katero vozlišče proglasimo za koren drevesa? Utemeljite odgovor. (iii) Ali obstaja primer, ko sta na grafu $G(V, E, w)$ drevesi $G_n(V_n, E_n)$ in $G_d(V_d, E_d)$ enaki, če je $|V|$ vsaj 6 in $w(u, v)$ ni trivialna funkcija, kar pomeni, da ni teža vseh povezav enaka? Utemeljite odgovor.

3. Grafek (*graphlet*) je majhen grafek kot je na primer na sl. 3.4 trikotnik na levi. Grafek se lahko pojavi v večjih grafih, kot se pojavi grafek trikotnik kar



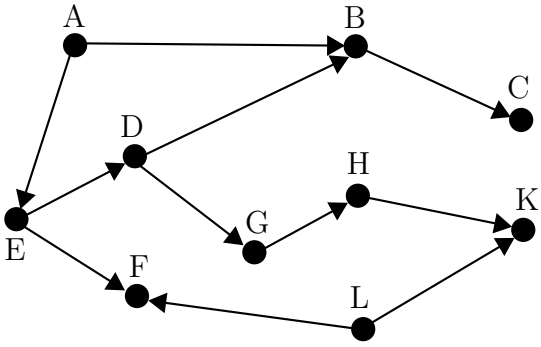
Slika 3.4: Grafek (trikotnik) in graf, v katerem se le-ta pojavi trikrat.

trikrat v desnem grafu na sl. 3.4. (i) Predlagajte postopek, kako poiščemo vsaj eno pojavitev grafka $G_g(V_g, E_g)$ v grafu $G(V, E)$.

Namig: Iščemo grafek, ki je majhen in ne kakšen velik podgraf.

(ii) Naj bo $|V_g| = n_g$, $|E_g| = m_g$, $|V| = n$ in $|E| = m$. Kakšna je časovna zahtevnost vašega algoritma?

Naloga 155. V tej nalogi se bomo ukvarjali s čim hitrejšim zaključkom dela, ki je razdeljeno na več opravil. Na sl. 3.5 vsako vozlišče predstavlja eno opravilo, povezave pa prikazujejo, katero opravilo mora biti prej zaključeno (vozlišče, iz katerega izhaja puščica), predno se lahko naslednje opravilo izvede (vozlišče, kamor kaže puščica). V splošnem imamo graf $G(V, E)$ ($|V| = n, |E| = m$).



Slika 3.5: Primer grafa.

Vprašanja:

1. Katero opravilo mora biti zaključeno prvo in katero je lahko zaključeno zadnje v primeru na sl. 3.5? Utemeljite odgovor.

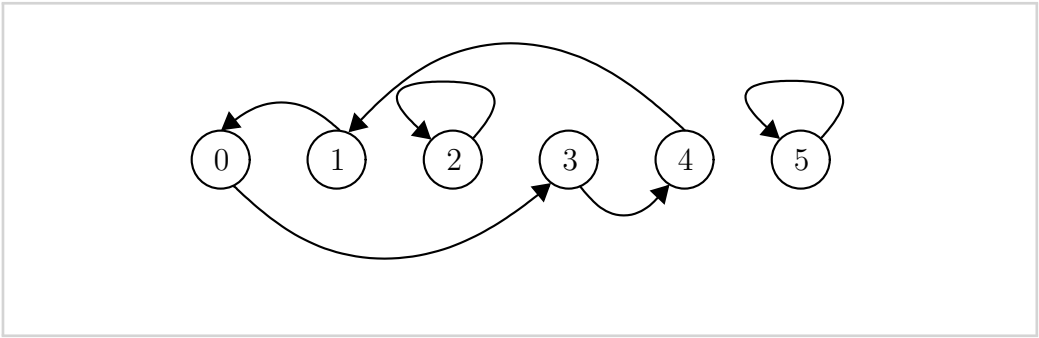
2. Predpostavimo, da vsako opravilo zahteva enako časa, da se ga izvede na enem procesorju. Recimo, da imamo na voljo en sam procesor in nas zanima, koliko časa potrebujemo, da opravimo vse delo. Podajte algoritem in njegovo časovno zahtevnost. Utemeljite odgovor.

3. Na koncu predpostavimo, da imamo poljubno število procesorjev, ki lahko hkrati izvajajo opravila in nas zanima, koliko najmanj časa potrebujemo sedaj pri danem grafu $G(V, E)$, da opravimo vse delo. Podajte algoritem in njegovo časovno zahtevnost. Utemeljite odgovor.

Naloga 156. Pri urejanju smo spoznali pojem *permutacijskega vektorja* π – vektor dolžine n, ki pove, da naj se element z indeksa i prestavi na indeks $\pi[i]$. Permutacijski vektor lahko predstavimo kot usmerjen graf. Na primer, vektor $\pi = [3, 0, 2, 4, 1, 5]$ je predstavljen na sl. 3.6. V grafu G imamo tri cikle in najdaljši je dolžine 5 ($0 \to 3 \to 4 \to 1 \to 0$), medtem ko sta druga dolžine 1.

Vprašanja:

1. (i) Če je dolžina permutacijskega vektorja n, kako veliki sta množici V in E? Utemeljite odgovor. (ii) Zapišite algoritem, ki poišče vse cikle v permutacijskem vektorju. Utemeljite njegovo pravilnost. (iii) Kakšna je časovna in prostorska zahtevnost vašega algoritma? Utemeljite odgovor.



Slika 3.6: Graf $G(V, E)$ permutacijskega vektorja π.

2. Kako bi povzporedili ta algoritem? Ali je kakšen drug algoritem primernejši za povzporejanje?

NAMIG: Učinkovitejše kot bo povzporejanje, več točk boste dobili.

3. Naj bosta π_R in π_G dva permutacijska vektorja[4] dolžine n. Vsak od vektorjev definira svoj graf $G_R(V_R, E_R)$ in $G_G(V_G, E_G)$, kjer $V_R = V_G$. Definirajmo unijo grafov $G = G_R \cup G_G$, kjer je $V = V_R = V_G$ in $E = E_R \cup E_G$. (i) Zapišite algoritem, ki bo poiskal v G najkrajši cikel, za katerega velja, da se bodo barve na povezavah izmenjevale: rdeča, zelena, rdeča in tako naprej. Utemeljite pravilnost vašega algoritma. (ii) Kakšna je časovna in prostorska zahtevnost vašega algoritma? Utemeljite odgovor. (iii) Zapišite algoritem, ki bo poiskal v G najdaljši cikel, za katerega velja, da se bodo barve na povezavah izmenjevale: rdeča, zelena, rdeča in tako naprej. Utemeljite pravilnost vašega algoritma.

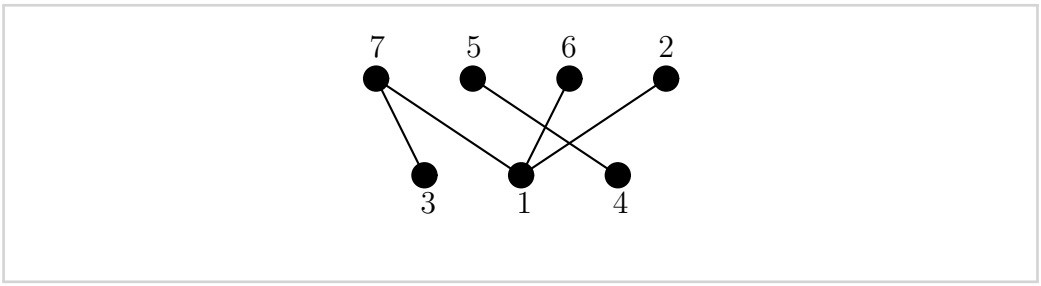
Naloga 157. Dvodelni (bipartitni) grafi so posebna oblika grafov, kjer lahko množico vozlišč V razdelimo na dve podmnožici V_1 in V_2 tako, da za vse povezave $(u, v) \in E$ velja $v \in V_1$ in $u \in V_2$. Primer takšnega grafa je na sl. 3.7.

VPRAŠANJA:

1. Popišite graf na sl. 3.7 z matriko sosednosti in še z incidenčno matriko.

2. Recimo, da imamo dvodelni graf $G(V, E)$, kjer je $|V| = n$ ter velja $|V_1| = n_1$ in $|V_2| = n_2$. Opišite, kako izgleda matrika sosednosti grafa G, in utemeljite svoj odgovor.

[4] R pomeni rdeči (*red*) in G zeleni (*green*).



Slika 3.7: Dvodelni graf.

NAMIG: Pomislite, ali lahko permutiramo vrstice in stolpce v matriki sosednosti.

3. Zapišite algoritem, ki preveri, ali je podani graf $G(V, E)$ dvodelen. Kakšna je njegova časovna zahtevnost?

Naloga 158. *Grafi.* Peter Zmeda se je odločil ponuditi storitev, ki bi izračunala, v katerem kolenu sta si dva Butalca v sorodu. Za definicijo kolena sorodstva je našel naslednji zapis[5]:

> Kolena ali stopnje sorodstva se lahko računa različno. Slovenski pravni red kolena sorodstva računa po t.i. romanskem sistemu. V praksi to izgleda takole:
>
> - brat - sestra: 2. koleno
> - stric/teta - nečak/inja: 3. koleno
> - stric/teta - pranečak/inja: 4. koleno
> - bratranci: 4. koleno
> - mali bratranci: 6. koleno
>
> Najlažje je to ponazoriti s primerom: npr. vnuk in ded sta sorodnika v ravni vrsti oz. liniji, ker je eden drugemu prednik oziroma potomec, torej ni skupnega prednika. Če začenjamo štetje kolen sorodstva z dedom, le-tega ne štejemo, štejemo vmesnega sorodnika (vnukovega očeta) in samega vnuka. To sta dva rojstva oz. osebi, torej sta si v sorodu v drugem kolenu. Nečak in stric pa sta

[5] `https://www.poravnava.si/kako-se-doloca-stopnjo-oziroma-kolena-sorodstva/`

> sorodnika v stranski vrsti, ker eden ni prednik ali potomec drugemu, torej imata skupnega prednika (npr. stričevega očeta oziroma nečakovega deda). Tu koleno določimo tako, da začnemo, na primer, pri stricu, katerega, kot že povedano ne vštevamo, in potem sledimu osebam do skupnega prednika ter od tega naprej po nečakovih prednikih v ravni vrsti. Torej: ded, nečakov oče/mati, nečak. To so tri osebe oziroma rojstva, če začenjamo štetje pri stricu. Nečak in stric sta torej sorodnika v 3. kolenu (po stranski vrsti).

Podatke bi vnašali Butalci sami in sicer tako, da bi vsak Butalec posredoval, kdo sta njegova oče in mati.
VPRAŠANJA:

1. Recimo, da so Butalci vnesli naslednje podatke[6]:

> $(1,5,3)$, $(7,8,4)$, $(9,2,7)$, $(1,5,3)$, $(4,6,9)$, $(5,10,9)$, $(10,6,1)$, $(3,2,9)$ in $(6,2,3)$,

kjer trojka (a,b,c) predstavlja, da sta b in c starša a. (i) V katerem kolenu sta si sorodnika 8 in 3? Utemeljite odgovor. (ii) V katerem kolenu sta si sorodnika 7 in 1? Utemeljite odgovor.

2. (i) Predlagajte strukturo, s katero bi modelirali podatke tako, da bi učinkovito odgovorili na poizvedbo `Koleno(a, b)`, ki vrne, v katerem kolenu sta si `a` in `b` v sorodu. (ii) Opišite algoritem, ki implementira omenjeno poizvedbo na predlagani strukturi. (iii) Kakšna je časovna zahtevnost vašega algoritma? Utemeljite odgovor.

3. (i) Na podatkih iz vprašanja 1 ugotovite, katera oseba ima največ otrok. (ii) V vprašanju 1 oznaki b in c predstavljata starša a, vendar ni določeno, kdo je mati in kdo oče. Tako naslednje trojice (a,b,c), (d,c,e) in (f,e,b) ne morejo nastopati hkrati. Zakaj? Utemeljite odgovor.

Naloga 159. Peter Zmeda zopet v problemih. Tokrat so ga poklicali na pomoč iz *Butalskega potniškega prometa – BPP*. Dali so mu naslednjo tabelo:

[6] Podatki so generirani in morda niso najbolj smiselni.

zač. čas	konč. čas	id	zač. čas	konč. čas	id
06:05	06:35	41	06:15	06:45	42
06:23	06:53	43	06:30	07:00	44
06:35	07:05	45	06:40	07:10	46
06:45	07:15	47	06:50	07:20	48
06:54	07:24	49	06:58	07:28	141
07:02	07:32	142	07:06	07:36	143
07:10	07:40	144	07:14	07:44	145
07:18	07:48	146	07:23	07:53	147
07:28	07:58	148			

V tabeli so navedeni (po vrsti) čas odhoda avtobusa z začetne postje, čas prihoda na končno postjo in id vožnje. Očitno je en avtobus lahko hkrati na samo eni vožnji.
Vprašanja:

1. Kolikšno je najmanjše število avtobusov, ki jih potrebuje *BPP*, da bo lahko opravil vse vožnje po zgornjem razporedu?

2. Pri zgornjem razporedu so vožnje urejene po času odhoda z začetne postaje. Zapišite (opišite) algoritem, ki v tako urejenem razporedu z n vožnjami poišče najmanjše število avtobusov, potrebnih za izvedbo vseh voženj. Utemeljite odgovor.

3. Sedaj pa predpostavimo, da razpored ni urejen. Peter se je seveda takoj domislil, da bi lahko potem najprej uredil razpored in uporabil algoritem iz prejšnje točke. Kakšna je časovna zahtevnost takšne rešitve?

4. [Dodatno] Predlagajte podatkovno strukturo, ki omogoča enako časovno zahtevnost, a ne potrebuje urejanja. Utemejite odgovor.

Naloga 160. V družabnih ali socialnih omrežjih je ena od lastnosti, da nekoga *poznaš*. Pri tem je lahko poznavanje vzajemno, se pravi, da velja, če Ana pozna Ceneta, potem tudi Cene pozna Ano, ali pa ne. Omejili se bomo na primer, kjer poznavanje ni vzajemno. Recimo, da v našem družabnem omrežju velja:

- Ana pozna Boruta in Erika,
- Borut pozna Erika in Ceneta,
- Cene pozna Danico,

- Danica pozna Ceneta in Ano ter
- Erik pozna Boruta, Ceneta in Danico.

Poleg tega imamo v omrežju operacijo `Predstavi(X, Y, Z)`, kjer `X` zaprosi `Y`, da mu predstavi `Z`, če seveda `Y` pozna `Z` in `X` pozna `Y`. Poleg tega imamo v omrežju še operacijo `Spozna(X, Y)`, ki vrne odgovor, če je kakorkoli možno, da bi `X` spoznal `Y`.
Vprašanja:

1. Ali sta v našem primeru dovoljeni operaciji

`Predstavi(Ana, Cene, Danica)` in
`Predstavi(Ana, Erik, Danica)`.

Utemeljite odgovor.

2. Kaj v našem omrežju vrneta klica

`Spozna(Ana, Danica)` in
`Spozna(Danica, Ana)`.

Utemeljite odgovor.

3. Omrežje, kjer lahko vsakdo spozna kogarkoli, imenujemo *mehka klika*. Ali je naše omrežje mehka klika? Utemeljite odgovor.

4. Zapišite algoritem, ki za poljubno družabno omrežje ugotovi, ali je mehka klika.

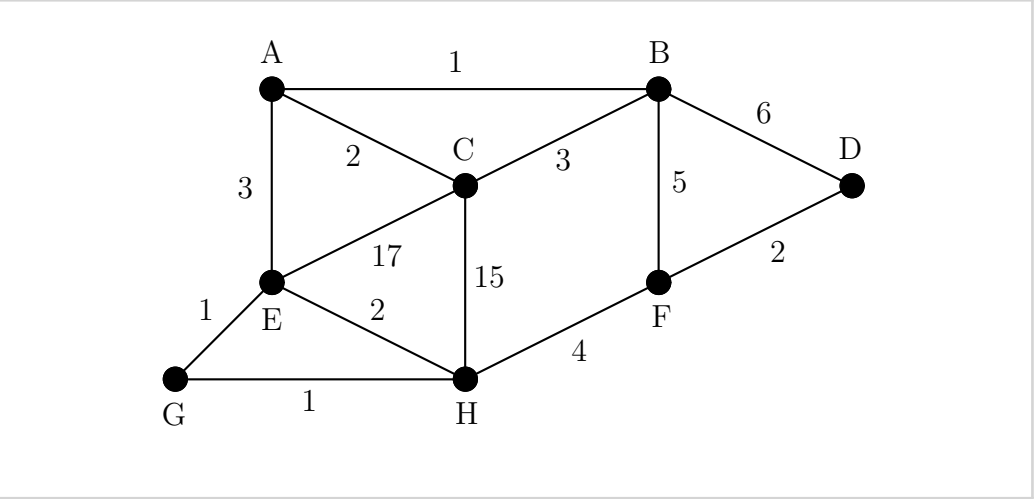
Naloga 161. Imamo graf $G(V, E)$ ($|V| = n$ in $|E| = m$), ki je predstavljen s seznami sosedov S, kjer je:

- $S[v].q$ število sosedov vozlišča v; in
- so $S[v].s[i]$ $(0 \leq i < S[v].q)$ uteži povezav $(v, S[v].s[i])$.

Vprašanja:

1. Za graf s sl. 3.8 zapišite celotno strukturo S.

2. Zapišite rekurzivni algoritem `Prestej(G, s, x)` za preiskovanje grafa `G` v globino, ki prične pregled v vozlišču `s` ter prešteje, koliko povezav v grafu ima utež večjo od `x`.



Slika 3.8: Primer grafa.

Namig: Pomagajte si z dodatnim poljem velikost n.

3. Na zadnjem kolokviju smo imeli graf s sl. 3.8, v katerem je bilo potrebno poiskati Eulerjev sprehod – Eulerjev sprehod je pot v grafu, pri kateri moramo prehoditi vsako povezavo natančno enkrat, lahko pa seveda obiščemo vozlišča večkrat. V grafu na sl. 3.8 obstaja Eulerjev sprehod. Ne obstaja pa Eulerjev obhod[7]. Kako je potrebno spremeniti graf (predlagajte konkretno spremembo), da bo obstajal tudi Eulerjev obhod?

Naloga 162. Imamo usmerjen graf na sl. 3.9.
Vprašanja:

1. Za graf s sl. 3.9 zapišite matriko sosednosti.

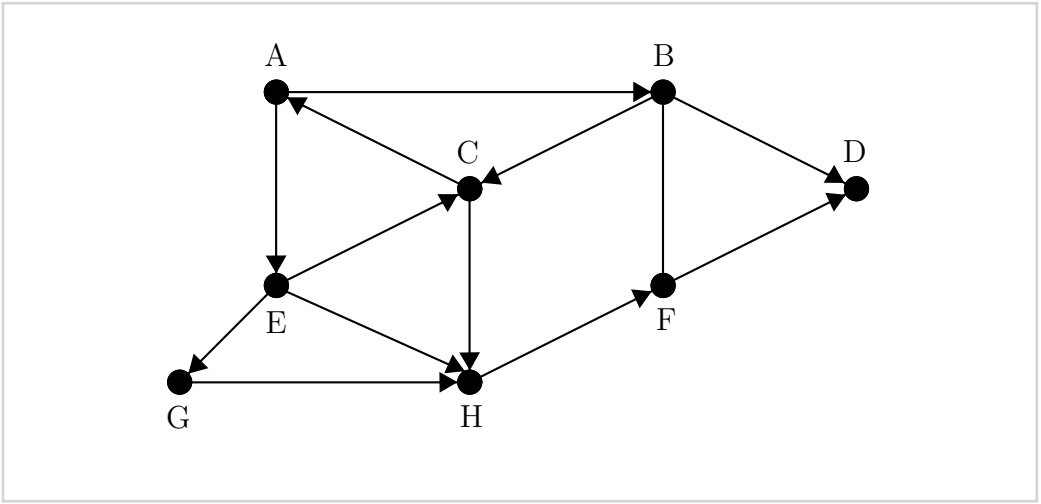
2. Na grafu s sl. 3.9 naredite obhod v globino pričenši v A in zapisujte posamezne korake oziroma vozlišča kot jih obiskujete. Sosede posameznega vozlišča obiskujete v leksikografskem vrstnem redu.

3. Peter Zmeda je zapisal naslednjo trditev: „Vsak neusmerjen graf ima vpeto drevo.“

Ali je trditev pravilna? Utemeljite svoj odgovor.

---

[7]Obhod je sprehod, kjer je začetno vozlišče hkrati tudi končno vozlišče.



Slika 3.9: Primer usmerjenega grafa.

Naloga 163. Dandanes lahko dokaj poceni kupujemo rabljene, a še delujoče računalnike. Peter je kupil 10 takšnih računalnikov in sicer po tri s takti 700MHz, 500MHz in 300MHz ter še enega malce sodobnejšega s taktom 1GHz. Problem, ki ga mora Peter rešiti, je obdelava velikega družabnega omrežja (*social network*), v katerem je $n > 10^9$ vozlišč.
Vprašanja:

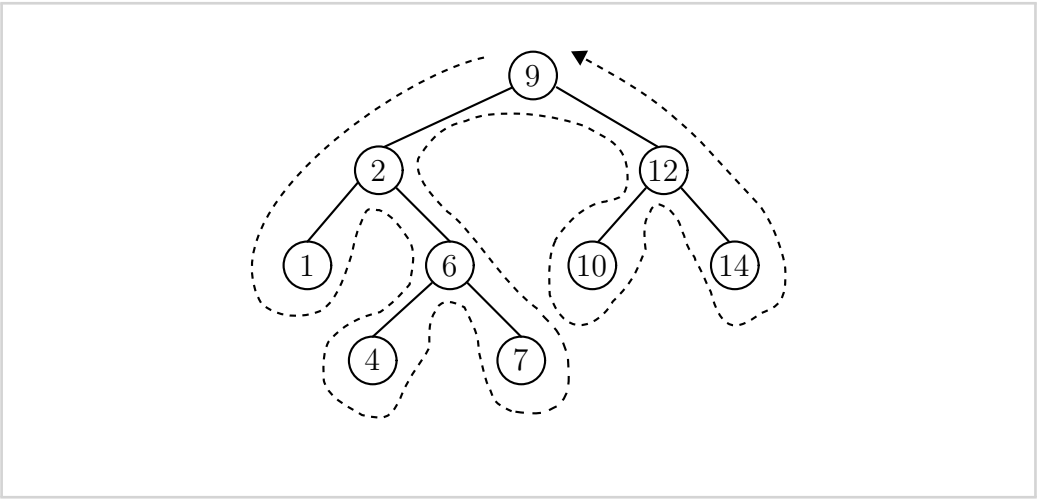
1. V družabnih omrežjih lahko vozlišče u spozna vozlišče v (vozlišči se povežeta), če obstaja vozlišče w, ki pozna tako u kot v. Prvi program, ki ga mora Peter napisati, je program, ki odgovori na vprašanje, ali lahko vozlišče v_1 spozna vozlišče v_2. (i) Zapišite algoritem, ki bo odgovoril na to vprašanje in (ii) utemeljite njegovo pravilnost. (iii) Ocenite njegovo časovno zahtevnost.

2. Kako bi lahko Peter pospešil delovanje svojega algoritma, če bi uporabil vse računalnike, ki jih je kupil?

Namig: Opišite, kako bi razporedili podatke med računalniki; za več točk pri razporejanju dela med računalniki upoštevajte različne sposobnosti računalnikov; več točk boste dobili, če boste natančneje opisali povzporejanje.

3. Stopnjo vozlišča v definiriamo kot število vozlišč u, ki jih vozlišče v pozna. Zapišite algoritem, ki izračuna povprečno stopnjo vozlišča v omrežju. Vaš algoritem naj predpostavi, da je omrežje podano z matriko sosednosti.

Naloga 164. Pri predmetu smo spoznali vrsto obhodov dreves in na sl. 3.10 imamo še enega, ki se imenuje Eulerjev obhod in tvori za drevo s slike nasle-



Slika 3.10: Eulerjev obhod drevesa.

dnji izpis: 9, 2, 1, 2, 6, 4, 6, 7, 6, 2, 9, 12, 10, 12, 14, 12, 9.
Vprašanja:

1. Recimo, da imamo v drevesu n elementov. Kako dolg je izhodni niz, ki ga dobimo po obhodu? Utemeljite odgovor.

2. Zapišite psevdokodo algoritma za izpis Eulerjevega obhoda drevesa t.

3. Recimo, da imamo izpis Eulerjevega obhoda. Ali lahko iz njega tvorimo nedvoumno nazaj izvorno drevo? Utemeljite odgovor.

Naloga 165. *Grafi.* Peter Zmeda je z neusmerjenim grafom modeliral podjetje Butale d.o.o. Vsak zaposleni predstavlja vozlišče, medtem ko je med vozliščema povezava samo tedaj, ko zaposlena poznata telefonsko številko drug drugega[8]. Se pa v podjetju držijo zelo strogega pravila, da lahko zaposleni vsako uro govori po telefonu največ z eno osebo. Recimo, da je graf $G(V, E)$ podan s seznami sosednosti ter je $|V| = n$ in $|E| = m$. Definirajmo še cikel. Za graf G pravimo, da ima cikel dolžine d, če v njem obstaja sprehod iz točke u preko vmesnih točk $u_1 u_2 \cdots u_{d-1}$ nazaj do vozlišča u in $u_i \neq u_j$ za vse točke na sprehodu razen za prvo in zadnjo.
Vprašanja:

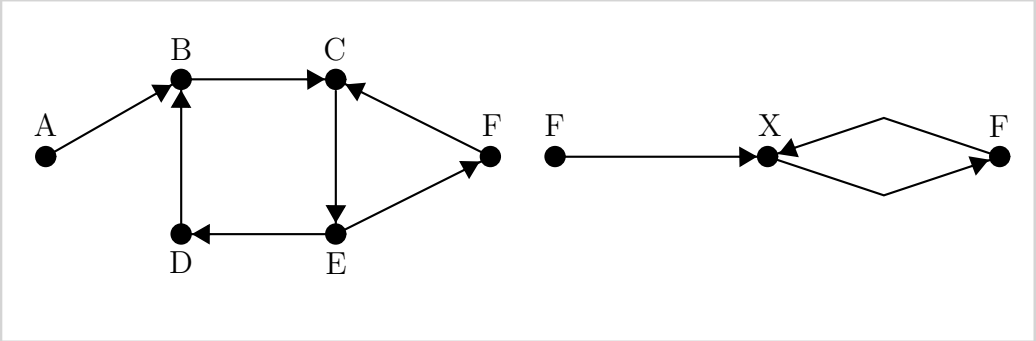
[8]Predpostavimo, da, če oseba A pozna telefonsko številko osebe B, potem tudi oseba B pozna številko osebe A.

1. Peter je izvedel, da je čvekavi Miha pred k urami izvedel neko neprijetno skrivnost, in sedaj bi Peter rad dobil seznam vseh zaposlenih, ki bi že lahko poznali skrivnost. Napišite/opišite algoritem, ki bo izpisal zahtevani seznam. Seveda se na seznamu nobeno ime ne sme pojaviti več kot enkrat. Kakšna je časovna zahtevnost vašega algoritma kot funkcija n, m in k?

2. Zapišite algoritem, ki v grafu G poišče cikel dolžine d, če obstaja. Utemeljite pravilnost vašega algoritma.

3. Kakšna je časovna zahtevnost vašega algoritma za iskanje cikla kot funkcija n, m in d?

Naloga 166. Na sl. 3.11 imamo najprej usmerjen graf, ki vsebuje cikel med



Slika 3.11: Sesedanje ciklov.

točkami $BCEDB$, ki se v drugem grafu sesede v eno samo točko X. V nalogi bomo imeli opravka z usmerjenim grafom $G(V, E)$, kjer je $|V| = n$ in $|E| = m$. Graf G naj bo podan z matriko sosednosti, oznake vozlišč pa so števila med 1 in n.

Vprašanja:

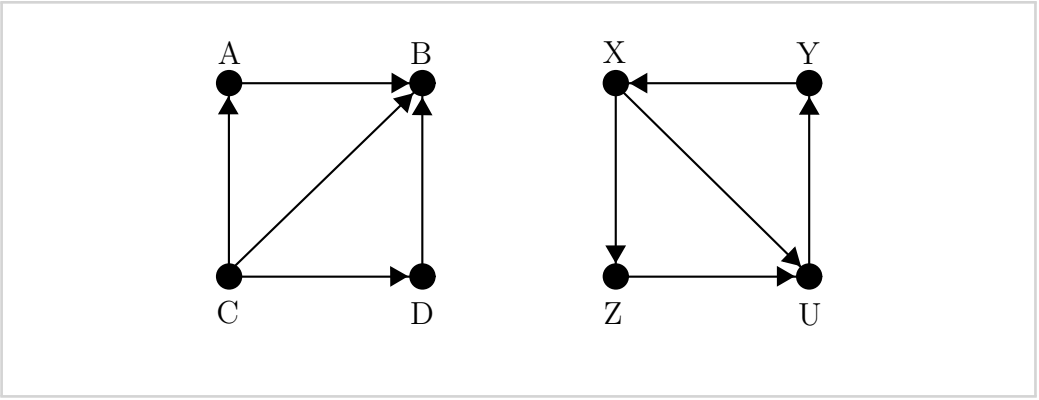
1. Naj bo $A \in V$. (i) Zapišite algoritem `Cikel(`G`, `A`)`, ki ugotovi, ali obstaja cikel v G, ki vsebuje točko A, in vrne seznam c vozlišč, ki tvorijo cikel. (ii) Kakšna je časovna zahtevnost vašega algoritma? Utemeljite odgovor. (iii) Ali obstaja algoritem s časovno zahtevnostjo $o(m)$? Utemeljite odgovor.

2. V grafu G imamo med točkama A in B po največ eno povezavo iz A v B in iz B v A (nimamo multigrafa). (i) Zapišite algoritem `Sesedi(`G`, `c`)`, ki v grafu G sesede cikel c, kjer je c rezultat funkcije `Cikel(`G`, `A`)`, v točko. (ii) Kakšna je časovna zahtevnost vašega algoritma? Utemeljite odgovor. (iii)

Funkciji `Cikel()` in `Sesedi()` sta že implementirani. Sestavite algoritem, ki sesede vse točke v grafu G.

3. Kot vemo, je iskanje Hamiltonovega cikla v grafu NP-poln problem. Peter Zmeda je rešil prejšnje vprašanje v tej nalogi in prišel na idejo, da preprosto sesede vse cikle v grafu in, če bi dobil samo eno točko, bi s tem našel Hamiltonov cikel. (i) Zapišite (narišite) primer grafa, kjer je njegov pristop pravilen. (ii) Zakaj v splošnem njegov pristop ni pravilen? Utemeljite odgovor. Najbolje s primerom grafa, kjer njegov pristop vrne napačen rezultat.

Naloga 167. *Grafi.* Na predavanjih smo omenjali pojem *topološke ureje nosti usmerjenega grafa.* Slednje v resnici pomeni, da lahko usmerjen graf $G(V, E)$ topološko uredimo, če lahko vsa vozlišča $v \in V$ razporedimo na premico tako, da so nato vse povezave $(u, v) \in E$ usmerjene v desno. Na sl. 3.12 imamo usmerjena grafa $G_L(V_L, E_L)$ (levi graf) in $G_D(V_D, E_D)$ (desni



Slika 3.12: Usmerjena grafa.

graf).

Vprašanja:

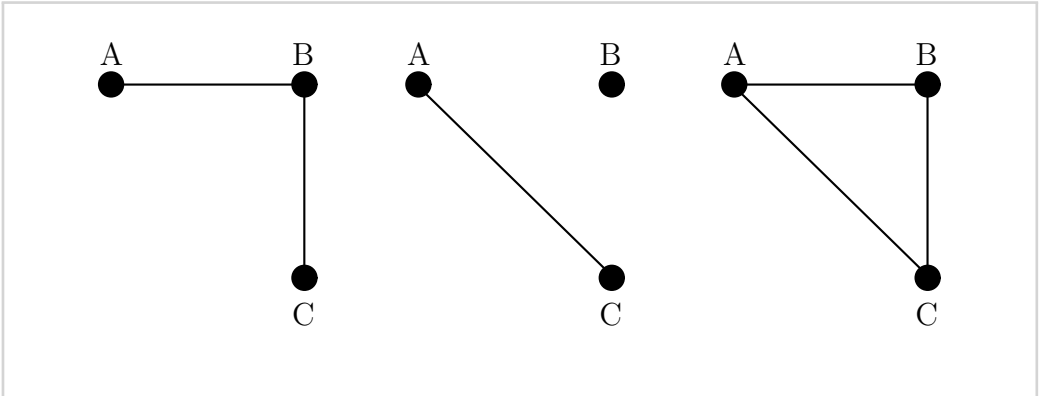
1. (i) Predstavite $G_L(V_L, E_L)$ in $G_D(V_D, E_D)$ najprej z matriko sosednosti in nato še s seznamom sosedov. (ii) Za $G_L(V_L, E_L)$ in za $G_D(V_D, E_D)$ ugotovite, če se ga da topološko urediti ali ne. Če se ga da, izpišite njegovo topološko urejenost in, če se ga ne da, utemeljite, zakaj se ga ne da.

2. Recimo, da imamo nek usmerjen graf $G(V, E)$. (i) Zapišite algoritem, ki ugotovi, ali se $G(V, E)$ da topološko urediti ali ne. (ii) Kakšna je njegova časovna in prostorska zahtevnost? Utemeljite odgovor.

NAMIG: Seveda, če je vaš algoritem rekurziven, morate pri prostorski zahtevnosti upoštevati, da globina rekurzije vpliva na zahtevnost.

3. Recimo, da se usmerjenega grafa $G(V, E)$ ne da topološko urediti. Slednje se lahko spremeni, če obrnemo smer kakšne povezave ali če povezavo izpustimo – katerikoli od opisanih operacij rečemo *popravek*. (i) Zapišite algoritem, ki s kar se da majhnim številom popravkov spremeni graf $G(V, E)$ tako, da se ga da topološko urediti. (ii) Utemeljite pravilnost vašega algoritma. (iii) Kakšna sta časovna in prostorska zahtevnost vašega algoritma? Utemeljite odgovor.

Naloga 168. *Algoritmi na grafih.* Na grafih bomo definirali dve operaciji. Prva operacija je `Krajšaj(G)`, ki iz grafa $G(V, E)$ naredi nov graf $G'(V, E')$, kjer velja, da je povezava $(u, v) \in E'$, če in samo če velja, da obstaja $x \in V$ tako, da sta povezavi $(u, x), (x, v) \in E$. Druga operacija pa je `Seštej(G1, G2)`, ki sešteje grafa $G_1(V, E_1)$ in $G_2(V, E_2)$ v graf $G_3(V, E_3)$, kjer velja, da $(u, v) \in E_3$, če $(u, v) \in E_1$ ali $(u, v) \in E_2$. Primer obeh operacij je na sl. 3.13. Pri tej nalogi naj bodo vsi grafi predstavljeni z matriko sosednosti.

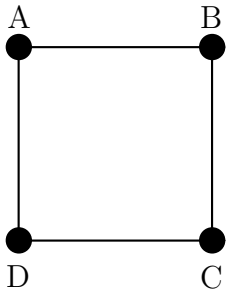


Slika 3.13: Operacija `Krajšaj` na levem grafu vrne srednji graf, medtem ko `Seštej` levega in srednjega grafa vrne desni graf.

VPRAŠANJA:

1. (i) Predstavite graf $G_1(V, E)$ na sl. 3.14 z matriko sosednosti. (ii) Na grafu $G_1(V, E)$ izvedite operacijo `Krajšaj` ter narišite graf, ki ga dobite. Napišite še zanj matriko sosednosti.

2. (i) Napišite funkcijo `Krajšaj(G)`, ki vrne okrajšani graf kot je definirano zgoraj. (ii) Kakšna je časovna zahtevnost vašega algoritma? Utemeljite



Slika 3.14: Graf $G_1(V, E)$ z $n = 4$ vozlišči.

odgovor. (iii) Napišite funkcijo `Seštej(G1, G2)`, ki vrne okrajšani graf. (iv) Kakšna je časovna zahtevnost vašega algoritma? Utemeljite odgovor.

3. Recimo, da imamo uspešno implementirani obe operaciji in poženemo naslednji algoritem (n je število vozlišč v grafu G):

```
KajNaredi(G):
  for i= 1...n do:
    G= Seštej(G, Krajšaj(G))
  return G
```

Kako izgleda graf, ki ga vrne funkcija `KajNaredi`? Utemeljite odgovor.

Naloga 169. Peter Zmeda je našel listek z naslednjo vsebino:

```
1:  2, 4, 6
2:  3
3:  7
4:  3, 5
5:  2, 3, 7
6:  7
```

Kmalu je ugotovil, da gre za zapis grafa s seznamom sosedov.
Vprašanja:

1. (i) Narišite graf, ki je podan z zgornjim zapisom, (ii) zapišite ga kot matriko sosednosti ter, (iii) ali imamo opravka z usmerjenim ali neusmerjenim grafom – utemeljite odgovor.

2. Graf je krepko povezan, če lahko iz vsakega vozlišča pridemo v poljubno drugo vozlišče. Zapišite algoritem, ki bo ugotovil, ali je graf krepko povezan.

3. Utemeljite pravilnost in ocenite časovno zahtevnost vašega algoritma iz prejšnjega vprašanja.

Namig: Za učinkovitejšo rešitev boste dobili več točk.

Naloga 170. *Grafi.* Imamo graf $G(V, E)$.
Vprašanja:

1. Recimo, da je graf G usmerjen. Poleg tega recimo, da je neko vozlišče $v \in V(G)$ dosegljivo iz vseh vozlišč V. Ali je graf G krepko povezan? Utemeljite odgovor.

2. Graf naj bo še naprej usmerjen. Definirajmo kvadrat grafa $H = G^2$, kjer velja, da je $(u, v) \in E(H)$, če obstaja v G takšno vozlišče $w \in V(G)$, da sta $(u, w), (w, v) \in E(G)$. (i) Narišite poljuben (majhen) usmerjen graf G ter ga kvadrirajte, se pravi narišite potem G^2. (ii) Recimo, da je graf podan s seznamom sosedov. Napišite algoritem, ki bo za poljuben graf $G(V, E)$ izračunal njegov kvadrat. (iii) Kakšna je časovna zahtevnost vašega algoritma?

3. Sedaj naj bo graf G neusmerjen. Odgovorite sedaj na vsa podvprašanja (od (i) do (iii)) iz prejšnjega vprašanja.

Naloga 171. Usmerjen graf $G(V, E)$, kjer je $V = \{\texttt{A}, \texttt{V}, \texttt{G}, \texttt{U}, \texttt{S}, \texttt{T}, \texttt{Z}\}$ množica vozlišč, je podan s seznami sosedov:

```
A:  G, S, Z          S:  Z          U:  S, A
G:  S                T:  G, U, V    V:  Z
```

Še dve definiciji. Podmnožico vozlišč grafa, za katero velja, da med katerimakoli vozliščema podmnožice obstaja pot, imenujemo *krepko povezana komponenta*. V splošnem imamo graf $G_S(V, E)$ in $X \subseteq V$. Potem lahko naredimo iz grafa G_S inducirani podgraf $G_I(X, Y)$, kjer $(u, v) \in Y$, če in samo če: $u, v \in X$ ter $(u, v) \in E$.
Vprašanja:

1. (i) Narišite graf, ki je podan z zgornjim zapisom. (ii) V grafu G poiščite največjo krepko povezano komponento.

2. (i) Naj bo $G_S = G$ in $X = \{\texttt{V}, \texttt{G}, \texttt{S}, \texttt{T}\}$. Zapišite induciran podgraf $G_I(X, Y)$. Podajte ga s seznami sosednosti. (ii) Naj bo G_S predstavljen s seznami sosedov. Zapišite algoritem, ki pri podanih G_S in X naračuna induciran podgraf G_I ter ga predstavi s seznami sosedov.

3. Utemeljite pravilnost in ocenite časovno zahtevnost vašega algoritma iz prejšnjega vprašanja.

Naloga 172. *Grafi. Bitcoin* je popularna internetna valuta, ki je skoraj povsem anonimna, saj se dejansko ne pozna ne plačevalca in ne plačnika. Njeno delovanje sloni na grafu transakcij $G_T(V, E)$, kjer je V množica vseh transakcij in E množica povezav med njimi. Povezava (u, v) je usmerjena in pomeni, da so se sredstva, ki so nastala v transakciji u, porabila v transakciji v. V transakciji se lahko porabijo sredstva iz večih transakcij in prav tako se lahko sredstva iz ene transakcije porabijo v večih transakcijah. Seveda, sredstva, ki so porabljena, se ne morejo porabiti še enkrat. Sredstva, ki so bila v neki transakciji in še niso bila porabljena, pravimo, da so shranjena v *denarnici*. Za potrebe te naloge tudi denarnice predstavljajo vozlišča v V (glej sl. 3.15).

Naj ima vozlišče $v \in V$ dve lastnosti (atributa) in sicer, čas t, ko je bila transakcija opravljena, in d, ki pove, ali gre za resnično transakcijo ali za denarnico.

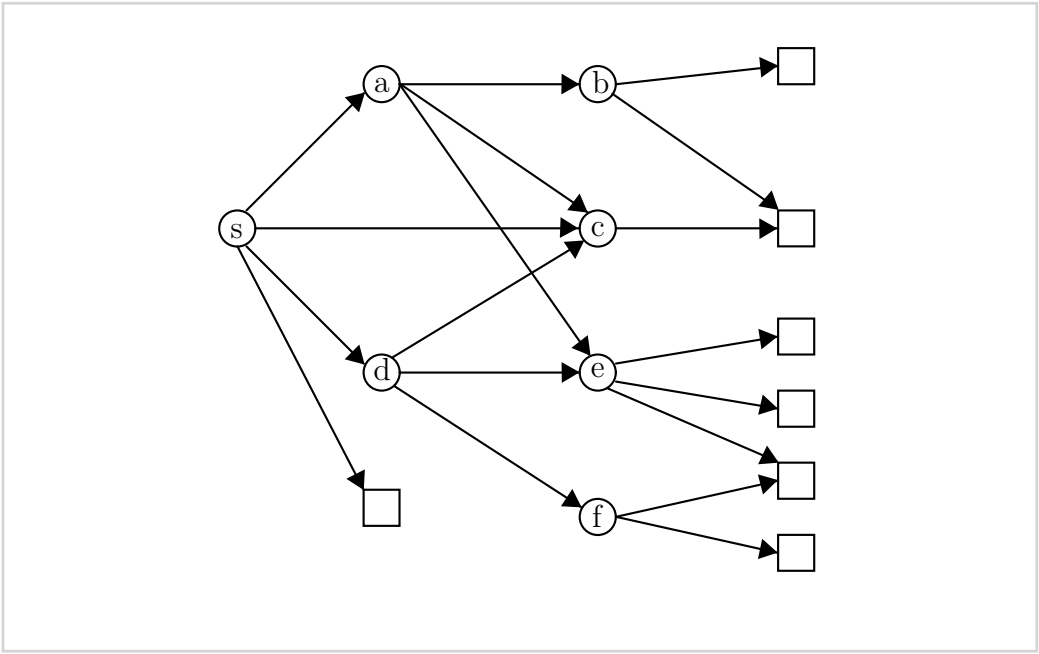
Vprašanja:

1. Napišite funkcijo `Najmlajsa(G, s)`, ki poišče tisto denarnico v grafu G, ki je dosegljiva iz transakcije s in je bila ustvarjena zadnja. Kakšna je časovna zahtevnost vašega algoritma? Utemeljite odgovor.

Namig: Denarnica je ustvarjena, ko je bila izvedena transakcija, ki jo je ustvarila.

2. Napišite še funkcijo `Najstarejsa(G, s)`, ki poišče denarnico v grafu G, ki je dosegljiva iz transakcije s in je najstarejša, ki še obstaja. Primerjajte časovno zahtevnost te funkcije s časovno zahtevnostjo funkcije iz prejšnjega podvprašanja.

3. Digitalni forenziki postanejo pozorni, če se transakcije dogajajo neobičajno. Eden od neobičajnih načinov je, da se dogajajo hitro ena za drugo.



Slika 3.15: Primer *BitCoin* grafa, kjer so transakcije krogi in denarnice kvadrati.

Napišite program `Sumljive(G, s)`, ki bo za graf G pričenši iz vozlišča s (i.) naračunal stopnjo sumljivosti za vsako dosegljivo transakcijo; in (ii.) vrnil seznam ali polje transakcij v padajočem vrstnem redu sumljivosti.

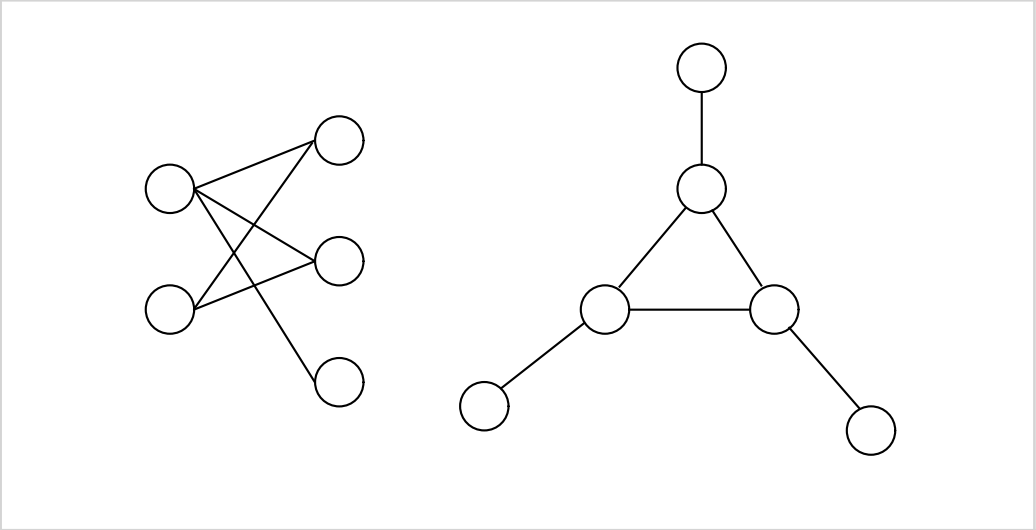
Sumljivost transakcije x glede na s definiramo kot $\frac{\text{razdalja najdalše poti od } s \text{ do } x}{x.t - s.t}$. Na sl. 3.15 je stopnja sumljivosti vozlišča e glede na s enaka $\frac{2}{e.t-s.t}$, ker je najdaljša razdalja od s do e enaka 2.

Naloga 173. Posebna oblika grafov so dvodelni grafi. Graf $G(V, E)$ je dvodelen, če lahko vozlišča V razdelimo na dve podmnožici V_1 in V_2, kjer je $V_1 \cap V_2 = \emptyset$ in $V_1 \cup V_2 = V$, za vsako povezavo $(u, v) \in E$ pa velja, da, če je $u \in V_1$, potem je $v \in V_2$, oziroma obratno. Na levem delu sl. 3.16 je primer dvodelnega grafa, kjer sta levi vozlišči v eni in desna v drugi množici.
Vprašanja:

1. Ali je desni graf na sl. 3.16 tudi dvodelen? Utemeljite odgovor. Če menite, da je, zapišite množici V_1 in V_2. Če menite, da ni, zapišite, čemu ni in ga s čim manj spremembami naredite dvodelnega.[9]

2. Imamo poljuben neusmerjen graf $G(V, E)$. (i) Napišite algoritem, ki ugo-

[9]Sprememba je dodajanje ali brisanje povezave.



Slika 3.16: Primer dvodelnega grafa (levo) in neznani graf (desno).

tovi, ali je graf dvodelen in, če je, naj izpiše V_1 in V_2. Predpostavite lahko, da so vozlišče oštevilčena od 1 do n. Kako so predstavljene povezave, si izberite sami. (ii) Kakšna je časovna in prostorska zahtevnost vašega algoritma? Utemeljite odgovor.

3. Ponovno imamo poljuben neusmerjen graf $G(V, E)$, za katerega vemo, da ni dvodelen. (i) Napišite algoritem, ki graf popravi tako, da postane dvodelen na netrivialen način.[10] Ponovno lahko predpostavite, da so vozlišča oštevilčena od 1 do n. Kako so predstavljene povezave, si ponovno izberite sami. (ii) Kakšna je časovna in prostorska zahtevnost vašega algoritma? Utemeljite odgovor.

Naloga 174. *Algoritmi na grafih in dinamično programiranje.*
Vprašanja:

1. Na predavanjih smo opisali različna vpeta drevesa. Narišite graf, ki bo imel različni minimalno vpeto drevo in vpeto drevo najkrajših poti iz enega vozlišča do ostalih vozlišč.

2. Narisani graf zapišite z incidenčno matriko in z matriko sosednosti.

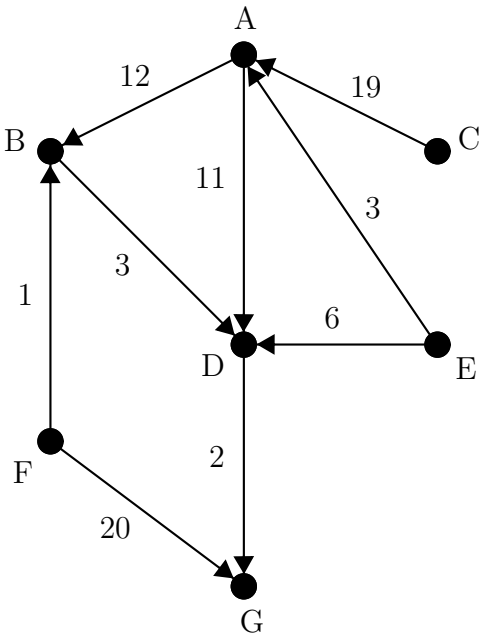
[10] Trivialen način pomeni, da, recimo, pobrišemo vse povezave.

3. Najkrajše poti med poljubnim parom vozlišč lahko poiščemo z dinamičnim programom (Floyd-Warshallov algoritem). Definiran je z naslednjo formulo:

$$\delta_{i,j}^{(n)} = \begin{cases} w_{i,j}, & \text{za } n = 0; \\ \min(\delta_{i,j}^{(n-1)}, \delta_{i,n}^{(n-1)} + \delta_{n,j}^{(n-1)}), & \text{sicer}, \end{cases}$$

kjer je $w_{i,j}$ cena povezave (v_i, v_j) in je $\delta_{i,j}^{(m)}$ najcenejša pot $v_i \rightsquigarrow v_j$, kjer so vozlišča na poti v_k (izključno začetek in konec), za katera velja $k \leq m$. (i) Zapišite algoritem po tej rekurzivni definiciji in uporabite tehniko pomnjenja. (ii) Kakšna je časovna in kakšna prostorska zahtevnost vašega algoritma? Utemeljite odgovor.

Naloga 175. V tej nalogi bomo uporabili grafu na sl. 3.17.



Slika 3.17: Primer grafa.

Vprašanja:

1. Na primeru grafa s sl. 3.17 poiščite drevo najkrajših povezav iz vozlišča od A do vseh ostalih vozlišč.

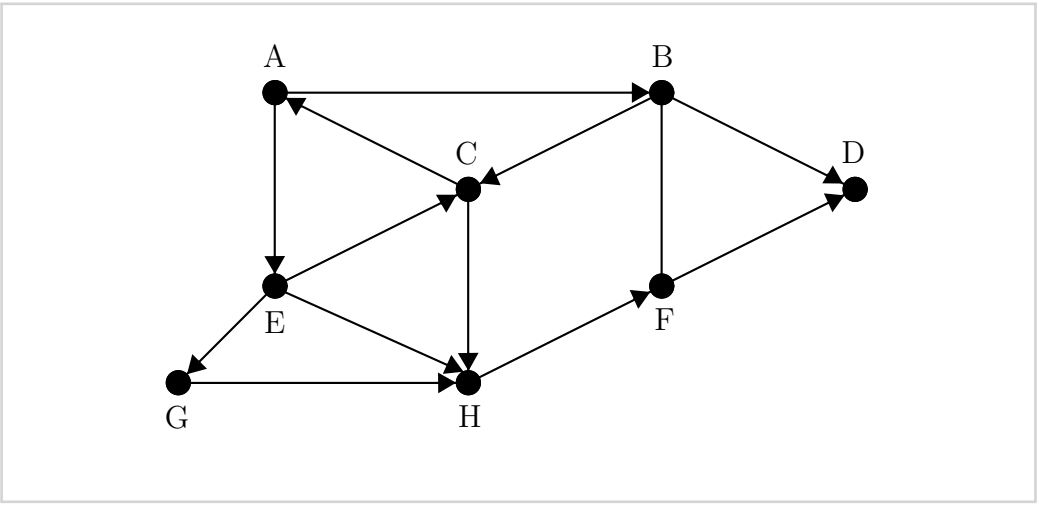
2. Topološko uredite točke iz grafa s sl. 3.17.

3. Premer grafa je definiran kot največja najkrajša razdalja med točkama grafa – z drugimi besedami, če poiščemo vse najkrajše razdalje med vsemi pari točk v grafu, potem je premer grafa tista razdalja, ki je najdaljša.

Graf s sl. 3.17 spremenimo tako, da postane neusmerjen in brez uteži na povezavah. V tako spremenjenem grafu izračunajte premer grafa ter med katerima točkama nastopi.

4. Napišite psevdokodo algoritma za iskanje premera grafa in ocenite njegovo časovno zahtevnost.

Naloga 176. Imamo usmerjen graf na sl. 3.18. Poleg tega definirajmo



Slika 3.18: Primer usmerjenega grafa.

podatkovno strukturo za predstavitev grafa $G(V, E)$ ($|V| = n$ in $|E| = m$) s seznami sosedov S, kjer je:

- $S[v].q$ število sosedov vozlišča v; in
- so $S[v].w[i]$ $(0 \leq i < S[v].q)$ uteži povezav $(v, S[v].s[i])$.

Vprašanja:

1. Za graf s sl. 3.18 zapišite vrednosti $S[C].q$, $S[D].q$ in $S[B].s[0]$. Utemeljite odgovor.

2. Na grafu s sl. 3.18 naredite obhod v širino pričenši v A in zapisujte posamezne korake oziroma vozlišča kot jih obiskujete.

3. Poiščite najkrajše poti iz vozlišča A do vseh ostalih vozlišč z uporabo Dijkstrovega algoritma. Zapisujte vozlišča, kot jih obiskujete pri posameznih korakih. Na koncu zapišite še dolžine najkrajših poti od A do vseh vozlišč.

Kaj opazite, ko primerjate odgovora na to in na prejšnje vprašanje? Primerjavo dobro utemeljite.

Naloga 177. Imamo graf $G(V, E)$ in v njem vozlišče s. Poleg tega imamo algoritem, ki ga poženemo na grafu G, kjer je `Queue` navadna vrsta:

```
1  Poisci(G, s)
2    Queue q;
3    V[*]= not visited;
4      q.Enqueue( (s, 0) );
5    while NOT q.Empty()
6      (u, d)= q.Dequeue();
7      V[u]= visited;
8      for v= all neighbours of u do
9        if V[v] == not visited then
10         q.Enqueue((v, d+1));
11   println(u, d)
```

Poleg tega imamo graf G kot je definiran na sl. 3.18.
Vprašanja:

1. Prikažite izvajanje algoritma `Poisci(G, C)`, pri čemer izpisujte vsebino uporabljenih podatkovnih struktur. Kaj izpiše algoritem na koncu?

2. Naj bo graf definiran kot $G(V, E)$, kjer je $|V| = n$ in $|E| = m$. Kakšna je časovna zahtevnost zgornjega algoritma? Utemeljite odgovor.

3. Kaj v resnici poišče zgornji algoritem – kaj izpiše? Utemeljite odgovor.

Naloga 178. Tokrat so Petra poklicali organizatorji njujorškega maratona, ki imajo posebno željo. Želijo namreč postaviti spletno stran, preko katere bi lahko pregledovali trenutni vrstni red tekačev. Od Petra želijo, da njegova rešitev podpira čim učinkoviteje naslednje funkcije:

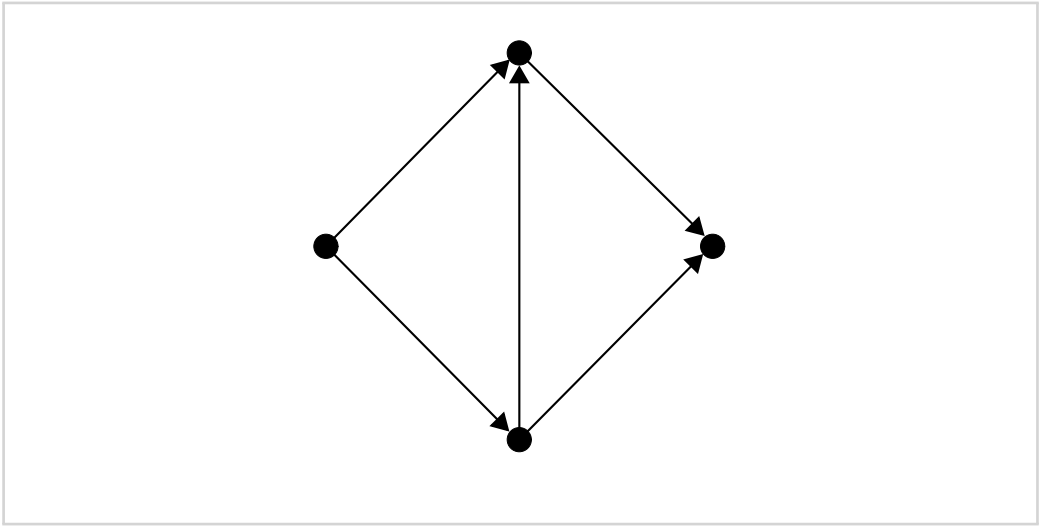
- `RaceStart()` – tekma se je pričela;
- `RaceEnded()` – tekma se je zaključila;
- `CurrentTime(who, time)` – ki osebi `who` popravi trenutni čas[11]; in
- `Place(who)` – ki vrne trenutno mesto tekmovalca `who`.

Spletna stran mora seveda delovati tudi še po zaključku tekme.
Vprašanja:

1. Opišite učinkovito podatkovno strukturo ali strukture, ki implementirajo zgornje operacije.

2. Nepovezano na zgornje vprašanje. Na predavanjih smo obravnavali Ford-Fulkersonov algoritem za izračunan največjega pretoka. Omenili smo tudi, da je časovna zahtevnost algoritma $O(mf)$, kjer je m število povezav in f velikost največjega pretoka, če so kapacitete vseh povezav cela števila. Narišite primer grafa, kjer bo algoritem dosegel to zgornjo mejo in utemeljite svoj odgovor.

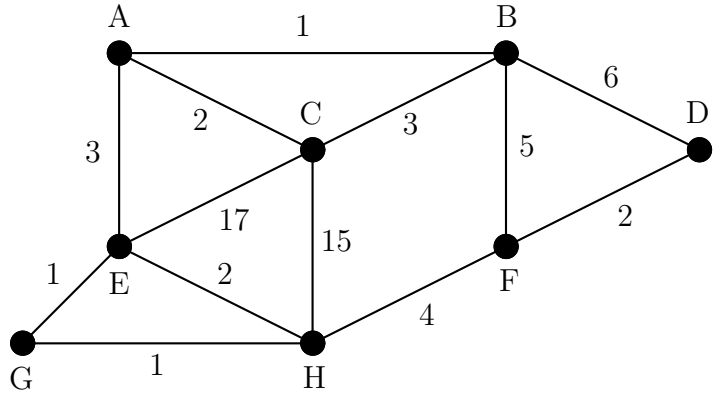
Namig: Topologija grafa je na sl. 3.19, dodajte samo kapacitete.



Slika 3.19: Primer grafa.

3. Imamo graf s sl. 3.20, v katerem poiščite Eulerjev sprehod. Eden od najbolj poznanih sprehodov v krepko povezanem usmerjenem grafu $G = (V, E)$ je Eulerjev sprehod. Pri tem sprehodu moramo prehoditi vsako povezavo

[11] Lahko predpostavite, da sta `who` in `time` celi števili – štartna številka in čas v sekundah.



Slika 3.20: Primer grafa.

natančno enkrat, lahko pa seveda obiščemo vozlišča večkrat. Utemeljite, zakaj ste pričeli in končali sprehod v vozlišču, v katerem ste ga začeli oziroma končali.

Naloga 179. Vračamo se h grafu na sl. 3.20, oziroma v splošnem h grafu $G(V, E)$, kjer $|V| = n$ in $|E| = m$.
Vprašanja:

1. Spoznali smo dva algoritma za izgradnjo najcenejšega vpetega drevesa. (i) Katera sta ta dva algortima? (ii) Kakšna je časovna zahtevnost vsakega od njiju? (iii) V čem se algoritma razlikujeta?

2. V grafu na sl. 3.20 poiščite najcenejše vpeto drevo. Pokažite izračun.

3. Recimo, da imajo v grafu vse uteži na povezavah težo 17. (i) Ali obstaja kakšen preprostejši algoritem za iskanje najcenejšega vpetega drevesa? Kakšna je njegova časovna zahtevnost? Utemeljite.

Naloga 180. Imamo graf na sl. 3.20, oziroma v splošnem graf $G(V, E)$, kjer je $|V| = n$ in $|E| = m$.
Vprašanja:

1. Spoznali smo dva algoritma za izgradnjo najcenejšega vpetega drevesa. (i) Katera sta ta dva algortima? (ii) Kakšna je časovna zahtevnost vsakega od njiju? (iii) V čem se algoritma razlikujeta?

2. V grafu na sl. 3.20 poiščite najcenejše vpeto drevo. Pokažite izračun.

3. Recimo, da imajo v grafu vse uteži na povezavah težo 7. (i) Ali obstaja kakšen preprostejši algoritem za iskanje najcenejšega vpetega drevesa? Kakšna je njegova časovna zahtevnosti? Utemeljite odgovor.

Naloga 181. Razpošiljanju (*multicasting*) v računalniških omrežjih deluje tako, da vsakdo, ki hoče poslati IP paket vsem vozliščem v skupini, le-tega najprej pošlje posebnemu vozlišču, ki se imenuje mesto zmenka (*rendezvous poin* – RP). Slednje vozlišče nato razpošlje paket vsem članom skupine.
Vprašanja:

1. Po kakšnem podgrafu naj razpošlje RP paket do vseh naročnikov, da bo optimiral dolžino poti? Utemeljite odgovor.

2. Imamo neusmerjen graf s sl. 3.21. V grafu poiščite minimalno vpeto drevo s korenom v vozlišču A. Pri iskanju zapisujte posamezne korake.

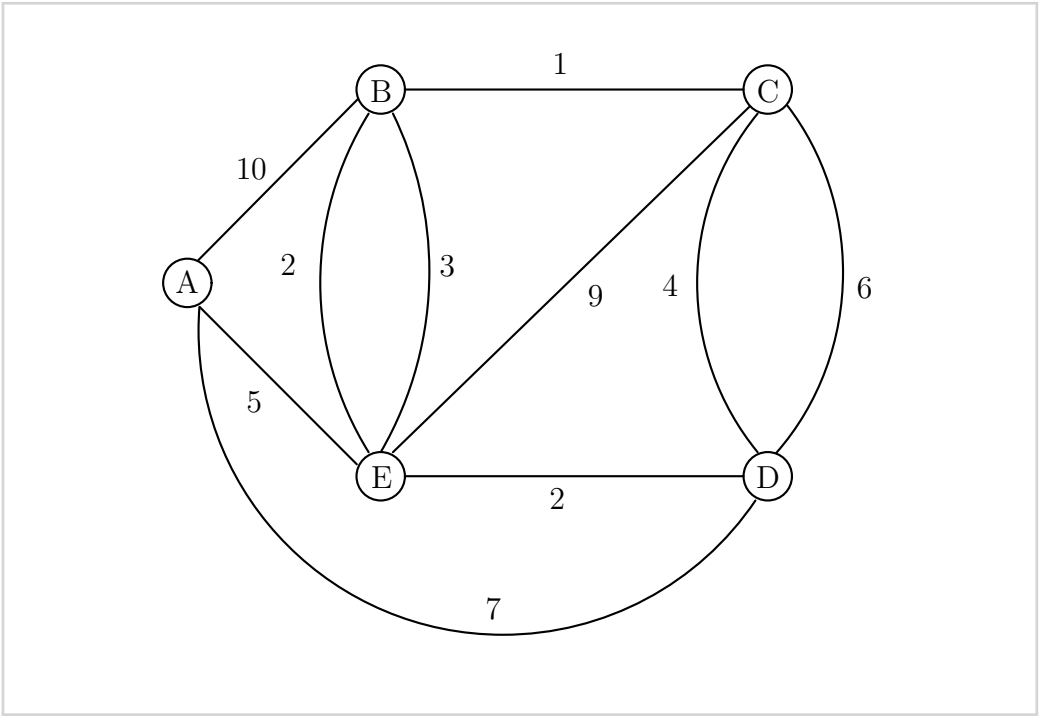
3. V grafu na sl. 3.21 poiščite še minimalno vpeto drevo s korenom v E. Ponovno utemeljite odgovor.

3.5 Črke, besede in besedila

Čeprav se iskanje vzorcev na prvi mah zdi kot problem, ki je vezan na besedila, ga srečamo še marsikje drugod. Primeri uporabe so v bioinformatiki, pri obdelavi časovnih vrst in, ne nazadnje, pri iskanju na svetovnem spletu. Očitno je učinkovitost iskalnih algoritmov ključnega pomena v vseh omenjenih primerih.

Cilj

Da znamo začrtati in implementirati učinkovite algoritme za iskanje vzorcev v besedilu.



Slika 3.21: Primer grafa.

Naloga 182. *Iskanje vzorca.*
Vprašanja:

1. Pri opisu algoritmov za iskanje vzorca v besedilu smo uporabljali koncept končnega avtomata. (i) Za vzorec TCGAC sestavite iskalni končni avtomat. (ii) Ali je vaš avtomat deterministično ali nedeterministično? Utemeljite odgovor.

2. Vzorci v bioloških zaporedjih niso vedno enolično določeni. Tako je lahko na posameznem mestu v vzorcu možnih več različnih znakov. Primer: v vzorcu

0	1	2	3	4	5	6
T	C/G	G	A/G	G	A/C/G/T	C

sta na mestu 1 možna znaka bodisi C ali G in podobno naprej. (i) Peter Zmeda je dobil idejo za definicijo nedeterminističnega končnega avtomata za iskanje tovrstnih vzorcev. Narišite ustrezni končni avtomat, pri čemer jasno označite njegova stanja in prehode. (ii) Kakšna je časovna zahtevnost iskanja vzorca v besedilu dolžine n z vašim nedeterminističnim končnim avtomatom? Utemeljite odgovor.

3. V bioinformatiki imamo pogosto opravka s problemom iskanja podobnosti med dvema vzorcema. Eden od načinov je z uporabo dinamičnega programiranja, ki smo ga že spoznali (razdalja urejanja in njemu sorodna algoritma za lokalno ter globalno poravnavo). Težava tega algoritma je, da ima časovno zahtevnost $O(n^2)$, kar je v praksi pogosto neuporabno, saj sta vzorec in/ali besedilo lahko dolga več milijard znakov. Zato potrebujemo hitrejši algoritem. Ena od možnosti je izračun točkovne matrike, kot smo jo spoznali v eni od domačih nalog. Opišite učinkovito podatkovno strukturo in algoritem, ki pri danem vzorcu p in besedilu t poišče najpopularnejšo diagonalo. Kakšna je časovna in prostorska zahtevnost vaše rešitve? Utemeljite odgovor.

Naloga 183. Pri analizi DNK imamo abecedo $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. Peter Zmeda je našel listek, na katerem je bil zapisan naslednji vzorec $p = \mathtt{GTGGCAG}$. Peter bi rad vzorec iskal v različnih besedilih.
Vprašanja:

1. (i) Najprej pomagajte Petru tako, da narišete nedeterministični končni avtomat, ki bo razpoznal besede, ki vsebujejo vzorec p. (ii) Utemeljite, zakaj je vaš avtomat nedeterminističen.

2. (i) Za vzorec p sestavite funkcijo δ deterministčnega končnega avtomata za iskanje, kot smo ga opisali na predavanjih. (ii) Recimo, da je dolžina vzorca m, dolžina besedila n in velikost abecede σ. Kakšna je časovna zahtevnost gradnje funcije δ? (iii) Utemeljite odgovor.

3. (i) Iz vzorca p zgradite priponsko drevo. (ii) Kateri vzorec dolžine 1 se največkrat ponovi? Kateri vzorec dolžine 2 se največkrat pojavi? Kako ste našli odgovor? (ii) Opišite postopek, kako ugotovimo, kateri vzorec dolžine k v besedilu dolžne n se največkrat ponovi.

Naloga 184. Peter Zmeda je našel listek z naslednjimi števili

$$5, 8, 66, 31, 27, 75, 29, 62, 36. \tag{3.9}$$

Peter se je domislil igre, pri kateri naključno generira m števil in se vpraša, če so ta števila *vsebovana* v enakem vrstnem redu med števili v (3.9). Na primer, za $m = 2$ in števili 27 in 29 je odgovor DA, za 29 in 27 pa NE.

Vprašanja:

1. (i) Kakšen je največji m za števila v (3.9), da bo odgovor še lahko DA? Utemeljite odgovor. (ii) Zapišite algoritem, ki pri dani množici N števil in množici M generiranih števil ugotovi, ali so števila v M vsebovana v N. (iii) Kakšno tehniko načrtovanja algoritmov ste uporabili? (iv) Kakšna je časovna zahtevnost vašega algoritma? Utemeljite odgovor.

2. Malce spremenimo igro. Sedaj imamo množico N z n števili in množico M z m števili in iščemo najdaljši niz števil, ki je vsebovan tako v N kot v M. (i) Naj množica N sestoji iz števila v (3.9), množica M pa iz števil 65, 66, 60, 27, 29, 88. Poiščite najdaljši niz, ki je vsebovan tako v N kot v M. (ii) Zapišite algoritem, ki najde takšen niz za poljuben N in M. (iii) Kakšno tehniko načrtovanja algoritmov ste uporabili?

3. Kakšna je časovna zahtevnost vašega algoritma? Utemeljite odgovor.

3.6 Naključnostni algoritmi

Največkrat zelo slabo poznamo porazdelitev vhodnih podatkov. Če pa vemo kaj več o sami porazdelitvi, lahko verjetnost in naključnost uporabimo kot orodje za načrtovanje in analizo algoritma tako, da postane njegovo izvajanje naključno. Naključnostni algoritem je torej algoritem, katerega izvajanje ni odvisno le od vhodnih podatkov, temveč tudi od vrednosti, ki jih ustvari generator naključnih števil. Posledično lahko takšna raba naključnosti spremeni sicer urejene podatke tako, da izgledajo kot naključni. Zaradi opisane preobrazbe lahko zato uporabimo preprostejše algoritme, ki se ob naključnih podatkih pričakovano obnašajo dokaj dobro.

Cilj

Da se zavedamo prednosti in slabosti uporabe naključnosti pri načrtovanju algoritmov in da znamo uporabiti verjetnost in naključnost za razvoj preprostih naključnostnih algoritmov.

Naloga 185. Na predavanjih smo z metodo *Monte Carlo* izračunali število π s pomočjo ploščine kroga. Poleg tega smo spoznali še metodo Las Vegas in kako povzporejamo takšne algoritme.
Vprašanja:

1. Podrobno *opišite in utemeljite* postopek, kako bi z metodo *Monte Carlo* izračunali π s pomočjo volumna krogle.

2. Natančnost metode *Monte Carlo* definira formula $O(\sqrt{1/N})$, kjer je N število iteracij. Kako se ta formula spremeni, če uporabimo p procesorjev, od katerih vsak izvede N iteracij? Utemeljite odgovor.

3. Na predavanjih smo tudi izvedeli, da je hitro urejanje (*QuickSort*) primer Las Vegas algoritma. Na predavanjih je bila tudi opisana splošna shema za povzporejanje algoritmov Las Vegas. Predlagajte, kako bi povzporedili hitro urejanje s p procesorji. Rešitev:

```
1 HitroUrejanje(polje A):
2   (A1, A2) := Razdeli(A)
3   vzporedno (HitroUrejanje(A1), HitroUrejanje(A2))
```

ne šteje.

Naloga 186. *Naključnostni algoritmi.* Na predavanjih smo spoznali dve vrsti naključnostnih algoritmov: Monte Carlo in Las Vegas.
Vprašanja:

1. Med letom smo spoznali algoritem `QuickSelect`. Ali je to algoritem tipa Monte Carlo ali tipa Las Vegas? Utemeljite odgovor.

2. Kako bi povzporedili ta algoritem? Ali je kakšen drug algoritem primernejši za povzporejanje?

Namig: Učinkovitejše kot bo povzporejanje, več točk boste dobili.

3. Peter Zmeda je navdušen nad naključnostnimi algoritmi. Tako se je odločil, da bo napisal naključnostni algoritem, ki išče v uteženem grafu $G(V, E)$, v katerem ni negativnih povezav, najkrajše poti iz vozlišča s do vseh ostalih vozlišč. Kaj menite, kako učinkovit bo njegov algoritem? Utemeljite svoj odgovor.

Naloga 187. Obstaja vrsta implementacij slovarja in med njimi tudi tista s preskočnim seznamom.
Vprašanja:

1. Ali gre v tem primeru za:

- Monte Carlo naključnostni algoritem; ali
- Las Vegas naključnostni algoritem; ali
- drug naključnostni algoritem[12]; ali
- ne gre za naključnostni algoritem?

Utemeljite odgovor.

2. Peter Zmeda je, hm, relativno priden študent, ki se hitro nauči, kako deluje takorekoč vsaka podatkovna struktura. Težavo ima potem v praksi, ker ne ve, kdaj kakšno podatkovno strukturo uporabiti. Recimo, da tehta med uporabo AVL drevesa in drevesa Patricia. Napišite in utemljite s *po dvema razlogoma* (situacijama), kdaj bi uporabil (i) AVL drevo in kdaj (ii) drevo Patricia.

3. Recimo, da imamo množico n števil in želimo izvedeti, ali sta dve števili enaki. (i) Zapišite algoritem za ta problem in utemeljite njegovo pravilnost. (ii) Kakšna je njegova časovna zahtevnost? (iii – dodatna) Koliko primerjav je najmanj potrebnih, da najdemo par enakih števil med n števili? Utemeljite odgovor.

Naloga 188. Peter Zmeda je našel listek z naslednjim programom:

```
1 int KajDela(A, k):
2   if A.length = 1 return A[0];
3   i= Random(A.length); p= A[i];
4   A1, A2= Razdeli(A, p);
5   if (k XXX A1.length) return KajDela(A1, k);
6   else return KajDela(A2, k-A1.length);
```

Na listku je bilo še pripisano, da funkcija `Random(p)` vrne naključno celo število večje ali enako 0 in manjše od p. Poleg tega funkcija `Razdeli(A, p)` tvori dve tabeli in sicer `A1` ter `A2` tako, da so v tabeli `A1` tisti elementi iz tabele `A`, ki so manjši ali enaki p.
Vprašanja:

1. Peter sumi, da funkcija poišče k-ti element po velikosti v tabeli `A`. (i) S čim naj nadomesti `XXX` v programu, da bo to res? (ii) Utemeljite odgovor.

2. Peter Zmeda je našel prevedeno knjižnico `mediana`. Ugotovil je, da vsebuje samo funkcijo `mediana(A)`, ki vrne srednji element polja `A`. (i) Kako

[12]Na primer obstajajo tudi Atlantic City naključnostni algoritmi.

lahko uporabi to funkcijo, da najde k-ti element za poljuben $1 \leq k \leq n$, kjer je n število elementov v tabeli `A`? (ii) Naj bo $f(n)$ časovna zahtevnost funkcije `mediana(A)`. Kakšna je časovna zahtevnost vaše rešitve? Utemeljite odgovor.

3. Zgornji program je naključnostni. (i) Za katero vrsto naključnostnega algoritma gre? (ii) Utemeljite odgovor. (iii) Kakšna je časovna zahtevnost algoritma? Bodite natančni pri opisu in utemeljitvi.

Naloga 189. *Iskanje praštevil.* Eden najstarejših postopkov za iskanje praštevil se imenuje Eratostenovo sito:

```
1 Presej(n):
2   for i= 1..n: prastevilo[i]= true
3   for i= 2..n:
4     j= 2*i
5     while j <= n:
6       prastevilo[j]= false
7       j= j+i
8   return prastevilo
```

Vprašanja:

1. Pokažite, da so na koncu izvajanja funkcije `Presej` tisti elementi polja `prastevilo`, za katere velja `prastevilo[j]==true`, v resnici praštevila.

2. Kakšna je časovna in prostorska zahtevnost algoritma `Presej`? Utemeljite odgovor.

3. Recimo, da imamo na voljo funkcijo `Prastevilo(x)`, ki, če x ni praštevilo, to vedno pravilno ugotovi in vrne `true`. Če pa odgovori `false`, je verjetnost p, da je x v resnici praštevilo. (i) Napišite algoritem, ki uporablja funkcijo `Prastevilo()` ter ugotovi, ali je x praštevilo z verjetnostjo $p' > p$. Utemeljite pravilnost vašega algoritma. (ii) Kako se da vaš algoritem povzporediti (*parallelize*)? Utemeljite odgovor.

Naloga 190. Eden od zadnjih izumov algoritmov se imenuje `Bogosort`:

```
1 Bogosort(A)
2   PONAVLJAJ brez konca:
3     Naključno izberi permutacijo π, dolžine n
4     ČE je polje π(A) urejeno POTEM vrni π(A)
```

V algoritmu je A polje števil dolžine n. Na primer, naj bo:

- $A = (23, 44, 11, 15)$ in
- naključna permutacija $\pi = (1, 3, 2, 4)$,
- potem je $\pi(A) = (23, 11, 44, 15)$ ter
- očitno to ni urejeno polje.

Vprašanja:

1. (i) Kakšna je časovna zahtevnost algoritma `Bogosort` v najslabšem primeru? Utemeljite odgovor. (ii) Kakšna je pričakovana časovna zahtevnost algoritma `Bogosort`? Utemeljite odgovor.

2. Ali je `Bogosort` Las Vegas ali Monte Carlo algoritem? Utemeljite odgovor.

3. (i) Predlagajte, kako povzporediti `Bogosort` (ne kakšen drug algoritem) na p procesorjih. Ocenjevalo se bo pravilnost vašega algoritma in časovno zahtevnost vašega algoritma. (ii) Kakšna je časovna zahtevnost vaše rešitve? Utemeljite odgovor.

3.7 P in NP

Nedeterminizem je močno orodje za načrtovanje algoritmov, s katerim se srečamo v tem razdelku. Po drugi strani se moramo v praksi spraševati, ali je algoritem uporaben, kar pomeni, da nam izračuna rešitev v doglednem času. V teoriji opredelimo algoritem kot uporaben, če ima polinomsko časovno zahtevnost.

Ko uparimo nedeterminizem in polinomskost, naletimo na oviro izvedbe nedeterministično polinomskih (NP) algoritmov na realnih računalnikih, ki pa so zgolj deterministični. Z drugimi besedami, na realnih računalnikih so praktični samo deterministično polinomski (P) algoritmi.

Cilj

Da smo sposobni razpoznati in pokazati, ali sodi nek problem v razred NP in kdaj problem sodi v razred P.

Naloga 191. Eden od problemov, ki smo jih obravnavali na predavanjih, je MAX-3-SAT. Za ta problem vemo, da je NP-poln.
Vprašanja:

1. Naslednje izraze obravnavajte kot primere MAX-3-SAT problema in jih rešite. Za vsako rešitev podajte dokazilo.

- $(x_2 \vee \overline{x_4} \vee \overline{x_1})\ \&\ (x_1 \vee x_2 \vee x_1)\ \&\ (\overline{x_3} \vee x_2 \vee \overline{x_2})\ \&\ (\overline{x_3} \vee \overline{x_1} \vee \overline{x_2})$
- $(x_1 \vee \overline{x_2} \vee x_4)\ \&\ (\overline{x_3} \vee \overline{x_1} \vee x_3)\ \&\ (x_2 \vee x_1 \vee \overline{x_2})\ \&\ (\overline{x_1} \vee x_4 \vee \overline{x_2})$
- $(\overline{x_1} \vee x_1 \vee x_2)\ \&\ (\overline{x_2} \vee \overline{x_3} \vee \overline{x_3})\ \&\ (\overline{x_3} \vee x_1 \vee x_2)\ \&\ (\overline{x_2} \vee x_1 \vee \overline{x_3})$

Namig: Rešitev problema MAX-3-SAT je število zadovoljivih stavkov.

2. (i) Napišite deterministični algoritem za reševanje problema MAX-3-SAT in utemeljite njegovo pravilnost. (ii) Kakšna je prostorska in kakšna časovna zahtevnost vašega programa? Utemeljite odgovora.

3. (i) Napišite Monte Carlo naključnostni algoritem za reševanje problema MAX-3-SAT in utemeljite, da je res tipa Monte Carlo. (ii) Kakšen je najboljši čas vašega algoritma in kakšen najslabši? Utemeljite odgovor. (iii) Nadgradite program v genetski program.

Namig: Nadgradnja bo vključevala ne več kot 15 vrstic. Verjetno manj.

4. [Dodatno] Kakšna je verjetnost, da vaš deterministični algoritem najde pravilno rešitev v k poskusih? Utemeljite odgovor. Komentirajte odgovor.

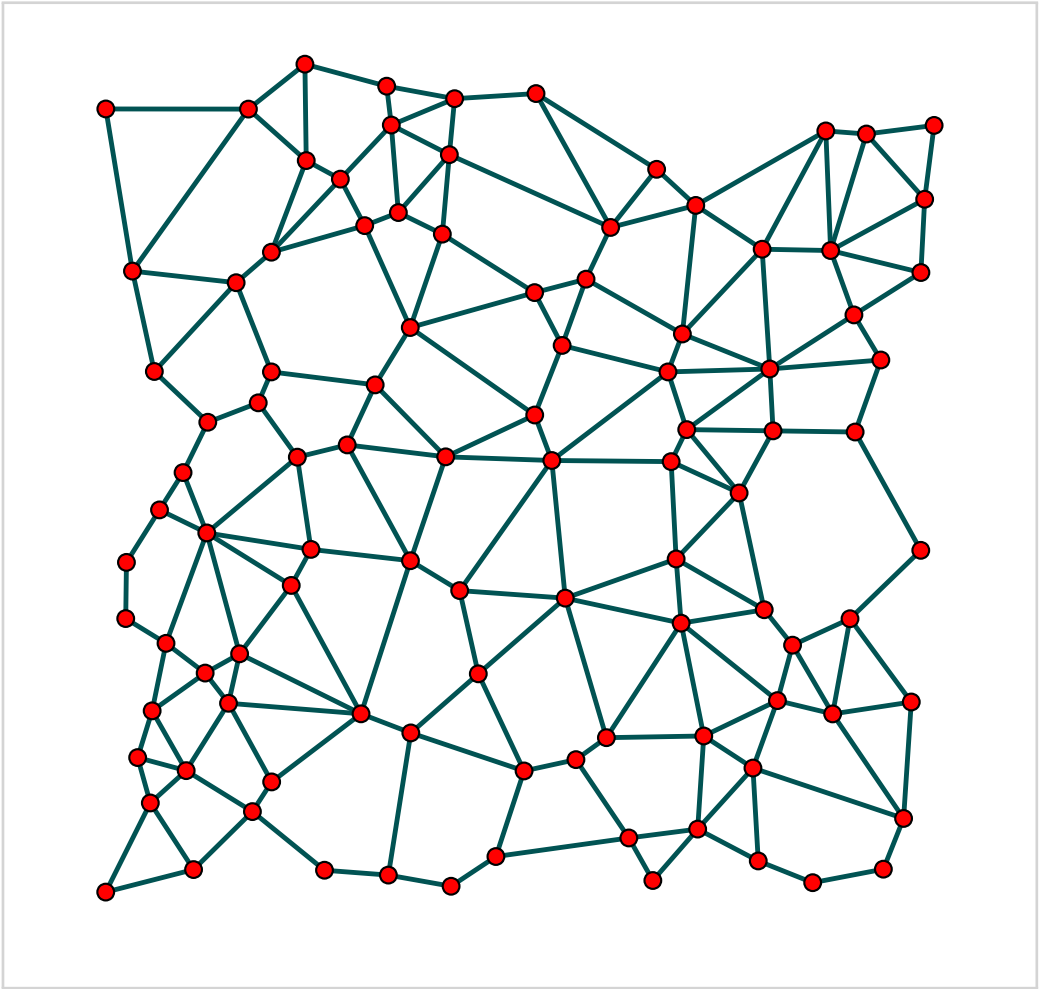
Naloga 192. Pri reševanju problem Hamiltonovega obhoda v grafu $G(V, E)$ iščemo v grafu takšen obhod, ki gre skozi vsa vozlišča V grafa G natančno enkrat[13]. Za problem Hamiltonovega obhoda HAM smo rekli, da je v NP.
Vprašanja:

1. Utemeljite, zakaj je problem Hamiltonovega obhoda v NP.

[13]Tukaj in v nadaljevanju naloge razumemo, da se pri obhodu vrnemo v začetno vozlišče, kar pomeni, da gremo skozi to vozlišče dvakrat.

NAMIG: Najprej zapišite, kdaj je problem v NP, in to definicijo uporabite za dokaz.

2. Malce posplošimo problem tako, da definiramo njegovo različico k-HAM, pri kateri iščemo v grafu $G(V, E)$ obhod, ki gre skozi vsaj k točk grafa G natanko enkrat. (i) V grafu na sl. 3.22 označite rešitev problema 28-HAM.



Slika 3.22: Primer grafa (Vir: wikimedia, Gabriel graph).

(ii) Recimo, da imamo algoritem, ki reši problem k-HAM v času $f(k, n)$. Uporabite ga za rešitev problema MAX-HAM, ki poišče v grafu najdaljši obhod, ki nobeno vozlišče ne obišče več kot enkrat. (iii) Kakšna je časovna zahtevnost vašega algoritma?

3. Petrov direktor je prebral, da za problem HAM ne poznamo učinkovite deterministične rešitve. Vendar je ugotovil, da bi bilo dovolj, če bi v svojem

grafu našel vsaj obhod dolžine 3. (i) Pomagajte Petru Zmedi in zapišite algoritem, za rešitev problema 3-HAM. (ii) Utemeljite njegovo pravilnost. (iii) Kakšna je njegova časovna zahtevnost?

4. [Dodatno] Zapišite in analizirajte rešitev algoritma za k-HAM, kjer naj bo k parameter. Komentirajte odgovor.

Naloga 193. Peter in Metka se igrata naslednjo igrico. Peter najprej zapiše množico A z n števili. Metka nato izbere poljubno število K in Peter mora iz A izbrati takšen $S \subseteq A$, da je vsota števil v S enaka K. Na primer, za $A = \{21, 72, 87, 13, 35, 37, 90, 49, 56, 95, 98, 78\}$ in $K = 288$ je $S = \{87, 13, 90, 98\}$. Vprašanja:

1. (i) Napišite deterministični algoritem, ki za poljuben A in K poišče S. (ii) Utemeljite njegovo pravilnost. (iii) Kakšna je njegova časovna zahtevnost?

2. (i) Napišite še enak nedeterministični algoritem. (ii) Utemeljite njegovo pravilnost. (iii) Kakšna je njegova časovna zahtevnost?

3. Spremenimo pravila tako, da mora Peter razdeliti A na disjunktni množici S_1 in S_2, kjer $A = S_1 \cup S_2$ in se vsoti elementov v posameznih množicah kar najmanj razlikujeta. Napišite algoritem sedaj in analizirajte njegovo zahtevnost.

Naloga 194. *Vrste s prednostjo in požrešnost.* Pri tej nalogi bomo dvakrat izvedli naslednje operacije:

> `Insert(23)`, `Insert(19)`, `Insert(17)`, `Insert(13)`, `Insert(11)`, `Insert(7)`, `Insert(5)`, `Insert(3)`, `Insert(2)` in `Insert(1)`.

Vprašanja:

1. Najprej imamo prazno leno binomsko minimalno-urejeno kopico in nad njo izvedite zgornje operacije. (i) Narišite stanje podatkovne strukture na koncu in izpišite njen potencial, kot smo ga definirali na predavanjih. (ii) Izvedite še operacijo `DeleteMin()`, narišite strukturo po tej operaciji, zapišite realno ceno operacije, zapišite amortizirano ceno in končni potencial.

2. Običajna dvojiška kopica ima v korenu vsake podkopice najmanjši element in je levo poravnana. *MinMax* kopica se razlikuje od navadne kopice

v tem, da loči podkopice, ki so na lihih in na sodih nivojih – koren je na lihem nivoju in njegova otroka sta na sodih nivojih. Kopica je še vedno levo poravnana, vendar velja, da je na lihih nivojih v korenu najmanjši element in na sodih največji element. (i) Nad prazno takšno kopico izvedite zgornje operacije. Narišite kopico na koncu. (ii) Kje vse se v takšni kopici lahko nahaja tretji najmanjši element? Utemeljite odgovor. (iii) Kje vse se v takšni kopici lahko nahaja drugi največji element? Utemeljite odgovor.

3. Najcenejši Hamiltonov obhod je definiran na grafu $G(V, E)$, kjer imajo povezave tudi uteži $w(u, v) \in \mathbb{R}$. (i) Peter Zmeda je prišel na idejo, da bi ta problem rešil s podobnim pristopom, kot je uporabljen v Kruskalovem algoritmu: uredimo povezave po velikosti; povezave jemljemo po vrsti iz urejenega seznama in jih dodajamo v konstruirano pot, če so dopustne[14]. (i) Menite, da bo algoritem deloval? Utemeljite svoj odgovor. (ii) Kateri problem je težji: iskanje Hamiltonovega obhoda ali iskanje najcenejšega Hamiltonovega obhoda? Utemeljite odgovor[15].

Naloga 195. *Optimizacijski problemi.* Peter Zmeda je v Butalah odprl lektorsko službo. Delo načrtuje samo za tekoči dan in sicer zjutraj dá povpraševanje za potrebe po prevodih, stranke pa mu pošljejo besedila in koliko so pripravljene plačati za posamezen prevod. Peter besedila pregleda in oceni, koliko časa bi potreboval za posamezen prevod. Na koncu se odloči, katera besedila bo sprejel v prevajanje in sicer tako, da bo imel čim večji zaslužek.
Vprašanja:

1. Danes je dobil sedem besedil, ki jih je pregledal. Poleg tega je dobil naslednje ocene časa za prevajanje posameznega besedila in ceno, ki jo je stranka pripravljena plačati:

besedilo	1	2	3	4	5	6	7
čas	13	19	7	7	5	5	3
plačilo	35	24	6	20	11	20	15

Svetujte Petru, ki ima 24 ur časa, za prevode katerih besedil naj se odloči, da bo njegov zaslužek čim večji. Utemeljite odgovor.

[14]Dopusten pomeni seveda tukaj nekaj drugega kot pri Kruskalu: da dodatek nove povezave ohranja konstruirano pot brez ciklov.

[15]Problem P_1 je vsaj tako težak kot problem P_2, če s pomočjo rešitve problema P_2 lahko rešimo problem P_1.

2. Napišite algoritem, ki bo pri n besedilih, kjer Peter za i-to besedilo potrebuje t_i časa in za prevod zasluži c_i, predlagal, *katera besedila* naj Peter sprejme v prevajanje, da bo njegov zaslužek največji. Na voljo ima T časa.

NAMIG: V zgornjem primeru je $n = 7$ in $T = 24h$.

3. Peter je razširil podjetje in je zaposlil še dva prevajalca. Napišite nedeterministični algoritem, ki razporedi delo vsem trem prevajalcem tako, da je skupni zaslužek vsaj C. Utemeljite pravilnost algoritma.

NAMIG: Razmislite, kaj je dokazilo.

Naloga 196. Poznamo dve osnovni predstavitvi grafa in sicer z matriko sosednosti in s seznami sosedov.

NAMIG: Nalogo boste veliko lažje rešili, če najprej v programskem jeziku ali kako drugače formalno zapišete eno in drugo predstavitev grafa.
VPRAŠANJA:

1. (i) Zapišite algoritem, ki pretvori predstavitev z matriko sosednosti v predstavitev s seznami sosedov. (ii) Kakšna je časovna zahtevnost vašega algoritma? Utemeljite odgovor.

2. Zapišite algoritem, ki poišče v grafu cikel, to je obhod po povezavah, ki se vrne v izvorno vozlišče in pri tem ne obišče nobenega vozlišča več kot enkrat.

3. Iskanje najdaljšega cikla v grafu je NP-poln problem, saj je najdaljši cikel v grafu Hamiltonov cikel in iskanje slednjega je NP-poln problem. Zapišite nedeterministični algoritem, ki najde najdaljši cikel v grafu.

NAMIG: Naloga je malce zahtevnejša. Za manj točk lahko zapišete nedeterminističen algoritem, ki poišče v grafu cikel dolžine k, kjer je k parameter in velja $1 \leq k \leq n$.

Naloga 197. Peter Zmeda ni od muh in včasih celo posluša predavanja. Na zadnjih predavanjih so govorili o obhodih po grafu. Govorili so o Hamiltonovem in Eulerjevem obhodu. Peter pa bi ne bil Peter, če bi res poslušal na predavanjih. Tako si je poleg obhodov zapomnil nekaj tudi o preiskovanju v globino in v širino.

Vprašanja:

1. Za nalogo je moral napisati algoritma, ki bi v grafu poiskala obhoda – seveda en algoritem za iskanje Hamiltonovega in en za iskanje Eulerjevega obhoda. Peter je za iskanje obakrat uporabil preiskovanje v širino. (i) Zapišite graf, ko sprehod v širino najde Hamiltonov obhod, in utemeljite pravilnost. (ii) Zapišite graf, ko sprehod v širino najde Eulerjev obhod, in utemeljite pravilnost. (iii) Zapišite graf, ko sprehod v širino ne najde Hamiltonovega obhoda, in utemeljite pravilnost. (iv) Zapišite graf, ko sprehod v širino ne najde Eulerjevega obhoda, in utemeljite pravilnost.

2. Recimo, da imamo drevo, kjer pa so povezave med starši in otroci podvojene – imamo tako dve povezavi med vsakim staršem in otrokom. Z drugimi besedami, vse povezave so podvojene. (i) Zapišite algoritem, ki poišče Eulerjev obhod v drevesu in izpiše vozlišča po vrsti, kot jih v obhodu obiščemo. (ii) Utemeljite pravilnost vašega algoritma. (iii) Kakšna je časovna zahtevnost vašega algoritma? Utemeljite odgovor.

3. Problem trgovskega potnika je definiran na naslednji način. Imamo n točk (x_i, y_i) v ravnini, pri čemer je razdalja med točkama evklidska. Trgovčev obhod je najkrajši obhod, ki obišče vse točke in se vrne v točko, kjer je pričel obhod. (i) Ali trgovčev obhod kakšno točko razen prve obišče večkrat? Utemeljite odgovor. (ii) Zapišite nedeterministični polinomski algoritem, ki najde takšen obhod. (iii) Utemeljite pravilnost in časovno zahtevnost vašega algoritma.

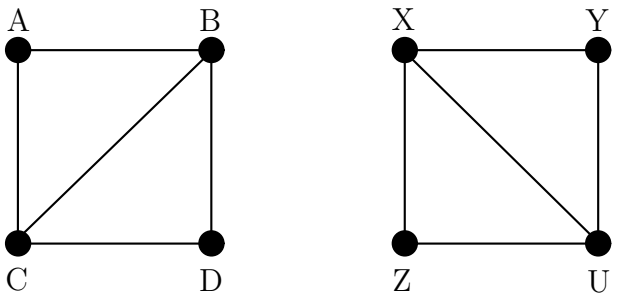
Namig: Za nekaj manj točk lahko zapišete nedeterministični polinomski algoritem, ki vrne obhod največ dolžine d.

Naloga 198. *Grafi.* Grafi lahko izgledajo različno, a v resnici niso, kot je preprost primer na sl. 3.23.
Vprašanja:

1. (i) Zapišite definicijo, kdaj sta grafa enaka. (ii) Utemeljite odgovor.

2. Peter Zmeda ima dva grafa $G_1(V_1, E_1)$ in $G_2(V_2, E_2)$, za katera trdi, da sta enaka. (i) Kakšno dokazilo naj vam poda, da mu boste verjeli? Utemeljite odgovor. (ii) Koliko časa boste potrebovali (časovna zahtevnost), da se prepričate v pravilnost njegove trditve?

3. Pavel si je zamislil naslednji algoritem, kako bi ugotovil, ali sta grafa enaka:



Slika 3.23: Dva enaka grafa, ki sta narisana različno.

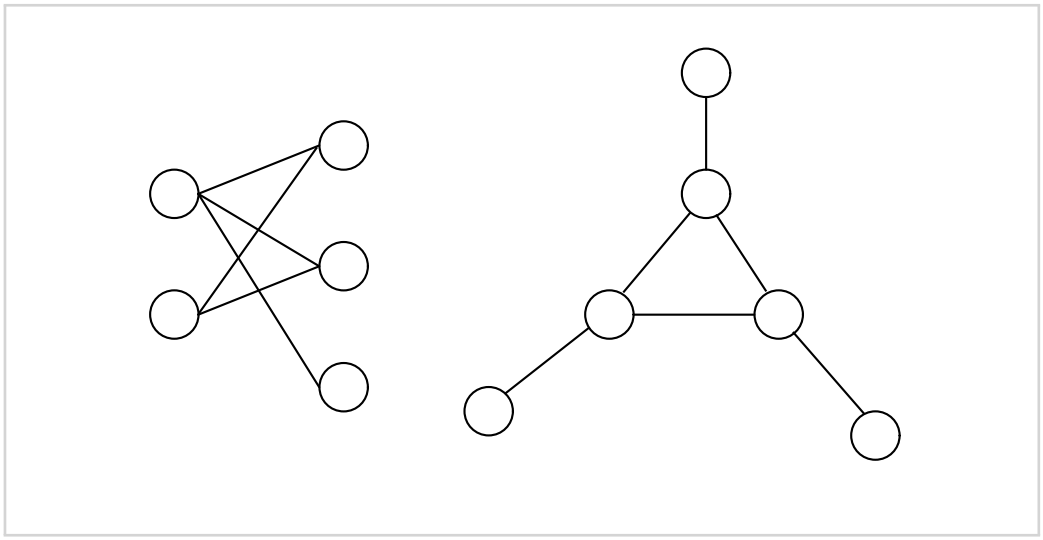
```
1 Enaka(G1, G2):
2   Uredi V1 glede na število sosednjih vozlišč,
3     če je več vozlišč z enakim številom sosedov,
4     uredi po oznakah vozlišč.
5   Uredi enako V2.
6   Preimenuj vozlišča V1 v oznake 1,...,|V1|.
7   Preimenuj vozlišča V2 v oznake 1,...,|V2|.
8   Preveri, ali sta grafa enaka.
```

Ali je njegov algoritem pravilen? Utemeljite odgovor.

Naloga 199. Družabno omrežje je v resnici en velik graf $G(V, E)$, kjer so vozlišča V posamezniki in povezave E poznanstva. Slednja naj bodo vzajemna (torej je graf neusmerjen). Peter se je prav tako priključil omrežju in kmalu spoznal, da ni samo pomembno koga poznaš, ampak da si del skupine, kjer se vsi med seboj poznajo. Takšni skupini recimo *druščina*.
Vprašanja:

1. Recimo, da sta grafa na sl. 3.24 dve družabni omrežji. (i) Kako veliki sta največji druščini v vsakem od njih? Zapišite ju in utemeljite odgovor. (ii) Koliko povezav najmanj je potrebno dodati v vsakem od grafov, da se največji druščini povečata? Utemeljite odgovor.

2. Recimo, da imamo graf $G(V, E)$ in $V_d \subseteq V$, ki predstavlja največjo druščino v grafu G. (i) Zapišite algoritem, ki s čim manj dodatnimi povezavami poveča druščino V_d. (ii) Kakšna je časovna in prostorska zahtevnost vašega algoritma? Utemeljite odgovor.



Slika 3.24: Dve družabni omrežji.

3. Ponovno imamo graf $G(V, E)$. Zapišite algoritem, ki za nek k (k je parameter algoritma) poišče v G druščino V_d, kjer $|V_d| \geq k$. Če ne gre z deterministič nim algoritmom, poskusite z nedeterminističnim.[16]

Naloga 200. Fiziki so res čudni tiči. Ti, s katerimi ima tokrat opravka naš Peter Zmeda, so posebej čudni. Še med seboj si namreč ne zaupajo. Tako je profesor Bučka odkril neko novo obliko letalskega krila, vendar noče nikakor izdati njegove oblike. Kljub temu želi, da mu krilo prebarvajo na ksanadu barvo. V barvarni bi radi naročili barvo, ki pa ni poceni, zato ne želijo naročiti preveč barve, da bi ostajala. Za pomoč so zaprosili Petra.
Vprašanja:

1. Peter je od dr. Bučke želel zgolj implementacijo naslednjega modula:

```
1 interface Krilo {
2       // skrajni x-koordinati krila
3   float Levo();
4   float Desno();
5       // skrajni y-koordinati krila
6   float Zgoraj();
7   float Spodaj();
```

[16]Slednji mora tvoriti dokazilo in morate pokazati, kako ga lahko v polinomskem času preverite.

```
8       // ali je točka (x, y) na krilu
9     boolean NaKrilu(float x, float y);
10  }
```

Pri tem je predpostavka, da je krilo položeno plosko na tla. Kaj menite, kakšen algoritem si je zamislil Peter za izračun površine krila? Opišite postopek vključno z zapisom algoritma.

2. Na predavanjih smo spoznali problem iskanja najcenejšega vpetega drevesa v grafu $G(V, E)$. To je optimizacijski problem. (i) Preoblikujte ga v odločitveni problem. (ii) Ali je vaš odločitveni problem v NP? Utemeljite odgovor.

3. Eden od zanimivih neuporabnih algoritmov je *Bogosort*, katerega psevdokoda je:

```
1   WHILE NOT urejeno(polje):
2     NaključnoPermutiraj(polje)
```

(i) Ali gre za Monte Carlo ali za Las Vegas algoritem? Utemeljite odgovor. (ii) Kakšna je: (a) najboljša, (b) najslabša in kakšna (c) pričakovana časovna zahtevnost algoritma? Utemeljite odgovor.

Naloga 201. Imamo množico $S = \{a_1, a_2, \ldots, a_n\}$ neurejenih števil in število t. Definirajmo odločitveni problem *vsota podmnožice*:

> Ali v S obstaja podmnožica števil, katerih vsota je t?

Za ta problem vemo, da je NP-poln.
Vprašanja:

1. Pokažite, da je definirani problem nedeterministično polinomski (NP).

2. Ali postane problem bistveno lažji, če so števila S urejena? Utemeljite odgovor!

3. (i) Definirajte optimizacijsko različico odločitvenega problema *vsota podmnožice* in utemeljite svojo definicijo. (ii) Opišite genetski algoritem za reševanje optimizacijskega problema.

Namig: Za genetski algoritem definirajte ocenitveno *fitness* funkcijo, kaj predstavlja gen in kakšne so tri osnovne operacije genetskega algoritma.

Naloga 202. Naš prijatelj Peter Zmeda je postavljen pred naslednji problem. Imamo množico A z n števili, ki jih moramo razdeliti na dve podmnožici B in C tako, da velja $B \cup C = A$ in $B \cap C = \emptyset$ ter bosta hkrati vsoti števil v obeh množicah enaki. Na primer, če je $A = \{1, 3, 4, 5, 6, 7\}$, potem sta množici $B = \{1, 5, 7\}$ in $C = \{3, 4, 6\}$ rešitev opisanega problema, saj je vsota števil v obeh 13. Po drugi strani pa množice $A_1 = \{1, 22, 33\}$ ne moremo razdeliti na ta način na dve podmnožici.
Vprašanja:

1. Zapišite nedeterministični polinomski algoritem, ki množico A velikosti n razdeli na podmnožici B in C, kot je definirano zgoraj.

2. (i) Zapišite naključnostni algoritem, ki razdeli množico A velikosti n na dve podmnožici, kot je zapisano zgoraj. (ii) Vaš algoritem bo verjetno tipa Las Vegas - zakaj?

3. Sedaj pa malce spremenimo definicijo problema in sicer tako, da iščemo B in C tako, da se vsoti čim manj razlikujeta (prim. 3-SAT in MAX-3-SAT). V zgornjem primeru bi bila delitev A_1 na $B_{1,1} = \{1\}$ in $C_{1,1} = \{22, 33\}$ slabša od $B_{1,2} = \{1, 33\}$ in $C_{1,1} = \{22\}$, ki pa je spet slabša od $B_{1,1} = \{1, 22\}$ in $C_{1,1} = \{33\}$. (i) Napišite Monte Carlo algoritem, ki poišče čim boljšo delitev. (ii) Zakaj je vaš algoritem tipa Monte Carlo?

Naloga 203. V Tepanjah imajo majhno logistično podjetje *DidelVozi*, ki oskrbuje pet strank, razdalje med njimi pa so podane z naslednjo matriko:

/	1	2	3	4	5
1	/	2	9	3	2
2	5	/	3	8	6
3	4	6	/	3	8
4	3	7	3	/	6
5	8	1	3	7	/

kjer prva vrstica pomeni, da je od stranke 1 do stranke 2 dolžina poti 2, od stranke 1 do stranke 3 je dolžina poti 9 in tako naprej. Obhod med strankami je dejansko zaporedje strank, kot jih obiščemo. Recimo, dva obhoda med zgornjimi strankami sta $r_1 = (1 \to 2 \to 3 \to 4 \to 5 \to 1)$ in $r_2 = (1 \to 2 \to 4 \to 5 \to 3 \to 1)$.
Vprašanja:

1. Kateri od obhodov r_1 in r_2 je krajši? Utemeljite odgovor. Poiščite še en obhod, ki je krajši od obeh in utemeljite odgovor.

2. Recimo, da imate na voljo funkcijo `Random(k)`, ki vrne naključno število med 1 in k. Napišite funkcijo `Obhod(n)`, ki tvori naključni obhod r med n strankami.

3. Lastnik podjetja boter Dideldač se je, kot se temu danes reče, odločil optimirati stroške podjetja. Prvi strošek, ki se ga je spravil zmanjšati, je bila poraba goriva. Način za zmanjšanje tega stroška je, da pri oskrbi strank prevoziš čim krajšo pot. Za iskanje takšne poti se je za pomoč obrnil na Petra, ki je napisal naslednji program:

```
int Najkrajsi() {
  r= Obhod(5);
  min= Cena(r);
  for (i=1; i <= 120; i++) {
    r= Obhod(5);
    c= Cena(r);
    if (c < r) min= c;
  }
}
```

V zgornji kodi funkcija `Cena(r)` izračuna dolžino obhoda r in $120 = 5!$. (i) Zgornji postopek uporablja naključnost in zato je bodisi tipa *Monte Carlo* bodisi *Las Vegas*. Katerega tipa je? Utemeljite odgovor. (ii) Ali je problem iskanja najkrajšega obhoda med strankami sploh v NP? Utemeljite odgovor.

Namig: Pri slednji utemeljitvi bodite zalo natančni.

(dodatno) Naštejte še vsaj dva drugačna načina zmanjašanja porabe goriva, ki ju lahko naredi boter Dideldač.

Naloga 204. Peter je zaposlen v podjetju Evu, ki razvaža dobrine po poslovalnicah. V svoji podatkovni bazi imajo seznam poslovalnic in seznam parov poslovalnic, med katerimi obstaja povezava. Peter je nekaj časa opazoval, kako vozniki razvažajo dobrine. Opazil je, da vozniki na isti poti večkrat obiščejo iste poslovalnice. Prišel je do spoznanja, da bi se razvoz pocenil, če bi odpravil večkratno obiskovanje istih poslovalnic. Zato se je odločil, da bo napisal program, ki bo vozniku pred odhodom iz garaže dal seznam poslovalnic s, kot naj jih po vrsti obišče.
Vprašanja:

1. Kako lahko voznik preveri, če je s, ki ga je dobil, pravilen? (i) Zapišite algoritem, ki to preveri in (ii) kakšna je časovna zahtevnost vašega algoritma.

2. Kot dober strokovnjak je Peter ocenil časovno zahtevnost svojega algoritma in ugotovil, da je $O(n4^n)$, kjer je n število poslovalnic. Pri EVU razvažajo dobrine po 12 poslovalnicah, medtem ko jih njihova velika konkurenca FZV razvaža po 756 poslovalnicah. Ali menite, da je Petrova rešitev primerna za FZV? Utemeljite odgovor.

3. (i) Opišite genetski algoritem, ki bi rešil Petrov problem. (ii) Ali je vaš algoritem tipa Monte Carlo ali Las Vegas? Utemeljite odgovor.

Stvarno kazalo

Stvarno kazalo po nalogah


function quicksort(array)
var list less, equal, g
if length(array) ≤ 1
return array
select a pivot value
for each x in
if x <
if x =
if x >
return