

Rešene naloge iz operacijskih raziskav  
za študente finančne matematike

Janoš Vidali

Ljubljana, 2022



# Kazalo

<b>Predgovor</b>	<b>5</b>
<b>1 Naloge</b>	<b>7</b>
1.1 Zahtevnost algoritmov . . . . .	7
1.2 Celoštevilsko linearno programiranje . . . . .	15
1.3 Teorija odločanja . . . . .	35
1.4 Dinamično programiranje . . . . .	53
1.5 Algoritmi na grafih . . . . .	75
1.6 CPM/PERT . . . . .	101
1.7 Upravljanje zalog . . . . .	109
1.8 Pretoki in prerezi . . . . .	113
1.9 Razdelitve . . . . .	125
<b>2 Rešitve</b>	<b>127</b>
2.1 Zahtevnost algoritmov . . . . .	127
2.2 Celoštevilsko linearno programiranje . . . . .	139
2.3 Teorija odločanja . . . . .	173
2.4 Dinamično programiranje . . . . .	215
2.5 Algoritmi na grafih . . . . .	267
2.6 CPM/PERT . . . . .	321
2.7 Upravljanje zalog . . . . .	335
2.8 Pretoki in prerezi . . . . .	343
2.9 Razdelitve . . . . .	373
<b>Literatura</b>	<b>377</b>



## Predgovor

Na pobudo izr. prof. dr. Arjane Žitnik, ki od študijskega leta 2017/18 predava predmet Operacijske raziskave za študente 2. letnika Finančne matematike, sem kot asistent pri tem predmetu začel zbirati naloge iz vaj, kolokvijev in izpitov ter njihove rešitve. Pričujočo zbirko tako sestavlja 243 nalog, ki sem jih "podedoval" od svojih predhodnikov, in ki sem jih sam sestavil od študijskega leta 2016/17 naprej, ko sem prevzel vodenje vaj pri tem predmetu. Naloge so razporejene glede na princip, po katerih jih rešujemo, kar ravno sledi snovi, kot se obravnava pri predmetu Operacijske raziskave (pri čemer se v zadnjih letih ne pokrije nalog iz zadnjih dveh razdelkov). Tako lahko študenti enostavno poiščejo naloge za snov, ki bi jo radi vadili. S pregledovanjem nalog lahko bralec seveda tudi hitro tudi dobi občutek, kateri način reševanja bi bil primeren za določen tip naloge.

Na tem mestu bi se rad zahvalil Arjani Žitnik za spodbudo ter predhodnim predavateljem Vladimirju Batagelju, Sergiu Cabellu, Martinu Juvanu in mojim predhodnikom na asistentskem mestu Jerneju Azariji, Ninu Bašiču, Boštjanu Gabrovšku, Davidu Gajserju, Blažu Jelencu, Matjažu Konvalinki, Alenu Orbaniču in Emilu Žagarju, ki so sestavili marsikatero nalogo in tudi napisali nekatere rešitve. Žal za vse naloge ni bilo mogoče določiti avtorja, zato je ta informacija v sami zbirki izpuščena – so pa avtorji, kjer so ti znani ali vsaj verjetni, zapisani v izvornih datotekah, ki so javno dostopne na [repozitoriju zbirke](#). Pri vsaki nalogi je tako naveden vir – večinoma gre za vaje, kolokvije in izpite pri Operacijskih raziskavah, nekaj nalog pa je tudi z vaj pri predmetu Optimizacijske metode ter iz literature, ki je navedena na koncu zbirke.

Rad bi se zahvalil tudi študentom pri predmetu Operacijske raziskave, ki so prispevali rešitve nalog in popravke, in sicer Juretu Babniku, Evi Deželak, Lari Jagodnik, Maksu Perbilu, Tjaši Renko, Mateju Rojcu, Vitu Rozmanu, Juretu Sternadu, Zali Stopar Špringer, Nejcu Ševerkarju, Janu Šifrerju in Anji Trobec. Upam, da je bila in bo tako njim in njihovim sošolcem kot tudi bodočim študentom Finančne matematike ta zbirka v pomoč pri študiju in morda tudi pri reševanju problemov v resničnem življenju.



# 1 Naloge

## 1.1 Zahtevnost algoritmov

### Naloga 1.1.

Vir: Vaje OR 21.2.2018

Naj bo  $A[1 \dots n][1 \dots n]$  matrika (tj., seznam seznamov) dimenzij  $n \times n$ . Dan je spodnji program:

```
for  $i = 1, \dots, n$  do
  for  $j = i + 1, \dots, n$  do
     $A[i][j] \leftarrow A[j][i]$ 
  end for
end for
```

- (a) Kaj počne zgornji program?
- (b) Oceni število korakov, ki jih opravi zgornji program, v odvisnosti od parametra  $n$ .

### Naloga 1.2.

Vir: Vaje OR 21.2.2018

Naj bo  $\ell[1 \dots n]$  seznam, ki ima na začetku vse vrednosti nastavljene na 0. Dan je spodnji program:

```
 $i \leftarrow 1$ 
while  $i \leq n$  do
   $\ell[i] \leftarrow 1 - \ell[i]$ 
  if  $\ell[i] = 1$  then
     $i \leftarrow 1$ 
  else
     $i \leftarrow i + 1$ 
  end if
end while
```

- (a) Kaj se dogaja, ko teče zgornji program?
- (b) Oceni število korakov, ki jih opravi zgornji program, v odvisnosti od parametra  $n$ .

### Naloga 1.3.

Vir: Vaje OR 21.2.2018

Algoritem BUBBLESORT uredi vhodni seznam  $\ell[1 \dots n]$  tako, da zamenjuje sosednje elemente v nepravem vrstnem redu:

```
function BUBBLESORT( $\ell[1 \dots n]$ )
   $z \leftarrow n$ 
  while  $z > 1$  do
     $y \leftarrow 1$ 
    for  $i = 2, \dots, z$  do
      if  $\ell[i - 1] > \ell[i]$  then
```

```

         $\ell[i-1], \ell[i] \leftarrow \ell[i], \ell[i-1]$ 
         $y \leftarrow i$ 
    end if
end for
 $z \leftarrow y-1$ 
end while
end function

```

- (a) Izvedi algoritem na seznamu [7, 11, 16, 7, 5].
- (b) Določi časovno zahtevnost algoritma.

**Naloga 1.4.**

Vir: Vaje OR 12.10.2016

Algoritem MERGESORT uredi vhodni seznam tako, da ga najprej razdeli na dva dela, nato vsakega rekurzivno uredi, nazadnje pa zlije dobljena urejena seznama.

- (a) S psevdokodo zapiši algoritem MERGESORT.
- (b) Izvedi algoritem na seznamu [7, 11, 16, 7, 5, 0, 14, 1, 19, 13].
- (c) Določi časovno zahtevnost algoritma.

**Naloga 1.5.**

Vir: Vaje OR 21.2.2018

Število  $n$  želimo razcepiti na dva netrivialna celoštevilka faktorja, kar storimo s sledečim algoritmom:

```

function RAZCEP( $n$ )
    for  $i = 2, \dots, \lfloor \sqrt{n} \rfloor$  do
        if  $n/i$  je celo število then
            return ( $i, n/i$ )
        end if
    end for
    return  $n$  je praštevilo
end function

```

Določi časovno zahtevnost algoritma. Ali je ta algoritem polinomski?

**Naloga 1.6.**

Vir: Vaje OR 21.2.2018

Zapiši rekurziven algoritem, ki na vhod dobi celo število  $n$  in teče v času  $O(\sqrt{n})$ . Uporaba korenjenja ni dovoljena.

**Naloga 1.7.**

Vir: Kolokvij OR 17.4.2014

Dani so končna neprazna množica  $S \subset \mathbb{N}$  moči  $n$ , število  $k \in \{1, 2, \dots, n\}$  ter algoritem ALG:



```

function ALG( $S, k$ )
   $x \leftarrow$  naključen element  $S$ 
   $S^+ \leftarrow \{y \in S \mid y > x\}$ 
   $S^- \leftarrow \{y \in S \mid y < x\}$ 
  if  $|S^-| < k - 1$  then
    return ALG( $S^+, k - |S^-| - 1$ )
  else if  $|S^-| = k - 1$  then
    return  $x$ 
  else
    return ALG( $S^-, k$ )
  end if
end function

```

Ugotovi, kaj je izhod algoritma pri danih vhodnih podatkih  $S$  in  $k$ . Oceni časovno zahtevnost algoritma v najslabšem in v povprečnem primeru.

### Naloga 1.8.

Vir: Kolokvij OR 25.11.2010

- Dokaži, da za poljubni konstanti  $a, b \in \mathbb{R}$ , kjer je  $b > 0$ , velja  $(n + a)^b = O(n^b)$ .
- Naj bo  $f$  naraščajoča funkcija. Ali velja  $g(n) = O(f(g(n)))$ ?
- Dokaži, da če  $T$  zadošča pogoju  $T(n) = 2T(\lceil n/2 \rceil) - 13$  za  $n \geq 2$ , potem je  $T(n) = O(n)$ .

### Naloga 1.9.

Vir: Kolokvij OR 5.4.2012

Naj bo  $G = (V, E)$  enostaven usmerjen graf (tj., brez zank in vzporednih povezav), podan z matriko sosednosti  $A = (a_{ij})_{i,j=1}^n$  ( $n = |V|$ ), kjer je  $a_{ij} = 1$ , če je  $ij \in E$ , in  $a_{ij} = 0$  sicer. Želimo ugotoviti, ali v  $G$  obstaja kak usmerjen trikotnik (tj., usmerjen cikel dolžine 3).

- Konrad je napisal naslednji algoritem, ki naj bi rešil ta problem:

```

for  $i = 1, \dots, n$  do
  for  $j = 1, \dots, n$  do
    for  $k = 1, \dots, n$  do
      for  $\ell = 1, \dots, n$  do
        if  $a_{ij} = a_{jk} = a_{k\ell} = 1 \wedge \ell = i$  then
          return TRUE
        end if
      end for
    end for
  end for
end for
return FALSE

```

Ali algoritem deluje pravilno? Kakšna je njegova časovna zahtevnost?

- V psevdokodi napiši algoritem, ki reši problem v času  $O(n^3)$ . Koliko korakov naredi tvoj algoritem v najboljšem in koliko v najslabšem primeru?

**Naloga 1.10.**

Vir: Izpit OR 26.6.2012

Dr. Kaczyński se ukvarja s podatkovno strukturo  $A$ , ki je zelo podobna tabeli (angl. *array*) celih števil. Vrednosti v posameznih celicah "tabele"  $A$  lahko beremo, ne moremo pa jih spreminjati. Elementi "tabele"  $A$  so indeksirani s celimi števili od 1 do  $n$ , kjer je  $n = A.length()$  indeks zadnjega elementa "tabele"  $A$ . Edina operacija (poleg dostopanja do posameznih elementov), ki jo lahko izvajamo nad  $A$ , je "obračanje podtabele". Če ima na začetku  $A$  vrednosti

$$A = [a_1, a_2, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_{j-1}, a_j, a_{j+1}, \dots, a_{n-1}, a_n],$$

po klicu ukaza  $A.reverse(i, j)$  izgleda takole:

$$A = [a_1, a_2, \dots, a_{i-1}, a_j, a_{j-1}, \dots, a_{i+1}, a_i, a_{j+1}, \dots, a_{n-1}, a_n].$$

Dr. Kaczyński se je lotil implementacije algoritmov nad to podatkovno strukturo. Najprej je seveda implementiral algoritem za urejanje:

```

n ← A.length()
for i = 2, ..., n do
  j ← i
  while j > 1 ∧ A[j - 1] > A[i] do
    j ← j - 1
  end while
  A.reverse(j, i)
  if j + 1 < i then

    end if
end for

```

- (a) V zgornji algoritem se je prikradla manjša tipkarska napaka, poleg tega pa je ena vrstica celo izginila. Popravi algoritem, da bo deloval pravilno.
- (b) Popravljeni algoritem izvedi na "tabeli"  $A = [5, 3, 12, 9, 1, 15]$ .

**Naloga 1.11.**

Vir: Izpit OR 9.7.2012

Dan je algoritem  $H$ , ki na vhod sprejme enostaven neusmerjen graf  $G = (V, E)$  (tj., brez zank in vzporednih povezav) ter različni vozlišči  $u, v \in V$ :

```

function H(G = (V, E), u, v)
  if |V| = 2 ∧ u ∈ ADJ(G, v) then
    return TRUE
  end if
  for w ∈ ADJ(G, u) \ {v} do
    if H(G - u, w, v) then
      return TRUE
    end if
  end for
  return FALSE
end function

```

Oznaka  $G - u$  predstavlja graf, ki ga dobimo iz  $G$  tako, da odstranimo vozlišče  $u$  in vse njegove povezave.

(a) Za katere vhode  $(G, u, v)$  algoritem  $H$  vrne TRUE?

**Namig:** pogledj, kako je z grafi z 2, 3, 4, 5, ... vozlišči, in poskusi posplošiti.

(b) Denimo, da je  $G = (V, E)$  graf z  $n$  vozlišči, pri čemer je vozlišče  $v$  izolirano (tj., nima nobene povezave), med vsakima drugima dvema vozliščema pa imamo povezavo. Pokaži, da se pri klicu  $H(G, u, v)$  (kjer je  $u \in V \setminus \{v\}$ ) zadnja vrstica funkcije  $H$  izvede  $\Theta((n-2)!)$ -krat.

### Naloga 1.12.

Vir: Izpit OR 4.9.2012

Dr. Kaczyński se ukvarja s podatkovno strukturo  $A$ , ki je zelo podobna tabeli (angl. *array*) celih števil. Vrednosti v posameznih celicah "tabele"  $A$  lahko beremo, ne moremo pa jih spreminjati. Elementi "tabele"  $A$  so indeksirani s celimi števili od 1 do  $n$ , kjer je  $n = A.length()$  indeks zadnjega elementa "tabele"  $A$ . Edina operacija (poleg dostopanja do posameznih elementov), ki jo lahko izvajamo nad  $A$ , je  $A.reverseStart(i)$ . Če ima na začetku  $A$  vrednosti

$$A = [a_1, a_2, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_{n-1}, a_n],$$

po klicu ukaza  $A.reverseStart(i)$  izgleda takole:

$$A = [a_i, a_{i-1}, \dots, a_2, a_1, a_{i+1}, \dots, a_{n-1}, a_n].$$

Dr. Kaczyński se je lotil implementacije algoritmov nad to podatkovno strukturo. Najprej je seveda implementiral algoritem za urejanje:

```

n ← A.length()
for i = n, ..., 2 do
  for j = 1, ..., i - 1 do
    if A[j] > A[i] then
      A.reverseStart(j)
      A.reverseStart(i)
    end if
  end for
end for
end for

```

(a) Oceni časovno zahtevnost zgornjega algoritma. Upoštevaj, da se operacija  $A.reverseStart(i)$  izvede v konstantnem času.

(b) Algoritem izvedi na "tabeli"  $A = [5, 9, 12, 7, 15]$ . Ali deluje pravilno?

(c) Napiši algoritem za urejanje, ki bo deloval pravilno. Njegova časovna zahtevnost ne sme biti slabša od časovne zahtevnosti algoritma dr. Kaczyńskiego.

**Naloga 1.13.**

Vir: Kolokvij OR 9.5.2013

Profesionalni programer je gospodu Velikozobcu prodal algoritem KOŠI. A Velikozobec je že v letih in je pozabil, kaj naj bi ta algoritem vračal. V prospektu je zasledil naslednjo psevdokodo algoritma:

**Vhod:**  $s[0 \dots n-1]$  tabela tromestnih števil

$T \leftarrow$  tabela velikosti 10, katere elementi so prazne vrste (podatkovne strukture)

```
for  $i = 0, 1, \dots, n-1$  do
  for  $i = 0, 1, \dots, 9$  do
    if zadnja številka  $s[i] = j$  then
       $T[j].append(s[i])$ 
    end if
  end for
end for
 $k \leftarrow 0$ 
for  $i = 0, 1, \dots, n-1$  do
  while  $T[k].isEmpty()$  do
     $k \leftarrow k + 1$ 
  end while
   $s[i] \leftarrow T[k].pop()$ 
end for
for  $i = 0, 1, \dots, n-1$  do
  for  $i = 0, 1, \dots, 9$  do
    if srednja številka  $s[i] = j$  then
       $T[j].append(s[i])$ 
    end if
  end for
end for
 $k \leftarrow 0$ 
for  $i = 0, 1, \dots, n-1$  do
  while  $T[k].isEmpty()$  do
     $k \leftarrow k + 1$ 
  end while
   $s[i] \leftarrow T[k].pop()$ 
end for
for  $i = 0, 1, \dots, n-1$  do
  for  $i = 0, 1, \dots, 9$  do
    if prva številka  $s[i] = j$  then
       $T[j].append(s[i])$ 
    end if
  end for
end for
 $k \leftarrow 0$ 
for  $i = 0, 1, \dots, n-1$  do
  while  $T[k].isEmpty()$  do
     $k \leftarrow k + 1$ 
  end while
   $s[i] \leftarrow T[k].pop()$ 
end for
return  $s$ 
```

(a) Algoritem izvedi na primeru

$$s = [146, 145, 359, 100, 359, 486, 200, 367, 980, 909, 257, 100].$$

Dovolj je napisati tabelo  $s$  po vsaki izmed zunanjih zank.

(b) Kakšna je časovna zahtevnost algoritma?

(c) Kaj algoritem vrača?

(d) Gospod Velikozobec je ugotovil, da se algoritem na vseh velikosti  $n = 10\,000$  izvede v približno 0.25 sekundah. Približno koliko časa potrebuje za izvajanje na vseh velikosti  $n = 50\,000$ ?

**Naloga 1.14.**

Vir: Izpit OR 28.8.2013

Asistent pri predmetu Operacijske raziskave je dobil idejo, kako bi predelal algoritem, ki z zlivanjem uredi seznam celih števil. Namesto, da na začetku seznam razdelimo na dva (približno) enako velika seznama, ga razdelimo na tri (približno) enako velike sezname. Le-te rekurzivno uredimo ter združimo ("zlijemo"), da dobimo urejen seznam.

Napiši psevdokodo algoritma časovne zahtevnosti  $O(n \log n)$ , ki uresniči asistentovo idejo. V psevdokodi lahko uporabiš algoritem ZLIJ, ki sprejme urejena seznama celih števil  $A$  in  $B$  dolžin  $m$  in  $n$  ter vrne urejen seznam števil iz  $A$  in  $B$  dolžine  $m + n$  v času  $O(m + n)$ . Psevdokode algoritma ZLIJ ni potrebno pisati. Utemelji, da je časovna zahtevnost tvojega algoritma res  $O(n \log n)$ .

**Naloga 1.15.**

Vir: Izpit OR 19.5.2015

Dana je zgornje trikotna matrika  $A \in \mathbb{Z}^{n \times n}$ , ki ima na diagonalni neničelne vrednosti, in vektor  $b \in \mathbb{Z}^n$ . V psevdokodi zapiši algoritem, ki sprejme  $A$  in  $b$  ter reši sistem enačb  $Ax = b$ . V psevdokodi lahko uporabiš le aritmetične operacije na številih, ne pa recimo operacije invertiranja matrike  $A$  (razen, če posebej napišeš psevdokodo za tako operacijo).

Kakšna je časovna zahtevnost tvojega algoritma?

**Naloga 1.16.**

Vir: Izpit OR 24.6.2015

Dan je algoritem, ki na vhod sprejme seznam števil  $A = [a_1, a_2, \dots, a_n]$  in vrne preurejen seznam  $A$ , pri čemer preurejanje poteka tako:

- Algoritem najprej primerja in uredi prvi dve števili v tabeli, nato drugo in tretje število, nato tretje in četrto število, ..., in nazadnje še  $(n - 1)$ -to in  $n$ -to število.

- Celoten opisan postopek iz prejšnje alineje algoritem ponavlja tako dolgo, dokler se med postopkom vsaj dvakrat dva elementa tabele zamenjata (tj., algoritem se ustavi, če se v ponovitvi prejšnje alineje zgodi največ ena zamenjava).

Ponazorimo delovanje algoritma na primeru vhoda  $A = [\pi, 7, 3, \sqrt{2}]$ . Potem se med izvajanjem algoritma seznam spreminja tako:

$$\begin{array}{lclclclclcl} A & \rightarrow & [\pi, 7, 3, \sqrt{2}] & \rightarrow & [\pi, 3, 7, \sqrt{2}] & \rightarrow & [\pi, 3, \sqrt{2}, 7] & \rightarrow & \\ & & [3, \pi, \sqrt{2}, 7] & \rightarrow & [3, \sqrt{2}, \pi, 7] & \rightarrow & [3, \sqrt{2}, \pi, 7] & \rightarrow & \\ & & [\sqrt{2}, 3, \pi, 7] & \rightarrow & [\sqrt{2}, 3, \pi, 7] & \rightarrow & [\sqrt{2}, 3, \pi, 7] & \rightarrow & \end{array}$$

- Podaj primer vhoda, kjer algoritem ne vrne urejene tabele. Na tem primeru zapiši zaporedje tabel med izvajanjem algoritma.
- Gornji algoritem zapiši v psevdokodi (ne prirejaj/izboljšuj algoritma). Kakšna je njegova časovna zahtevnost?  
**Namig:** ali je po tem, ko se prvič izvede prva alineja v opisu algoritma, katero število že na istem mestu, kot bi bilo v urejeni tabeli?
- Recimo, da na vhod dobimo tabelo dolžine  $n$ . Kakšni morajo biti vhodni podatki, da algoritem naredi najmanjše število primerjav? Kolikšno je v tem primeru to število?

## 1.2 Celoštevilsko linearno programiranje

### Naloga 2.1.

Vir: Vaje OR 16.3.2016

Občina Ljubljana želi projekte iz množice  $\mathcal{P} = \{p_1, p_2, \dots, p_n\}$ , pri čemer ima na voljo  $M$  evrov kapitala. V želji po razvoju regije želijo, da se v sklopu sponzoriranih projektov ustvari vsaj  $N$  delovnih mest. Projekt  $p_i$  ( $1 \leq i \leq n$ ) potrebuje  $d_i$  evrov finančne pomoči in zaposli  $a_i$  ljudi. Na občini so ocenili, da ima projekt  $p_i$  ob uspešnem dokončanju donos  $c_i$  evrov. Katere projekte naj sponzorira, da bo donos čim večji? Na smiseln način modeliraj opisani problem z linearnim programom.

### Naloga 2.2.

Vir: Vaje OR 16.3.2016

Obravnavajmo posplošen scenarij iz naloge 2.1.

- (a) Denimo, da so projekti lahko med seboj odvisni. Imejmo množico  $V \subseteq \mathcal{P}^2$ , ki določa, da za vsak par projektov  $(p_i, p_j) \in V$  velja, da lahko projekt  $p_i$  sponzoriramo le, če sponzoriramo tudi projekt  $p_j$ .
- (b) Nekateri izmed projektov so lahko v konfliktu. Naj bo  $S \subseteq 2^{\mathcal{P}}$  družina množic, ki določa, da so za vsako množico  $H \in S$  projekti iz  $H$  med seboj v konfliktu (tj., hkrati lahko sponzoriramo le enega izmed njih.)

Opiši, kako bi modelirali opisane omejitve.

### Naloga 2.3.

Vir: Vaje OR 16.3.2016

(problem prevoza in skladiščenja dobrin)

V Evropski uniji je na voljo  $n$  skladišč, pri čemer znašajo stroški najema  $i$ -tega skladišča  $f_i$  (ne glede na zasedenost), vsako skladišče pa lahko hrani enoto dobrine. Imamo  $m$  strank, ki jim dostavljamo dobrine, pri čemer  $c_{ij}$  ( $1 \leq i \leq n$ ,  $1 \leq j \leq m$ ) predstavlja strošek dostave dobrine stranki  $j$  iz skladišča  $i$ . Predpostavimo tudi, da ima vsaka stranka določeno potrebo  $d_j$ , ki ponazarja število enot dobrine, ki jo potrebuje. V katerih skladiščih naj hranimo dobrine, da bodo skupni stroški najema in dostave čim manjši? Na smiseln način modeliraj opisani problem z linearnim programom.

### Naloga 2.4.

Vir: Vaje OR 16.3.2016

Definiraj problem dominacijske množice v grafu in zapiši celoštevilski linearni program, ki rešuje opisani problem.

### Naloga 2.5.

Vir: Vaje OR 12.10.2016

Napiši linearni program, ki modelira iskanje največjega prirejanja v dvodelnem grafu.

**Naloga 2.6.**

Vir: Vaje OR 16.3.2016

Napiši linearni program, ki modelira iskanje najkrajše poti med danima vozliščema  $u$  in  $v$  v usmerjenem grafu  $G$ .

**Naloga 2.7.**

Vir: Izpit OR 28.8.2013

Kukavica bo izlegla 16 jajc in jih podtaknila v 12 gnezd, ki pripadajo dvema taščicama, štirim vrtnim penicam, trem travniškim cipam, dvema belima pastiricama in eni sovi. V vsako gnezdo lahko izleže največ tri jajca, pri čemer je verjetnost, da mladiči v gnezdju  $i$  preživijo, enaka  $p_{ij}$ , kjer je  $j$  število podtaknjenih jajc v gnezdju  $i$  (preživijo bodisi vsi ali noben mladič v posameznem gnezdju). Pri vsaki od petih vrst ptic želi izleči vsaj eno jajce, pri taščicah pa želi izleči strogo več jajc kot pri belih pastiricah. Poleg tega pri drugi beli pastirici ne bo odložila jajca, če bo pri prvi taščici odložila dve jajci ali več. Kukavica želi maksimizirati pričakovano število preživelih mladičev.

Zapiši problem kot celoštevilski linearni program.

**Naloga 2.8.**

Vir: Kolokvij OR 31.5.2012

Vinar Janez je pridelal 2000 litrov rumenega muškata, 10000 litrov laškega rizlinga in 5000 litrov renškega rizlinga. Njegovi kupci so bara Kocka in Luka ter župnišče Sv. Martin in občina Duplek. Vsak od njih je pripravljen kupiti največ določeno količino vina po fiksni ceni, ne glede na sorto:

kupec	Kocka	Luka	župnišče	občina
cena za liter	1.0	1.1	1.5	1.8
največja količina v litrih	15000	5000	500	1000

Janez se je odločil, da bo vsako sorto prodal največ enemu kupcu, in sicer v maksimalni količini (če kupec ne kupi vsega vina iste sorte, ga Janez ohrani zase). Župan pravi, da občina drugega vina kot renškega rizlinga ne bo kupila. Bar Luka želi rumeni muškat, če bar Kocka dobi laški rizling. Pri Kocki so se dogovorili, da če občina in župnišče ne kupijo nič, tudi oni ne bodo kupili ničesar. Janezova žena pa vztraja, da če kupec  $A$  kupi sorto  $s_A$  in kupec  $B$  kupi sorto  $s_B$ , potem naj sorta  $s_C$  ostane doma ali jo kupi kupec  $C$  (za neke  $A, B, C$ ). Kako naj Janez proda vino, da bo čim več zaslužil?

Zapiši problem kot celoštevilski linearni program.

**Naloga 2.9.**

Vir: Vaje OR 16.3.2016

Napiši celoštevilski linearni program, ki poišče razdalje od danega vozlišča  $u$  v grafu  $G$ .



**Naloga 2.10.**

Vir: Vaje OR 16.3.2016

Napiši linearni program, ki modelira določanje kromatičnega števila grafa.

**Naloga 2.11.**

Vir: Vaje OR 16.3.2016

(problem trgovskega potnika)

Danih je  $n$  mest na zemljevidu. Strošek potovanja iz mesta  $i$  v mesto  $j$  je  $c_{ij}$  ( $1 \leq i, j \leq n$ ). Trgovski potnik želi obiskati vseh  $n$  mest, pri tem pa minimizirati strošek potovanja. Na smiseln način modeliraj opisani problem z linearnim programom.

**Naloga 2.12.**

Vir: Vaje OR 23.3.2016

S celoštevilskim linearnim programom modeliraj problem iskanja minimalnega vpetega drevesa v grafu.

**Naloga 2.13.**

Vir: Izpit OR 15.12.2016

Na oddelku za matematiko je zaposlenih  $n$  asistentov, ki jim moramo dodeliti vaje pri  $m$  predmetih. Za asistenta  $i$  ( $1 \leq i \leq n$ ) naj bosta  $a_i$  in  $b_i$  najmanjše in največje število ur, ki jih lahko izvaja, ter  $N_i \subseteq \{1, 2, \dots, m\}$  množica predmetov, ki jih ne želi izvajati. Za predmet  $j$  ( $1 \leq j \leq m$ ) naj bo  $c_j$  število skupin za vaje pri predmetu, ter  $u_j$  število ur vaj na skupino. Poleg tega vemo, da sta asistenta  $p$  in  $q$  skregana, zato pri nobenem predmetu ne smeta oba izvajati vaj.

Predmete želimo asistentom dodeliti tako, da bomo ob upoštevanju njihovih želja minimizirali največje število različnih predmetov, ki smo jih dodelili posameznemu asistentu.

Zapiši celoštevilski linearni program, ki modelira zgoraj opisani problem.

**Namig:** napiši program s spremenljivko  $t$ , ki je dopusten natanko tedaj, ko vsak asistent dobi največ  $t$  različnih predmetov, in potem minimiziraj  $t$ .

**Naloga 2.14.**

Vir: Izpit OR 31.1.2017

Imamo  $m$  opravil, ki jih želimo razporediti med  $n$  strojev. Vsak stroj lahko hkrati opravlja le eno opravilo, vsa opravila pa trajajo eno časovno enoto, neodvisno od stroja. Če stroj  $i$  ( $1 \leq i \leq n$ ) uporabimo za vsaj eno opravilo, plačamo ceno  $c_i$  (cena ostane enaka, če na istem stroju naredimo več opravil). Skupni stroški ne smejo preseči količine  $C$ . Dani sta še množici parov  $P$  in  $S$ , pri čemer  $(j, k) \in P$  pomeni, da mora biti opravilo  $j$  dokončano pred začetkom opravila  $k$ ,  $(j, k) \in S$  pa pomeni, da se opravila  $j$  in  $k$  ne smeta izvajati hkrati. Imamo še dodaten pogoj, ki zahteva, da je lahko v posamezni časovni enoti lahko aktivnih največ  $A$  strojev. Minimizirati želimo čas dokončanja zadnjega stroja.

Zapiši celoštevilski linearni program, ki modelira zgoraj opisani problem.

**Naloga 2.15.**

Vir: Izpit OR 15.3.2017

**(problem polnjenja košev)**

Imamo neskončno košev  $B_1, B_2, \dots$ , od katerih ima vsak velikost  $V > 0$ , in  $n$  predmetov pozitivnih velikosti  $s_1, s_2, \dots, s_n$ , od katerih je vsaka največ  $V$ . Predmete želimo postaviti v koše tako, da porabimo čim manj košev, pri čemer noben koš ne vsebuje predmetov skupne velikosti več kot  $V$ . Na primer, če  $V = 10$  ter  $s_1 = 4$ ,  $s_2 = 5$  in  $s_3 = 7$ , lahko to storimo z dvema košema, medtem ko z enim samim košem to ni mogoče. Zapiši celoštevilski linearni program, ki modelira opisani problem.

- (a) Zapiši celoštevilski linearni program, ki modelira zgornji problem.
- (b) Denimo, da imamo seznam  $L$  s pari predmetov, ki jih ne smemo postaviti v isti koš. Če imamo na primer  $L = \{(1, 2), (3, 4), (1, 4)\}$ , potem predmetov 1 in 2 ne moremo postaviti skupaj, prav tako ne predmetov 3 in 4 oziroma 1 in 4.

Dopolni zgornji celoštevilski linearni program tako, da bo vključeval te omejitve.

**Naloga 2.16.**

Vir: Izpit OR 10.7.2017

Za hitrejšo nalaganje posnetkov na YouTube se uporabljajo krajevni strežniki, do katerih lahko uporabniki na določeni lokaciji hitreje dostopajo kakor do glavnega strežnika, ki vsebuje vse posnetke. Vendar pa imajo krajevni strežniki omejen prostor, zato je potrebno ugotoviti, kateri posnetki naj se naložijo na katere krajevne strežnike.

Denimo, da imamo poleg glavnega strežnika še  $k$  krajevnih strežnikov,  $m$  internetnih ponudnikov in  $n$  posnetkov. Naj bo  $c_h$  ( $1 \leq h \leq k$ ) prostor v megabajtih, ki je na voljo na krajevni strežnik  $h$ . Z indeksom 0 označimo glavni strežnik – lahko torej predpostavljamo  $c_0 = \infty$ . Nadalje naj bo  $s_j$  ( $1 \leq j \leq n$ ) velikost posnetka  $j$ , prav tako v megabajtih,  $\ell_{hi}$  ( $0 \leq h \leq k$ ,  $1 \leq i \leq m$ ) latenca (zakasnitev pri prenosu v milisekundah) ponudnika  $i$  do strežnika  $h$ , in  $r_{ij}$  ( $1 \leq i \leq m$ ,  $1 \leq j \leq n$ ) število zahtevkov za posnetek  $j$ , ki jih pričakujejo od ponudnika  $i$ . Vsi parametri so cela števila.

Pri vsakem zahtevku bo posnetek poslan iz strežnika z najmanjšo latenco, ki vsebuje želeni posnetek. Lahko predpostavljamo, da ima za vsakega ponudnika glavni strežnik največjo latenco (torej  $\ell_{0i} \geq \ell_{hi}$  za vsaka  $1 \leq h \leq k$ ,  $1 \leq i \leq m$ ). Določiti želimo, katere posnetke naj naložimo na posamezen krajevni strežnik, da minimiziramo vsoto pričakovanih latenc pri vseh zahtevkih. Posamezen posnetek lahko seveda naložimo tudi na več krajevnih strežnikov, ali pa na nobenega (v tem primeru bo poslan iz glavnega strežnika).

Zapiši celoštevilski linearni program, ki modelira zgoraj opisani problem.

**Naloga 2.17.**

Vir: Izpit OR 29.8.2017

Distributer ima  $A$  zabojev avokadov in  $B$  zabojev banan, ki jih bo prodal  $n$  trgovcem. Trгоvec  $i$  ( $1 \leq i \leq n$ ) plača  $a_i$  evrov za zaboj avokadov in  $b_i$  evrov za zaboj banan, skupaj pa bo porabil največ  $c_i$  evrov. Distributer bo zaboje dostavil s tovornjaki, v katerih je lahko največ  $K$  zabojev (ne glede na vsebino). Če nekemu trgovcu dostavi  $t$  zabojev, bo torej opravljenih  $\lceil t/K \rceil$  voženj. Vsaka vožnja do trgovca  $i$  (ne glede na to, koliko je poln tovornjak) ga stane  $d_i$  evrov. Poleg tega bo trgovec  $p$  kupil samo banane ali samo avokade, trgovec  $q$  pa bo kupil vsaj en zaboj avokadov, če bo tudi trgovec  $r$  kupil vsaj en zaboj avokadov. Distributer želi zaboje razdeliti med trgovce tako, da bo maksimiziral svoj dobiček – torej vsoto cen, ki jih plačajo trgovci, zmanjšano za stroške dostave. Lahko predpostavljaš, da so vse cene pozitivne.

Zapiši celoštevilski linearni program, ki modelira zgoraj opisani problem.

**Naloga 2.18.**

Vir: Kolokvij OR 23.4.2018

Pri izvedbi projekta bo potrebno narediti  $n$  nalog, ki jih bomo dodelili delavcem. Vsako nalogo bo opravil natanko en delavec, vsak delavec pa lahko hkrati izvaja samo eno nalogo. Naj bo  $t_i \in \mathbb{N}$  število časovnih enot, ki ga posamezen delavec potrebuje za dokončanje naloge  $i$  ( $1 \leq i \leq n$ ). Vsaka naloga mora biti opravljena v enem kosu (če torej začnemo z nalogo  $i$  v času  $s$ , bo ta dokončana v času  $s + t_i$ , brez možnosti prekinitve in kasnejšega dokončanja). Celoten projekt mora biti dokončan v času  $T$ . Dana je še množica parov  $S$ , kjer  $(i, j) \in S$  pomeni, da se naloga  $j$  ne sme začeti, preden se zaključi naloga  $i$  (lahko se pa  $j$  začne izvajati v trenutku, ko se  $i$  konča).

Delavce bomo najeli preko podjetja za posredovanje dela, to pa nam zaračuna fiksno ceno na najetega delavca (za ustrezna plačila delavcem bodo tako poskrbeli oni). Minimizarati želimo torej število najetih delavcev. Zapiši celoštevilski linearni program, ki modelira zgoraj opisani problem.

**Naloga 2.19.**

Vir: Izpit OR 11.6.2018

Avtobusno podjetje želi uvesti novo avtobusno linijo. Dan je neusmerjen enostaven graf  $G = (V, E)$ , katerega vozlišča predstavljajo postajališča, povezave pa ceste med njimi. Za vsako povezavo  $uv \in E$  poznamo še čas  $c_{uv} \in \mathbb{N}$ , v katerem avtobus pride od  $u$  do  $v$ .

Dan je še seznam postajališč  $p_1, p_2, \dots, p_n \in V$ , ki jih linija mora obiskati v tem vrstnem redu – začeti mora v  $p_1$  in končati v  $p_n$ , na poti med dvema postajališčema iz seznama pa lahko obišče tudi druga postajališča. Linija lahko vsako postajališče obišče največ enkrat (ko doseže končno postajališče, se vrne po isti poti do začetnega). Skupno trajanje vožnje (v eno smer) je lahko največ  $T$ . Podjetje želi določiti linijo tako, da bo obiskala čim več postaj.

Zapiši celoštevilski linearni program, ki modelira zgoraj opisani problem.

**Namig:** poskrbi, da linija nima ciklov.

**Naloga 2.20.**

Vir: Izpit OR 5.7.2018

Nogometni klub pred novo sezono prenavlja svoj igralski kader. Naj bo  $A$  množica trenutnih igralcev kluba,  $B$  pa množica igralcev, za katere se v klubu zanimajo ( $A$  in  $B$  sta disjunktni množici). Za vsakega igralca  $i \in A$  imajo s strani kluba  $j$  ( $1 \leq j \leq n$ ) ponudbo v višini  $p_{ij} \in \mathbb{N}$  (če se klub  $j$  ne zanima za igralca  $i$ , lahko predpostaviš  $p_{ij} = 0$ ), za vsakega igralca  $i \in B$  pa bodo morali plačati odkupnino  $r_i \in \mathbb{N}$ , če ga hočejo pripeljati v klub. Vsakega igralca  $i \in A$  lahko seveda prodamo kvečjemu enem klubu. Skupni stroški trgovanja (tj., razlika stroškov nakupov in dobička od odprodaj) ne smejo preseči  $S$ , število igralcev v klubu pa mora ostati enako kot pred trgovanjem.

Za vsakega igralca  $i \in A \cup B$  poznamo njegov količnik kvalitete  $q_i$ , vsak pa pripada natanko eni izmed množic  $G$  (vratarji),  $D$  (branilci),  $M$  (vezni igralci) in  $F$  (napadalci). Število igralcev kluba v vsaki izmed teh množic se lahko spremeni (poveča ali zmanjša) za največ 1. Poleg tega lahko igralca  $a, b \in A$  prodamo le istemu klubu – ali pa oba igralca ostaneta v klubu. Doseči želimo, da bo kvaliteta kluba čim večja – maksimizirati želimo torej vsoto količnikov  $q_i$  za igralce v klubu.

Zapiši celoštevilski linearni program, ki modelira zgoraj opisani problem.

**Naloga 2.21.**

Vir: Kolokvij OR 24.1.2011

Podjetje dobiva po pošti čeke iz celotne Evrope. Čas potovanja čeka je odvisen od kraja, od koder je bil ček poslan, in od kraja, kamor ček potuje. Na primer, ček, poslan iz vzhodne Evrope v Berlin, v povprečju potuje 5 dni – podjetje mora torej čakati 5 dni, preden lahko ček unovči in razpolaga z denarjem. Iz severne Evrope je v povprečju dnevno poslanih za 70 000€ čekov, iz zahodne Evrope za 50 000€, iz vzhodne 60 000€, iz južne pa 40 000€. Podjetje želi obračunavati čeke kar se da hitro, saj izgubi 20% vrednosti čeka na letni ravni. Podjetje lahko postavi podružnice (in s tem poštne predale) v Londonu, Berlinu, Budimpešti in/ali Madridu. Strošek vzdrževanja ene podružnice znaša 50 000€ letno. Časi potovanja čekov so razvidni iz spodnje tabele:

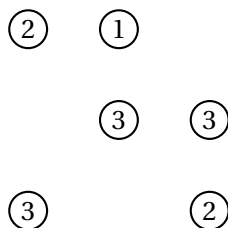
	London	Berlin	Budimpešta	Madrid
Severna Evropa	2	6	8	6
Zahodna Evropa	6	2	5	5
Vzhodna Evropa	8	5	2	5
Južna Evropa	8	5	5	2

Primer: če severna Evropa pošilja čeke v Budimpešto, bo v obtoku v povprečju za 560 000€ ( $= 8 \cdot 70\,000\text{€}$ ) čekov, kar z 20% izgubo pomeni 112 000€ izgube na letni ravni. Zanima nas, kje naj podjetje postavi podružnice, da bodo stroški čim manjši. Zastavi nalogo kot celoštevilski linearni program.

**Naloga 2.22.**

Vir: Izpit OR 9.2.2011

Selektor košarkaške reprezentance bi rad sestavil petčlansko začetno postavo, ki bo imela povprečno višino kar se da visoko. Na voljo ima sledeče igralce:



Slika 1: Otoki za nalogo 2.23.

Igralec	Višina	Pozicija
Anže	213	Center
Borut	208	Center
Ciril	203	Krilo
David	192	Krilo
Emil	196	Krilo
Filip	197	Obramba
Gorazd	200	Obramba
Hugo	195	Obramba

Pri izbiri mora selektor upoštevati naslednje pogoje:

- v začetni postavi morajo biti zastopane vse tri pozicije,
- v rezervi mora biti bodisi Ciril bodisi Filip, ne pa oba,
- v začetni postavi je lahko največ en center,
- če začne Borut ali David, mora Hugo ostati v rezervi.

Formuliraj problem kot celoštevilski linearni program.

### Naloga 2.23.

Vir: Izpit OR 28.6.2011

Otoke (vozlišča) na sliki 1 je potrebno povezati z mostovi po sledečih pravilih:

- mostovi potekajo samo v vodoravni ali navpični smeri,
- med dvema otokoma smeta biti največ dva mostova,
- številka na otoku pove, koliko mostov naj vodi na ta otok.

Formuliraj problem kot celoštevilski linearni program.

**Naloga 2.24.**

Vir: Izpit OR 9.7.2012

Družina Vinar iz Vurberka že več kot 200 let prideluje vino. Njihovi začetki segajo še v čas Habsburžanov, ko je Janez Vinar od grofa Herbersteina v dar prejel najlepši vinograd na Vurberku. Janez je kmalu začel dobivati naročila od okoliških gostincev in vsako leto je moral razmisliti, kako bo razdelil svoj pridelek med porabnike. Danes njegov daljni potomec (prav tako Janez) nadaljuje s tradicijo.

Če je letina dobra (predpostavimo, da je vsako leto dobra letina), pridelava Janez 2000 litrov rumenega muškata, 10 000 litrov laškega rizlinga in 5 000 litrov renškega rizlinga. Njegovi standardni odkupniki sta bara Kocka in Luka ter župnišče Sv. Martin in občina Duplek. V spodnji tabeli je prikazano, koliko litrov vina je vsak pripravljen (največ) kupiti.

kupec	Kocka	Luka	občina	župnišče
količina v litrih	15 000	5 000	1 000	500

Teče drugo leto Janezovega vinogradništva<sup>1</sup>. To leto se je odločil, da bo vino prodajal le v flaškonih po 10 litrov in postavil fiksne cene, vidne v spodnji tabeli.

sorta	rumeni muškat	laški rizling	renski rizling
cena za flaškon	12€	8€	15€

A ne gre brez omejitev. Vsak bar zahteva, da Janezu skupno plača največ toliko, kot mu plačata skupaj župnišče in občina. V župnišču želijo 500 litrov vina, ki naj ne bo laški rizling. V občini želijo vsaj 500 litrov renškega rizlinga, drugih sort pa sploh ne bodo kupovali. Nazadnje se je oglasila še Janezova žena, ki želi ohraniti doma vsaj toliko sorte A, kot je Janez proda kupcema B in C skupaj. Toda parametrov v ženini zahtevi nam Janez ne želi zaupati. Napiši tak celoštevilski linearni program (tako da lahko Janez vstavi ustrezne ženine parametre), ki bo Janezu drugo leto pomagal pri prodaji vina, tako da bo njegov zaslužek kar največji.

**Naloga 2.25.**

Vir: Izpit OR 4.9.2012

Minolovec je stara Microsoftova računalniška igrice. Dano je minsko polje v obliki kariraste mreže dimenzij  $n \times m$ , v katerem se nahaja  $p$  min. Slika 2 predstavlja delno odprto minsko polje, ki vsebuje  $p = 40$  min.

V vsaki nedokončani igri je neka celica bodisi odprta bodisi zaprta. V nekatereh zaprtih celicah se nahajajo mine. V odprtih celicah so številke od 0 do 8 (številke 0 na sliki 2 niso prikazane), ki povedo, koliko min se nahaja v okolici te celice (tj., na sosednjih 8 celicah). Za razliko od Minolovca na računalniku lahko pri nas tudi celica s številom 0 meji na zaprto celico.

Delno odprto minsko polje (levo) ima pri  $p = 7$  več možnih rešitev. Dve sta prikazani na desni strani:

<sup>1</sup> Glej nalogo 2.8.

2	3	4		
0	0	2		
0	0	1		

2	3	4		
0	0	2		
0	0	1		

2	3	4		
0	0	2		
0	0	1		

Naslednje delno odprto minsko polje pa ni veljavno (ne glede na  $p$ ):

2	2	2	
0	0	2	

- (a) Opiši celoštevilski linearni program, ki bo povedal, ali je neko tako delno odprto minsko polje veljavno, tj., ali je možno  $p$  min postaviti na polje tako, da bo v okolici vsake odprte celice ustrezno število min.
- (b) Ali bi lahko s pomočjo celoštevilskega linearnega programa ugotovil, kolikšno je največje možno število min pri nekem delno odprtem polju?  
**Namig:** morda zadošča, če malenkost popraviš program iz prejšnje točke.
- (c) Kako bi s pomočjo zgornjega celoštevilskega linearnega programa poiskal celice, ki nujno vsebujejo mino (tj., pri vseh možnih rešitvah se v tistih celicah nahajajo mine)?

Pri pisanju celoštevilskih linearnih programov si lahko pomagaš z oznakami:

- $\mathcal{O}$ : množica vseh odprtih celic,
- $\mathcal{Z}$ : množica vseh zaprtih celic,
- $N(s)$ : množica sosedov celice  $s$ .

### Naloga 2.26.

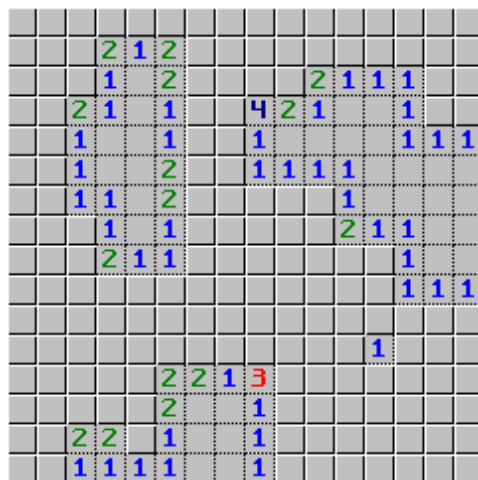
Vir: Kolokvij OR 17.4.2014

Danih je  $n$  praznih škatel z volumni  $v_1, v_2, \dots, v_n$ . Dana so naslednja pravila pospravljanja škatel:

- več manjših škatel lahko zložimo v večjo, če njihov skupni volumen ne presežemo volumna večje škatle, in
- škatle, ki ima znotraj sebe že kakšno škatlo, ne smemo več postaviti znotraj večje škatle.

Zanima nas takšno pospravljanje škatel, da bo skupen volumen škatel, ki ostanejo, čim manjši. Formuliraj nalogo kot celoštevilski linearni program.

Na primer, če imamo škatle z volumni 2, 3, 6, 8, 9, potem je ena možna rešitev (ne nujno optimalna!), da zložimo škatli z volumnoma 2 in 3 znotraj škatle z volumnom 6 ter škatlo z volumnom 8 znotraj največje škatle. Ta razporeditev ima skupni volumen  $6 + 9 = 15$ .



Slika 2: Primer delno odprtega polja za nalogo 2.25.

### Naloga 2.27.

Vir: Izpit OR 14.6.2013

(problem kombinatorične dražbe)

Pohorski škratje so se v osrednji sobani svojega podzemnega kraljestva zbrali na dražbi. Vodja dražbe prodaja predmete  $A = \{1, 2, \dots, k\}$ , ki so jih škratje uspeli "pridobiti" od ljudi. Ponudbe so pari  $(B_i, c_i)$ , kjer je  $B_i \subseteq A$  in je  $c_i > 0$  cena ponudbe. Škrat, ki je ponudil  $j$ -to ponudbo, je torej pripravljen za predmete  $B_i$  plačati  $c_i$  (če bi dobil le nekaj od predmetov iz  $B_i$ , ne bi plačal nič). Vodja dražbe prejme  $k$  ponudb  $\{(B_i, c_i)\}_{i=1}^k$ , od vsakega škrate največ eno. Predmete želi razdeliti med ponudnike, da bo iztržil kar največ.

- Predstavi problem kot celoštevilski linearni program!
- Kako se celoštevilski linearni program spremeni, če dodamo naslednja pogoja?
  - Če je ponudba 1 uspešna, potem mora biti uspešna tudi ponudba 2 ali ponudba 3.
  - Če so uspešne vse ponudbe s sodim indeksom, potem morajo biti neuspešne vsaj tri ponudbe z lihim indeksom.

### Naloga 2.28.

Vir: Izpit OR 5.6.2014

Dana je množica števil  $A = \{a_1, a_2, \dots, a_n\}$ . Iščemo podmnožico  $L \subseteq A$ , za katero doseže izraz

$$\sum_{a_i \in L} (a_i)^2$$

maksimalno vrednost, pri čemer morajo biti izpolnjeni pogoji:



- če  $a_1 \in L$ , potem  $a_2 \notin L$ ,
- če  $a_3 \in L$  in  $a_5 \in L$ , potem  $a_7 \in L$ ,
- če  $|L| \geq 5$ , potem  $a_4 \notin L$ , ter
- v  $L$  je več pozitivnih kot negativnih števil.

Problem formuliraj kot celoštevilski linearni program.

**Naloga 2.29.**

Vir: Izpit OR 24.6.2014

Dan imamo tovor, ki ga sestavlja množica  $n$  predmetov s celoštevilskimi masami  $m_1, m_2, \dots, m_n$ . Tovor želimo razdeliti na 5 tovornjakov tako, da bo obremenitev najbolj obremenjenega tovornjaka čim manjša. Problem formuliraj kot celoštevilski linearni program.

Dodatno formuliraj še naslednja pogoja:

- Tovor na prvem tovornjaku ne sme presegati mase  $M$ .
- Na vsaj enem tovornjaku tovor ne sme presegati mase  $M$ .

**Naloga 2.30.**

Vir: Izpit OR 25.8.2014

Naj bodo v vrsto dolžine  $n$  razporejena števila  $1, 2, \dots, n$  (v poljubnem vrstnem redu). Pri izbiri  $i$ -tega mesta dobimo število točk, ki je enako minimumu med vsoto števil strogo levo od  $i$ -tega in vsoto števil strogo desno od  $i$ -tega mesta.

- Določi vrstni red števil, da bo povprečni dobiček pri naključnem izbiranju mest čim večji. Problem formuliraj kot celoštevilski linearni program.
- Formuliraj še pogoj, ki pravi, da morata biti števili 1 in 2 sosednji.

**Naloga 2.31.**

Vir: Izpit OR 19.5.2015

Podjetje Nijke proizvaja moške copate. Po svetu ima več tovarn, vsaka je na stopnji razvoja 1, 2, 3 ali 4. Koliko dobička prinese tovarna na posamezni stopnji razvoja v posamezni državi ter koliko tovarn na posamezni stopnji razvoja ima trenutno vsaka država, je razvidno iz tabele 1.

V naslednjem letu lahko nadgradimo do 15 tovarn, v vsaki državi največ 3. Vsako tovarno lahko nadgradimo največ za eno stopnjo. Dodatno moramo upoštevati še naslednje omejitve.

- Slovenija mora imeti več tovarn stopnje 4 kot Hrvaška in več tovarn stopnje 3 kot Italija.
- Število nadgradenj tovarn v EU (države nad črto tabeli 1) mora biti vsaj tolikšno kot število nadgradenj tovarn izven EU.

država	dobiček				število tovarn			
	st. 1	st. 2	st. 3	st. 4	st. 1	st. 2	st. 3	st. 4
Avstrija	3	4	7	10	0	0	0	5
Slovenija	3	3	5	8	0	1	2	1
Italija	4	4	6	8	1	1	1	1
Madžarska	5	7	8	10	0	1	2	1
Hrvaška	3	4	5	7	1	2	1	0
Iran	6	10	10	10	3	0	4	10
Irak	2	4	4	6	5	3	0	0
Afganistan	6	6	5	2	20	0	0	0
Turkmenistan	6	7	7	6	10	5	5	0
Pakistan	6	7	7	7	10	7	2	0
Armenija	6	7	9	10	3	0	1	0
Azerbajdžan	6	7	9	8	0	1	0	0

Tabela 1: Podatki za nalogo 2.31.

- Če Azerbajdžan nadgradi svojo tovarno in Armenija ne nadgradi tovarne stopnje 3, potem mora Armenija nadgraditi vse tri tovarne stopnje 1.
- Iran mora imeti vsaj toliko tovarn stopnje 4 kot vse države EU skupaj.

Napiši celoštevilski linearni program, ki upošteva opisane omejitve in pove, kje in koliko tovarn moramo nadgraditi, da bomo imeli kar največji dobiček.

### Naloga 2.32.

Vir: Izpit OR 24.6.2015

Krt Črnko že vrsto let domuje pod zelenim travnikom. Zgradil si je že pravo kraljestvo rovov, ki ga lahko predstavimo z grafom  $G = (V, E)$ : vozlišča grafa so križišča rovov, povezave pa so rovi. Črnkov travnik je pred kratkim kupila velika korporacija in se odločila, da bo Črnka pregnala s plinom. A slednji se ne bo pustil kar tako odgnati in namerava na križišča svojega kraljestva postaviti zaščitne maske.

- (a) Pomagaj Črnku tako, da napišeš celoštevilski linearni program, ki za graf  $G$  določi križišča, kamor naj Črnko postavi maske, da jih bo porabil čim manj in da bo imelo vsako križišče masko, ali pa bo masko imelo sosednje križišče.
- (b) Kako se CLP spremeni, če se Črnko odloči:
- da bo masko gotovo postavil v križišče  $s \in V$ , ne pa v križišče  $t \in V$ ?
  - da bosta največ dve izmed križišč  $s_1, s_2, \dots, s_k \in V$  imeli masko?
  - da bo v primeru, ko bo v križišču  $v_1 \in V$  maska, v križišču  $v_2 \in V$  pa ne, postavil masko v križišče  $v_3 \in V$  ali v križišče  $v_4 \in V$ ?

**Naloga 2.33.**

Vir: Izpit OR 28.8.2015

Poslovnež Kopalec je lastnik velike evropske verige bazenov. Trenutno ima v lasti 20 bazenov, ki jih označimo kar s števili od 1 do 20. Vsi razen bazenov 1, 2 in 3 so v “osnovnem” stanju, tj., okoli njih ni zgrajen noben večji kompleks. Bazeni 1, 2 in 3 so pa v “nadgrajenem” stanju, kar pomeni, da je okoli njih zgrajen večji kompleks, seveda v lasti gospoda Kopalca.

V naslednjem 5-letnem obdobju bo gospod Kopalec nadgradil nekaj svojih bazenov, ki so trenutno v osnovnem stanju, nekaj bazenov pa bo prodal. Napiši celoštevilski linearni program, ki bo gospodu Kopalcu pomagal maksimizirati število nadgrajenih bazenov po koncu 5-letnega obdobja. Pri tem upoštevaj naslednje omejitve:

- Gospod Kopalec bo nadgradil največ dvakrat toliko bazenov, kot jih bo prodal.
- Gospod Kopalec ne bo prodal bazenov, ki so nadgrajeni ali jih bo nadgradil.
- Če bo gospod Kopalec nadgradil bazen 8, potem bo nadgradil tudi bazena 14 in 11, ne bo pa prodal bazena 15.
- Gospod Kopalec ne bo nadgradil bazenov 4, 5, 6 in 7, če ne bo prodal bazenov 15, 16, 17, 18, 19 in 20 (tj., če ne proda nobenega od bazenov 15–20, potem ne bo nadgradil nobenega od bazenov 4–7).
- Za neka  $a$  in  $b$  ( $4 \leq a, b \leq 20$ ), ki ju želi gospod Kopalec določiti kasneje, bo prodal bazen  $a$  in nadgradil bazen  $b$ .
- Ženi gospoda Kopalca je všeč bazen  $c$  ( $1 \leq c \leq 20$ ) tak, kot je. Zato ga gospod Kopalec ne bo prodajal, niti nadgrajeval.

**Naloga 2.34.**

Vir: Izpit OR 12.5.2016

Dan je enostaven graf  $G = (V, E)$  in funkcija uteži na povezavah  $t : E \rightarrow \mathbb{R}_{>0}$ . Podmnožici  $M \subseteq E$  pravimo *prirejanje*, če za vsaki različni povezavi  $e, e' \in M$  velja  $e \cap e' = \emptyset$  (tj., nobeni dve povezavi iz  $M$  nimata skupnega krajišča). Prirejanje  $M$  je *maksimalno*, če ne obstaja tako prirejanje  $M'$ , da velja  $M \subset M'$ . *Teža* prirejanja  $M$  je vsota uteži njegovih povezav, torej  $\sum_{e \in M} t(e)$ .

Formuliraj celoštevilski linearni program, ki za dani vhodni graf  $G$  poišče maksimalno prirejanje najmanjše teže.

**Naloga 2.35.**

Vir: Izpit OR 24.5.2016

Dan je graf  $G = (V, E)$ . Razbitju  $V_1, V_2, \dots, V_k$  množice vozlišč  $V$  pravimo *k-barvno pakiranje*, če velja  $d_G(u, v) > i$  za vsaki različni vozlišči  $u, v \in V_i$  ( $1 \leq i \leq k$ ) – tj., poljubni dve vozlišči iz  $V_i$  sta v grafu  $G$  oddaljeni za več kot  $i$ . Minimalnemu številu  $k$ , za katero obstaja *k-barvno pakiranje*, pravimo *mavrično pakirno število*.

in ga označimo s  $\chi_p(G)$ . Napiši celoštevilski linearni program, ki za dani graf  $G$  izračuna  $\chi_p(G)$ .

**Naloga 2.36.**

Vir: Izpit OR 29.8.2016

Naj bo  $G = (V, E)$  enostaven graf. Barvanju vozlišč grafa  $c_k : V \rightarrow \{1, 2, \dots, k\}$  pravimo *k-uravnoreženo barvanje*, če ima naslednji lastnosti:

- Če sta vozlišči  $u$  in  $v$  sosedni v  $G$ , potem velja  $c_k(u) \neq c_k(v)$ .
- Za vsaka  $i, j \in \{1, 2, \dots, k\}$  velja

$$\left| |\{v \in V \mid c_k(v) = i\}| - |\{v \in V \mid c_k(v) = j\}| \right| \leq 1$$

– tj., število vozlišč, pobarvanih z barvama  $i$  in  $j$ , se razlikuje za največ 1.

Napiši celoštevilski linearni program, ki ima dopustno rešitev natanko tedaj, ko za vhodni graf  $G$  obstaja *k-uravnoreženo barvanje*. Predpostaviš lahko, da je tudi  $k$  vhodni podatek. Koliko pogojev in koliko spremenljivk ima dobljeni celoštevilski linearni program?

**Naloga 2.37.**

Vir: Teorija OR 5.7.2013

Zapiši naslednji problem kot celoštevilski linearni program.

$$\begin{aligned} & \min f(x) + g(y) + h(z) \quad \text{p.p.} \\ & f(x) = \begin{cases} 10 & \text{če } x = 0, \text{ in} \\ 10 + 5x & \text{če } x > 0 \end{cases} \\ & g(y) = \begin{cases} 10 & \text{če } y = 0, \text{ in} \\ 5 + 5y & \text{če } y > 0 \end{cases} \\ & h(z) = \begin{cases} 8 & \text{če } z = 0, \text{ in} \\ 10 + 4z & \text{če } z > 0 \end{cases} \\ & x + y + z \geq 100 \\ & (x \geq 5 \wedge y \geq 5) \vee (x \geq 5 \wedge z \geq 5) \vee (y \geq 5 \wedge z \geq 5) \\ & x, y, z \geq 0, \quad x, y, z \in \mathbb{Z} \end{aligned}$$

**Naloga 2.38.**

Vir: Teorija OR 12.7.2017

Zapiši naslednji problem kot celoštevilski linearni program.

$$\begin{aligned} & \min f_1(x_1) + f_2(x_2) \quad \text{p.p.} \\ & f_1(x_1) = \begin{cases} 0 & \text{če } x_1 = 0, \text{ in} \\ 10 + 5x_1 & \text{če } x_1 > 0 \end{cases} \end{aligned}$$

1	2		
		2	1
	4		
3			

Slika 3: Primer sudokuja za nalogo 2.39.

$$f_2(x_2) = \begin{cases} 0 & \text{če } x_2 = 0, \text{ in} \\ 20 + 4x_2 & \text{če } x_2 > 0 \end{cases}$$

$$100x_1 + 150x_2 \geq 10000$$

$$x_1, x_2 \geq 0$$

**Naloga 2.39.**

Vir: Teorija OR 5.9.2017

Obravnavamo manjši sudoku velikosti  $4 \times 4$  s slike 3. Vanj vpisujemo števila med 1 in 4. Vsako število se pojavi natanko enkrat v vsaki vrstici, vsakem stolpcu in vsakem od 4 kvadratov velikosti  $2 \times 2$ , omejenih z debelejšo črto. Opiši celoštevilski linearni program za reševanje takega problema oz. za določanje, da rešitev ne obstaja. Koliko spremenljivk in pogojev ima linearni program?

**Naloga 2.40.**

Vir: Kolokvij OR 19.4.2019

Na turnirju poleg tebe sodeluje še  $n$  igralcev, pri čemer se bo vsak igralec soočil z vsakim drugim igralcem v enem dvoboju. Vsak igralec na začetku dobi  $M$  žetonov, za katere se (po začetni fazi zbiranja informacij) vnaprej odloči, kako jih bo razdelil med dvoboje. V dvoboju zmaga igralec z večjim številom vloženih žetonov. Če oba igralca vložita enako število žetonov, je izid dvoboja izenačen. Zmagovalec dvoboja dobi 3 točke, poraženec 0 točk, v primeru izenačenega izida pa vsak dvobojevalec dobi 1 točko. Cilj vsakega igralca je zbrati čim večje število točk.

- (a) Denimo, da za vsakega igralca izveš, kako namerava igrati. Naj bo torej  $c_i$  število žetonov, ki jih bo  $i$ -ti nasprotnik ( $1 \leq i \leq n$ ) vložil v dvoboj proti tebi. Svojo strategijo želiš načrtovati tako, da bi dosegel čim večje število točk. Zapiši celoštevilski linearni program, ki modelira ta problem.
- (b) Celoštevilkemu linearnemu programu iz prejšnje točke dodaj omejitve, ki modelirajo sledeče pogoje:
- Proti nobenemu od nasprotnikov iz množice  $A$  ne smeš doseči izenačenega izida.

- Izgubiš lahko kvečjemu proti dvema nasprotnikoma iz množice  $B$ .
- Če izgubiš proti nasprotniku  $u$ , moraš proti nasprotnikoma  $v$  in  $w$  doseči vsaj 4 točke.
- Več kot  $t$  žetonov lahko vložiš v največ  $k$  dvobojev.
- Izgubiti moraš vsaj  $\ell$  dvobojev.

**Naloga 2.41.**

Vir: Izpit OR 3.6.2019

V trgovini z oblačili sestavljajo ponudbo za poletno sezono. Odločijo se lahko za nakup kolekcij  $n$  različnih proizvajalcev, pri čemer za posamezen izvod kolekcije  $i$ -tega proizvajalca ( $1 \leq i \leq n$ ) plačajo ceno  $c_i$  evrov, od nje pa si obetajo zaslužek  $d_i$  evrov. Kolekcijo  $i$ -tega proizvajalca lahko kupijo v največ  $k_i$  izvodih (vsak izvod je nedeljiva enota), z njeno prisotnostjo v ponudbi (ne glede na količino) pa si obetajo povečano vidnost in s tem dodaten zaslužek  $e_i$  evrov (npr. iz prodaje stalne ponudbe in starih zalog).

Za nakup kolekcij imajo na voljo  $C$  evrov, iz marketinških razlogov pa želijo kupiti kolekcije največ  $K$  različnih proizvajalcev. Poleg tega proizvajalec  $p$  zahteva, da če kupijo kak izvod njegove kolekcije, je morajo kupiti v vsaj toliko izvodih kot kolekciji proizvajalcev  $q$  in  $r$  skupaj (če pa ne kupijo nobenega izvoda kolekcije proizvajalca  $p$ , te omejitve ni). V trgovini želijo maksimizirati pričakovani dobiček (tj., razliko zasluga od prodaje s stroški nakupa).

Zapiši celoštevilski linearni program, ki modelira zgoraj opisani problem.

**Naloga 2.42.**

Vir: Izpit OR 27.8.2019

Glasbeni festival bo trajal  $m$  dni, vsak dan pa bo nastopilo  $k$  glasbenih skupin. Organizator izbira nastopajoče izmed  $n$  glasbenih skupin (kjer je  $n > km$ ). Na voljo ima  $M$  evrov za plačilo glasbenikom, pri čemer  $i$ -ta skupina ( $1 \leq i \leq n$ ) računa  $m_i$  evrov za nastop, pripeljala pa bo  $a_{ij}$  obiskovalcev, če bo nastopila  $j$ -ta po vrsti v dnevu ( $1 \leq j \leq k$ ). Vsaka skupina lahko seveda nastopi samo enkrat, prav tako naenkrat nastopa samo ena skupina. V posameznem dnevu se skupine zvrstijo glede na njihovo ceno (tj., zadnja nastopa skupina, ki največ računa), organizator pa želi zagotoviti, da bo vsak dan festivala prišlo vsaj toliko ljudi kot prejšnji dan.

Skupina  $r$  pogojuje svoj nastop s tem, da skupina  $s$  ne nastopa na festivalu. Skupina  $t$  bo nastopila samo kot glavna skupina (tj., zadnja v posameznem dnevu), razen če nastopi neposredno pred skupino  $u$  (lahko predpostaviš, da velja  $m_t \leq m_u$ ) – ali pa sploh ne bo nastopila. Skupini  $v$  in  $w$  ne smeta nastopati na isti dan, saj se njuni oboževalci med seboj ne marajo.

Organizator želi določiti spored tako, da bo na festival prišlo čim več ljudi. Zapiši celoštevilski linearni program, ki modelira opisani problem.

**Naloga 2.43.**

Vir: Izpit OR 22.6.2020

Iz kriznih žarišč COVID-19 po svetu se na Kitajsko vrača  $n$  strokovnjakov, ki jih želijo oblasti razporediti po  $m$  karantenskih centrih s kapacitetami  $k_1, k_2, \dots, k_m$ , pri čemer velja  $\sum_{h=1}^m k_h \geq n$ . S pomočjo naprednih tehnologij so za vsak par strokovnjakov  $i, j \in \{1, 2, \dots, n\}$  izračunali verjetnost  $p_{ij} \in [0, 1]$ , da je prišlo do prenosa virusa med  $i$  in  $j$  (lahko predpostaviš, da za vsaka  $i, j$  velja  $p_{ij} = p_{ji}$  ter  $p_{ii} = 0$ ). Poiskati želijo tako razporeditev, da bo vsota vrednosti  $p_{ij}$  vseh parov  $(i, j)$ , ki bodo nastanjeni v istem karantenskem centru, čim manjša.

Pri tem imajo nekaj omejitev. Identificirali so množico  $A$  strokovnjakov, ki so se nahajali na območjih, kjer je prišlo do mutacije virusa – ti ne smejo biti nastanjeni skupaj s strokovnjaki izven množice  $A$  (lahko so pa člani množice  $A$  nastanjeni po različnih centrih). Poleg tega so sestavili množico  $B$  parov strokovnjakov v bližnjem sorodstvu – za vsak par  $(i, j) \in B$  velja, da mora biti nastanjen v istem centru.

Zapiši celoštevilski linearni program, ki modelira opisani problem.

**Namig:** uporabi spremenljivke, ki za par oseb povedo, ali se nastanita v nekem centru.

**Naloga 2.44.**

Vir: Kolokvij OR 12.4.2021

V večjem podjetju so se odločili za nabavo licenc za programsko opremo, in sicer bodo za vsakega zaposlenega iz množice  $Z$  nabavili natanko eno licenco. Razvijalec svojo programsko opremo ponuja v različicah  $1, 2, \dots, n$  z različnimi nabori funkcionalnosti. V podjetju so identificirali množico  $F$  funkcionalnosti, ki jih zanimajo, ter določili vrednosti  $p_{zf}$  in  $q_{fi}$  ( $z \in Z, f \in F, 1 \leq i \leq n$ ), ki povedo sledeče:

$$p_{zf} = \begin{cases} 1 & \text{če zaposleni } z \text{ potrebuje funkcionalnost } f, \text{ in} \\ 0 & \text{sicer,} \end{cases}$$

ter

$$q_{fi} = \begin{cases} 1 & \text{če različica } i \text{ ponuja funkcionalnost } f, \text{ in} \\ 0 & \text{sicer.} \end{cases}$$

Cena, ki jo v podjetju plačajo za posamezno licenco, je odvisna od števila licenc, ki jih kupijo za posamezno različico programske opreme: za posamezno licenco za različico  $i$  plačajo ceno  $c_{ij}$ , če kupijo vsaj  $r_j$  takih licenc ( $1 \leq j \leq m$ ). Lahko predpostaviš, da velja  $1 = r_1 < r_2 < \dots < r_m$  in  $c_{i1} \geq c_{i2} \geq \dots \geq c_{im} > 0$  ( $1 \leq i \leq n$ ).

- (a) Zapiši celoštevilski linearni program, ki modelira iskanje najcenejšega izbora licenc, tako da bo vsak zaposleni dobil različico programske opreme, ki ponuja vse funkcionalnosti, ki jih potrebuje.

**Namig:** uvedi spremenljivke, ki za vsakega zaposlenega, različico in ceno povedo, če naj v podjetju kupijo tako licenco.

- (b) Celoštevilskemu linearnemu programu iz točke (a) dodaj omejitve za sledeče dodatne pogoje.

- Zaposlena  $a, b \in Z$  morata dobiti licenco za enako različico programske opreme.
- Vsako funkcionalnost, ki jo ima na voljo katerikoli zaposleni, mora v svoji različici programske opreme imeti na voljo tudi direktor  $d \in Z$ .
- Če ima katerikoli zaposleni licenco za tako različico, ki ponuja funkcionalnost  $e \in F$ , potem mora funkcionalnost  $e$  biti na voljo vsem zaposlenim.

#### Naloga 2.45.

Vir: Izpit OR 3.6.2021

Igor je zadnje mesece pridno zbiral kupone, s katerimi lahko pridobi različne popuste v bližnji trgovini. Zanima ga, koliko kosov posameznega artikla naj kupi in katere kupone naj unovči, da bo zadostil svojim potrebam in da bo nakup čim cenejši.

V trgovini prodajajo artikle iz množice  $A$ , pri čemer posamezen kos artikla  $i \in A$  prodajajo po ceni  $c_i$ . Igor si je pripravil seznam stvari, ki jih potrebuje: identificiral je paroma disjunktno množico  $B_h \subseteq A$  in količine  $n_h$  ( $1 \leq h \leq m$ ), ki povedo, vsaj koliko kosov artiklov iz množice  $B_h$  mora kupiti (tj., posamezna množica  $B_h$  podaja artikle istega tipa – posamezen artikel lahko kupi v več kosih, količino  $n_h$  pa lahko doseže tudi z nakupom različnih artiklov iz  $B_h$ ). Zaloge artiklov v trgovini so dovolj velike, tako da Igor glede tega ne skrbi.

Pri nakupu lahko Igor uporabi tudi kupone, pri čemer množica  $K_j$  ( $1 \leq j \leq k$ ) pove, katere artikle mora kupiti, da lahko unovči  $j$ -ti kupon. Z unovčenjem  $j$ -tega kupona si pribori popust v višini  $p_j$  za vsak nabor izdelkov iz množice  $K_j$ . Pri tem velja omejitev, da lahko pri posameznem artiklu unovči največ en kupon.

**Primer:** denimo, da imamo  $B_1 = \{t, u\}$ ,  $B_2 = \{v, w\}$ ,  $n_1 = n_2 = 3$  in  $K_1 = \{u, v\}$ ,  $K_2 = \{u, w\}$  in  $K_3 = \{v, t\}$ . Potem Igor zadosti svojim potrebam, če se odloči za nakup treh kosov artikla  $u$ , dveh kosov artikla  $v$  in enega kosa artikla  $w$ . Tretjega kupona v tem primeru ne more unovčiti, saj ne kupi nobenega kosa artikla  $t$ . Če se odloči za unovčenje prvega kupona, potem za svoj nakup plača  $3c_u + 2c_v + c_w - 2p_1$ , saj je nabor artiklov iz  $K_1$  pokrit dvakrat. Pri tem ne more dodatno unovčiti tudi drugega kupona, saj je pri artiklu  $u$  že unovčil en kupon.

Zapiši celoštevilski linearni program, ki modelira zgoraj opisani problem. Pri tem lahko predpostaviš, da so vse vrednosti  $c_i$  ( $i \in A$ ) in  $p_j$  ( $1 \leq j \leq k$ ) nenegativne. Prav tako lahko predpostaviš, da noben kupon ne prinaša popusta, ki bi presegal skupne cene artiklov, ki jih je treba kupiti za njegovo unovčenje – torej  $p_j \leq \sum_{i \in K_j} c_i$  ( $1 \leq j \leq k$ ).

#### Naloga 2.46.

Vir: Kolokvij OR 31.3.2022

Skupina  $m$  vlagateljev pripravlja ponudbe za prevzem deležev  $n$  zagonskih podjetij. Naj bo  $k_i$  skupni kapital, ki ga ima  $i$ -ti vlagatelj na voljo, in  $c_{ij}$  znesek, ki ga je pripravljen vložiti v  $j$ -to podjetje, da pridobi delež  $p_{ij} > 0$  ( $1 \leq i \leq m$ ,  $1 \leq j \leq n$ )



– odkupil bo natanko tak delež po ponujeni ceni, ali pa ničesar. Poleg tega naj bo  $q_j$  ( $1 \leq j \leq n$ ) največji delež  $j$ -tega podjetja, ki ga je vlagatelj pripravljen odprodati njegov lastnik (deleže podjetja lahko odproda nobenemu, enemu ali več vlagateljem). Vlagatelji želijo pripraviti tako ponudbo, ki bo upoštevala njihove želje in omejitve ter bo maksimizirala najmanjše število podjetij, katerih delež bo odkupil posamezen vlagatelj (če delež nekega podjetja odkupi več vlagateljev, se pri vsakem šteje enkrat) – tj., želimo, da tisti vlagatelj, ki odkupi delež v najmanjšem številu podjetij, to stori pri čim večjem številu podjetij.

- (a) Zapiši celoštevilski linearni program, ki modelira zgornji problem.

**Namig:** maksimiziraj spremenljivko, ki je navzgor omejena s številom podjetij, katerih delež odkupi posamezen vlagatelj.

- (b) Vlagatelji so pripravili svojo ponudbo, a lastniki zagonskih podjetij z njo niso najbolj zadovoljni, zato bi radi sestavili protiponudbo, ki bo upoštevala tudi njihove pogoje. Lastnik  $j$ -tega podjetja ( $1 \leq j \leq n$ ) ocenjuje njegovo vrednost na  $v_j$  in tako ne bo sprejel ponudbe, s katero bi delež podjetja prodal pod sorazmerno ceno (tj., če želijo vlagatelji skupaj odkupiti delež  $r$ , potem morajo skupaj plačati vsaj  $r v_j$ ); prav tako ne bo sprejel ponudbe, s katero ne bi prejel skupnega zneska vsaj  $u_j$  (lahko pa ne odproda nobenega deleža in tako ostane brez dobička).

Zgornji celoštevilski linearni program dopolni z omejitvami, ki bodo upoštevale tudi želje lastnikov zagonskih podjetij.



### 1.3 Teorija odločanja

#### Naloga 3.1.

Vir: Vaje OR 30.3.2016

Na ulici nas ustavi neznanec in nam predlaga met kovanca. Če pade grb, nam izplača 250 000 €, če pade glava, pa mi njemu 100 000 €. Ali naj sprejmemo ponudbo?

#### Naloga 3.2.

Vir: [1, Naloga 4.2]

Trgovina pri pekarni kupuje žemlje po 0.2 € in jih prodaja po 0.4 €. Skozi leta poslovanja so izračunali naslednjo porazdelitev za prodajo žemljic.

Prodaja	50	60	70	80	90	100
Verjetnost	0.1	0.15	0.3	0.2	0.15	0.1

Če žemelj zmanjka, naročijo pri pekarni razliko, pri čemer jih žemlja tedaj stane 0.3 €. Ob koncu dneva jim pekarna odkupi presežek po 0.15 € na žemljo. Koliko žemelj se trgovini splača kupiti?

#### Naloga 3.3.

Vir: Izpit OR 5.6.2014

Pacient ima na voljo operacijo. Brez operacije bo živel natanko 3 mesece. Z uspešno opravljeno operacijo bo živel natanko 12 mesecev. Operacija je neuspešna z verjetnostjo 0.3 (v tem primeru pacient dočaka 0 mesecev). Cilj pacienta je maksimiranje pričakovane življenjske dobe.

- (a) Ali naj pacient sprejme operacijo?
- (b) Pacient lahko opravi predhodni test, ki z zanesljivostjo 0.9 napove uspešnost operacije, vendar z verjetnostjo 0.005 pacient zaradi komplikacij med testom umre. Ali naj pacient opravi test?

Nariši odločitveno drevo in odločitve sprejmi na podlagi izračunanih verjetnosti!

#### Naloga 3.4.

Vir: Vaje OR 30.3.2016

Podjetje je razvilo produkt, za katerega je konkurenca pripravljena plačati 15 M€. Če se odločijo samostojno prodajati produkt, jih vzpostavitev proizvodnje stane 6 M€, za vsak uspešno prodan produkt pa dobijo 600 €. Računajo, da bi z verjetnostjo 0.5 investicija uspela in bi prodali 100 000 produktov, z verjetnostjo 0.5 pa bi projekt propadel in bi prodali zgolj 10 000 produktov. Podjetje se lahko odloči tudi za neodvisno raziskavo trga. Ta stane 1 M€, z verjetnostjo 2/3 pa bo pravilno napovedala uspeh projekta (ne glede na to, ali bi ta uspel ali ne). Kako naj se podjetje odloči?

**Naloga 3.5.**

Vir: Izpit OR 24.6.2015

Veliki koncert skupine FiM se bo odvijal v dvorani s 100 neoznačenimi sedeži. Prireditelj se lahko odloči, da proda 100, 101, 102 ali 103 karte (povpraševanja je dovolj). Znanе so verjetnosti  $p_0 = 0.2$ ,  $p_1 = 0.3$ ,  $p_2 = 0.4$  in  $p_3 = 0.1$ , kjer je  $p_i$  verjetnost, da  $i$  kupcev kart ne pride na koncert (ne glede na število prodanih kart). Vsaka prodana karta prinese 10€ dobička, vsak obiskovalec, ki ne bo mogel v dvorano, pa pomeni 30€ stroškov (ker je že plačal 10€ za karto, ima torej organizator 20€ izgube). Koliko kart naj prireditelj proda, da bo pričakovani dobiček čim večji?

**Naloga 3.6.**

Vir: Kolokvij OR 31.5.2012

Rexhep Bajrami bi se rad naslednja štiri leta ukvarjal s prodajo sadja in zelenjave (po štirih letih mu poteče delovna viza). Rad bi najel parcelo za stojnico, ki bo stala 6000€. Če je lokacija dobra, bo imel 12000€ dobička, če pa je lokacija slaba, bo imel le 3000€ dobička. Ocenjuje, da je z verjetnostjo  $2/3$  lokacija dobra, z verjetnostjo  $1/3$  pa slaba.

- Z odločitvenim drevesom opiši njegove možnosti in ugotovi, kako naj se odloči ter kakšen dobiček naj pričakuje.
- Za nasvet lahko vpraša znanca Seada, ki "ima nos" za tovrstne posle. Sead mu lahko da nasvet, a zanj zahteva 1200€. Dobro je znano, da ima Sead naslednje pogojne verjetnosti  $P(\text{Seadovo mnenje} \mid \text{kakovost parcele})$ :

	dobra	slaba
priporoča	$2/3$	$1/2$
odsvetuje	$1/3$	$1/2$

Ali naj vpraša Seada za nasvet? Kakšen je pričakovani dobiček?

**Naloga 3.7.**

Vir: Izpit OR 15.3.2017

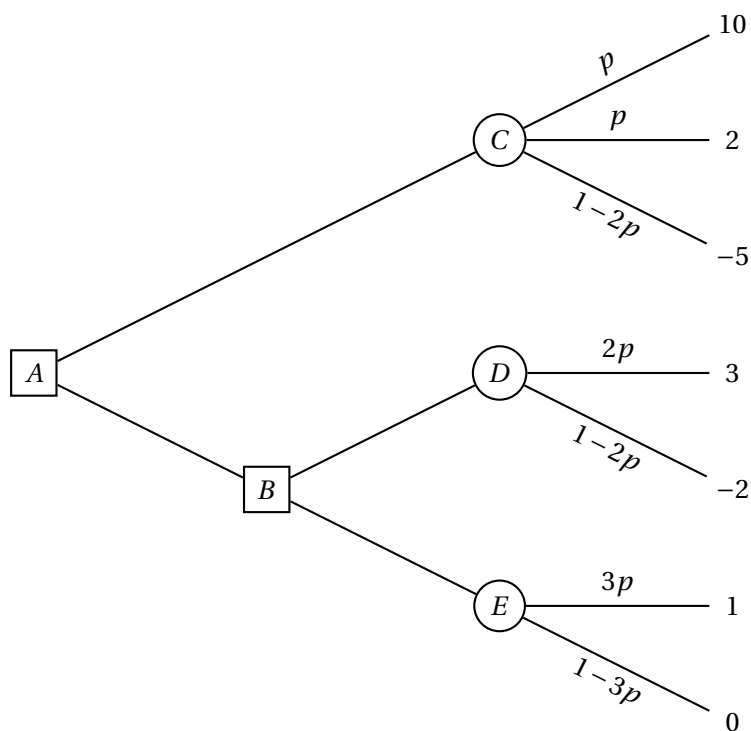
Imaš odločitveno drevo s slike 4, a nisi prepričan glede vrednosti  $p \in [0, 1/3]$ . Pričakovano vrednost želiš maksimizirati. Poišči optimalne odločitve glede na vrednost  $p$ .

**Naloga 3.8.**

Vir: Izpit OR 15.12.2016

Mudi se ti na izpit, a ravno v trenutku, ko prideš na postajo Konzorcij, odpelje avtobus številka 1. Na prikazovalniku se izpiše, da bo naslednji avtobus številka 1 prispel čez 10 minut, naslednji avtobus številka 6 čez 6 minut, naslednji avtobus številka 14 pa čez 2 minuti.

Avtobusa 1 in 6 ob ugodnih semaforjih potrebujeta 6 minut do postaje pri FE, pri čemer se lahko čas vožnje zaradi rdeče luči na semaforju pri FF podaljša za 1 minuto. Verjetnosti, da bo rdečo luč imel avtobus 1, da bo rdečo luč imel avtobus



Slika 4: Odločitveno drevo za nalogo 3.7.

6, ter da bosta oba avtobusa imela zeleno luč, so enake  $1/3$  (zaradi majhnega razmaka se ne more zgoditi, da bi oba avtobusa naletela na rdečo luč). Avtobus številka 1 nadaljuje pot do postaje pri FME, za kar potrebuje še 2 minuti.

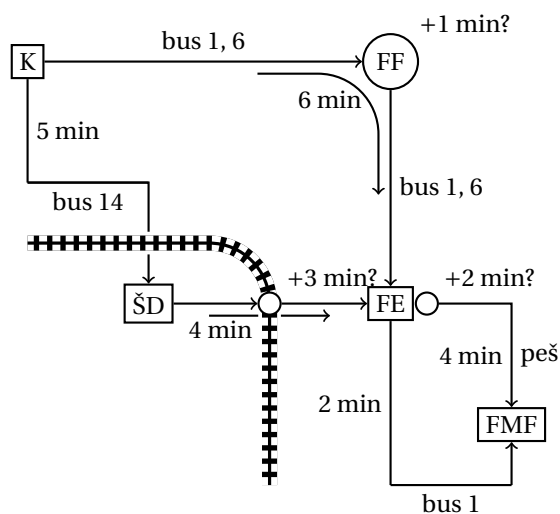
Avtobus številka 14 potrebuje 5 minut do postaje pri študentskih domovih, od tam pa greš peš do postaje pri FE, za kar potrebuješ še 4 minute. Pri tem prečkaš železnico – če mimo pripelje vlak (kar se zgodi z verjetnostjo 0.05), se čas hoje podaljša za 3 minute. Ko prideš na postajo pri FE (ne glede na to, ali si prišel z avtobusom 6 ali 14), te čakajo še 4 minute hoje do FME, vendar moraš najprej prečkati Tržaško cesto. Če je na semaforju rdeča luč (kar se zgodi z verjetnostjo 0.9, neodvisno od drugih dogodkov), se lahko odločiš, da 2 minuti počakaš na zeleno luč in potem nadaljuješ peš, ali pa da greš nazaj do postaje in počakaš na avtobus številka 1 (ki bo, tako kot prej, vozil še 2 minuti do FME).

Kakšne bodo tvoje odločitve, da bo pričakovano trajanje poti do FME čim krajše? Nariši odločitveno drevo in odločitve sprejmi na podlagi izračunanih verjetnosti! Glej sliko 5 za shemo možnih poti.

### Naloga 3.9.

Vir: Izpit OR 31.1.2017

Dve podjetji bosta predstavili konkurenčna izdelka. Možnost imaš kupiti delnice prvega podjetja za ceno 10000€ ali delnice drugega podjetja po ceni 5000€,



Slika 5: Shema možnih poti za nalogo 3.8.

lahko pa se seveda odločiš tudi, da delnic ne kupiš. Ocenjuješ, da bo z verjetnostjo 0.4 uspelo prvo podjetje, z verjetnostjo 0.1 bo uspelo drugo podjetje, z verjetnostjo 0.5 pa ne bo uspelo nobeno izmed njiju (ne more se zgoditi, da bi obe uspeli). Ob uspehu prvega podjetja se bo vrednost njihovih delnic potrojila, ob uspehu drugega podjetja pa se bo vrednost njihovih delnic popeterila – če si lastiš delnice uspešnega podjetja, jih torej lahko prodaš, pri čemer bo torej dobiček enak dvakratniku oziroma štirikratniku vloženega zneska. Delnic neuspešnega podjetja ne bo želel nihče kupiti, tako da je v tem primeru vložen znesek izgubljen.

Za mnenje lahko povprašaš tržnega izvedenca, ki bo po opravljeni raziskavi povedal, katero od dveh podjetij ima večje možnosti za uspeh. Če bo uspešno prvo podjetje, bo to pravilno napovedal z verjetnostjo 0.8, če pa bo uspešno drugo podjetje, bo to pravilno napovedal z verjetnostjo 0.7. V primeru, ko podjetji ne bosta uspeli, bo z verjetnostjo 0.4 večje možnosti pripisal prvemu, z verjetnostjo 0.6 pa drugemu podjetju. Za svoje mnenje izvedenec računa 1 000€.

Kakšne bodo tvoje odločitve, da bo tvoj pričakovani dobiček po odprodaji delnic čim večji? Nariši odločitveno drevo in odločitve sprejmi na podlagi izračunanih verjetnosti. Pričakovani dobiček tudi izračunaj.

### Naloga 3.10.

Vir: Izpit OR 10.7.2017

Proti računalniškemu programu igraš Texas hold 'em poker. Pravila igre tukaj niso pomembna. Ker imaš dostop do kode programa, poznaš logiko, po kateri se ravna. V trenutni igri si vložil 30 žetonov, enako tudi nasprotnik. Nasprotnik z verjetnostjo 0.6 meni, da so odprte karte ugodnejše zate, z verjetnostjo 0.4 pa, da so ugodnejše zanj (sam si ne ustvariš nobenega mnenja). V prvem primeru je verjetnost, da so dejansko tvoje karte boljše, enaka 0.8, v drugem pa le 0.1.

Nasprotnik se bo sedaj odločil, ali naj vloži še 10 žetonov. Sam se lahko nato odločiš, ali boš vložil 0, 10 ali 20 žetonov (skupni vložek bo torej 30, 40 ali 50 žetonov). Če je tvoj vložek manjši od nasprotnikovega, je igra izgubljena in izgubiš do sedaj vloženo. Če je tvoj vložek enak nasprotnikovemu, z nasprotnikom pogleda karte in tako določita zmagovalca. Če je tvoj vložek višji od nasprotnikovega, ima ta možnost odstopiti (tako pridobiš nasprotnikov vložek), ali pa izenačiti, nakar se zmagovalec določi na podlagi kart. Če zmagaš, pridobiš nasprotnikov vložek, če izgubiš, pa izgubiš svojega.

V spodnji tabeli so zbrane verjetnosti dogodkov v odvisnosti od nasprotnikovega mnenja glede kart. Verjetnosti navedenih dogodkov pri istem mnenju so med seboj neodvisne.

dogodek \ nasprotnikovo mnenje	ugodnejše karte	zate za nasprotnika
dejansko imaš boljše karte	0.8	0.1
nasprotnik vloži 10 žetonov po razkritju karte	0.3	0.8
nasprotnik izenači skupni vložek 40 žetonov	0.2	0.7
nasprotnik izenači skupni vložek 50 žetonov	0.1	0.8

Na primer:

$$\Pr[\text{nasprotnik izenači skupni vložek 40 žetonov} \mid \text{nasprotnikovo mnenje je "ugodnejše karte zate"}] = 0.2.$$

Kakšne bodo tvoje odločitve, da bo tvoj pričakovani dobiček po koncu igre čim večji? Nariši odločitveno drevo in odločitve sprejmi na podlagi izračunanih verjetnosti. Pričakovani dobiček tudi izračunaj.

### Naloga 3.11.

Vir: Izpit OR 29.8.2017

Dano je odločitveno drevo s slike 6, pri čemer velja  $0 \leq p \leq 1/4$ . Pričakovano vrednost želimo maksimizirati. Poišči optimalne odločitve in pričakovano vrednost v odvisnosti od vrednosti parametra  $p$ .

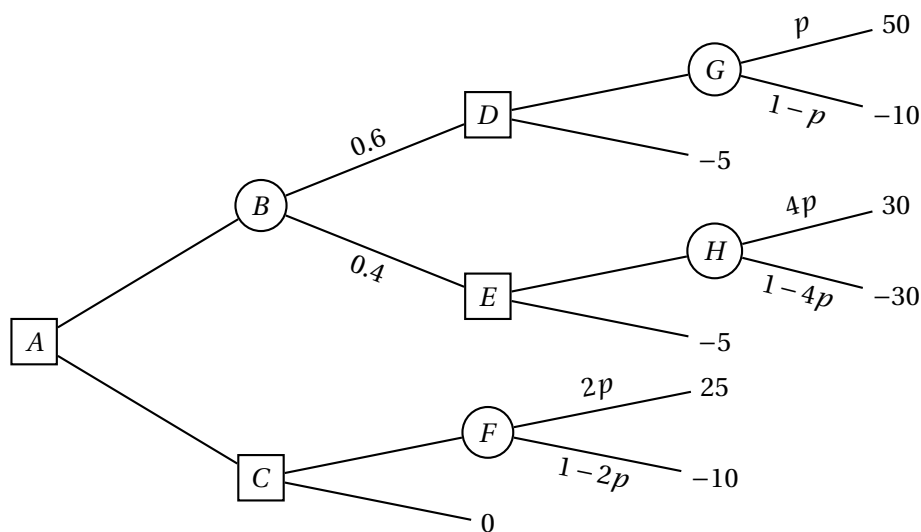
### Naloga 3.12.

Vir: Kolokvij OR 23.4.2018

Mladi podjetnik je razvil inovativen izdelek in se odloča za nadaljnje korake pri njegovem trženju. Naenkrat lahko naroči izdelavo 500 izdelkov po ceni 10 000€ ali 1 000 izdelkov po ceni 18 000€, lahko se pa odloči tudi, da izdelave ne naroči. Podjetnik ocenjuje, da je izdelek tržno zanimiv z verjetnostjo 0.8. Odloča se, ali naj posamezen izdelek prodaja po ceni 50€ ali 60€, pri čemer so v spodnji tabeli zbrana pričakovana števila prodanih kosov v odvisnosti od teh pogojev.

	cena 50€	cena 60€
tržno zanimiv	650	550
tržno nezanimiv	250	100

Predpostavi, da se bodo prodali vsi izdelani kosi, če je število teh manjše od pričakovane prodaje pri danih pogojih.



Slika 6: Odločitveno drevo za nalogo 3.11.

- (a) Kako naj se podjetnik odloči, da bo pričakovani zaslužek čim večji? Nariši odločitveno drevo in ga uporabi pri sprejemanju odločitev.
- (b) V zadnjem trenutku je podjetnik izvedel, da bo Podjetniški pospeševalnik objavil razpis za nagrado za najboljši izdelek. Razpisni pogoji zahtevajo, da se za potrebo kontrole kvalitete izdela vsaj 1 000 izdelkov, od katerih komisija izbere 20 za dejansko kontrolo, z ostalimi pa lahko prijavitelj prosto razpolaga. Če podjetnik zmaga, bo dobil nagrado v višini  $k$ , pri čemer  $k \in [1\,000\text{€}, 5\,000\text{€}]$  še ni znan, poleg tega pa si obeta tudi 20% povečanje pričakovanega števila prodanih kosov (v vseh zgoraj omenjenih pogojih). Če ne zmaga, se pričakovanja ne spremenijo. Podjetnik ocenjuje, da je verjetnost zmage enaka 0.6, če je izdelek tržno zanimiv, in 0.1, če izdelek ni tržno zanimiv.

Naj se podjetnik prijavi na razpis? Nariši odločitveno drevo in odločitve sprejmi v odvisnosti od parametra  $k$ !

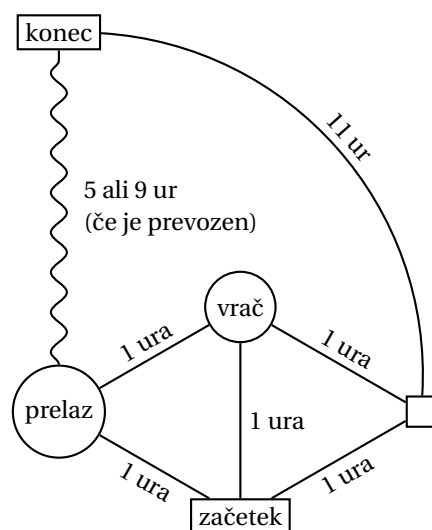
### Naloga 3.13.

Vir: Izpit OR 11.6.2018

Podajamo se na pot v mesto na drugi strani gorovja. Lahko se odločimo za pot po cesti okoli gorovja, za kar bomo porabili 12 ur. Vendar pa nas najkrajša pot vodi čez prelaz, ki je eno uro stran od začetne lokacije. Žal pa so razmere v gorah nepredvidljive: ocenjujemo, da bo z verjetnostjo 0.2 prelaz čist in ga bomo lahko prevozili v 5 urah, z verjetnostjo 0.1 bo delno zasnežen in ga bomo prevozili v 9 urah, z verjetnostjo 0.7 pa bo neprevozen, zaradi česar se bomo morali vrniti na začetek in iti okoli gorovja (skupno trajanje poti bo v tem primeru torej 14 ur).

Edini, ki nam lahko kakorkoli pomaga pri oceni vremenskih razmer na prelazu, je lokalni vrač, ki pa živi v dolini pod goro in ne uporablja sodobne teh-





Slika 7: Shema možnih poti za nalogo 3.13.

nologije, s katero bi ga lahko kontaktirali. Rade volje pa nam bo povedal, ali na gori vlada mir, če se na poti do prelaza ustavimo pri njem. Pogojne verjetnosti njegovega odgovora v odvisnosti od razmer na prelazu so podane v spodnji tabeli.

$P(\text{vračev odgovor} \mid \text{razmere na prelazu})$	čist	delno zasnežen	neprevozen
na gori vlada mir	0.9	0.5	0.1
na gori divja vojna	0.1	0.5	0.9

Do vrača imamo eno uro vožnje, do prelaza pa potem še eno uro. Če se po obisku vrača odločimo za pot okoli gorovja, bomo za nadaljnjo pot porabili 12 ur.

Kako se bomo odločili? Nariši odločitveno drevo in ga uporabi pri sprejemanju odločitev. Izračunaj tudi pričakovano trajanje poti. Glej sliko 7 za shemo možnih poti.

### Naloga 3.14.

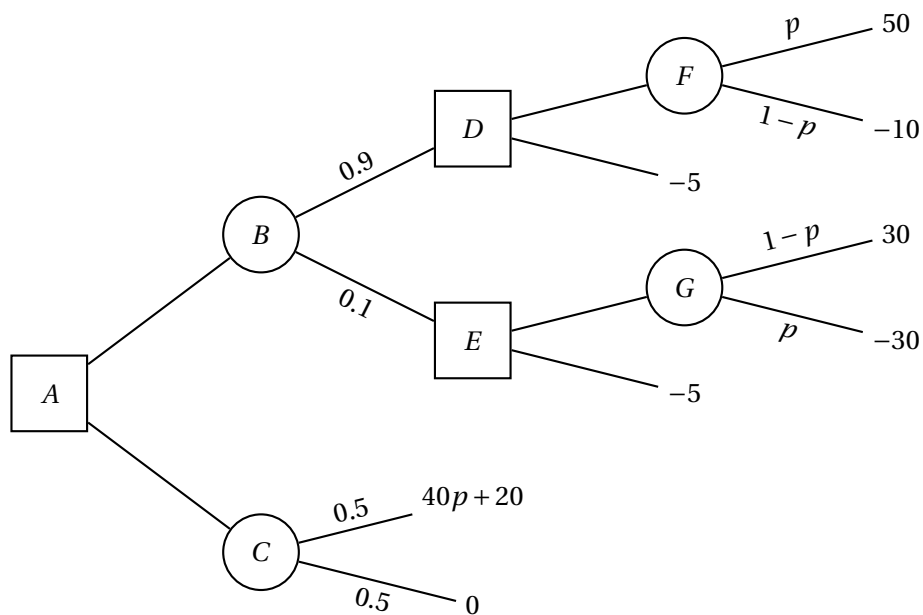
Vir: Izpit OR 28.8.2018

Dano je odločitveno drevo s slike 8, pri čemer velja  $0 \leq p \leq 1$ . Pričakovano vrednost želimo maksimizirati. Poišči optimalne odločitve in pričakovano vrednost v odvisnosti od vrednosti parametra  $p$ .

### Naloga 3.15.

Vir: Kolokvij OR 26.1.2010

V ugledni banki nameravajo okrepiti upravni odbor z direktorjem za informacijske tehnologije. Za svetovanje in pridobivanje kandidatov za to mesto so prosili znano agencijo za kadrovske svetovanje *Kadria d.o.o.* Ti so jim priporočili prodornega g. Miho Ajkalaja. Po intervjuju z g. Ajkalajem je direktor ocenil, da bo g. Ajkalaj z verjetnostjo  $p = 3/5$  v enem letu uspešen pri svojem delu. Če bo uspešen, bo banka dodatno prislužila 2 100 000€, sicer pa bo pridelala dodatno izgubo 800 000€ (izobraževanje, vpeljava, odpravnina, ...).



Slika 8: Odločitveno drevo za nalogo 3.14.

- (a) Modeliraj problem v okviru teorije odločanja (stanja, odločitve). Kakšno odločitev svetuješ direktorju glede na pričakovni dobiček? Kako je odločitev odvisna od verjetnosti  $p$ ? Nariši graf, ki prikazuje optimalni pričakovani dobiček v odvisnosti od  $p$ .
- (b) Za dodatnih 80 000€ lahko agencija z g. Ajkalajem opravi dodatne inteligence in psihološke teste, s katerimi lahko ugotovi, ali bo g. Ajkalaj uspešen. Statistični povzetek uspešnosti metode v preteklosti je naslednji:

$P(\text{izid testa} \mid \text{posl. rezultat})$ Poslovni rezultat	Izid testa	
	pozitiven	negativen
uspešen	7/8	1/8
neuspešen	1/4	3/4

Izračunaj EVPI in EVE ter komentiraj, ali je smiselno izvesti dodatna testiranja.

- (c) Nariši drevo odločitev in ugotovi, ali naj direktor izvede testiranje in če ga, kako naj se odloči glede zaposlitve g. Ajkalaja.

**Naloga 3.16.**

Vir: Izpit OR 28.6.2010

Na naftni ploščadi podjetja BP je prišlo do nekontroliranih izpustov nafte iz vr-tine v zalivu. Po vseh mogočih podvigih, da bi zamašili vrtino, se je obupano vodstvo podjetja pod pritiski predsednika bližnje države začelo dobivati z ru-skimi strokovnjaki za mašenje vrtin s pomočjo jedrskih eksplozij pod vodstvom prof. dr. Mikhaila Rasturoviča Totalkova. Vodstvo BP ocenjuje, da bo ekipa dr. To-talkova z verjetnostjo  $p = 3/7$  uspešno zamašila vrtino. Tako bo podjetje imelo sicer samo 2.8 milijarde \$ dobička, sicer pa bi zaradi upada dobička in povzročene škode utrpeli izgubo 700 milijonov \$.

- (a) Modeliraj problem v okviru teorije odločanja (stanja, odločitve). Kakšno odločitev svetuješ vodstvu BP? Kako je odločitev odvisna od verjetnosti  $p$ ? Nariši graf, ki prikazuje optimalni pričakovani dobiček v odvisnosti od  $p$ .
- (b) Za dodatnih 90 000 \$ lahko podjetje BP naroči študijo pri kitajskem ljudskem inštitutu za naftne vrtine iz mesta Lanzhou, ki ocenjuje uspešnost rizičnih projektov, in bi ocenilo, ali bo ekipa dr. Totalkova uspešna. Statistični povzetek uspešnosti raziskav inštituta v preteklosti je naslednji:

Rezultat projekta	Rezultat raziskave inštituta	
	pozitiven	negativen
uspešen	7/9	2/9
neuspešen	1/3	2/3

Izračunaj EVPI in EVE ter komentiraj, ali je smiselno izvesti dodatno raziska-vo.

- (c) Nariši drevo odločitev in ugotovi, ali naj podjetje naroči raziskavo in če jo, kako naj se odloči glede mašenja vrtine.

**Naloga 3.17.**

Vir: Izpit OR 15.9.2010

Letalska družba namerava nabaviti že rabljeno letalo, ki stane 170 000€. Oce-njujejo, da bodo z njim imeli, če je odlično ohranjeno, 1 000 000€ dobička, če je zadovoljivo ohranjeno, 340 000€ dobička, in če je slabo ohranjeno, le 10 000€ dobička. Verjetnosti, da je letalo odlično, zadovoljivo ali slabo ohranjeno, so zaporedoma 0.2, 0.3 in 0.5.

- (a) Modeliraj problem v okviru teorije odločanja (stanja, odločitve). Kakšno odločitev svetuješ vodstvu družbe?
- (b) Družba lahko naroči oceno letala pri izvedenski firmi, ki zahteva za svoje poročilo 10 000€. Vodstvo družbe takole ocenjuje pogoje verjetnosti:

$P(\text{rezultat poročila} \mid \text{kakovost letala})$	odlično	zadovoljivo	slabo
ugodno	0.9	0.6	0.1
neugodno	0.1	0.4	0.9

Kako naj se vodstvo družbe odloči?

**Naloga 3.18.**

Vir: Kolokvij OR 24.1.2011

Študent tretjega letnika finančne matematike se mora odločiti, ali bi nadaljeval študij na drugi stopnji. Ocenjuje, da bo, če študij uspešno zaključi, v življenju zaslužil 200 000 € več, kot če študij zaključi že po prvi stopnji. Če pa študija ne zaključi uspešno, bo imel zaradi stroškov študija in izgubljenega dohodka 40 000 € izgube. Verjetnost, da bo študij na drugi stopnji uspešno zaključil, je 80%.

- (a) Modeliraj problem v okviru teorije odločanja (stanja, odločitve). Kakšno odločitev svetuješ študentu?
- (b) Matematični oddelek ponuja dodatno testiranje, ki študentom pomaga pri odločitvi, ali naj nadaljujejo študij. Test stane 500 evrov, iz izkušenj kolegov pa študent ocenjuje, da so pogojne verjetnosti naslednje:

$P(\text{rezultat testa} \mid \text{uspešnost študija})$	uspešen	neuspešen
pozitiven	19/20	1/10
negativen	1/20	9/10

Ali naj se študent prijavi na dodatno testiranje?

**Naloga 3.19.**

Vir: Izpit OR 9.2.2011

Direktorica banke se mora odločiti, ali bi stranki, računalniškemu podjetju, odobrila posojilo v vrednosti 100 000 €. Po izkušnjah banke so računalniška podjetja neuspešna z verjetnostjo 20%, povprečno uspešna z verjetnostjo 50% in uspešna z verjetnostjo 30%. Če damo podjetju kredit in se izkaže za neuspešno, bomo imeli v povprečju za 15 000 € izgube, če je povprečno uspešno, bomo imeli 10 000 € dobička, če pa je uspešno, bomo imeli 20 000 € dobička.

- (a) Modeliraj problem v okviru teorije odločanja (stanja, odločitve). Kakšno odločitev svetuješ direktorici banke? Kolikšen je EVPI?
- (b) Za ceno 5 000 € lahko najamemo podjetje, ki natančno preuči računalniško podjetje in poda svojo oceno: negativno, nevtravno ali pozitivno. Po podatkih banke so pogojne verjetnosti  $P(\text{rezultat testa} \mid \text{uspešnost podjetja})$  naslednje:

	neuspešno	povprečno uspešno	uspešno
negativno	1/2	2/5	1/5
nevtravno	2/5	1/2	2/5
pozitivno	1/10	1/10	2/5

Izračunaj EVE in nariši odločitveno drevo. Kakšno odločitev svetuješ direktorici banke?

**Naloga 3.20.**

Vir: [4, Naloga 6.9]

Janez želi naložiti vsoto 1000€ v banko za dobo petih let. Odloča se med tem, da bi jo vezal za pet let (obrestna mera 5%) ali pa petkrat zaporedoma po eno leto (obrestna mera 4%). Če denar veže za pet let, vmes pa varčevanje prekine, mu pripada le obrestna mera 3%. Ocenjuje, da lahko v naslednjih petih letih pride do naslednjih situacij:

- varčeval bo pet let, pri tem se obrestna mera ne spremeni,
- varčeval bo pet let, obrestna mera se po treh letih poveča za 30%,
- varčeval bo pet let, obrestna mera se po treh letih zmanjša za 20%,
- varčeval bo tri leta.

Opiši, kako naj se odloči glede na posamezne kriterije (optimist, pesimist, Laplace, Savage). Določi vrednosti parametra  $\alpha$ , pri katerih je po Hurwiczevem kriteriju možnih več enakovrednih odločitev.

**Naloga 3.21.**

Vir: Kolokvij OR 17.4.2014

Na letalu je 100 sedežev. Dane so naslednje verjetnosti za število potnikov, ki ne pridejo na vkrcanje, pri čemer je  $P(i)$  verjetnost, da natanko  $i$  potnikov ne pride na vkrcanje:

$$P(0) = 0.25, \quad P(1) = 0.5, \quad P(2) = 0.25.$$

- (a) Koliko letalskih kart naj proda letalska družba, če vsaka prodana karta prinese 100€ dobička, vsak nezadovoljen potnik, ki ne dobi sedeža, pa 200€ izgube?
- (b) Za 200€ lahko izvedemo predhodno analizo, ki nam napove število potnikov, ki jih ne bo na vkrcanju. Zanesljivost analize je podana z verjetnostmi

$$(p_{ij})_{i,j=0}^2 = \begin{bmatrix} 0.7 & 0.1 & 0 \\ 0.2 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.9 \end{bmatrix},$$

kjer je  $p_{ij}$  verjetnost, da analiza napove odpoved  $i$  potnikov v primeru, ko imamo dejansko  $j$  odpovedi. Ali naj letalska družba izvede analizo pred prodajo kart?

**Naloga 3.22.**

Vir: Izpit OR 14.6.2013

Fenko, brat škrate Bolfenka, je najboljši alkimist, kar jih je družina imela. Preživlja se z izdelovanjem zlata iz granodiorita<sup>2</sup>. Postopek izdelave zlata je naslednji: ob 23:00 mora nastaviti sveže izkopen granodiorit (1 kg svežega granodiorita

---

<sup>2</sup>Granodiorit je zelo razširjena magmatska kamnina na Pohorju, znana tudi kot pohorski tonalit.

je vreden 2 gulda<sup>3</sup>) na točno določeno mesto pod milo nebo<sup>4</sup>. Če nastavljene kamnine noben škrat, srna ali človek ne vidi, se le-ta v dveh urah spremeni v zlato (za 1 kg granodiorita dobi 10 g zlata). Verjetnost, da se to zgodi (tj., da nastane zlato), je  $2/5$ . To zlato lahko proda škrtu zlatarju za gulde (in to je edini možni način porabe zlata). Za 10 g zlata dobi 4 gulde. Granodiorit, ki je enkrat že bil nastavljen, se ne bo nikoli več spremenil v zlato. Fenko lahko kilogram "porabljenega" granodiorita proda za 1 guld.

Predpostavimo, da Fenko zapravlja denar le za izdelavo zlata iz granodiorita, da zmeraj nastavi celoštevilsko mnogo kilogramov granodiorita in ima zmeraj celoštevilsko mnogo guldov (npr. s tremi guldi lahko kupi največ 1 kg granodiorita, en guld mu ostane).

Trenutno ima Fenko 5 guldov, nič zlata in nič granodiorita. Koliko kilogramov granodiorita naj v naslednjih dveh nočeh nastavi pod milo nebo, da bo imel kar največje pričakovano število guldov? V odgovoru napiši, koliko granodiorita naj nastavi prvo noč in koliko drugo noč. Odgovor bo odvisen od tega, ali se mu ponoči granodiorit spremeni v zlato ali ne.

### Naloga 3.23.

Vir: [1, Naloga 4.3]

Ljubo prodaja časopise po centru Ljubljane. Vsak dan se mora odločiti, koliko časopisov naj naroči. Za vsak naročen časopis plača 0.8€, za vsak prodan časopis pa iztrži 1€. Za neprodane časopise ne dobi povrnjenega denarja. Vsak dan je verjetnost, da bo imel  $i$  kupcev, enaka  $p_i$ , kjer je  $p_6 = 0.1$ ,  $p_7 = 0.2$ ,  $p_8 = 0.3$ ,  $p_9 = 0.3$  in  $p_{10} = 0.1$ .

- Kakšno odločitev svetuješ Ljubu in zakaj?
- Kolikšen zaslužek lahko pričakuje v mesecu, ko časopis izide petindvajsetkrat?

### Naloga 3.24.

Vir: Izpit OR 24.6.2014

Tovarna od dobavitelja dobi paket 10 komponent  $A$ . Tovarna sestavlja izdelke  $B$ , kjer gre v vsak izdelek  $B$  natanko ena komponenta  $A$ . Dane so verjetnosti

$$p_i = P(\text{med 10 komponentami } A \text{ je natanko } i \text{ pokvarjenih}),$$

kjer je  $p_1 = 0.7$ ,  $p_2 = 0.2$  in  $p_3 = 0.1$ . Stroški pregleda posamezne komponente so 45€. Stroški popravila izdelka  $B$  zaradi vgrajene okvarjene komponente  $A$  so enaki 350€.

- Denimo, da ima tovarna na izbiro samo dve možnosti. Prva je, da pregleda vseh 10 dobavljenih izdelkov. Druga možnost pa je, da ne pregleda nobenega dobavljenega izdelka. Katera odločitev tovarni povzroči manj stroškov?

<sup>3</sup>Guld je denarna enota, ki jo uporabljajo pohorski škratje.

<sup>4</sup>Kam je treba nastaviti granodiorit, je odvisno tudi od položaja planetov.

- (b) Naj ima sedaj tovarna na voljo drugačni izbiri. Prva je, da naključno izbere eno izmed 10 komponent  $A$ , jo pregleda in po potrebi zamenja, ter gre nato direktno v proizvodnjo izdelkov  $B$ . Druga možnost pa je, da pregleda vseh 10 izdelkov. Katera odločitev tovarni povzroči manj stroškov?

**Naloga 3.25.**

Vir: Izpit OR 25.8.2014

V škatli so štirje ponarejeni kovanci, vredni 0€, in en zlat kovanec, vreden 100€. Na slepo lahko izbereš po en kovanec, pri čemer moraš za vsako izbiranje plačati 30€. Določi pričakovani profit, ki ga dosežeš pri igranju te igre.

**Naloga 3.26.**

Vir: Izpit OR 12.5.2016

Smo v letu 2500 in turistične rakete že potujejo na Luno. LunaAirways oglašuje let, na katerem je lahko 100 potnikov. LunaAirways računa, da bodo vsa mesta zasedena. Z leti izkušenj vedo, da nekatere potnike zagrabi strah in tako ne pridejo na potovanje. Ocenjujejo, da je verjetnost  $p_i$ , da natanko  $i$  potnikov ( $0 \leq i \leq 5$ ) ne pride na potovanje, naslednje:

$i$	od 0 do 3	od 4 do 5
$p_i$	0.2	0.1

S prodajo karte ima družba 300 denot dobička. Za vsakega potnika, ki bo moral menjati let, ima LunaAirways 400 denot stroškov. Koliko kart naj proda LunaAirways, če želi maksimizirati pričakovani dobiček?

**Naloga 3.27.**

Vir: Izpit OR 24.5.2016

Kavarna *maφja* je razvila recept za novo torto. Slaščičarna *sladkeoperacijske-raziskave* je za ekskluzivno pravico do recepta pripravljena plačati 15002€. Če se kavarna *maφja* odloči za samostojno prodajo torte, jih začetni vložek stane 10000€, za vsako prodano torto pa zaslužijo 25€. Po njihovi presoji je verjetnost uspeha recepture (prodajo 100000 tort) enaka 0.4, verjetnost propada (prodali bi zgolj 10000 tort) pa 0.6. Kavarna se lahko odloči, da zaprosi za pomoč tudi podjetje, ki na osnovi degustacij torte sestavi mnenje o uspehu recepta. Svetovanje jih stane 35000€, natančnost svetovanja pa opisuje spodnja tabela.

$P(\text{mnenje svetovalca} \mid \text{uspeh recepta})$	Recept uspe	Recept ne uspe
Ugodno	$\frac{3}{4}$	$\frac{1}{2}$
Neugodno	$\frac{1}{4}$	$\frac{1}{2}$

Kako naj se kavarna *maφja* odloči? Zakaj?

**Naloga 3.28.**

Vir: Izpit OR 24.5.2016

Organizatorji olimpijskih iger so zgradili dvorano za judo, ki sprejme 1 000 ljudi. Popularnost juda je na vrhuncu, zato organizatorji pričakujejo, da bodo vse karte zakupljene. Zaradi straha pred virusom zika so analitiki za  $0 \leq i \leq 5$  ( $i \in \mathbb{Z}$ ) izračunali verjetnosti  $p_i$ , da se natanko  $i$  izmed imetnikov kart ne pojavi na prireditvi. Verjetnosti so zbrane v spodnji tabeli.

$i$	0	1	2	3	4	5
$p_i$	0.01	0.19	0.3	0.4	0.05	0.05

S prodajo vstopnice imajo organizatorji 1 000 denot dobička. Za vsakega obiskovalca, ki bo ostal brez sedeža in bo moral v VIP sekcijo dvorane, imajo 1 400 denot stroškov. Koliko kart naj organizatorji olimpijskih iger prodajo, če želijo maksimizirati pričakovani dobiček?

**Naloga 3.29.**

Vir: Kolokvij OR 19.4.2019

V manjšem podjetju so razvili nov okus sadnega soka. Proizvedli so ga že 100 000 litrov, ko so prejeli ponudbo multinacionalke, da odkupijo vso razpoložljivo količino po ceni 1€ na liter. Če se ne odločijo za odprodajo, bodo sok sami pakirali in ponudili na trgu po ceni 3€ na liter. Zagon pakiranja stane 1 000€, pakiranje enega litra soka pa 0.5€. V podjetju ocenjujejo, da bo z verjetnostjo 0.7 produkt uspešen in ga bodo vsega prodali, z verjetnostjo 0.3 pa bo neuspešen in ga bodo prodali le 10 000 litrov.

Pri podjetju imajo na voljo ravno dovolj časa, da pred odločitvijo pakirajo in na regionalnem trgu ponudijo 1 000 litrov soka po akcijski ceni 2.5€ na liter. Ocenjujejo, da bi v primeru uspeha produkta z verjetnostjo 0.8 tudi akcija uspela in bi tako razprodali akcijske zaloge, v primeru neuspeha produkta pa bi se to zgodilo le z verjetnostjo 0.1. Če akcija ne bi uspela, bi prodali le 100 litrov soka. Če se po akcijski ponudbi odločijo za samostojno prodajo, bodo seveda morali še enkrat plačati stroške zagona pakiranja, v nasprotnem primeru pa bo multinacionalka odkupila preostalih 99 000 litrov še nepakiranega soka.

**Opomba:** količina, prodana v akcijski ponudbi, je vključena v končno prodano količino. Če torej v akciji prodajo vseh 1 000 litrov soka po ceni 2.5€ na liter, ga bodo v primeru neuspeha produkta v redni prodaji prodali le še 9 000 litrov po ceni 3€ na liter, v primeru uspeha pa še 99 000 litrov.

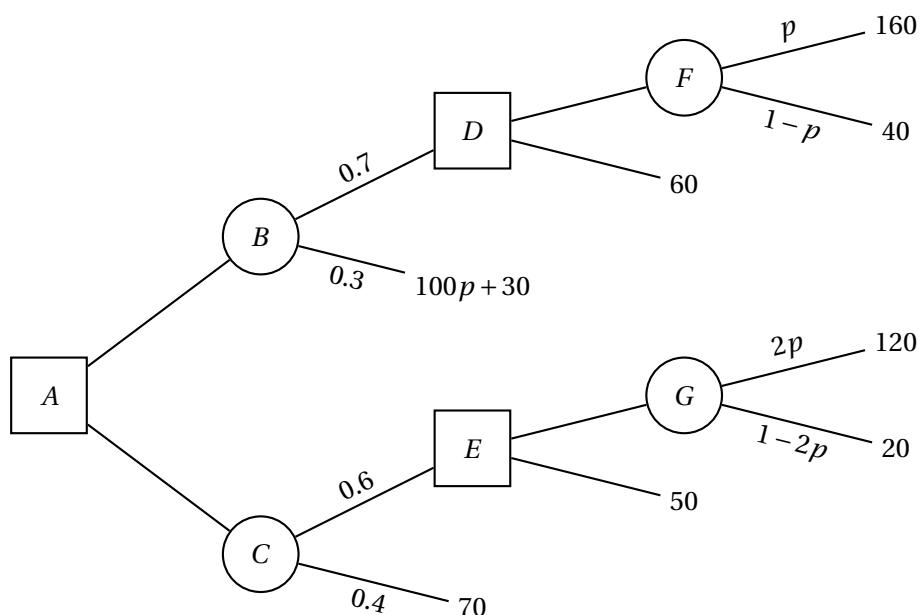
Kako naj se pri podjetju odločijo, da bo pričakovani zaslužek čim večji? Nariši odločitveno drevo in ga uporabi pri sprejemanju odločitev.

**Naloga 3.30.**

Vir: Izpit OR 3.6.2019

Podjetje je od konkurenta v finančnih težavah odkupilo tovarno čokolade, ki že nekaj časa ni bila v uporabi. Ker že prejemale nova naročila, želijo v naslednjih 8 tednih proizvesti čim večjo količino čokolade. Ocenjujejo, da so z verjetnostjo 0.4 stroji v dobrem stanju in lahko proizvedejo 10 000 kg čokolade na teden, z





Slika 9: Odločitveno drevo za nalogo 3.31.

verjetnostjo 0.6 pa so v slabem stanju in lahko proizvedejo le 1 000 kg čokolade na teden.

Odločijo se lahko, da pred zagonom proizvodnje na strojih izvedejo servis. Izvedba servisa lahko poteka gladko in konča v enem tednu, ali pa se pojavijo težave in tako servis traja tri tedne (za proizvodnjo tako ostane le še 7 oziroma 5 tednov). Če so stroji v dobrem stanju, bo servis z verjetnostjo 0.9 potekal gladko, z verjetnostjo 0.1 pa se bodo pojavile težave – v obeh primerih pa bodo stroji po servisu ostali v dobrem stanju. Če so stroji v slabem stanju, bo servis z verjetnostjo 0.2 potekal gladko in jih tedaj z verjetnostjo 0.7 spravil v dobro stanje, z verjetnostjo 0.8 pa se bodo pojavile težave – tedaj je verjetnost, da bodo stroji po servisu v dobrem stanju, enaka 0.5. Po zagonu proizvodnje te zaradi previsokih stroškov ni mogoče predčasno prekiniti; stroški servisa pa niso pomembni, saj ga bodo morali izvesti kasneje, če se zanj ne odločijo takoj.

Kako naj se v podjetju odločijo? Nariši odločitveno drevo in ga uporabi pri sprejemanju odločitev. Izračunaj tudi pričakovano količino proizvedene čokolade.

### Naloga 3.31.

Vir: Izpit OR 19.6.2019

Dano je odločitveno drevo s slike 9, pri čemer velja  $0 \leq p \leq 1/2$ . Pričakovano vrednost želimo minimizirati. Poišči optimalne odločitve in pričakovano vrednost v odvisnosti od vrednosti parametra  $p$ .

**Naloga 3.32.**

Vir: Izpit OR 4.6.2020

Na trajektu stoji 200 avtomobilov, vozovnica pa stane 35€. Pri družbi, ki upravlja trajekt, se lahko odločijo prodati 200, 201, 202 ali 203 vozovnice. Naj bo  $p_i$  verjetnost, da  $i$  avtomobilov z vozovnicami ne pride do trajekta (tj.,  $p_0$  je verjetnost, da vsi pridejo):

$i$	0	1	2	3
$p_i$	0.3	0.4	0.2	0.1

Z vsakim avtomobilom, ki pride do trajekta in se ne more vkrcati, ima družba 60€ stroškov (skupaj torej 25€ izgube).

- (a) Koliko vozovnic naj proda družba, da bo imela čim večji dobiček?
- (b) Na trajektu je na voljo še eno mesto, ki pa ni najbolj varno<sup>5</sup> – avtomobil, ki se pelje na njem, bo z verjetnostjo 0.001 (neodvisno od ostalih dejavnikov) med potjo padel v morje. V tem primeru ima družba dodatnih 40000€ stroškov. Ali se družbi izplača uporabiti to mesto? Koliko vozovnic naj tedaj proda?

**Naloga 3.33.**

Vir: Izpit OR 25.8.2020

Janez je podedoval zemljo, na kateri si želi zgraditi hišo. Ve, da mora pred začetkom gradnje pridobiti gradbeno dovoljenje, zaradi nerazumljivih birokratskih postopkov in nejasne razmejitve parcel pa ne ve, ali morda potrebuje tudi soglasja sosedov. Lahko se odloči, da takoj zaprosi za gradbeno dovoljenje – ocenjuje, da bo z verjetnostjo  $1/2$  vloga uspešna in bo po 30 dneh lahko začel graditi, sicer pa mu bodo vlogo zavrnil in bo moral pridobiti soglasja, kar bo celoten postopek podaljšalo na 55 dni. Lahko pa pred vložitvijo prošnje pridobi soglasja vseh sosedov – v tem primeru bo celoten postopek trajal 45 dni.

Janez pozna tudi odvetnika Bineta, ki lahko pred sprožitvijo postopkov preuči situacijo, za kar potrebuje 7 dni. Spodaj so zbrane pogojne verjetnosti za Binetovo mnenje glede na to, ali so soglasja potrebna.

$P(\text{Binetovo mnenje} \mid \text{potreba po soglasjih})$	so potrebna	niso potrebna
ugodno	$\frac{1}{3}$	$\frac{3}{4}$
neugodno	$\frac{2}{3}$	$\frac{1}{4}$

Kako naj se Janez odloči, da bo pričakovani čas do začetka gradnje hiše čim krajši? Nariši odločitveno drevo in ga uporabi pri sprejemanju odločitev. Izračunaj tudi pričakovani čas.

<sup>5</sup>Seveda bo to mesto ostalo prazno, če bo to mogoče.

**Naloga 3.34.**

Vir: Kolokvij OR 12.4.2021

Janez si je ravnokar odgovoril na vprašanje iz naloge 3.33, ko je ugotovil, da so uradi zaradi epidemiološke situacije zaprti in da postopka ne bo mogel začeti v naslednjih  $t$  dneh, kjer je  $t \in [7, 15]$  (vrednost  $t$  bo znana v kratkem). Na srečo lahko Janez v tem času še vedno pridobi soglasja sosedov, za kar potrebuje 15 dni, prav tako lahko mu lahko pomaga Bine, ki za izdelavo svojega mnenja še vedno potrebuje 7 dni. Po izteku  $t$  dni bo lahko Janez sprožil postopek (seveda bo pred tem počakal na Binetovo mnenje oziroma soglasja sosedov, če se za to odloči), ki bo trajal 55 dni, če se izkaže, da so soglasja potrebna in jih pred tem še ni pridobil, oziroma 30 dni sicer. Vse verjetnosti ostajajo nespremenjene.

Nariši odločitveno drevo in ga uporabi pri sprejemanju odločitev v odvisnosti od vrednosti parametra  $t$ .



## 1.4 Dinamično programiranje

### Naloga 4.1.

Vir: Izpit OR 15.9.2010

Na avtocestni odsek dolžine  $M$  kilometrov želimo postaviti oglasne plakate. Dovoljene lokacije plakatov določa urad za oglaševanje in so predstavljene s števili  $x_1, x_2, \dots, x_n$ , kjer  $x_i$  ( $1 \leq i \leq n$ ) predstavlja oddaljenost od začetka odseka v kilometrih. Profitabilnost oglasa na lokaciji  $x_i$  določa vrednost  $v_i$  ( $1 \leq i \leq n$ ). Urad za oglaševanje podaja tudi omejitev, da mora biti razdalja med oglasi vsaj  $d$  kilometrov. Oglase želimo postaviti tako, da bodo čim bolj profitabilni.

- (a) Napiši rekurzivne enačbe za opisani problem.
- (b) Reši problem za parametre  $M = 20$ ,  $d = 5$ ,  $n = 8$ ,  $(x_i)_{i=1}^n = (1, 2, 8, 10, 12, 14, 17, 20)$  in  $(v_i)_{i=1}^n = (8, 8, 12, 10, 7, 5, 6, 10)$ .
- (c) Napiši algoritem, ki poišče najbolj profitabilno postavitev oglasov za dane parametre. Kakšna je njegova časovna zahtevnost?

### Naloga 4.2.

Vir: Vaje OR 6.4.2016

Imamo nahrbtnik nosilnosti  $M$  kilogramov. Danih je  $n$  objektov z vrednostmi  $v_i$  in težami  $t_i$  ( $1 \leq i \leq n$ ). Problem nahrbtnika sprašuje po izbiri predmetov  $I \subseteq \{1, 2, \dots, n\}$ , ki maksimizira njihovo skupno vrednost pri omejitvi  $\sum_{i \in I} t_i \leq M$ .

- (a) Napiši rekurzivne enačbe za opisani problem.
- (b) Z uporabo rekurzivnih enačb reši problem za parametre  $M = 8$ ,  $n = 8$ ,

$$(v_i)_{i=1}^n = (9, 9, 8, 11, 10, 15, 3, 12) \quad \text{in} \\ (t_i)_{i=1}^n = (3, 5, 1, 4, 3, 8, 2, 7).$$

### Naloga 4.3.

Vir: Vaje OR 6.4.2016

Dana je matrika  $A = (a_{ij})_{i,j=1}^{m,n}$ . Poiskati želimo pot minimalne vsote, ki se začne v levem zgornjem kotu (pri  $a_{11}$ ) in konča v desnem spodnjem kotu (pri  $a_{mn}$ ). Dovoljeni so zgolj premiki v desno in navzdol.

- (a) Napiši rekurzivne enačbe za opisani problem.
- (b) Reši problem za matriko

$$A = \begin{pmatrix} 131 & 673 & 234 & 103 & 18 \\ 201 & 96 & 342 & 965 & 150 \\ 630 & 803 & 746 & 422 & 111 \\ 537 & 699 & 497 & 121 & 956 \\ 805 & 732 & 524 & 37 & 332 \end{pmatrix}.$$

- (c) Na osnovi rekurzivnih enačb napiši algoritem, ki reši opisani problem. Oceni tudi njegovo časovno zahtevnost v odvisnosti od  $m$  in  $n$ .

**Naloga 4.4.**

Vir: Vaje OR 6.4.2016

Dan je niz  $S = a_1 a_2 \dots a_n$ , kjer so  $a_i$  ( $1 \leq i \leq n$ ) elementi neke končne abecede. Nizu  $a_j a_{j+1} \dots a_k$ , kjer je  $1 \leq j \leq k \leq n$ , pravimo *strnjen podniz* niza  $S$ . S pomočjo dinamičnega programiranja napiši algoritem, ki določi najdaljši palindromski strnjen podniz v  $S$ .

**Naloga 4.5.**

Vir: Vaje OR 6.4.2016

Dana je matrika  $A = (a_{ij})_{i,j=1}^{m,n}$ . Poiskati želimo strnjeno podmatriko matrike  $A$  z največjo vsoto komponent.

(a) Napiši rekurzivne enačbe za opisani problem.

(b) Reši problem za matriko

$$A = \begin{pmatrix} 1 & -1 & 2 & 4 \\ -3 & -2 & 8 & 2 \\ -3 & 2 & -2 & 4 \\ 1 & -5 & -1 & -2 \end{pmatrix}.$$

(c) Napiši algoritem, ki reši opisani problem. Oцени tudi njegovo časovno zahtevnost v odvisnosti od  $m$  in  $n$ .

**Naloga 4.6.**

Vir: Vaje OR 6.4.2016

Na voljo imamo kovance z vrednostmi  $1 = v_1 < v_2 < \dots < v_n$  in vsoto  $C$ , ki jo želimo izplačati s čim manjšim številom kovancev. Predpostavljamo, da imamo dovolj velik nabor kovancev.

(a) S pomočjo dinamičnega programiranja reši problem v splošnem.

(b) Poišči izplačilo z najmanjšim številom kovancev za  $C = 25$ ,  $n = 4$  in  $(v_i)_{i=1}^n = (1, 2, 5, 7)$ .

**Naloga 4.7.**

Vir: Vaje OR 6.4.2016

Na ulici je  $n$  vrstnih hiš, pri čemer je v  $i$ -ti hiši  $c_i$  denarja. Tat se odloča, katere izmed hiš naj oropa. Vsak oropan stanovalec to sporoči svojim sosedom, zato tat ne sme oropati dveh sosednjih hiš. Ker je tat poslušal predmet Operacijske raziskave, pozna dinamično programiranje. Pokaži, kako naj tat določi, katere hiše naj oropa.

**Naloga 4.8.**

Vir: Kolokvij OR 31.5.2012

Imamo hlod dolžine  $\ell$ , ki bi ga radi razžagali na  $n$  označenih mestih  $0 < x_1 < x_2 < \dots < x_n < \ell$ . Eno rezanje stane toliko, kolikor je dolžina hloda, ki ga režemo. Ko hlod prerežemo, dobimo dva manjša hloda, ki ju režemo naprej. Poiskati želimo zaporedje rezanj z najmanjšo ceno.

- (a) S pomočjo dinamičnega programiranja reši problem v splošnem. Oцени tudi njegovo časovno zahtevnost.
- (b) Reši problem pri podatkih  $\ell = 10$  in  $(x)_{i=1}^4 = (3, 5, 7, 8)$ .

**Naloga 4.9.**

Vir: [3, Problem 11.3-1]

Lastnik verige  $n$  trgovin z živili je kupil  $m$  zabojev svežih jagod. Naj bo  $p_{ij}$  pričakovani dobiček v trgovini  $j$ , če tja dostavimo  $i$  zabojev. Zanima nas, koliko zabojev naj gre v vsako trgovino, da bomo imeli čim večji zaslužek. Zaradi logističnih razlogov zabojev ne želimo deliti.

- (a) Z dinamičnim programiranjem reši problem za podatke  $m = 5$ ,  $n = 3$  in  $p_{ij}$  iz sledeče tabele:

$p_{ij}$	1	2	3
0	0	0	0
1	5	6	4
2	9	11	9
3	14	15	13
4	17	19	18
5	21	22	20

- (b) Napiši algoritem, ki reši opisani problem v splošnem.

**Naloga 4.10.**

Vir: [3, Problem 11.3-8]

Podjetje bo kmalu uvedlo nov izdelek na zelo konkurenčen trg, zato trenutno pripravlja marketinško strategijo. Odločili so se, da bodo izdelek uvedli v treh fazah. V prvi fazi bodo pripravili posebno začetno ponudbo z močno znižano ceno, da bi privabili zgodnje kupce. Druga faza bo vključevala intenzivno oglaševalsko kampanjo, da bi zgodnje kupce prepričali, naj izdelek še vedno kupujejo po redni ceni. Znano je, da bo ob koncu druge faze konkurenčno podjetje predstavilo svoj izdelek. Zato bo v tretji fazi okrepljeno oglaševanje z namenom, da bi preprečili beg strank h konkurenci.

Podjetje ima za oglaševanje na voljo 4 milijone evrov, ki jih želimo čim bolj učinkovito porabiti. Naj bo  $m$  tržni delež v procentih, pridobljen v prvi fazi,  $f_2$  delež, ohranjen po drugi fazi, in  $f_3$  delež, ohranjen po tretji fazi. Maksimizirati želimo končni tržni delež, torej količino  $m f_2 f_3$ .

- (a) Denimo, da želimo v vsaki fazi porabiti nek večkratnik milijona evrov, pri čemer bomo pri prvi fazi porabili vsaj milijon evrov. V spodnji tabeli so zbrani vplivi porabljenih količin na vrednosti  $m$ ,  $f_2$  in  $f_3$ .

$M\text{€}$	$m$	$f_2$	$f_3$
0	–	0.2	0.3
1	20	0.4	0.5
2	30	0.5	0.6
3	40	0.6	0.7
4	50	–	–

Kako naj razdelimo sredstva?

- (b) Denimo sedaj, da lahko v vsaki fazi porabimo poljubno pozitivno količino denarja (seveda glede na omejitve skupne porabe). Naj bodo torej  $x_1$ ,  $x_2$  in  $x_3$  količine denarja v milijonih evrov, ki jih porabimo v prvi, drugi in tretji fazi. Vpliv na tržni delež je podan s formulami

$$m = x_1(10 - x_1), \quad f_2 = 0.4 + 0.1x_2, \quad \text{in} \quad f_3 = 0.6 + 0.07x_3.$$

Kako naj sedaj razdelimo sredstva?

#### Naloga 4.11.

Vir: Izpit OR 26.6.2012

Nori profesor Boltežar stanuje v stolpnici z  $n$  nadstropji, oštevilčenimi od 1 do  $n$ . Nori stanovalci tega bloka radi mečejo cvetlične lončke z balkonov. Boltežar bi rad ugotovil, katero je najvišje nadstropje, s katerega lahko pade cvetlični lonček, ne da bi se razbil. Jasno je, da če se lonček razbije pri padcu iz  $k$ -tega nadstropja, potem se razbije tudi pri padcu s  $(k + 1)$ -tega nadstropja. Če bi Boltežar imel le en cvetlični lonček, bi ga lahko metal po vrsti od najnižjega nadstropja navzgor, dokler se ne bi razbil. V najslabšem primeru bi lonček torej vrgel  $n$  krat (možno je, da bi lonček preživel tudi padec iz najvišjega nadstropja).

Ker ima Boltežar doma  $k$  cvetličnih lončkov, lahko do rezultata pride tudi z manjšim številom metov. S pomočjo dinamičnega programiranja bi rad poiskal strategijo metanja, ki bi minimizirala število potrebnih metov v najslabšem primeru.

- (a) Napiši rekurzivne enačbe za opisani problem.
- (b) Napiši algoritem, ki reši opisani problem. Oceni tudi njegovo časovno zahtevnost v odvisnosti od  $n$  in  $k$ .



**Naloga 4.12.**

Vir: [3, Problem 11.2-2]

Vodja prodaje pri založniku učbenikov za fakulteto ima na voljo 6 trgovskih potnikov, ki jim želi dodeliti eno od treh regij, v kateri bodo delovali. V vsaki regiji mora delovati vsaj en trgovski potnik. Naj bo  $p_{ij}$  pričakovana porast v prodaji v regiji  $j$ , če bo tam delovalo  $i$  trgovskih potnikov:

$p_{ij}$	1	2	3
1	35	21	28
2	48	42	41
3	70	56	63
4	89	70	75

Reši problem s pomočjo dinamičnega programiranja.

**Naloga 4.13.**

Vir: [3, Problem 11.3-16]

Dan je sledeči nelinearni program.

$$\begin{aligned} \max \quad & 2x_1^2 + 2x_2 + 4x_3 - x_3^2 \\ & 2x_1 + x_2 + x_3 \leq 4 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Reši ga s pomočjo dinamičnega programiranja.

**Naloga 4.14.**

Vir: [3, Problem 11.4-1]

Igralec na srečo bo odigral tri partije s svojimi prijatelji, pri čemer lahko vsakič stavi na svojo zmago. Stavi lahko katerokoli vsoto denarja, ki jo ima na voljo – če izgubi partijo, zastavljeno vsoto izgubi, sicer pa tako vsoto pridobi. Pri vsaki partiji sta verjetnosti zmage in poraza enaki  $1/2$ . Na začetku ima 75€, na koncu pa želi imeti 100€ (ker igra s prijatelji, noče imeti več kot toliko).

Z dinamičnim programiranjem poišči strategijo stavljenja, ki maksimizira verjetnost, da bo na koncu imel natanko 100€.

**Naloga 4.15.**

Vir: [2, Exercise 6.14]

Imamo pravokoten kos blaga dimenzij  $m \times n$ , kjer sta  $m$  in  $n$  pozitivni celi števili, ter seznam  $k$  izdelkov, pri čemer potrebujemo za izdelek  $h$  pravokoten kos blaga dimenzij  $a_h \times b_h$  ( $a_h, b_h$  sta pozitivni celi števili), ki ga prodamo za ceno  $c_h > 0$ . Imamo stroj, ki lahko poljuben kos blaga razreže na dva dela bodisi vodoravno, bodisi navpično. Začetni kos blaga želimo razrezati tako, da bomo lahko naredili izdelke, ki nam bodo prinašali čim večji dobiček. Pri tem smemo izdelati poljubno število kosov posameznega izdelka. Kose blaga lahko seveda tudi obračamo (tj., za izdelek  $h$  lahko narežemo kos velikosti  $a_h \times b_h$  ali  $b_h \times a_h$ ).

Zapiši rekurzivne enačbe za reševanje danega problema. Razloži, kaj predstavljajo spremenljivke, v kakšnem vrstnem redu jih računamo, ter kako dobimo optimalno rešitev.

**Naloga 4.16.**

Vir: Izpit OR 15.12.2016

Oceni časovno zahtevnost algoritma, ki sledi iz rekurzivnih enačb za nalogo 4.15. Reši problem za podatke  $m = 5, n = 3, k = 4$ ,  $(a_h)_{h=1}^k = (2, 3, 1, 2)$ ,  $(b_h)_{h=1}^k = (2, 1, 4, 3)$  in  $(c_h)_{h=1}^k = (6, 3, 5, 7)$ .

**Naloga 4.17.**

Vir: Izpit OR 31.1.2017

Dano je zaporedje  $n$  realnih števil  $a_1, a_2, \dots, a_n$ . Želimo poiskati strnjeno podzaporedje z največjim produktom – t.j., taka indeksa  $i, j$  ( $1 \leq i \leq j \leq n$ ), da je produkt  $a_i a_{i+1} \cdots a_{j-1} a_j$  čim večji.

- (a) Zapiši rekurzivne enačbe za reševanje danega problema. Razloži, kaj predstavljajo spremenljivke, v kakšnem vrstnem redu jih računamo, ter kako dobimo optimalno rešitev.

**Namig:** posebej obravnavaj pozitivne in negativne delne produkte.

- (b) Oceni časovno zahtevnost algoritma, ki sledi iz zgoraj zapisanih enačb.

- (c) S svojim algoritmom reši problem za zaporedje

$$0.9, -2, -0.6, -0.5, -2, 5, 0.1, 3, 0.5, -3.$$

**Naloga 4.18.**

Vir: Izpit OR 15.3.2017

Za zaporedje števil  $x_1, x_2, \dots, x_m$  pravimo, da je *oscilirajoče*, če velja  $x_i < x_{i+1}$  za vse sode  $i$  in  $x_i > x_{i+1}$  za vse lihe  $i$ . S pomočjo dinamičnega programiranja zasnuj algoritem, ki v polinomskem času izračuna dolžino najdaljšega oscilirajočega podzaporedja zaporedja celih števil  $a_1, a_2, \dots, a_n$ .

**Naloga 4.19.**

Vir: Izpit OR 10.7.2017

V podjetju imajo na voljo  $m$  milijonov evrov sredstev, ki jih bodo vložili v razvoj nove aplikacije. Denar bodo porazdelili med tri skupine. Naj bodo  $x_1, x_2$  in  $x_3$  količine denarja (v milijonih evrov), ki jih bodo dodelili razvijalcem, oblikovalcem in marketingu. Vrednosti  $x_1, x_2, x_3$  niso nujno cela števila. Razvijalci morajo dobiti vsaj  $a_1$  milijonov evrov, potencial, ki ga ustvarijo, pa je  $p_1 = n_1 + k_1 x_1$ . Oblikovalci morajo dobiti vsaj  $a_2$  milijonov evrov, potencial, ki ga ustvarijo, pa je  $p_2 = n_2 + k_2 x_2$ . Marketing mora dobiti vsaj  $a_3$  milijonov evrov, ustvari pa faktor  $p_3 = n_3 + k_3 x_3$ . Pričakovani dobiček v milijonih evrov se izračuna po formuli  $d = (p_1 + p_2) p_3$ . V podjetju bi radi sredstva porazdelili med skupine tako, da bo pričakovani dobiček čim večji.

- (a) Zapiši rekurzivne enačbe za reševanje danega problema.

- (b) Z zgoraj zapisanimi enačbami reši problem pri podatkih  $m = 15$ ,  $a_1 = 4$ ,  $n_1 = 3$ ,  $k_1 = 1.5$ ,  $a_2 = 3$ ,  $n_2 = 4$ ,  $k_2 = 2$ ,  $a_3 = 2$ ,  $n_3 = 0.4$  in  $k_3 = 0.3$ .

**Naloga 4.20.**

Vir: Izpit OR 29.8.2017

V veliki multinacionalni korporaciji želijo, da bi se zakonodaja spremenila v njihov prid. V ta namen so najeli  $m$  lobistov, ki se bodo pogajali z  $n$  političnimi strankami, da pridobijo njihovo podporo pri spremembi zakonodaje. Vsak lobist se bo pogajal s samo eno stranko; k vsaki stranki lahko pošljejo več lobistov. Naj bo  $p_{ij}$  ( $0 \leq i \leq m$ ,  $1 \leq j \leq n$ ) verjetnost, da pridobijo podporo stranke  $j$ , če se z njo pogaja  $i$  lobistov (lahko predpostaviš  $p_{i-1,j} \leq p_{ij}$  za vsaka  $i, j$ ). Verjetnosti za različne stranke so med seboj neodvisne. Maksimizirati želijo verjetnost, da bodo lobisti pridobili podporo vseh  $n$  političnih strank.

- (a) Zapiši rekurzivne enačbe za reševanje danega problema.
- (b) Naj bo  $m = 6$  in  $n = 3$ . K vsaki stranki želijo poslati vsaj enega lobista (tj.,  $p_{0j} = 0$  za vsak  $j$ ), vrednosti  $p_{ij}$  za  $i \geq 1$  pa so podane v spodnji tabeli.

$p_{ij}$	1	2	3
1	0.2	0.4	0.3
2	0.5	0.5	0.4
3	0.7	0.5	0.8
4	0.8	0.6	0.9

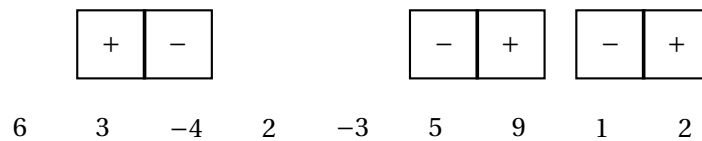
Za dane podatke reši problem z zgoraj zapisanimi enačbami.

**Naloga 4.21.**

Vir: Kolokvij OR 23.4.2018

Imamo zaporedje  $n$  polj, pri čemer je na  $i$ -tem polju zapisano število  $a_i$ . Na voljo imamo še  $\lfloor n/2 \rfloor$  domin, z vsako od katerih lahko pokrijemo dve sosednji polji. Vsaka domina je sestavljena iz dveh delov: na enem je znak  $+$ , na drugem pa znak  $-$ . Posamezno polje lahko pokrijemo z le eno domino; če sta pokriti dve sosednji polji, morata biti pokriti z različnima znakoma (bodisi z iste, bodisi z druge domine). Iščemo tako postavitev domin, ki maksimizira vsoto pokritih števil, pomnoženih z znakom na delu domine, ki pokriva število. Pri tem ni potrebno, da uporabimo vse domine. Primer je podan na sliki 10.

- (a) Zapiši rekurzivne enačbe za reševanje danega problema. Razloži, kaj predstavljajo spremenljivke, v kakšnem vrstnem redu jih računamo, ter kako dobimo optimalno rešitev.  
**Namig:** posebej obravnavaj dva primera glede na postavitev zadnje domine.
- (b) Oцени časovno zahtevnost algoritma, ki sledi iz zgoraj zapisanih enačb.
- (c) S svojim algoritmom poišči optimalno pokritje za primer s slike 10.



Slika 10: Primer dopustnega (ne nujno optimalnega) pokritja za nalogo 4.21. Vsota tega pokritja je  $3 - (-4) - 5 + 9 - 1 + 2 = 12$ . Če bi eno od zadnjih dveh domin obrnili (zamenjala bi se znaka), dobljeno pokritje ne bi bilo dopustno, saj bi dve zaporedni polji bili pokriti z enakima znakoma.

#### Naloga 4.22.

Vir: Izpit OR 5.7.2018

Vlagatelj ima na voljo 50 milijonov evrov sredstev, ki jih lahko porabi za donosno, a tvegano naložbo. Ocenjuje, da bi se mu ob uspehu naložbe vložek povrnil petkratno, verjetnost uspeha pa ocenjuje na 0.6. Zaradi tveganja se lahko odloči za zavarovanje naložbe, pri čemer ima ponudbi dveh zavarovalnic, ki mu proti plačilu ustrezne premije ponujata povračilo dela vložka, če bo naložba neuspešna. Vlagatelj lahko del sredstev obdrži tudi zase (tj., ga ne porabi za naložbo ali premijo).

Naj bodo torej  $x_1, x_2, x_3$  vrednosti v milijonih evrov, ki zaporedoma predstavljajo količine, ki jih vlagatelj obrzi zase, porabi za naložbo, in plača za zavarovalniško premijo. Pričakovana vrednost naložbene strategije vlagatelja (tj., količina denarja, ki jo ima na koncu) je potem

$$x_1 + x_2(0.6 \cdot 5 + 0.4q(x_3)),$$

kjer  $q(x_3)$  predstavlja delež vložka, ki ga glede na vloženo premijo zavarovalnica povrne ob neuspehu naložbe.

Vlagatelj ima dve ponudbi konkurenčnih zavarovalnic. Zavarovalnica Zvezna d.z.z. za premijo v višini  $x_3$  milijonov evrov ponuja povračilo deleža  $0.15x_3$  celotne naložbe v primeru neuspeha, pri čemer je največja možna premija 4 milijone evrov. Zavarovalnica Diskretna d.d.z. pa ponuja le tri možne premije:

premija	delež povračila ob neuspešni naložbi
1 milijon evrov	0.1
2 milijona evrov	0.35
3 milijoni evrov	0.5

Pogodbo smemo skleniti samo pri eni zavarovalnici.

- Zapiši definicijo funkcije  $q(x)$  skupaj z izbiro najugodnejše zavarovalnice pri vsakem  $x$ .
- Zapiši rekurzivne formule za določitev strategije vlaganja, ki nam bo prinesla največji pričakovani dobiček.
- S pomočjo zgornjih rekurzivnih enačb ugotovi, kako naj ravna vlagatelj, da bo imel čim večji dobiček.

**Naloga 4.23.**

Vir: Izpit OR 28.8.2018

Pri direkciji za ceste načrtujejo nov avtocestni odsek dolžine  $M$  kilometrov. Ob cesti želijo zgraditi počivališča tako, da je razdalja med dvema zaporednima počivališčema največ  $K$  kilometrov. Prav tako mora biti prvo počivališče največ  $K$  kilometrov od začetka, zadnje pa največ  $K$  kilometrov od konca avtocestnega odseka. Naj bodo  $x_1 < x_2 < \dots < x_n$  možne lokacije počivališč (v kilometrih od začetka avtocestnega odseka), in  $c_i$  ( $1 \leq i \leq n$ ) cena izgradnje počivališča na lokaciji  $x_i$ . Postavitev počivališč želijo izbrati tako, da bo skupna cena izgradnje čim manjša.

- Zapiši rekurzivne enačbe za reševanje danega problema. Razloži, kaj predstavljajo spremenljivke, v kakšnem vrstnem redu jih računamo, ter kako dobimo optimalno rešitev.
- Oceni časovno zahtevnost algoritma, ki sledi iz zgoraj zapisanih enačb.
- S pomočjo rekurzivnih enačb reši zgornji problem za podatke

$$\begin{aligned} M &= 100, & (x_i)_{i=1}^8 &= (5, 12, 22, 34, 49, 65, 83, 91), \\ K &= 30, & (c_i)_{i=1}^8 &= (18, 11, 21, 16, 23, 15, 19, 13). \end{aligned}$$

**Naloga 4.24.**

Vir: Kolokvij OR 26.1.2010

Pri neznanem problemu, ki smo se ga lotili z dinamičnim programiranjem, smo prišli do naslednje rekurzivne zveze:

$$c_{ij} = \max\{c_{i-1,j-1}, c_{i+1,j-1} + 2c_{i-1,j+1}\}.$$

Izračunati želimo vrednost  $c_{mn}$ . Katere vrednosti  $c_{ij}$ , pri čemer je  $(i, j)$  zunaj  $\{1, \dots, m\} \times \{1, \dots, n\}$ , moramo poznati, da bo  $c_{mn}$  enolično določen? Poišči čim manjšo takšno množico.

**Naloga 4.25.**

Vir: Kolokvij OR 26.1.2010

Upravljalca nove moderne visoko tehnološko opremljene kongresne dvorane v Kongresnem centru Grebka Pohleparja zaračunava za najem dvorane 5€/min. Povpraševanje za najem nove dvorane je ogromno, a se na žalost ponudbe časovno prekrivajo, dvorano pa seveda lahko najame le ena stranka naenkrat. Tako se je upravljalca že prvi dan znašel pod goro ponudb, med katerimi je napol v paniki izbiral, bodimo pošteni, tako bolj "čez palec". A za kongresni center, ki nosi ime takega velikana ekonomije, se spodobi, da si uredi vse segmente poslovanja čim bolj optimalno in je kot tak zgled drugim. Predpostavimo, da ima upravljalca v trenutku odločanja podatke v obliki intervalov  $(z_i, k_i)$ ,  $i = 1, \dots, n$ , kjer  $z_i$  in  $k_i$  predstavljata začetni in končni čas ponudbe  $i$ . Ponudbi se prekrivata, če je njun presek neprazen časovni interval.

- (a) Denimo, da uredimo intervale ponudb glede na začetne čase ponudb od najmanjšega do največjega. Potem izberemo prvi interval, vse intervale, ki ga prekrivajo, pa zavržemo, ter postopek ponovimo rekurzivno. Upravljalcu se ta postopek zdi optimalen. Kaj pa meniš ti?
- (b) Denimo, da so intervali ponudb urejeni kot v prejšnji točki ter da je interval  $I$  zadnji interval v neki optimalni izbiri OPT. Premisli, kateri intervali so lahko kandidati za predzadnji interval v izbiri OPT. Na podlagi tega premisleka sestavi postopek, ki s pomočjo dinamičnega programiranja najde optimalno rešitev problema. Postopek utemelji.
- (c) Kakšna je časovna zahtevnost tvojega algoritma v najslabšem primeru?
- (d) Z uporabo razvitega algoritma reši problem za podatke  $(15, 60)$ ,  $(60, 180)$ ,  $(30, 90)$ ,  $(120, 210)$ ,  $(165, 255)$ ,  $(75, 135)$ ,  $(30, 105)$ ,  $(90, 150)$ .

**Naloga 4.26.**

Vir: Kolokvij OR 26.1.2010

Jatifi Bajrami se ukvarja s preprodajo pomaranč. Naslednja sezona je razdeljena na  $n$  mesecev. Za vsak mesec so znana predvidena povpraševanja  $r_1, \dots, r_n$ . Nabavni stroški v mesecu  $i$  so sestavljeni iz fiksnih stroškov  $K_i$  ter stroškov za nabavo sadja. Na začetku meseca  $i$  stane škatla pomaranč  $c_i$ . Če Jatifi takrat kupi  $m$  škatel, je za nabavo sadja torej moral odšteti  $K_i + mc_i$  denarnih enot. Če se Jatifi zaradi nižje cene v nekem mesecu odloči nakupiti vnaprej za več mesecev, mora plačati skladiščenje za škatle pomaranč za preostale mesece, pri čemer je cena skladiščenja za škatlo v mesecu  $i$  enaka  $h_i$ . Jatifi kupuje izključno, ko ve, da so mu zaloge popolnoma pošle, in od tega ne odstopa, saj se mu zdi škoda, da bi v skladišču hranil stare pomaranče. Pomaranče prodaja ves čas po isti ceni. Jatifi bi rad maksimiziral dobiček in zadovoljil celotno povpraševanje.

- (a) Predlagaj rešitev problema s pomočjo dinamičnega programiranja. Pri tem so vhodni podatki  $n$  mesecev,  $r = (r_i)_{i=1}^n$ ,  $K = (K_i)_{i=1}^n$ ,  $c = (c_i)_{i=1}^n$  ter  $h = (h_i)_{i=1}^n$ . Kakšno časovno zahtevnost ima tvoj algoritem?
- (b) Naj bo  $n = 4$ ,  $r = (20, 40, 15, 10)$ ,  $K = (2, 3, 4, 2)$ ,  $c = (3, 4, 3, 4)$  ter  $h = (1, 1, 1, 2)$ . Poišči optimalno strategijo kupovanja pomaranč.

**Naloga 4.27.**

Vir: Izpit OR 5.2.2010

Dana je rekurzija

$$c_{ij} = \min_{i \leq k \leq j} \left( c_{i-1,k} + \sum_{u=i}^k a_{iu} \right), \quad c_{0j} = 1, \quad (1 \leq i \leq j \leq n)$$

kjer je  $A = (a_{ij})_{i,j=1}^n$  matrika realnih števil. Pokaži in zelo natančno utemelji, da lahko izračunaš  $c_{nn}$  v času  $O(n^3)$ .

**Naloga 4.28.**

Vir: Izpit OR 5.2.2010

Na znanem TV kanalu HOPtv bodo ob sredah ob 17h predvajali humanitarno igro Prodajalec™, pri kateri igralec prodaja predmete tipa  $X$  in predmete tipa  $Y$  v določenem zaporedju  $P$ , v studio pa po vrsti prihajajo potencialni kupci. Organizatorji vnaprej določijo zaporedje kupcev in  $i$ -temu kupcu določijo tudi tip predmeta  $T_i$ . Kupec  $i$  lahko sam poda le ceno  $c_i$ , ki jo je pripravljen plačati za predmet. Organizatorji na podlagi tega določijo zaporedje  $P$  tako, da je izvedljiva prodaja vseh predmetov v zaporedju  $P$ . Podatkov  $P$ ,  $T$  in  $c$  igralec seveda ne pozna vnaprej. Kupci prihajajo v studio in igralec se odloča, ali jim za ponujeno ceno trenutni predmet proda ali ne. Pri tem mu svetuje in vpliva na njegov humanitarni čut znani voditelj Joras Š. Če igralec trenutnega predmeta ne proda  $i$ -temu kupcu, pride naslednji kupec, če pa ga proda, pa prisluži za humanitarne namene  $c_i$  enot denarja in naslednjemu kupcu poskusi prodati naslednji predmet v zaporedju  $P$ . V sezoni zmaga igralec, katerega prodaja je procentualno največja glede na optimalno prodajo. Na HOPtv so na žalost kupili le okrnjeno licenco za igro Prodajalec™, ki ne vključuje programske opreme za preračun optimalnih rešitev, tako da te prosijo za pomoč. Za primer: če so podatki  $T = XYXXY$ ,  $c = (2, 2, 7, 1, 1)$  in  $P = XY$ , je optimalna rešitev prodaja  $X$  tretjemu kupcu ter prodaja  $Y$  petemu kupcu, s katero prislužimo  $c_3 + c_5 = 7 + 1 = 8$  enot denarja.

- (a) S pomočjo dinamičnega programiranja izpelji algoritem, ki za dane  $T$ ,  $c$  in  $P$  poišče optimalno rešitev. Podaj ustrezno rekurzivno zvezo ter opiši, kako rekonstruiramo optimalno rešitev.  
**Namig:** zgleduj se po algoritmu za najdaljše skupno podzaporedje.
- (b) Kakšna je časovna zahtevnost tvojega algoritma v najslabšem primeru?
- (c) Izvedi algoritem na zgornjih podatkih in tako preveri, da je opisana rešitev v zgornjem primeru res optimalna.

**Naloga 4.29.**

Vir: Izpit OR 28.6.2010

Podatke (nize) stiskamo na naslednji način. Podana je tabela  $m$  nizov, dolžina vsakega je kvečjemu  $k$ . Radi bi zakodirali podatkovni niz  $D$  dolžine  $n$  z najkrajšim možnim zaporedjem nizov iz tabele. Npr., če naša tabela vsebuje nize  $(a, ba, abab, b)$  in bi radi zakodirali niz  $bababbaababa$ , je najboljši način kodiranja zaporedje  $(b, abab, ba, abab, a)$ , torej s petimi kodnimi besedami.

- (a) S pomočjo strategije dinamičnega programiranja izpelji čim bolj učinkovit algoritem, ki poišče najkrajše kodiranje danega niza  $D$ , če kodiranje za tak niz obstaja.
- (b) Demonstriraj uporabo algoritma na podanem nizu  $bababbaababa$  s podano tabelo  $(a, ba, abab, b)$ .
- (c) Kakšna je časovna zahtevnost tvoje rešitve?

**Naloga 4.30.**

Vir: Kolokvij OR 24.1.2011

Danih imamo  $n$  kock, ki jih želimo zložiti v čim višji stolp. Dimenzije  $i$ -te kocke so  $a_i \times b_i \times c_i$ , kjer je  $a_i$  dolžina,  $b_i$  širina in  $c_i$  višina. Kock ne smemo rotirati, kocko  $i$  pa lahko postavimo na kocko  $j$  samo, če je  $a_i < a_j$  in  $b_i < b_j$ . Brez škode za splošnost lahko predpostavimo, da je  $a_1 \geq a_2 \geq \dots \geq a_n$ . S pomočjo dinamičnega programiranja poišči najvišji možni stolp.

**Naloga 4.31.**

Vir: Izpit OR 9.2.2011

Znanstveniki proučujejo novo vrsto bakterije. V DNK te bakterije so našli naslednje zaporedje nukleotidov:

AACAGTTA.

Sumijo, da je to različica gena že znane bakterije, ki ima zaporedje

ACCATGTA.

Če sta zaporedji dovolj podobni, obstaja verjetnost, da imata gena podobni funkciji. Dovoljene so naslednje operacije:

- substitucija ( $ACT \rightarrow AGT$ ),
  - vstavljanje ( $AC \rightarrow AGC$ ),
  - izbris ( $AGC \rightarrow AC$ ) in
  - transpozicija ( $AT \rightarrow TA$ ).
- (a) Pomagaj znanstvenikom in napiši postopek, ki bo preštel najmanjše število operacij, s katerim pridemo od enega zaporedja nukleotidov do drugega.
- (b) Postopek iz prejšnje točke izvedi nad podanima zaporedjema in ugotovi število razlik.

**Naloga 4.32.**

Vir: Izpit OR 28.6.2011

Dano imamo zaporedje simbolov  $\top$  (resnično) in  $\perp$  (neresnično). Med vsakima simboloma imamo veznik  $\wedge$  (prvi in drugi),  $\vee$  (prvi ali drugi) ali  $\oplus$  (prvi ali drugi, ne pa tudi oba), npr.  $\top \wedge \top \oplus \perp$ . V zaporedje postavljamo oklepaje, in sicer tako, da se znotraj vsakega para oklepajev (tj., oklepaj in ustrezni zaklepaj) pojavita dva podizraza, ločena z veznikom. Podizraz je lahko posamezen simbol ali par oklepajev s svojo vsebino. Izračunaj število postavitvev oklepajev, da je logična vrednost celotnega izraza  $\top$ , in število postavitvev oklepajev, da je rezultat  $\perp$ .

**Namig:** definiraj  $T_{ij}$  in  $F_{ij}$  kot število takih postavitvev oklepajev med  $i$ -tim in  $j$ -tim simbolom.



**Naloga 4.33.**

Vir: Kolokvij OR 31.5.2012

Samostojna umetnica z nadimkom Stekelce se preživlja z barvanjem na steklo. Tokrat je dobila naročilo za izdelavo dveh zelo posebnih pobarvanih steklenih kroglic. Plačilo je dobila vnaprej, stroške pa bo imela le z nabavo stekla, saj ima barv in čopičev že dovolj na zalogi. Če ne uspe izdelati obeh kroglic v dogovorjenem času, bo morala plačati 800€ kazni. Steklene kroglice lahko naroči v steklarni, in sicer jo vsaka stane 100€. Steklarna ima stroške vsakič, ko zažene proizvodnjo teh kroglic, zato ji za vsako naročilo (ne glede na število naročenih kroglic) zaračuna dodatnih 300€. Kljub temu, da mora Stekelce zmeraj plačati vse dobljene kroglice, je za vsako le 50% verjetnost, da bo zanjo uporabna. Neuporabne kroglice lahko vrže kar v smeti, odvečne uporabne kroglice pa zanjo prav tako niso vredne nič in jih bo brez stroškov zavrnila.

- (a) Ker je naročilo časovno omejeno, ima Stekelce čas za največ dve naročili v steklarni. Po koliko kroglic naj vsakič naroči, da bodo pričakovani stroški čim manjši? (Število naročenih kroglic pri drugem naročilu bo seveda odvisno od števila dobrih kroglic pri prvem naročilu.)
- (b) Kaj pa, če steklarna v prvi seriji za naročilo namesto 300€ zahteva le 150€, v drugi seriji pa stroški naročila ostanejo 300€?

**Naloga 4.34.**

Vir: Izpit OR 9.7.2012

Dan je številski trikotnik

$$\begin{array}{ccccccc}
 & & & & a_{11} & & \\
 & & & & a_{21} & a_{22} & \\
 & & a_{31} & & a_{32} & a_{33} & \\
 & \ddots & & \vdots & & \vdots & \ddots \\
 a_{n1} & & a_{n2} & & \cdots & a_{n,n-1} & a_{nn}
 \end{array} ,$$

kjer so  $a_{ij} \in \mathbb{Z}$ . Spust v takem trikotniku je pot od vrha do dna – t.j., zaporedje  $(a_{i,j_i})_{i=1}^n$ , kjer je  $j_1 = 1$  in  $j_i \in \{j_{i-1}, j_{i-1} + 1\}$  za vsak  $i = 2, \dots, n$ . Teža spusta je vsota števil, po katerih poteka spust (torej  $\sum_{i=1}^n a_{i,j_i}$ ).

S pomočjo dinamičnega programiranja opiši postopek, ki poišče najtežji spust v danem številskem trikotniku.

**Naloga 4.35.**

Vir: Izpit OR 4.9.2012

Antonio se preživlja z graviranjem na zlato. Ravnokar je dobil naročilo za graviranje dveh zelo umetelno izdelanih zaročnih prstanov. Če ne uspe izdelati prstanov v dogovorjenem času, bo moral plačati 1 600€ kazni. Prstane lahko naroči pri zlatarju, ki ga je izbral naročnik. Ta zlatar bo izdelavo vsakega prstana zaračunal po 100€, varovana dostava pa (ne glede na število prstanov) stane 200€. Ker prstani niso čisto homogeni, je verjetnost, da graviranje uspe, le 50%. Ves ostali

material (neuspeli prstani, opilki ipd.) bo moral na koncu vrniti zlatarju (tako je določeno v pogodbi). Ker je naročilo časovno omejeno, ima časa za največ 2 naročila pri zlatarju. Koliko prstanov naj vsakič naroči, da bodo pričakovani stroški čim manjši?

**Naloga 4.36.**  
(pravična delitev)

Vir: Kolokvij OR 17.4.2014

Dana je množica naravnih števil  $\{a_1, a_2, \dots, a_n\}$ . Naš cilj je razdeliti to množico na dva dela  $S_1$  in  $S_2$  tako, da bo absolutna vrednost razlike med vsoto elementov v  $S_1$  in  $S_2$  čim manjša. S pomočjo dinamičnega programiranja opiši postopek, ki izračuna minimalno vrednost te razlike.

**Naloga 4.37.**

Vir: Izpit OR 28.8.2013

Za šestimi gorami poteka tekmovanje v plezanju na goro Lep Trikotnik. Vsak odsek te gore ima prirejeno težavnost od  $-4$  do  $4$ . Goro lahko ponazorimo s številskim trikotnikom

$$\begin{array}{ccccccc}
 & & & a_{11} & & & \\
 & & & a_{21} & & a_{22} & \\
 & & a_{31} & & a_{32} & & a_{33} \\
 & \ddots & & \vdots & & \vdots & \ddots \\
 a_{n1} & & a_{n2} & & \cdots & & a_{n,n-1} & & a_{nn}
 \end{array} ,$$

kjer so  $a_{ij} \in \{-4, -3, \dots, 3, 4\}$  težavnosti odsekov. *Veljavna pot* v takem trikotniku je pot od vznožja do vrha, ki gre zmeraj desno navzgor ali levo navzgor – t.j., zaporedje  $(a_{n-i,j_i})_{i=0}^{n-1}$ , kjer je  $1 \leq j_i \leq n-i$  in  $j_i \in \{j_{i-1}-1, j_{i-1}\}$  za vsak  $i = 0, 1, \dots, n-1$ . *Teža* veljavne poti je vsota težavnosti odsekov, po katerih poteka pot (torej  $\sum_{i=0}^{n-1} a_{n-i,j_i}$ ).

Na tekmovanje sta se med drugimi prijavila tudi gornika Marin in Mirko. V okviru svojih sposobnosti želita izbrati kar najtežjo pot do vrha. Oba zmoreta plezati po odsekih težavnosti 3 ali manj.

- Marin ni dovolj izkušen, da bi lahko plezal po odsekih težavnosti 4. Opiši, kako lahko najdemo najtežjo veljavno pot za Marina, oz. mu povemo, da s svojim znanjem ne more priti do vrha po veljavni poti!
- Gornik Mirko je izkušenejši od Marina. Odseke težavnosti 4 lahko prepleza, a ne more preplezati dveh zaporednih odsekov te težavnosti. Opiši, kako lahko najdemo najtežjo veljavno pot za Mirka, oz. mu povemo, da s svojim znanjem ne more priti do vrha po veljavni poti!

**Naloga 4.38.**

Vir: Izpit OR 5.6.2014

Dano je zaporedje števil  $c_1, c_2, \dots, c_n$ . Igralca  $A$  in  $B$  igrata igro, pri kateri  $A$  bodisi z začetka bodisi s konca danega zaporedja izbere eno (prej še neizbrano) število,  $B$  pa tako izbira dvakrat zapored. Igrata, dokler števil ne zmanjka. Zmaga igralec z največjo skupno vsoto izbranih števil. Denimo, da začne igralec  $A$ . S pomočjo dinamičnega programiranja določi maksimalni znesek, ki ga  $A$  lahko doseže, če tudi drugi igralec igra optimalno.

**Naloga 4.39.**

Vir: Izpit OR 24.6.2014

Kmet redi ovce, pri čemer se vsako leto njihovo število podvoji: če ima na začetku leta  $N$  ovac, jih ima ob koncu leta  $2N$ . Denimo, da ima kmet  $k_0$  ovac na začetku leta 0. Ob prehodu iz leta  $i - 1$  v leto  $i$  se vsakič odloči, koliko ovac bo prodal, pri čemer je profit enak  $p_i$  za vsako prodano ovco. Ob začetku leta  $n$  hoče imeti prodane vse ovce.

- (a) S pomočjo dinamičnega programiranja opiši algoritem, ki za dane podatke  $n, k_0$  in  $p_1, p_2, \dots, p_n$  izračuna maksimalni profit, ki ga lahko kmet doseže po  $n$  letih.
- (b) Reši nalogo za  $n = 3$ ,  $k_0 = 2$ ,  $p_1 = 100\text{€}$ ,  $p_2 = 130\text{€}$  in  $p_3 = 120\text{€}$ .

**Naloga 4.40.**

Vir: Izpit OR 25.8.2014

Na voljo imamo  $w$  delavcev, ki jih želimo razporediti med  $N$  opravil. Dane imamo verjetnosti  $p_k(j)$ , da bo  $k$ -to opravilo uspešno opravljeno, če na njem dela  $j$  delavcev.

- (a) S pomočjo dinamičnega programiranja opiši algoritem, ki izračuna maksimalno verjetnost, da bo vseh  $N$  opravil uspešno opravljenih (opravila so med seboj neodvisna).
- (b) Izračunaj maksimalno verjetnost uspešno opravljenih opravil in določi število delavcev na posameznem opravilu za  $N = 3$ ,  $w = 5$  in verjetnosti iz spodnje tabele:

$j$	0	1	2	3	4	5
$p_1(j)$	0	0.5	0.6	0.7	0.8	0.85
$p_2(j)$	0	0.4	0.7	0.8	0.9	0.95
$p_3(j)$	0	0.3	0.6	0.8	0.85	0.85

**Naloga 4.41.**

Vir: Izpit OR 24.6.2015

Tajni agent slovenske obveščevalne službe nam je prinesel besedilo v neznanem tujem jeziku, brez ločil in brez presledkov. V bistvu smo dobili en zelo dolg niz znakov (recimo dolžine  $n$ ). Ker v obveščevalni službi sumijo, da je besedilo v turkmensčini, so nam prinesli elektronski seznam veljavnih turkmenskih besed (ki so ga dobili iz skeniranja več turkmenskih spletnih strani ter iz slovarja njihovega knjižnega jezika). Ta seznam besed nam pomaga, da lahko hitro preverimo, ali je dana beseda veljavna.

Naša naloga je, da preverimo, ali lahko besedilo razdelimo na zaporedje veljavnih turkmenskih besed. Reši nalogo s pomočjo dinamičnega programiranja.

**Naloga 4.42.**

Vir: Izpit OR 28.8.2015

V podjetju Hierarhija so delavci razvrščeni hierarhično tako, da za vsaka dva delavca vemo, kdo je komu nadrejen. Če povemo še bolj natančno, delavce lahko postavimo v vrsto tako, da so za vsakega delavca levo od njega podrejeni, desno pa nadrejeni.

Recimo, da je v podjetju  $n$  delavcev, označimo jih z  $d_1, d_2, \dots, d_n$ . Predpostavimo tudi, da so že urejeni hierarhično, tj., delavec  $d_i$  je podrejen delavcem  $d_{i+1}, d_{i+2}, \dots, d_n$ . Vsak delavec se boji prvih nekaj delavcev, ki so mu podrejeni, da bi ga prehiteli v hierarhiji. Naj bo  $k_i \in \{0, 1, \dots, i-1\}$  stopnja ogroženosti delavca  $d_i$  ( $1 \leq i \leq n$ ). Stopnja ogroženosti nam pove, koliko delavcev, ki so podrejeni delavcu  $d_i$ , se ta delavec "boji". Z drugimi besedami, delavec  $d_i$  se boji delavcev  $d_{i-1}, d_{i-2}, \dots, d_{i-k_i}$ .

Z dinamičnim programiranjem poišči tako največjo skupino delavcev, da se nihče v tej skupini ne bo bal drugega iz te skupine. Dovolj je zapisati rekurzivno enačbo z začetnimi pogoji. Kako lahko izvemo, kateri delavci so v tej (največji) skupini?

**Naloga 4.43.**

Vir: Izpit OR 12.5.2016

Dan je seznam števil  $A = [a_1, a_2, \dots, a_n]$ . Želimo poiskati maksimalno vsoto takega podzaporedja v  $A$ , ki ne vsebuje dveh zaporednih členov seznama.

- (a) Napiši rekurzivne enačbe za reševanje opisanega problema.
- (b) S pomočjo dinamičnega programiranja napiši algoritem, ki rešuje dani problem. Kakšna je njegova časovna zahtevnost?

**Naloga 4.44.**

Vir: Izpit OR 24.5.2016

Dan je seznam pozitivnih celih števil  $S = [a_1, a_2, \dots, a_n]$ , ki ga interpretiramo na naslednji način. Element  $a_i$  pomeni, da se lahko iz  $i$ -te pozicije v seznamu premaknemo na pozicije  $a_{i+1}, a_{i+2}, \dots, a_{i+a_i}$ . Naj bo  $f(S)$  minimalno število korakov, ki so potrebni, da se iz elementa  $a_1$  premaknemo v element  $a_n$ . Na

primer, če je  $S = [1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9]$ , potem je  $f(S) = 3$ , saj lahko opravimo skoke  $a_1 = 1 \rightarrow a_2 = 3 \rightarrow a_4 = 8 \rightarrow a_{11} = 9$ .

- (a) Napiši rekurzivne enačbe za računanje funkcije  $f(S)$ .
- (b) S pomočjo dinamičnega programiranja napiši algoritem, ki rešuje dani problem. Kakšna je njegova časovna zahtevnost?

**Naloga 4.45.**

Vir: Izpit OR 24.5.2016

Dana je matrika  $A$  dimenzij  $n \times n$ , katere elementi so 0 in 1. *Strnjena podmatrika*  $A'$  matrike  $A$  je matrika, določena z zaporednimi vrsticami in stolpci matrike  $A$ . *Velikost* matrike je število elementov v matriki. Velikost podmatrike dimenzij  $n_1 \times n_2$  je torej  $n_1 n_2$ . Naj bo  $N(A)$  največja velikost take podmatrike  $A'$  matrike  $A$ , da  $A'$  vsebuje samo enice. Na primer:

$$N\left(\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}\right) = 4, \quad N\left(\begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}\right) = 3, \quad \text{in} \quad N\left(\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}\right) = 2.$$

- (a) Napiši rekurzivne enačbe za računanje  $N(A)$ .
- (b) Napiši algoritem za računanje  $N(A)$  s časovno zahtevnostjo največ  $O(n^4)$ .

**Naloga 4.46.**

Vir: [5, §18, Example 5]

Podjetje ima na voljo 6000€ za investiranje. Na voljo so tri investicije, pri katerih je donosnost (v 1000€) enaka

$$\begin{aligned} r_1(d_1) &= \begin{cases} 7d_1 + 2, & \text{če } d_1 \geq 1, \text{ in} \\ 0 & \text{sicer;} \end{cases} \\ r_2(d_2) &= \begin{cases} 3d_2 + 7, & \text{če } d_2 \geq 1, \text{ in} \\ 0 & \text{sicer;} \end{cases} \quad \text{ozioroma} \\ r_3(d_3) &= \begin{cases} 4d_3 + 5, & \text{če } d_3 \geq 1, \text{ in} \\ 0 & \text{sicer;} \end{cases} \end{aligned}$$

kjer so  $d_1, d_2, d_3$  vložki v vsako investicijo v 1000€. Kako naj podjetje investira svoj denar, da bo zaslužek čim večji?

**Naloga 4.47.**

Vir: Kolokvij OR 19.4.2019

Gradimo avtocesto skozi puščavo in želimo zagotoviti, da bo v celoti pokrita z mobilnim signalom. Cesta je ravna in dolga  $n$  milj ( $n \in \mathbb{Z}$ ), na vsako miljo pa imamo možnost postaviti bazno postajo z dosegom 1 miljo. Cena postavitve bazne postaje na  $i$ -ti milji je podana s parametrom  $a_i$  ( $0 \leq i \leq n$ ). Predpostaviš lahko, da so vse cene pozitivne. Pri obstoječi infrastrukturi je pokrita le začetna točka avtoceste (tj., milja 0). Poiskati želimo torej čim cenejšo postavitev baznih postaj, da je vsaka točka avtoceste pokrita s signalom.

**Primer:** če postavimo postaji na milji 0 in 3, potem je interval  $(1, 2)$  nepokrit. Če pa npr. postajo namesto na milji 0 postavimo na milji 1, smo tako v celoti pokrili interval  $[0, 4]$ .

- (a) Zapiši rekurzivne enačbe za reševanje danega problema. Razloži, kaj predstavljajo spremenljivke, v kakšnem vrstnem redu jih računamo, ter kako dobimo optimalno rešitev.
- (b) Oцени časovno zahtevnost algoritma, ki sledi iz zgoraj zapisanih enačb.
- (c) S svojim algoritmom poišči optimalno postavitev postaj za  $n = 10$  ter

$$(a_i)_{i=0}^{10} = (4, 6, 1, 10, 14, 21, 15, 6, 10, 3, 2).$$

- (d) Denimo, da lahko postavimo tudi večje bazne postaje z dosegom 2 milji, pri čemer je cena postavitve take postaje na  $i$ -ti milji podana s parametrom  $b_i$  ( $0 \leq i \leq n$ ). Zapiši rekurzivne enačbe, ki bodo upoštevale tudi to možnost.
- (e) Problem iz točke (d) reši s podatki iz točke (c) in

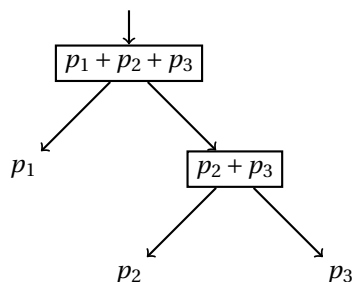
$$(b_i)_{i=0}^{10} = (10, 12, 3, 18, 24, 25, 20, 11, 16, 7, 4).$$

**Naloga 4.48.**

Vir: Izpit OR 19.6.2019

Za novo postavljeno obrtno cono želimo napeljati vodovodno omrežje. Imamo  $n$  delavnic v ravni vrsti, za  $i$ -to delavnico pa imamo podano porabo vode  $p_i$ . Iz javnega vodovoda bodo napeljali cev, preko katere bo priteklo dovolj vode za vseh  $n$  delavnic, mi pa želimo postaviti razdelilnike, ki bodo poskrbeli za ustrezno razdelitev vode med delavnice. Vsak razdelilnik ima eno vhodno cev in dve izhodni, vsaka od njiju pa bo pripeljala vodo do zaporednih delavnic (po potrebi preko nadaljnjih razdelilnikov). Cena postavitve razdelilnika je sorazmerna porabi delavnic, ki jim služi. Razdelilnike želimo postaviti tako, da bo skupna cena čim manjša.

**Primer:** na sliki 11 je prikazana možna postavitev razdelilnikov za tri delavnice. Odločimo se lahko, ali bomo najprej razdelili vodo delavnici 1 in delavnicama 2,3, ali pa bomo vodo peljali naprej do delavnic 1,2 in do delavnice 3 (ne moremo pa deliti na delavnici



Slika 11: Primer postavitve razdelilnikov za nalogo 4.48.

1, 3 in delavnico 2). Če se odločimo za prvi primer, potem do delavnice 1 ne potrebujemo drugih razdelilnikov, še enega pa moramo postaviti, da razdelimo vodo do delavnic 2 in 3. Cena postavitve prvega razdelilnika je tako  $p_1 + p_2 + p_3$  (ne glede na odločitev), za drugega pa je v tem primeru cena  $p_2 + p_3$ .

- (a) Naj bo  $c_i = \sum_{h=1}^i p_h$  ( $0 \leq i \leq n$ ). Zapiši rekurzivne enačbe za čim učinkovitejši izračun teh vrednosti.
- (b) Zapiši rekurzivne enačbe za reševanje danega problema. Razloži, kaj predstavljajo spremenljivke, v kakšnem vrstnem redu jih računamo, ter kako dobimo optimalno rešitev. Kakšna je časovna zahtevnost algoritma?
- Namig:** pomagaj si z vrednostmi  $c_i$  iz prejšnje točke.
- (c) S pomočjo rekurzivnih enačb reši zgornji problem za  $n = 6$  in

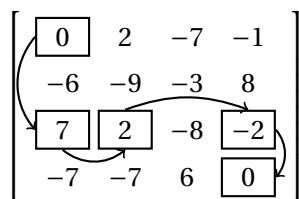
$$(p_i)_{i=1}^6 = (4, 19, 17, 7, 5, 9).$$

#### Naloga 4.49.

Vir: Izpit OR 4.6.2020

Dana je matrika  $A$  dimenzij  $m \times n$ . Iščemo tako zaporedje skokov po matriki, pri čemer začnemo levo zgoraj in končamo desno spodaj, v vsakem koraku pa skočimo za vsaj eno mesto desno ali dol, pri katerem je vsota obiskanih mest čim večja. Drugače povedano, iščemo tako zaporedje indeksov  $(i_1, j_1), (i_2, j_2), \dots, (i_k, j_k)$  (za nek  $k \geq 1$ ), za katere velja  $(i_1, j_1) = (1, 1)$ ,  $(i_k, j_k) = (m, n)$  in  $(i_h < i_{h+1} \wedge j_h = j_{h+1}) \vee (i_h = i_{h+1} \wedge j_h < j_{h+1})$  za vse  $h$  ( $2 \leq h \leq k$ ), ki maksimizira vsoto  $\sum_{h=1}^k A_{i_h, j_h}$ . Primer je podan na sliki 12.

- (a) Zapiši začetne pogoje in rekurzivne enačbe za reševanje opisanega problema ter določi, v kakšnem vrstnem redu računamo spremenljivke in kako dobimo maksimalno vsoto. Kakšna je časovna zahtevnost algoritma, ki sledi iz rekurzivnih enačb?
- (b) Reši problem za matriko s slike 12 z uporabo rekurzivnih enačb.



Slika 12: Primer matrike dimenzij  $4 \times 4$  za nalogo 4.49 skupaj z dopustno (ne nujno optimalno!) rešitvijo. Vsota obiskanih mest je v tem primeru  $0 + 7 + 2 + (-2) + 0 = 7$ .

#### Naloga 4.50.

Vir: Izpit OR 27.8.2019

Vlagatelj ima na voljo  $m$  kapitala, del katerega bo vložil v eno izmed dveh konkurenčnih podjetij, ki razvijata podobna produkta (v obe ne more vložiti), preostanek pa bo namenil za marketinško kampanjo, ki bo promovirala produkt izbranega podjetja. Naj bodo  $x_1$ ,  $x_2$  in  $x_3$  količine denarja (te so lahko poljubna nenegativna realna števila), ki jih bo vložil v prvo oziroma drugo podjetje in v marketing, ter določimo

$$p_i = \begin{cases} n_i + k_i x_i, & \text{če } x_i \geq a_i, \text{ in} \\ 0 & \text{sicer} \end{cases} \quad (i = 1, 2, 3)$$

kot faktor, ki ga ustvari vsako izmed njih. Pričakovani dobiček se izračuna po formuli  $d = (p_1 + p_2)p_3$ , pri čemer bo eden od  $p_1$  in  $p_2$  enak 0. Vlagatelj bi rad sredstva porazdelil tako, da bo pričakovani dobiček čim večji.

- Zapiši enačbe za reševanje danega problema. Lahko predpostaviš, da velja  $p_i \geq 0$  za vse vrednosti  $x_i$  ( $i = 1, 2, 3$ ).
- Z zgoraj zapisanimi enačbami reši problem pri podatkih  $m = 10$ ,  $a_1 = 4$ ,  $n_1 = 8$ ,  $k_1 = 4$ ,  $a_2 = 5$ ,  $n_2 = 12$ ,  $k_2 = 2$ ,  $a_3 = 3$ ,  $n_3 = 4$  in  $k_3 = 1$ .

#### Naloga 4.51.

Vir: Kolokvij OR 12.4.2021

Zgraditi želimo avtocestni viadukt dolžine  $\ell$  metrov, ki bo prečkal dolino s težavnim terenom. Geologi in arhitekti so že pregledali teren in identificirali  $n$  mest, kamor bi lahko postavili podporne stebre. Vsako mesto je predstavljeno s številom  $x_i$  ( $1 \leq i \leq n$ ), ki predstavlja oddaljenost v metrih od začetka viadukta, pri čemer velja  $0 = x_1 < x_2 < \dots < x_n = \ell$ . Gradnja stebra na mestu  $x_i$  nas stane  $c_i$  (pri čemer velja  $c_1 = c_n = 0$ , saj prvi in zadnji steber predstavljata začetno in končno točko viadukta), razdalja do sosednjih stebrov ne sme preseči  $r_i$  – t.j., če zgradimo stebra na mestih  $x_i$  in  $x_j$  ter nobenega drugega vmes, mora veljati  $|x_i - x_j| \leq r_i$  in  $|x_i - x_j| \leq r_j$ . Poiskati želimo čim cenejšo postavitev stebrov, da bo mogoče v celoti zgraditi viadukt.



- (a) Zapiši rekurzivne enačbe za reševanje danega problema. Razloži, kaj predstavljajo spremenljivke, v kakšnem vrstnem redu jih računamo, ter kako dobimo optimalno rešitev.
- (b) Zapiši psevdokodo algoritma, ki sledi iz zgoraj zapisanih enačb, in oceni njegovo časovno zahtevnost. Algoritem naj vrne samo najmanjšo ceno postavitve stebrov, ne pa tudi same izbire mest za postavitev.
- (c) S svojim algoritmom poišči optimalno postavitev stebrov pri podatkih  $\ell = 1000$ ,  $n = 12$  in

$$\begin{aligned}(x_i)_{i=1}^{12} &= \{0, 49, 60, 119, 258, 309, 509, 688, 753, 857, 887, 1000\}, \\(c_i)_{i=1}^{12} &= \{0, 8, 9, 32, 14, 18, 9, 27, 47, 16, 22, 0\}, \\(r_i)_{i=1}^{12} &= \{300, 76, 75, 217, 251, 277, 273, 181, 168, 193, 137, 300\}.\end{aligned}$$

#### Naloga 4.52.

Vir: Kolokvij OR 31.3.2022

Dana je matrika  $M$  dimenzij  $2 \times n$ , pri čemer naj bodo  $a_i$  in  $b_i$  ( $1 \leq i \leq n$ ) vrednosti v prvi oziroma drugi vrstici ter  $i$ -tem stolpcu matrike  $M$ :

$$M = \begin{bmatrix} a_1 & a_2 & \dots & a_{n-1} & a_n \\ b_1 & b_2 & \dots & b_{n-1} & b_n \end{bmatrix}$$

Poiskati želimo tako podmnožico  $S$  komponent matrike  $M$  z največjo vsoto, tako da je mogoče med poljubnima elementoma v  $S$  priti samo s premiki levo, desno, gor ali dol (ne diagonalno!) tako, da ne zapustimo množice  $S$ .

**Primer** dopustne (ne nujno optimalne!) rešitve (elementi  $S$  so v okvirčkih):

$$\begin{bmatrix} \boxed{4} & \boxed{4} & -3 & \boxed{4} & -3 \\ -1 & \boxed{-2} & \boxed{4} & \boxed{-4} & \boxed{2} \end{bmatrix}$$

Rešitev bi še vedno bila dopustna, če bi iz  $S$  odstranili elementa v prvem oziroma zadnjem stolpcu. Če bi pa odstranili katerega od elementov v spodnji vrstici srednjih treh stolpcev, pa rešitev ne bi bila dopustna, saj bi množica  $S$  razpadla na dva dela, med katerima ne bi mogli prehajati z dovoljenimi premiki.

- (a) Zapiši rekurzivne enačbe, s katerimi poiščemo največjo vsoto elementov v množici  $S$ . Razloži, kaj predstavljajo spremenljivke, v kakšnem vrstnem redu jih računamo, ter kako dobimo optimalno rešitev.

**Namig:** uporabi različne spremenljivke glede na izbiro komponent v stolpcu.

- (b) Oceni časovno zahtevnost algoritma, ki sledi iz zgoraj zapisanih enačb.
- (c) S svojim algoritmom poišči optimalno rešitev (tj., največjo vsoto in še samo množico  $S$ ) na zgornjem primeru.

**Naloga 4.53.**

Vir: Kolokvij OR 31.3.2022

Vojaška enota je dobila nalogo, da prevzame nadzor nad pomembno oskrbovalno potjo. V ta namen morajo zasedati  $n$  mest, ki ležijo ob poti. Naloge se bodo lotili v  $n$  korakih: v prvem koraku bodo zajeli eno od mest, v vsakem naslednjem koraku pa bodo zasedli eno od mest, ki je sosednje tistim, ki jih že zasedajo (tj., če po  $i$  korakih zasedajo mesta  $j, j+1, \dots, j+i-1$ , bodo v  $(i+1)$ -vem koraku zasedli mesto  $j-1$  ali  $j+i$ ). Naj bo  $s_i$  količina sredstev, ki jih potrebujejo, da uspešno zasedejo  $i$ -to mesto (ta sredstva se pri tem porabijo),  $t_i$  pa količina sredstev, ki jih lahko pridobijo po uspešni zasedbi  $i$ -tega mesta (ta sredstva lahko potem porabljajo v sledečih korakih). Poveljnik enote želi določiti tak vrstni red zasedanja mest, pri katerem bo potrebno na začetku vložiti čim manj sredstev (seveda je potrebno pred vsakim korakom imeti zadostna sredstva – ta torej v nobenem trenutku ne smejo biti negativna).

- (a) S psevdokodo ali rekurzivnimi enačbami podaj čim bolj učinkovit algoritem za reševanje zgornjega problema. Jasno naj bo, v kakšnem vrstnem redu poteka računanje ter kako sestavimo končno rešitev.

**Namig:** algoritem naj izračuna tako minimalna potrebna sredstva za zasedbo izbranih mest, kot tudi sredstva, ki ostanejo na voljo po njihovi zasedbi.

- (b) Oцени časovno zahtevnost svojega algoritma.
- (c) S svojim algoritmom poišči optimalno rešitev za podatke  $n = 5$  in

$$(s_i)_{i=1}^5 = \{8, 10, 9, 11, 9\},$$

$$(t_i)_{i=1}^5 = \{6, 4, 7, 12, 3\}.$$

## 1.5 Algoritmi na grafih

### Naloga 5.1.

Vir: Vaje OR 20.4.2016

Zasnuj podatkovno strukturo za grafe, ki temelji na matrični predstavitvi. Podatkovna struktura naj ima sledeče metode:

- $\text{INIT}(G)$ : ustvarjanje praznega grafa
- $\text{ADDVERTEX}(G, u)$ : dodajanje novega vozlišča
- $\text{ADDEDGE}(G, u, v)$ : dodajanje nove povezave
- $\text{DELEDGE}(G, u, v)$ : brisanje povezave
- $\text{DELVERTEX}(G, u)$ : brisanje vozlišča
- $\text{ADJ}(G, u)$ : seznam sosedov danega vozlišča

Za vsako od naštetih metod podaj tudi njeno časovno zahtevnost v odvisnosti od števila vozlišč, števila povezav in stopenj vhodnih vozlišč. Oceni tudi prostorsko zahtevnost celotne strukture.

### Naloga 5.2.

Vir: Vaje OR 20.4.2016

Zasnuj podatkovno strukturo za grafe, ki temelji na seznamih sosedov. Zapiši metode kot pri nalogi 5.1 ter oceni njihovo časovno zahtevnost in prostorsko zahtevnost celotne strukture.

### Naloga 5.3.

Vir: Vaje OR 20.4.2016

Kako moramo spremeniti strukturi iz nalog 5.1 in 5.2, da bosta predstavljali digrafe?

### Naloga 5.4.

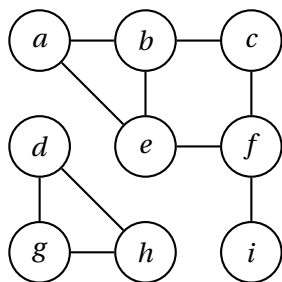
Vir: Vaje OR 20.4.2016

Napiši algoritem, ki za vhodni graf  $G$  določi, ali ima trikotnik. Katero podatkovno strukturo za grafe boš uporabil?

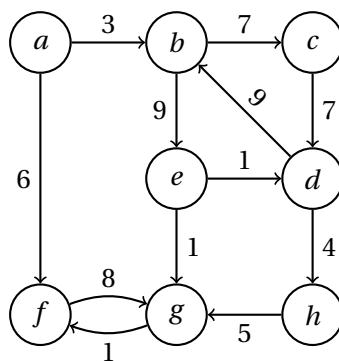
### Naloga 5.5.

Vir: Vaje OR 4.5.2016

Dan je digraf  $D = (V, E)$ . Pravimo, da je vozlišče  $v \in V$  *zvezda* digrafa  $D$ , če ima izhodno povezavo do vseh ostalih vozlišč in v digrafu  $D$  ni drugih povezav. Napiši algoritem, ki poišče zvezdo danega digrafa, če ta obstaja.



Slika 13: Graf za nalogi 5.6 in 5.12.



Slika 14: Graf za nalogo 5.8.

**Naloga 5.6.**

Vir: Vaje OR 30.11.2016

Na grafu s slike 13 izvedi iskanje v širino. V primerih, ko imaš več enakovrednih izbir, upoštevaj abecedni vrstni red. Za vsako povezavo določi, ali se nahaja v drevesu iskanja v širino.

**Naloga 5.7.**

Vir: Vaje OR 4.5.2016

Zapiši algoritem, ki za vhodni graf  $G$  določi njegov premer.

**Naloga 5.8.**

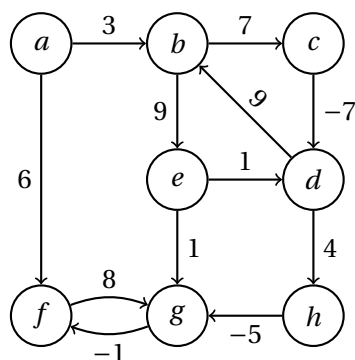
Vir: Vaje OR 7.12.2016

S pomočjo Dijkstrovega algoritma določi razdalje od vozlišča  $a$  do ostalih vozlišč v grafu s slike 14.

**Naloga 5.9.**

Vir: Vaje OR 4.5.2016

Naj bo  $G = (V, E)$  graf, za katerega so dolžine povezav določene s funkcijo  $\ell : E \rightarrow \mathbb{R}$  (tj., dolžine so lahko tudi negativne). Definirajmo še funkcijo  $\ell' : E \rightarrow \mathbb{R}$  tako, da velja  $\ell'(e) = \ell(e) - \min \{ \ell(f) \mid f \in E \}$  (dolžine, določene z  $\ell'$ , so torej nenegativne). Dokaži ali ovrzi: drevo najkrajših poti, ki ga Dijkstrov algoritem ustvari ob vходу  $(G, \ell')$ , je tudi drevo najkrajših poti za graf  $G$  z dolžinami povezav, določenimi z  $\ell$ .



Slika 15: Graf za nalogi 5.13 in 5.25.

**Naloga 5.10.**

Vir: [2, Exercise 4.13]

Denimo, da imamo neusmerjen graf  $G = (V, E)$ , katerega vozlišča predstavljajo mesta, povezave pa predstavljajo ceste, ki jih povezujejo. Za vsako povezavo  $e \in E$  poznamo njeno dolžino  $\ell_e$  (v kilometrih).

Priti želimo iz mesta  $s$  v mesto  $t$ . V vsakem mestu je bencinska črpalka, ob cestah pa teh ni. Žal imamo na voljo samo star avto, ki lahko s polnim rezervoarjem prepelje le  $L$  kilometrov.

- Zapiši algoritem, ki v linearnem času poišče pot, ki jo lahko prevozimo z našim avtom, oziroma ugotovi, da ta ne obstaja.
- Izkaže se, da z našim avtom te poti ne moremo prevoziti, zato se odločimo za nakup novega. Zapiši algoritem, ki v času  $O(m \log n)$  določi najmanjše število prevoženih kilometrov, ki naj jih avto zmore z enim polnjenjem, da bo pot od  $s$  do  $t$  mogoča.

**Naloga 5.11.**

Vir: Vaje OR 7.12.2016

Zasnui različico Dijkstrovega algoritma za iskanje najkrajše poti med vozliščema  $s$  in  $t$  v grafu  $G$ , ki iskanje hkrati začne v vozliščih  $s$  in  $t$ . Kdaj naj se iskanje konča in kako naj se poišče rešitev?

**Naloga 5.12.**

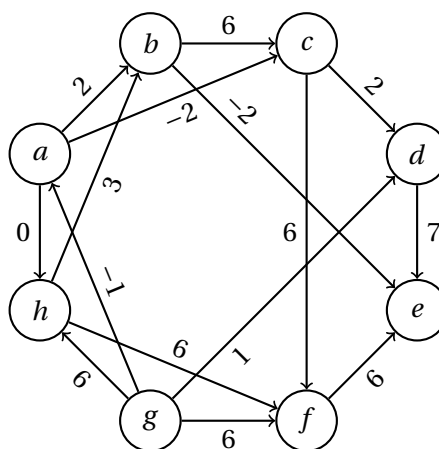
Vir: [2, Exercise 3.1]

Na grafu s slike 13 izvedi iskanje v globino. V primerih, ko imaš več enakovrednih izbir, upoštevaj abecedni vrstni red. Za vsako povezavo določi, ali se nahaja v drevesu iskanja v globino.

**Naloga 5.13.**

Vir: Vaje OR 21.5.2018

S pomočjo Bellman-Fordovega algoritma določi razdalje od vozlišča  $a$  do ostalih vozlišč v grafu s slike 15.



Slika 16: Graf za nalogo 5.14.

**Naloga 5.14.**

Vir: Vaje OR 7.12.2016

Dan je usmerjen acikličen graf s slike 16.

- (a) Poišči topološko ureditev vozlišč zgornjega grafa.
- (b) Poišči najkrajšo pot od vozlišča  $g$  do vozlišča  $e$ .
- (c) Poišči najdaljšo pot od vozlišča  $g$  do vozlišča  $e$ .

**Naloga 5.15.**

Vir: Kolokvij OR 9.5.2013

Oviratlon je tekalna preizkušnja na 8 do 10 kilometrov dolgi poti z različnimi ovirami. Zanima nas, na koliko različnih načinov lahko pridemo od štarta do cilja. Dan je utežen usmerjen acikličen graf  $G$  ter vozlišči  $s$  in  $t$ , ki predstavljata štart oziroma cilj. Uteži na povezavah nam predstavljajo, na koliko načinov jih lahko prečkamo.

- (a) Zapiši algoritem, ki reši dani problem. Kakšna je njegova časovna zahtevnost?
- (b) Reši nalogo za graf s slike 17.

**Naloga 5.16.**

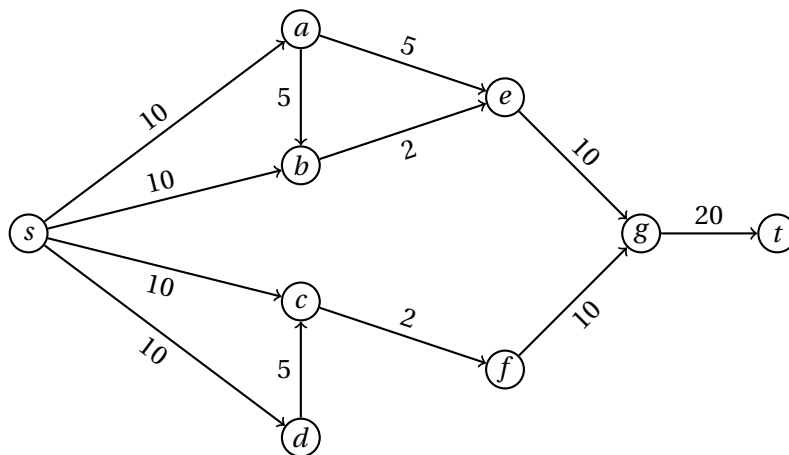
Vir: Vaje OR 14.12.2016

Zapiši celoštevilski linearni program, ki določi topološko urejanje grafa.

**Naloga 5.17.**

Vir: Vaje OR 14.12.2016

Zapiši algoritem, ki ugotovi, ali ima graf več kot eno topološko ureditev.



Slika 17: Graf za nalogo 5.15.

**Naloga 5.18.**

Vir: Izpit OR 15.12.2016

Lovec na zaklade se z bogatim ulovom vrača iz Kalifornije nazaj domov v Chicago, pri čemer mora seveda prečkati Divji zahod. Potoval bo s kočijo, pri čemer bo vsak dan potoval med dvema mestoma in nato prespal. Zaradi varnosti se bo držal samo državnih cest, ki so varne. Toda mesta, kjer bo prespal, niso povsem varna. Za vsako mesto pozna verjetnosti, da ga tam ne bodo oropali (te so med seboj neodvisne). Tako bi želel načrtovati najvarnejšo pot domov – torej pot z največjo verjetnostjo, da ga pri nobenem postanku ne bodo oropali.

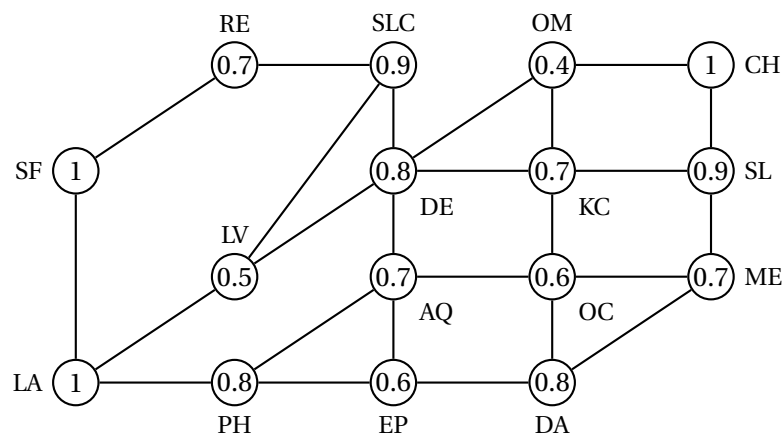
- Mesta in ceste med njimi lahko predstavimo z vozlišči in povezavami v ne-usmerjenem grafu  $G$ , verjetnosti pa kot teže vozlišč. Opiši, kako lahko za dani graf  $G$  z uteženimi vozlišči učinkovito poiščemo ustrezno pot med danima vozliščema  $s$  in  $t$  z uporabo variante Dijkstrovega algoritma, ter utemelji njegovo ustreznost. Lahko predpostaviš, da sta teži začetnega in končnega vozlišča enaki 1.
- Reši problem za graf s slike 18, pri čemer naj se pot začne v LA in konča v CH. Zadostovalo bo, če verjetnosti računaš na 3 decimalke natančno.

**Naloga 5.19.**

Vir: Izpit OR 15.3.2017

Dan je usmerjen graf  $G = (V, E)$  s pozitivnimi dolžinami povezav. Dano imamo še vozlišče  $s \in V$  ter vrednost  $\delta_v$  za vsako vozlišče  $v \in V$ . Natančno opiši algoritem (z besedami ali psevdokodo), ki v času  $O(m)$  (kjer je  $m = |V| + |E|$ ) preveri, ali velja  $\delta_v = d_G(s, v)$  za vse  $v \in V$  – t.j., v linearnem času preveri, ali so vrednosti  $\delta_v$  enake razdalji med vozliščema  $s$  in  $v$  v grafu  $G$ .

Lahko predpostaviš, da so vsa vozlišča grafa  $G$  dosegljiva iz vozlišča  $s$ . Utemelji pravilnost meje za časovno zahtevnost algoritma.



Slika 18: Graf za nalogo 5.18.

**Naloga 5.20.**

Vir: Izpit OR 10.7.2017

Z električnim vozilom se odpravljamo na počitnice. Vozilo moramo vsako noč napolniti, zato smo si pripravili seznam krajev in cestnih povezav med njimi, ki jih lahko prevozimo v enem dnevu. Poiskati želimo pot od začetne točke do destinacije, ki bo imela čim manjše število postankov (tj., bo trajala čim manj dni).

- Predstavi problem v jeziku grafov in predlagaj algoritem za njegovo reševanje.
- Na koncu počitnic razmišljamo o poti nazaj. Spet bi radi naredili čim manj postankov, a se pri tem ne želimo ustaviti v nobenem kraju, kjer smo se ustavili na poti naprej. Dopolni zgornji algoritem, da bo našel še ustrezno pot nazaj.
- S pomočjo zgornjih algoritmov poišči najkrajšo pot od LJ do AM in najkrajšo pot nazaj v grafu s slike 19, ki ne gre čez kraje iz prejšnje poti.

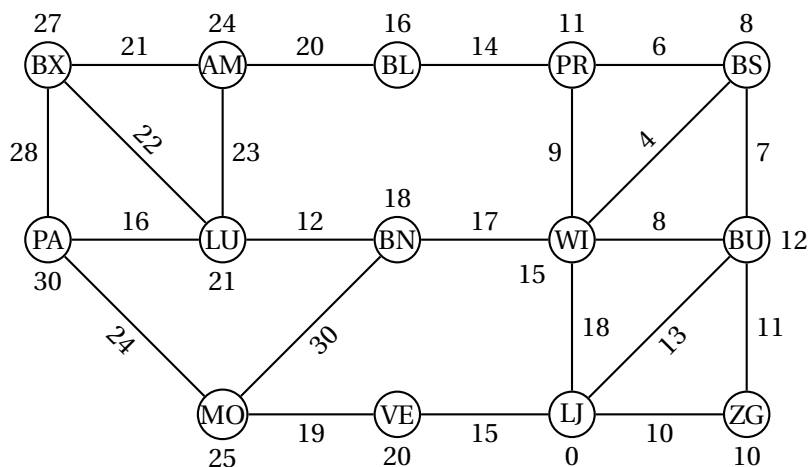
**Naloga 5.21.**

Vir: Kolokvij OR 11.6.2018

Dan je povezan neusmerjen enostaven graf  $G = (V, E)$  (tj., brez zank in večkratnih povezav). *Prerezno vozlišče* v grafu  $G$  je tako vozlišče  $u \in V$ , da graf  $G - u$  (tj., graf  $G$  brez vozlišča  $u$  in povezav s krajiščem  $u$ ) ni več povezan. Poiskati želimo seznam prereznih vozlišč grafa  $G$ .

Pri iskanju si bomo pomagali s preiskovanjem v globino. Ob prvem obisku vozlišča  $u$  s predhodnikom  $v$  se tako pokliče funkcija  $\text{PREVISIT}(u, v)$ , ob njegovem zadnjem obisku pa funkcija  $\text{POSTVISIT}(u, v)$ . Če je  $u$  koren preiskovalnega drevesa, potem ima  $v$  vrednost  $\text{NULL}$ . Predpostavi, da imaš v obeh funkcijah dostop do seznama *izhod*, kamor bo treba dodati najdena presečna vozlišča. Prav tako imata lahko obe funkciji dostop do drugih pomožnih spremenljivk.





Naj bo  $\ell_u$  globina vozlišča  $u$  v drevesu iskanja v globino (tj., razdalja od korena do  $u$  v drevesu iskanja v globino). Za vsako vozlišče  $u$  definiramo vrednost  $p_u$  kot najmanjšo globino vozlišč, ki so v grafu  $G$  sosedna (ali enaka) vozlišču  $u$  ali njegovim potomcem v drevesu iskanja v globino.

- (a) Za graf na sliki 20 nariši drevo iskanja v globino (v njem označi tudi povratne povezave, npr. s črtkano črto) in določi njegova prerezna vozlišča. Upoštevaj abecedni vrstni red obiskovanja vozlišč. Za vsako vozlišče  $u$  določi še vrednosti  $\ell_u$  in  $p_u$ .

**Namig:** vrednosti  $p_u$  najprej določi za vozlišča z večjo globino.

- (b) Napiši rekurzivno formulo za vrednost  $p_y$ .

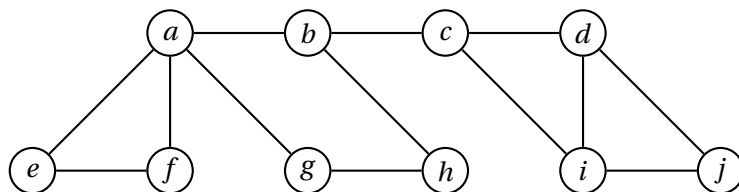
**Namig:** loči med sosedi  $v$  vozlišča  $u$  v grafu  $G$  (pišeš lahko  $u \sim v$ ) in njegovimi neposrednimi nasledniki  $w$  v preiskovalnem drevesu ( $u \rightarrow w$ ).

- (c) Natančno opiši funkcijo  $\text{PREVISIT}(u, v)$  (z besedami ali psevdokodo), ki poskrbi za izračun vrednosti  $\ell_u$ .

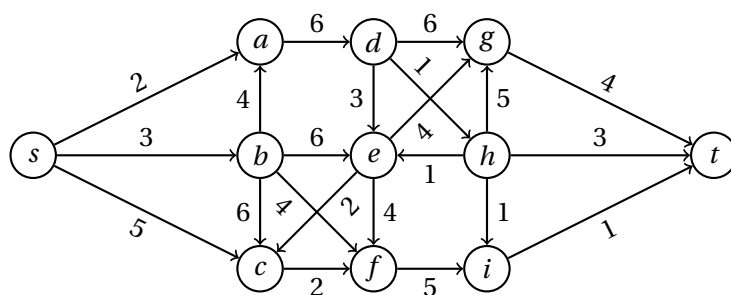
- (d) Natančno opiši funkcijo  $\text{POSTVISIT}(u, v)$  (z besedami ali psevdokodo), ki naj za vozlišče  $u$  izračuna vrednost  $p_u$  in ugotovi, ali je  $u$  prerezno vozlišče, in ga v tem primeru doda v *izhod*. Predpostavi, da imaš globine vozlišč že poračunane.

**Namig:** obravnavaj dve možnosti – ko je  $u$  koren drevesa, in ko  $u$  ni koren drevesa. Kako v vsakem od teh primerov ugotoviš, ali je  $u$  prerezno vozlišče?

- (e) Oceni časovno zahtevnost celotnega algoritma.



Slika 20: Graf za nalogo 5.21.



Slika 21: Graf za nalogo 5.22.

**Naloga 5.22.**

Vir: Izpit OR 5.7.2018

Dan je utežen usmerjen acikličen graf s slike 21.

- Poišči topološko ureditev grafa s slike 21.
- Poišči najcenejšo pot od vozlišča  $s$  do vozlišča  $t$  v grafu s slike 21.
- Naj bo  $G = (V, E)$  usmerjen acikličen graf z nenegativno uteženimi povezavami ter  $s, t \in V$  njegovi vozlišči. Algoritem  $\mathcal{A}$  se po grafu  $G$  sprehaja po naslednjem pravilu: začne v vozlišču  $s$ , v vsakem koraku pa se iz vozlišča  $u$  premakne v njegovega izhodnega sosedo  $v$  z verjetnostjo

$$p_{uv} = \frac{\ell_{uv}}{\sum_{u \rightarrow w} \ell_{uw}},$$

kjer je  $\ell_{uv}$  teža povezave od  $u$  do  $v$ . Algoritem  $\mathcal{A}$  se ustavi, ko doseže vozlišče  $t$ .

Natančno opiši (z besedami ali psevdokodo), kako bi v času  $O(m)$  (kjer je  $m = |V| + |E|$ ) za vsako vozlišče  $u \in V$  določil verjetnost  $q_u$ , da algoritem  $\mathcal{A}$  obišče vozlišče  $u$ . Verjetnosti za graf s slike 21 ni potrebno računati.

**Naloga 5.23.**

Vir: Izpit OR 28.8.2018

Odpravljamo se na pot, ki bo trajala več dni. Pripravili smo si seznam krajev in povezav med njimi, ki jih lahko prevozimo v enem dnevu. Za vsako povezavo poznamo stroške prevoza, prav tako pa za vsak kraj poznamo še stroške nočitev. Poiskati želimo čim cenejšo pot od začetne točke do destinacije (tj., skupna cena prevozov in nočitev naj bo čim manjša).

- (a) Predstavi problem v jeziku grafov in predlagaj čim bolj učinkovit algoritem za njegovo reševanje.
- (b) S pomočjo zgornjega algoritma poišči najcenejšo pot od LJ do BX v grafu s slike 19. Na povezavah so napisani stroški prevozov med krajema (veljajo za obe smeri), pri vozliščih pa stroški prenočitve v kraju.

**Naloga 5.24.**

Vir: Kolokvij OR 19.11.2009

Naj bo  $G$  graf z uteženimi povezavami, kjer so uteži nenegativne. Dokaži ali ovrzi naslednje trditve:

- (a) Če v grafu obstaja enolična najkrajša povezava, potem je ta povezava vsebovana v vsakem drevesu najkrajših poti.
- (b) Če v grafu obstaja enolična najkrajša povezava, potem je ta vključena v vsako minimalno vpeto drevo.
- (c) Naj bo  $e$  povezava v grafu  $G$ , ki je izmed povezav na nekem ciklu  $C$  najdaljša. Če iz  $G$  odstranimo povezavo  $e$ , dobimo graf  $G'$ . Pokaži, da je poljubno minimalno vpeto drevo v  $G'$  tudi minimalno vpeto drevo v  $G$ .

**Naloga 5.25.**

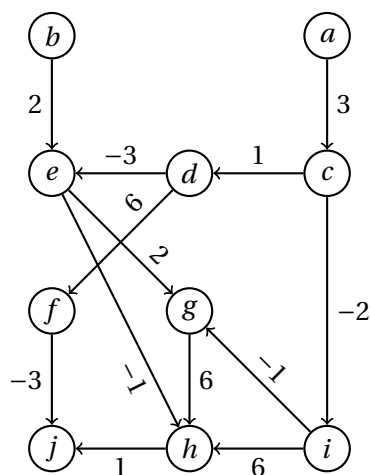
Vir: Vaje OR 17.5.2021

S pomočjo Floyd-Warshallovega algoritma poišči najkrajše poti med vsemi pari vozlišč v grafu s slike 15.

**Naloga 5.26.**

Vir: Kolokvij OR 19.11.2009

Zaradi naravnih nesreč so reševalci prejeli klice za  $n$  ponesrečenih oseb iz različnih lokacij, ki jih je potrebno prepeljati v  $k$  bolnišnic po danem cestnem omrežju. Ponesrečence lahko prepeljemo do bolnišnic, ki so oddaljene za največ pol ure urgentne vožnje (za dane ponesrečence imamo torej v splošnem več izbir za bolnišnice). Da pa bolnišnic ne bi preobremenili, zahtevamo, da v vsako bolnišnico pride največ  $\lceil \frac{n}{k} \rceil$  ponesrečencev. Zanima nas, ali je to izvedljivo. Modeliraj problem in ga prevedi na kakšen znan problem.



Slika 22: Graf za nalogo 5.27.

**Naloga 5.27.**

Vir: Kolokvij OR 19.11.2009

Janez je prekaljen poslovnež, ki hoče vedno zaslužiti kar največ. A Janez lahko naenkrat prevzame le en posel. Če se en posel začne med izvajanjem nekega drugega posla, novega posla ne more prevzeti. Poslovne priložnosti so predstavljene z grafom. Vozlišča predstavljajo stanja, v katerih Janez izbira med posli, izhodne povezave iz vsakega vozlišča pa predstavljajo posle, ki se jih lahko loti. Cene povezav predstavljajo dobiček pri poslu oziroma izgubo, če je cena negativna (včasih je potrebno sprejeti tudi kak posel, ki nosi izgubo, da se lahko prebijemo do dobičkonosnega posla...). V vsakem stanju lahko Janez izbere katerega koli izmed poslov, ki pripadajo izhodnim povezavam. Tekom svoje poslovne kariere se lahko Janez v določenem stanju znajde tudi večkrat. Janez se trenutno nahaja v izbranem vozlišču grafa.

- (a) Za poljuben graf poslovnih priložnosti ugotovi, kakšen problem moraš na njem rešiti, da boš lahko pravilno svetoval Janezu, da bo uresničil svoje poslovne ambicije. Predlagaj algoritem, s katerim rešiš problem, in oceni časovno zahtevnost predlagane rešitve. Janeza med drugim še posebej zanima, ali mu graf poslovnih priložnosti zagotavlja stalno pridobivanje poslov in dobičkonosno poslovanje za celotno kariero.
- (b) Za graf poslovnih priložnosti s slike 22 izberi ustrezen algoritem in ga izvedi ter izračunaj, koliko največ lahko zasluži Janez, če "vstopi v igro" v stanju *a* ali *b*. Upoštevaj lastnosti spodnjega grafa in navedi, kateri algoritem je to. Ali graf predstavlja poslovne priložnosti, ki bodo Janezu zagotovile trajno dobičkonosno poslovanje?

**Naloga 5.28.**

Vir: Izpit OR 5.2.2010

Neusmerjeno omrežje  $G$  s pozitivnimi razdaljami na povezavah razdelimo na dve disjunktne množici vozlišč  $A$  in  $B$ , ki pokrivata vsa vozlišča. Naj bo  $e$  najkrajša povezava z enim krajiščem v  $A$  in drugim v  $B$ . Pokaži ali ovrzi:

- (a) Vsaka najkrajša pot med kakim vozliščem iz  $A$  in kakim vozliščem iz  $B$  vsebuje povezavo  $e$ .
- (b) Obstaja najkrajša pot med nekima vozliščema v grafu  $G$ , ki vsebuje povezavo  $e$ .
- (c) Za vsak par vozlišč  $a \in A$  in  $b \in B$  obstaja najkrajša pot, ki vsebuje povezavo  $e$ .

**Naloga 5.29.**

Vir: Izpit OR 28.6.2010

Na univerzi na grškem otoku Lenoritas so zaradi gospodarske krize uvedli varčevalne ukrepe, ki vključujejo tudi nov režim plačevanja za izpite, ki jih pazijo asistenti. Asistentom sta ostali le dve boniteti: nadomestilo za izvedbo strnjene zaporedja izpitov ter nadomestilo za vsako "luknjo", daljšo od pol ure, med strnjenimi zaporedji izpitov. Ostale bonitete, kot so trinajsta in štirinajsta plača ter dodatek za točnost na delovnem mestu, je univerza pred kratkim ukinila. Tako se univerzi splača, da uporabi čim manj asistentov, ki pazijo strnjena zaporedja izpitov brez "lukenj". Za vsak dan so termini izpitov podani v naprej. Podatki za ponedeljek so (7, 8), (8, 9), (9, 10), (8, 12), (10, 13), (12, 15), (13, 15), (9, 12), (7, 10).

Ne glede na ekonomičnost se je dekan odločil, da bo v ponedeljek vodji sindikata asistentov dr. Dusanikusu Semolitakisu zagodel in mu dal kar najdaljše možno strnjeno zaporedje izpitov. Izberi ustrezen algoritem za izvedbo te naloge s kar se da najboljšo časovno zahtevnostjo (in jo tudi navedi), ter s pomočjo le-tega reši dekanov "problem" za ponedeljek.

*Nadaljevanje sledi v nalogi 8.3.*

**Naloga 5.30.**

Vir: Izpit OR 15.9.2010

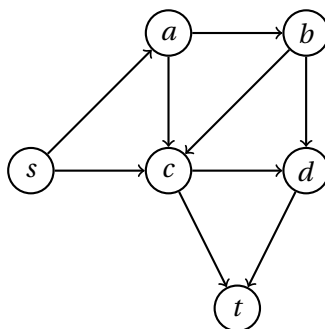
Z usmerjenim grafom je podano omrežje enosmernih prehodov med letališkimi terminali (glej npr. sliko 41). Načrtovalci niso čisto gotovi, ali je možno prehajati med vsemi terminali. Predlagaj algoritem, ki bi ugotovil, ali je vedno možen prehod med poljubnima izbranimi terminaloma. Kakšna je časovna zahtevnost predlaganega algoritma?

*Nadaljevanje sledi v nalogi 8.5.*

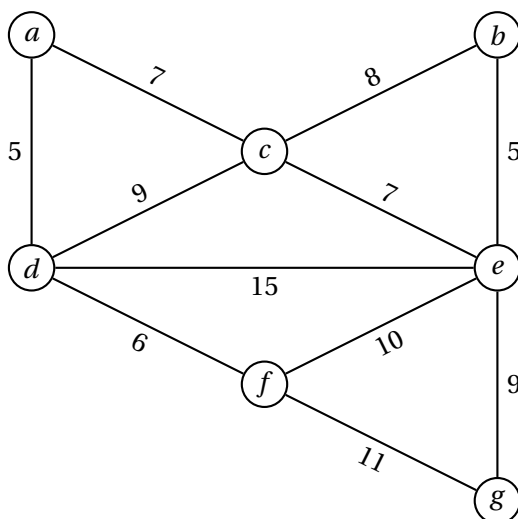
**Naloga 5.31.**

Vir: Kolokvij OR 25.11.2010

Napiši psevdokodo algoritma, ki v acikličnem usmerjenem grafu prešteje število poti od danega vozlišča  $s$  do vseh vozlišč grafa. Algoritem nato uporabi na grafu s slike 23.



Slika 23: Graf za nalogo 5.31.



Slika 24: Graf za nalogo 5.32.

**Naloga 5.32.**

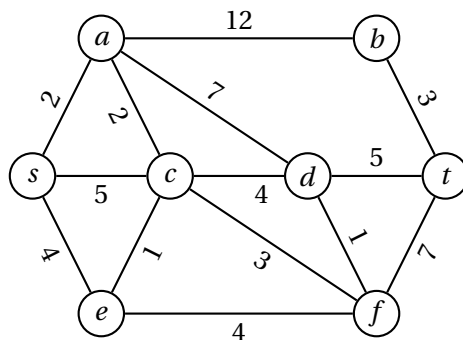
Vir: Kolokvij OR 25.11.2010

Otoke z grafa na sliki 24 želimo povezati z mostovi, tako da bo skupna cena gradnje čim manjša. Cena gradnje mostu je milijon evrov na kilometer (razdalje v kilometrih so označene na sliki; na povezavah, ki niso narisane, zaradi tehničnih preprek ne moremo zgraditi mostov). Med katerimi otoki naj zgradimo mostove? Pri iskanju odgovora uporabi primeren algoritem in zapiši vse vmesne rezultate. Kakšen je odgovor, če je za vsak zgrajeni most potrebno dobiti potrdilo inšpektorjev, ti pa za vsak pregled zaračunajo milijon evrov?

**Naloga 5.33.**

Vir: Izpit OR 28.6.2011

Radi bi zgradili cesto med točkama  $s$  in  $t$ . Stroški vsakega možnega cestnega odseka so prikazani na grafu na sliki 25. Kje naj poteka cesta, da bodo stroški gradnje najmanjši možni?



Slika 25: Graf za nalogo 5.33.

### Naloga 5.34.

Vir: Kolokvij OR 5.4.2012

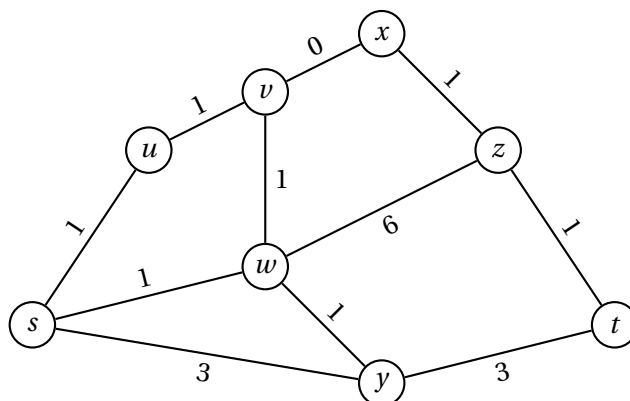
Na kongresu se je zbralo  $n$  delegatov. Problem je v tem, da ne govorijo vsi skupnega jezika. Za vsakega delegata poznamo seznam jezikov, ki jih govori, skupaj s stopnjo znanja za vsak posamični jezik. Stopnja znanja je število med 1 in 100. Stopnja 1 pomeni, da oseba jezik popolnoma obvlada, stopnja 100 pa pomeni, da pozna zgolj nekaj osnovnih fraz. Recimo, da bi oseba  $A$  rada osebi  $B$  posredovala neko sporočilo. Če znata skupni jezik, se lahko pogovorita neposredno. Lahko pa oseba  $A$  pošlje sporočilo preko enega ali več posrednikov.

(a) Radi bi, da oseba  $B$  prejme sporočilo v najkrajšem možnem času (pri čemer lahko sporočilo potuje preko enega ali več posrednikov). Če osebi  $X$  in  $Y$  govorita skupen jezik ter sta stopnji obvladovanja tega jezika  $s_X$  za osebo  $X$  in  $s_Y$  za osebo  $Y$ , prenos sporočila traja  $\max\{s_X, s_Y\}$  časovnih enot. (Če dve osebi ne obvladata dovolj dobro skupnega jezika, si lahko pomagata z opisovanjem pojmov, mahanjem rok, risanjem ipd.) Formuliraj zgornjo nalogo kot problem iskanja najkrajše poti v ustreznem grafu.

(b) Dan je naslednji seznam delegatov:

Delegat	Jeziki
Frank	(angleščina, 5), (španščina, 10), (ruščina, 80)
Ivan	(ruščina, 5), (španščina, 20), (angleščina, 95)
Paul-Henri	(francoščina, 5), (nemščina, 85), (angleščina, 95)
Brigitte	(nizozemščina, 10), (nemščina, 15)
Andrej	(slovenščina, 5), (nemščina, 10), (latinščina, 90)
Wolfgang	(nemščina, 5), (angleščina, 90)
Jafar	(arabščina, 5), (ruščina, 10), (nizozemščina, 30), (francoščina, 80)

Kako lahko Frank najhitreje posreduje informacijo Wolfgangu?



Slika 26: Graf za nalogo 5.35.

**Naloga 5.35.**

Vir: Izpit OR 9.7.2012

Čmrlja Gaber in Bor slovita kot najhitrejša letalca med čmrlji. Nedavno je Gaber izzval Bora na dirko čez travnik, kjer pot ni vnaprej določena. Bor je izziv sprejel in dogovorila sta se za pravili:

- določila sta točki  $s$  in  $t$ : začneta v  $s$ , in kdor prvi pride v  $t$ , zmaga;
- določila sta tudi graf poti  $G$ , po katerih lahko letita. Povezave v tem grafu sta določila tako, da je čas letenja (če ni rožic na povezavi) od enega do drugega krajišča za vse povezave enak (čmrlja sta enako hitra). Vozlišča v tem grafu pa sta določila tako, da na njih ne raste nobena rožica.

Znano je, da če katerikoli čmrlj prileti do kake rožice, se na njej zadrži enako dolgo, kot potrebuje za prelet ene povezave brez rožic. Rožice izven poti bosta čmrlja z lahkoto spregledala, saj gre za prestižno tekmo.

Tvoja naloga je, da Gabru pomagaš do zmage. Če torej poznaš graf  $G$ , vozlišči  $s$  in  $t$  ter število rožic na vsaki povezavi, katero pot naj izbere?

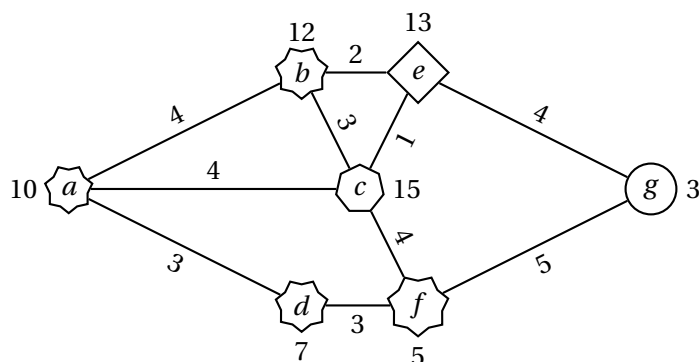
- Prevedi zgornji problem na kak znan problem in predlagaj znani algoritem, s katerim ga lahko rešiš.
- Reši problem za graf s slike 26, kjer oznake na povezavah povedo, koliko rožic se nahaja vzdolž povezave.

**Naloga 5.36.**

Vir: Izpit OR 4.9.2012

Žabec Rok in žabica Neli živita v mlaki, na kateri plavajo lokvanjevi listi. Na enem listu sedi Rok, na drugi strani mlake pa je list, na katerem počiva Neli. Rok bi rad po listih priskakljal k Neli. Z enim skokom lahko premosti razdaljo največ  $d$ . Čisto vseeno mu je, koliko skokov naredi in koliko je skupna preskakana razdalja. Ali jo lahko obišče? Lokvanjevi listi so podani kot seznam koordinat v  $\mathbb{R}^2$ . Listi so majhni v primerjavi z razdaljami med njimi, zato jih lahko obravnavas kot točke.





Slika 27: Graf za nalogo 5.37.

- Formuliraj zgornji problem kot problem na grafih in predlagaj algoritem, s katerim ga lahko rešiš.
- Reši problem za  $d = 3$  in lokvanje na koordinatah

$(0, 0), (2, 1), (4, 1), (2, 4), (7, 4), (1, 6), (5, 6), (3, 8), (8, 8),$

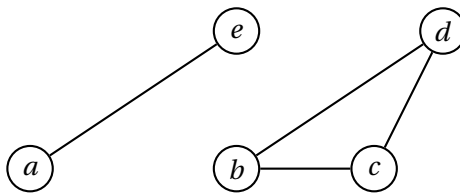
pri čemer Rok sedi na lokvanju  $(0, 0)$ , Neli pa na  $(8, 8)$ .

### Naloga 5.37.

Vir: Izpit OR 14.6.2013

Družina škrata Bolfenka živi v rovih nekje na Pohorju. Odločili so se, da v sobane napeljejo optični internet. Ker je družina velika, potrebujejo algoritem, s katerim bi določili, po katerih rovih naj poteka optični kabel in kje naj se optični kabel priključi na zunanje ("človeško") optično omrežje. Dan je povezan graf  $G$  s pozitivnimi utežmi na povezavah. Vozlišča grafa so sobane, kjer želimo imeti dostop do optičnega interneta, povezave nam povedo, kje lahko potegnemo kabel, uteži na povezavah pa nam povedo ceno polaganja kabla (v guldih). Za vsako sobano je znano, kolikšna je cena priklopa na zunanje optično omrežje iz te sobane. Ker se škratje skrivajo, bodo naredili priklop na zunanji internet iz natanko ene sobane. Pomagaj škratom najti rove, po katerih naj napeljejo optični kabel, in sobano, iz katere naj se priključijo na zunanji internet, da bo vsaka sobana imela dostop do optičnega interneta in bo cena čim manjša!

- Opiši algoritem, ki učinkovito reši zgoraj opisani problem.
- Predlagani algoritem izvedi na grafu s slike 27. Številke pri vsakem vozlišču pomenijo ceno priklopa na zunanje omrežje iz sobane.



Slika 28: Graf za nalogo 5.38.

**Naloga 5.38.**

Vir: Izpit OR 24.6.2013

Profesionalni programer je vladi Republike Slovenije prodal algoritem PoG. A na pristojnem ministrstvu so pozabili, kdo je ta algoritem naročil, in zato ne vedo, kaj naj bi algoritem vračal. V prospektu so zasledili naslednjo psevdokodo algoritma:

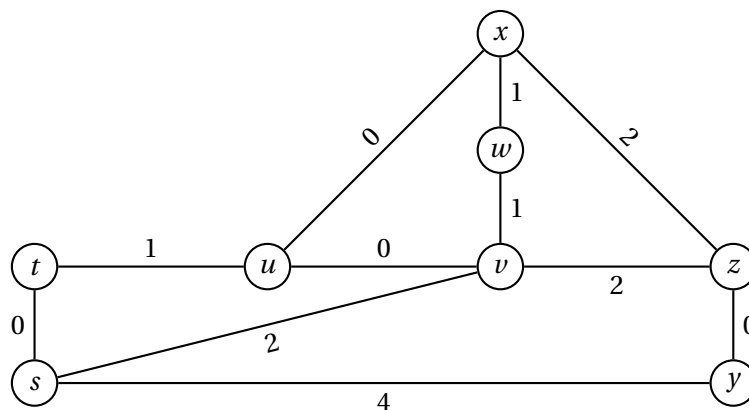
**Vhod:** neusmerjen graf  $G = (V, E)$

```

for  $v \in V$  do
    obarvaj  $v$  sivo
end for
 $Q \leftarrow$  nova vrsta
 $s \leftarrow 0$ 
for  $v \in V$  do
    if  $v$  je siv then
         $s \leftarrow s + 1$ 
         $Q.append(v)$ 
        while  $\neg Q.isEmpty()$  do
             $w \leftarrow Q.pop()$ 
            for  $u \in \text{ADJ}(G, w)$  do
                if  $u$  je siv then
                     $Q.append(u)$ 
                    obarvaj  $u$  rumeno
                end if
            end for
        end while
    end if
end for
return  $s$ 

```

- Algoritem izvedi na grafu s slike 28, tj., zabeleži vsako spremembo barve nekega vozlišča. Koliko je takrat vrednost števca?
- Kaj algoritem vrača? Odgovor utemelji.
- Kakšna je časovna zahtevnost algoritma? Odgovor utemelji.



Slika 29: Graf za nalogo 5.39.

**Naloga 5.39.**

Vir: Izpit OR 24.6.2013

Računalniško omrežje sestavlja  $n$  računalnikov. Predstavljeno je z neusmerjenim povezanim grafom  $G$ , ki ima na povezavah nenegativne uteži. Utež na povezavi  $uv$  nam pove, koliko časa (v milisekundah) potrebuje informacija, da pride neposredno od računalnika  $u$  do računalnika  $v$ . Vsak računalnik potrebuje natanko 1 ms, da informacijo pošlje naprej. Naj bo  $T_{uv}$  najkrajši čas, v katerem lahko računalnik  $v$  dobi informacijo od računalnika  $u$ . Ta čas vsebuje 1 ms, ki jo računalnik  $u$  porabi za pošiljanje informacije, in ne vsebuje 1 ms za računalnik  $v$ , saj slednjemu informacije ni več potrebno poslati naprej. Tudi naš računalnik je priklopljen na to omrežje – recimo, da je predstavljen z vozliščem  $s$ . Zanima nas, kateri računalnik od nas najkasneje dobi informacije, tj., pri katerem  $v$  je dosežena maksimalna vrednost  $T_{sv}$ , in kolikšen je ta maksimum.

- Opiši algoritem časovne zahtevnosti največ  $O(n^2)$ , ki reši dani problem. Na vhod naj sprejme graf  $G$  in vozlišče  $s$ .
- Reši nalogo na grafu s slike 29 tako, da bo postopek jasen in pravilen.

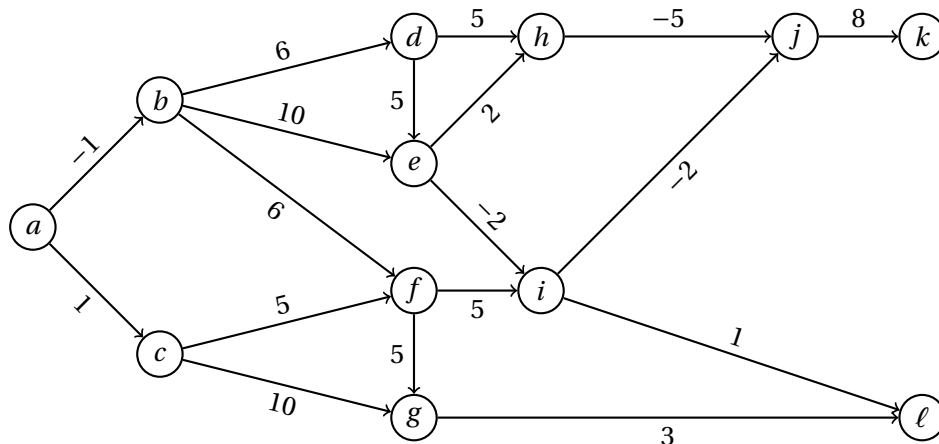
**Naloga 5.40.**

Vir: Izpit OR 24.6.2013

Potujoči trgovec Marco načrtuje pot, na kateri bo imel kar največji dobiček.

Dan je usmerjen aciklični graf  $G = (V, E)$  z utežmi na povezavah. Uteži predstavljajo dobiček na povezavi (oz. izgubo, če so negativne). Marco bo začel svojo pot v enem izmed vozlišč grafa in se po usmerjenih povezavah odpravil do končnega vozlišča (začetek in konec nista znana, moramo ju še določiti). Pomagaj Marcu najti najboljšo pot (vključno z začetnim in končnim vozliščem)! Kakšen dobiček si lahko obeta?

- Opiši algoritem, ki reši dani problem, ter obravnavaj njegovo časovno zahtevnost.



Slika 30: Graf za nalogo 5.40.

**Namig:** uporabi dinamično programiranje, da bo tvoj algoritem tekel v času  $O(|E| + |V|)$ .

- (b) Reši nalogo za graf s slike 30 (razloži, kako si prišel do rezultata, oz. jasno označi vmesne rezultate).

**Naloga 5.41.**

Vir: Izpit OR 5.6.2014

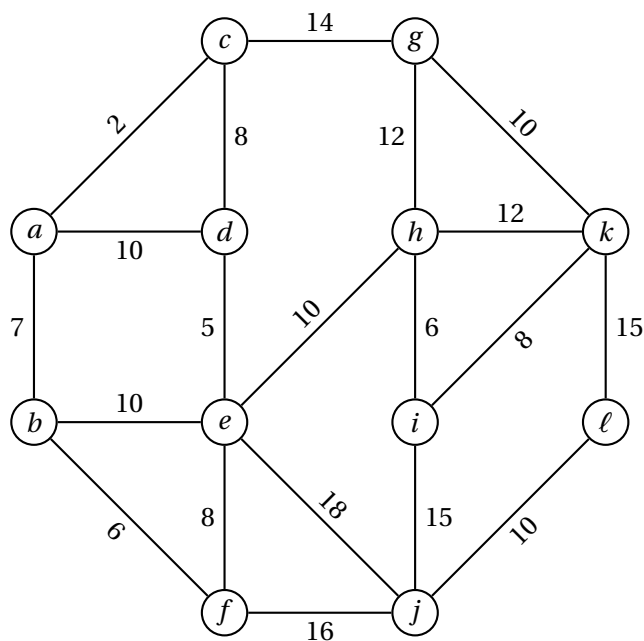
Dan je graf mest in cestnih povezav med njimi, prikazan na sliki 31. Predvidena je obnova nekaterih cest, pri čemer je strošek obnove sorazmeren z dolžino ceste.

- (a) Določi ceste, ki jih je potrebno obnoviti, da bo strošek obnove čim manjši, ter da bo med poljubnima dvema mestoma obstajala povezava preko samih obnovljenih cest.
- (b) Zaradi nižanja stroškov preuči še naslednjo možnost. Določi ceste, ki jih je potrebno obnoviti, da bo strošek obnove čim manjši, ter da bo med poljubnima dvema mestoma obstajala povezava preko samih obnovljenih cest in največ ene neobnovljene ceste.

**Naloga 5.42.**

Vir: Izpit OR 12.5.2016

Napiši algoritem, ki na vhod sprejme neusmerjen graf  $G = (V, E)$  in povezavo  $e \in E$  ter pove, ali obstaja cikel v  $G$ , ki vsebuje povezavo  $e$ .



Slika 31: Graf za nalogo 5.41.

**Naloga 5.43.**

Vir: Izpit OR 29.8.2017

Peter zaključuje študij na Fakulteti za alternativno znanost. Opravi je že vse obvezne predmete, za pristop k zaključnemu izpitu pa mora opraviti še nekaj izbirnih predmetov. To bi rad storil čim hitreje. V tabeli 2 so naštet izbirni predmeti skupaj s trajanjem (v tednih, od pristopa do uspešnega opravljanja) in predmeti, h katerim lahko pristopi po uspešnem opravljanju. K predmetom  $a$ ,  $c$  in  $f$  lahko pristopi že takoj, za pristop k ostalim predmetom pa zadostuje, če je opravljen eden od pogojev.

- Topološko uredi ustrezni graf in ga nariši.
- Katere predmete naj Peter opravi, da bo lahko čim prej pristopil k zaključnemu izpitu? Koliko časa bo za to potreboval? Natančno opiši postopek iskanja odgovora.

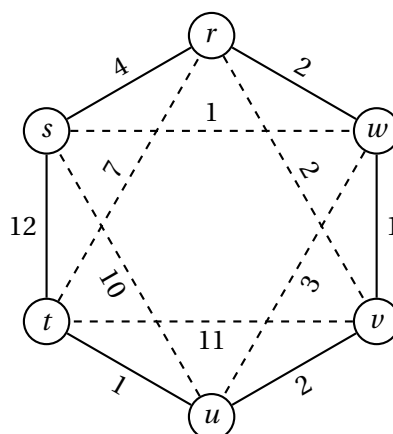
**Naloga 5.44.**

Vir: Kolokvij OR 3.6.2019

Dan je neusmerjen utežen graf  $G = (V, E)$  z nenegativnimi cenami povezav  $L_e$  ( $e \in E$ ). Naj bosta  $A$  in  $B$  disjunktni množici povezav, tako da velja  $E = A \cup B$ . Želimo najti najcenejšo *alternirajočo* pot med danima vozliščema  $s, t \in V$  – torej takšno, v kateri se povezave iz  $A$  in iz  $B$  izmenjujejo (ni pomembno, ali začnemo oziroma končamo s povezavo iz množice  $A$  ali  $B$ ). Posamezno vozlišče se lahko v alternirajoči poti pojavi tudi večkrat.

Oznaka	Predmet	Trajanje	Zadosten pogoj za
<i>a</i>	Alternativna zgodovina	5	<i>i, j</i>
<i>b</i>	Astrološki praktikum	7	<i>d, g</i>
<i>c</i>	Diskretna numerologija	9	<i>b</i>
<i>d</i>	Filozofija magije	4	<i>z</i>
<i>e</i>	Kvantno pravo	8	<i>b, h</i>
<i>f</i>	Postmoderna ekonomija	4	<i>e</i>
<i>g</i>	Telepatija in telekineza	5	<i>z</i>
<i>h</i>	Teorija antigravitacije	4	<i>g</i>
<i>i</i>	Teorije zarote	5	<i>h</i>
<i>j</i>	Ufologija II	10	<i>k</i>
<i>k</i>	Uvod v kriptozoologijo	6	<i>z</i>
<i>z</i>	Zaključni izpit	/	/

Tabela 2: Podatki za nalogo 5.43.



Slika 32: Graf za nalogo 5.44(b).

- (a) Predlagaj čim učinkovitejši algoritem za reševanje danega problema. Kakšna je njegova časovna zahtevnost?

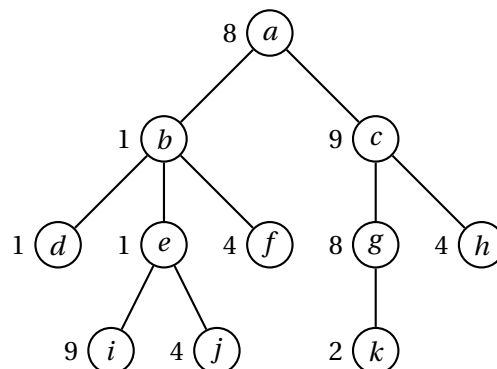
**Namig:** grafu  $G$  priredi usmerjen graf  $G'$ , v katerem bodo vse poti od  $s$  do  $t$  ustrezale alternirajočim potem v  $G$ . Po potrebi lahko vozlišča tudi podvojiš.

- (b) S svojim algoritmom poišči najcenejšo alternirajočo pot od  $s$  do  $t$  v grafu s slike 32. Povezave iz množice  $A$  so označene s polno, povezave iz množice  $B$  pa s črtkano črto. Iz rešitve naj bo jasno, kako poteka izvajanje algoritma.

#### Naloga 5.45.

Vir: Kolokvij OR 3.6.2019

Neodvisna množica vozlišč grafa  $G = (V, E)$  je taka množica  $S \subseteq V$ , da sta poljubni vozlišči iz množice  $S$  nesosedni v  $G$ , torej  $uv \notin E$  za vsaka  $u, v \in S$ .



Slika 33: Drevo za nalogo 5.45(e).

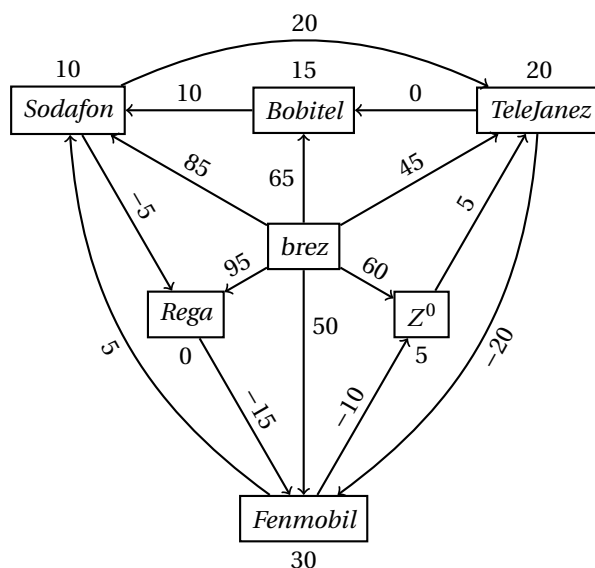
Dano je drevo  $T = (V, E)$  in uteži vozlišč  $c_v$  ( $v \in V$ ). V drevesu  $T$  želimo najti *najtežjo neodvisno množico* – torej tako množico vozlišč  $S \subseteq V$ , ki maksimizira vsoto njihovih uteži, torej vrednost  $\sum_{u \in S} c_u$ .

- Denimo, da je drevo  $T$  podano kot neusmerjen graf, predstavljen s seznamami sosedov. Razloži, kako lahko sestaviš slovar *pred*, ki za vsako vozlišče  $v \in V$  določa njegovega prednika, če za koren izbereš vozlišče  $r \in V$ . Koren  $r$  je lahko izbran poljubno, zanj pa velja  $\text{pred}[r] = \text{NULL}$ . Kako iz seznamov sosedov in slovarja *pred* ugotovimo, katera vozlišča so neposredni nasledniki danega vozlišča v drevesu?
- Napiši rekurzivne enačbe za reševanje problema najtežje neodvisne množice v drevesu  $T$ . Razloži, kaj predstavljajo spremenljivke, v kakšnem vrstnem redu jih računamo, ter kako dobimo optimalno rešitev. Predpostaviš lahko, da imaš poleg seznamov sosedov in uteži vozlišč drevesa  $T$  na voljo tudi slovar *pred* kot v točki (a).  
**Namig:** za vsako vozlišče uporabi dve spremenljivki – eno za primer, ko je vozlišče izbrano, in eno za primer, ko ni.
- Natančno opiši postopek (z besedami ali psevdokodo), ki iz zgoraj izračunanih vrednosti sestavi najtežjo neodvisno množico v  $T$ .
- Oceni časovno zahtevnost algoritma iz točk (b) in (c).
- S svojim algoritmom poišči najtežjo neodvisno množico na drevesu s slike 33. Iz rešitve naj bo jasno, kako poteka izvajanje algoritma.

#### Naloga 5.46.

Vir: Izpit OR 19.6.2019

Trg mobilne telefonije je zelo konkurenčen, zato si mobilni operaterji na vse kriptične prizadevajo pridobiti stranke svojih konkurentov. Tako zelo, da lahko v nekaterih primerih stranka s preходом h konkurentu celo zasluži. Da pa vendarle



Slika 34: Graf za nalogo 5.46.

operaterji sami ne bi imeli prevelike izgube, stranke zadržujejo z različnimi vezavami, poleg tega pa novim strankam (tistim, ki še niso imele mobilnega telefona) krepko zaračunajo priklop.

Stanje trga zberemo v utežen usmerjen graf  $G = (V, E)$ , kjer vozlišča predstavljajo operaterje, utež  $L_{uv}$  povezave  $uv \in E$  pomeni strošek prehoda od operaterja  $u$  k operaterju  $v$ , utež  $c_v$  vozlišča  $v \in V$  pa pomeni strošek, ki nastane tekom trajanja vezave pri operaterju  $v$  (ko preidemo k operaterju  $v$ , imamo torej  $c_v$  dodatnih stroškov). Če povezave  $uv$  ni v grafu, potem neposreden prehod od  $u$  k  $v$  ni mogoč. Uteži vozlišč so nenegativne, uteži povezav pa so lahko tudi negativne (tj., če ob prehodu zaslužimo). Poiskati želimo najcenejše zaporedje prehodov med operaterji (tj., tako, pri katerem imamo najmanj stroškov), če začnemo pri operaterju  $s$  in končamo pri operaterju  $t$ .

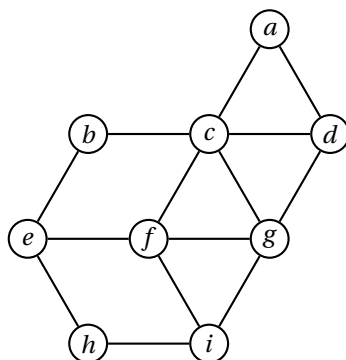
- Predlagaj čim bolj učinkovit algoritem za reševanje zgornjega problema. Kakšna je njegova časovna zahtevnost? Lahko predpostaviš, da velja  $c_s = c_t = 0$ .
- Denimo, da trenutno še nimamo mobilnega telefona. Naši izračuni kažejo, da je dolgoročno najugodnejša ponudba operaterja *Rega*, tako da bi radi s čim manjšimi stroški postali njihov naročnik. S pomočjo algoritma iz točke (a) poišči ustrezno zaporedje prehodov na grafu s slike 34.

#### Naloga 5.47.

Vir: Izpit OR 4.6.2020

Dana sta neutežen neusmerjen graf  $G = (V, E)$  z  $n$  vozlišči in  $m$  povezavami ter vozlišče  $s \in V$ . Za vsako vozlišče  $v \in V$  nas zanima, koliko najkrajših poti od  $s$  do





Slika 35: Graf za nalogo 5.47.

$\nu$  je v grafu  $G$  (tj., koliko je takšnih poti od  $s$  do  $\nu$ , katerih dolžina je enaka razdalji med  $s$  in  $\nu$  v grafu  $G$ ).

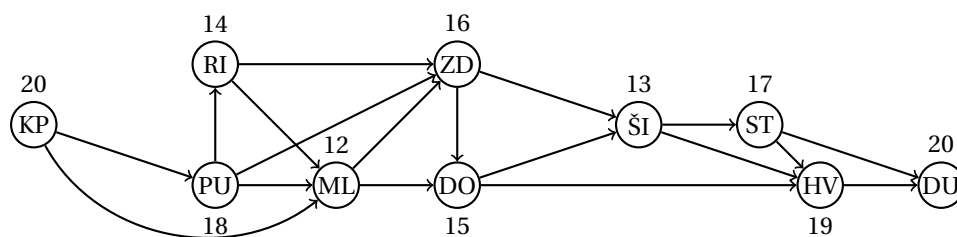
- Čim bolj natančno opiši algoritem, ki reši zgornji problem v času  $O(m)$ .
- Uporabi svoj algoritem, da za vsako vozlišče v grafu s slike 35 poiščeš število najkrajših poti od vozlišča  $i$ . Natančno zabeleži, kaj se zgodi v vsakem koraku algoritma.

#### Naloga 5.48.

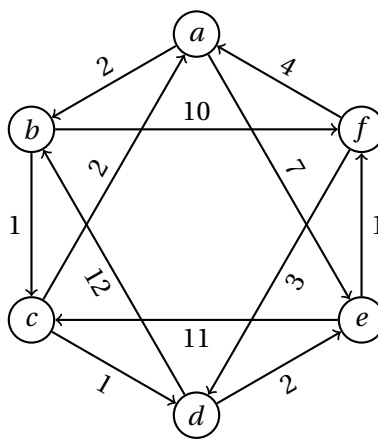
Vir: Izpit OR 22.6.2020

Večja skupina prijateljev želi preživeti počitnice na jadrnici. Ker pa je na sami jadrnici le manjše število ležišč, so se odločili, da bodo prenočevali na kopnem vzdolž poti (vključno z začetnim in končnim krajem). Sestavili so usmerjen aciklični graf  $G = (V, E)$  z  $n$  vozlišči in  $m$  povezavami, v katerem vozlišča predstavljajo kraje, kjer se lahko ustavijo, vozlišči  $u, v \in V$  pa sta povezani z usmerjeno povezavo  $uv \in E$ , če lahko v enem dnevu plujejo od kraja  $u$  do kraja  $v$ . Začetna in končna točka poti sta predstavljeni z vozliščema  $s, t \in V$ . Poleg tega za vsako vozlišče  $u \in V$  poznajo še število  $k_u$ , ki predstavlja, koliko oseb lahko prespi v kraju  $u$ . Sestaviti želijo tako pot od  $s$  do  $t$ , da bo lahko na jadrnici potovalo čim več prijateljev (tj., maksimizirati želijo minimalno vrednost  $k_u$  vseh obiskanih krajev  $u$  na poti). Dolžina poti pri tem ni pomembna.

- Natančno opiši čim učinkovitejši algoritem (z besedami ali psevdokodo) za reševanje zgornjega problema in določi njegovo časovno zahtevnost.
- Uporabi svoj algoritem, da v grafu  $G = (V, E)$  s slike 36 poiščeš tako pot od KP do DU, da bo lahko na jadrnici potovalo čim večje število prijateljev. Vrednosti  $k_u$  ( $u \in V$ ) so zapisane ob vsakem vozlišču. Natančno zabeleži, kaj se zgodi v vsakem koraku algoritma.



Slika 36: Graf za nalogo 5.48(b).



Slika 37: Graf za nalogo 5.49(b).

**Naloga 5.49.**

Vir: Kolokvij OR 3.6.2021

Dan je usmerjen utežen graf  $G = (V, E)$  s cenami povezav  $c_e$  ( $e \in E$ ) ter njegovo vozlišče  $s$  in celo število  $t$  ( $0 \leq t < n = |V|$ ). V grafu  $G$  želimo poiskati najkrajše poti od vozlišča  $s$  do ostalih vozlišč grafa, ki vsebujejo največ  $t$  povezav.

- (a) Predlagaj čim učinkovitejši algoritem za reševanje danega problema. Kakšna je njegova časovna zahtevnost?

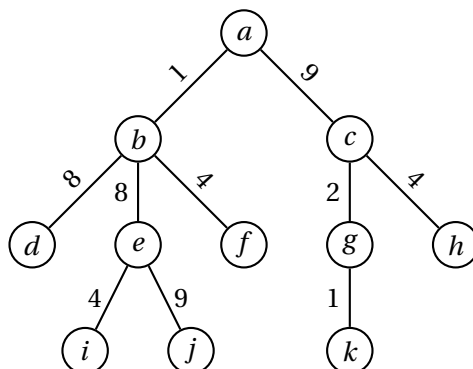
**Opozorilo:** pazi, da bo mogoče iz izhoda rekonstruirati vse želene poti.

- (b) S svojim algoritmom poišči najkrajše poti od vozlišča  $a$  do ostalih vozlišč v grafu s slike 37, ki vsebujejo največ 4 povezave. Iz rešitve naj bo jasno, kako poteka izvajanje algoritma.

**Naloga 5.50.**

Vir: Kolokvij OR 3.6.2021

*Prيرهjanje* v grafu  $G = (V, E)$  je taka množica  $M \subseteq E$ , da je vsako vozlišče  $u \in V$  krajišče največ ene povezave iz  $M$ .



Slika 38: Drevo za nalogo 5.50(d).

Dano je drevo  $T = (V, E)$  in uteži povezav  $c_e$  ( $e \in E$ ). V drevesu  $T$  želimo najti *najtežje prirejanje* – torej tako množico povezav  $M \subseteq E$ , ki maksimizira vsoto njihovih uteži, torej vrednost  $\sum_{e \in M} c_e$ .

- (a) Napiši rekurzivne enačbe za reševanje problema najtežjega prirejanja v drevesu  $T$ . Razloži, kaj predstavljajo spremenljivke, v kakšnem vrstnem redu jih računamo, ter kako dobimo optimalno rešitev. Predpostaviš lahko, da imaš poleg seznamov sosedov in uteži povezav drevesa  $T$  na voljo tudi tabelo *pred* (glej nalogo 5.45(a)), ki za vsako vozlišče  $v \in V$  določa njegovega prednika, če za koren izbereš vozlišče  $r \in V$ . Koren  $r$  je lahko izbran poljubno, zanj pa velja  $pred[r] = \text{NULL}$ .

**Namig:** za vsako vozlišče uporabi dve spremenljivki – eno za primer, ko je vozlišče krajišče povezave iz prirejanja, in eno za primer, ko ni.

- (b) Natančno opiši postopek (z besedami ali psevdokodo), ki iz zgoraj izračunanih vrednosti sestavi najtežje prirejanje v  $T$ .
- (c) Oceni časovno zahtevnost algoritma iz točk (a) in (b).
- (d) S svojim algoritmom poišči najtežje prirejanje na drevesu s slike 38. Iz rešitve naj bo jasno, kako poteka izvajanje algoritma.



faza	opis	trajanje	pogoj	min. trajanje	cena za dan manj
<i>a</i>	gradnja kleti	10 dni	/	7 dni	200
<i>b</i>	gradnja pritličja	6 dni	<i>a</i>	5 dni	100
<i>c</i>	gradnja prvega nadstropja	7 dni	<i>b, f</i>	5 dni	150
<i>d</i>	gradnja strehe	8 dni	<i>c, e</i>	6 dni	160
<i>e</i>	gradnja desnega podpornega stebra	13 dni	<i>a</i>	9 dni	250
<i>f</i>	gradnja glavnega podpornega stebra	14 dni	/	11 dni	240
<i>g</i>	gradnja baročnega stolpa pred hišo	30 dni	/	25 dni	300

Tabela 3: Podatki za nalogi 6.1 (prvi štirje stolpci) in 6.2.

## 1.6 CPM/PERT

### Naloga 6.1.

Vir: Kolokvij OR 9.5.2013

Gradbinec in samooklicani arhitekt Brezzobec se je odločil, da bo postavil zelo posebno hišo. Gradnja bo imela sedem glavnih faz, ki so opisane v tabeli 3.

- Kdaj je lahko hiša najhitreje zgrajena? Katere faze so kritične?
- Koliko je kritičnih poti in katere so?
- Katero opravilo je najmanj kritično? Najmanj kritično je opravilo, katerega trajanje lahko največ podaljšamo, ne da bi vplivali na trajanje gradnje.
- Brezzobčev brat je ponudil pomoč pri največ eni fazi gradnje. Slovi po tem, da pri fazi, pri kateri pomaga, zmanjša čas izvajanja za 10%. Pri kateri fazi naj pomaga, da bo čas gradnje čim krajši?

### Naloga 6.2.

Vir: Vaje OR 28.5.2018

Brezzobčev bratranec ima podjetje, ki lahko pomaga pri gradnji, vendar za vsak dan krajšanja posamezne faze zahteva ustrezno plačilo (glej zadnja dva stolpca tabele 3). Brezzobca zanima način, kako bi s čim manjšimi stroški čas gradnje zmanjšal na 27 dni. Zapiši linearni program za ta problem.

### Naloga 6.3.

Vir: Izpit OR 19.5.2015

Dinamika priprave dveh palačink z dvema kuharjema je opisana v tabeli 4.

- Topološko uredi ustrezni graf in ga nariši.
- Določi kritična opravila in kritično pot ter trajanje priprave.
- Katero opravilo je najmanj kritično?

	aktivnost	trajanje	predhodna opravila
<i>a</i>	nakup moke, jajc in mleka	5 min	<i>c</i>
<i>b</i>	rezanje sira	3 min	/
<i>c</i>	vožnja do trgovine	5 min	/
<i>d</i>	čiščenje mešalnika	2 min	<i>e</i>
<i>e</i>	mešanje sestavin	5 min	<i>a</i>
<i>f</i>	pečenje prve palačinke	2 min	<i>e</i>
<i>g</i>	mazanje prve palačinke z marmelado	1 min	<i>f, j</i>
<i>h</i>	pečenje palačinke (s sirom)	3 min	<i>b, f</i>
<i>i</i>	pomivanje posode	8 min	<i>g, h</i>
<i>j</i>	odpiranje marmelade	1 min	/

Tabela 4: Podatki za naloge 6.3, 6.4 in 6.5.

**Naloga 6.4.**

Vir: Vaje OR 18.5.2020

Določi skupne, proste, varnostne in neodvisne rezerve opravil iz naloge 6.3.

**Naloga 6.5.**

Vir: Vaje OR 14.12.2016

Določi razpored opravil iz naloge 6.3, pri čemer en kuhar prevzame opravila na kritični poti, drugi pa naj čim kasneje začne in čim prej konča.

**Naloga 6.6.**

Vir: [3, Problem 10.4-4]

Pri gradbenem podjetju razmišljajo, da bi se prijavili na razpis za prenovno avtocestnega viadukta. Identificirali so pet nalog, ki so opisane v tabeli 5.

Če bodo izbrani za izvedbo del, si obetajo zaslužek v višini 250 000 €. Če del ne bodo končali v roku 11 tednov, bodo morali plačati pogodbeno kazen v višini 500 000 €.

- Za vsako opravilo določi pričakovano trajanje in varianco.
- Topološko uredi ustrezni graf in ga nariši.
- Določi pričakovano kritično pot ter trajanje izvedbe.
- Oceni verjetnost, da se bo projekt zaključil v 11 tednih. Naj se podjetje prijavi na razpis?

**Naloga 6.7.**

Vir: Izpit OR 31.1.2017

Izdelati želimo terminski plan za izdelavo spletne aplikacije. V tabeli 6 so zbrana opravila pri izdelavi.

- Topološko uredi ustrezni graf in ga nariši.
- Določi kritična opravila in kritično pot ter čas izdelave.

naloga	najkrajše trajanje	najbolj verjetno trajanje	najdaljše trajanje	predhodna opravila
<i>a</i>	3 tedni	4 tedni	5 tednov	/
<i>b</i>	2 tedna	2 tedna	2 tedna	<i>a</i>
<i>c</i>	3 tedni	5 tednov	6 tednov	<i>b</i>
<i>d</i>	1 teden	3 tedni	5 tednov	<i>a</i>
<i>e</i>	2 tedna	3 tedni	5 tednov	<i>b, d</i>

Tabela 5: Podatki za nalogo 6.6.

opravilo	opis	trajanje	pogoji
<i>a</i>	natančna opredelitev funkcionalnosti	15 dni	/
<i>b</i>	programiranje uporabniškega vmesnika	40 dni	<i>k</i>
<i>c</i>	programiranje skrbniškega vmesnika	25 dni	<i>a, m</i>
<i>d</i>	programiranje strežniškega dela	30 dni	<i>a, m</i>
<i>e</i>	integracija uporabniškega vmesnika s strežnikom	20 dni	<i>b, d</i>
<i>f</i>	alfa testiranje	20 dni	<i>c, e</i>
<i>g</i>	beta testiranje	30 dni	<i>f, h</i>
<i>h</i>	pridobivanje testnih uporabnikov	45 dni	<i>a</i>
<i>i</i>	vnos zadnjih popravkov	10 dni	<i>g</i>
<i>j</i>	izdelava uporabniške dokumentacije	35 dni	<i>b</i>
<i>k</i>	dizajniranje uporabniškega vmesnika	15 dni	<i>a</i>
<i>ℓ</i>	nabava računalniške opreme	20 dni	/
<i>m</i>	postavitev strežnikov	10 dni	<i>ℓ</i>

Tabela 6: Podatki za nalogo 6.7.

- (c) Katero opravilo je najmanj kritično? Najmanj kritično je opravilo, katerega trajanje lahko najbolj podaljšamo, ne da bi vplivali na trajanje izdelave.

**Naloga 6.8.**

Vir: Kolokvij OR 11.6.2018

Izdelati želimo terminski plan za organizacijo konference. V tabeli 7 so zbrana opravila pri organizaciji.

- (a) Topološko uredi ustrezni graf in ga nariši. Za trajanja opravil vzemi pričakovana trajanja po modelu PERT.
- (b) Določi pričakovano kritično pot in čas izdelave.
- (c) Katero opravilo je (ob zgornjih predpostavkah) najmanj kritično? Najmanj kritično je opravilo, katerega trajanje lahko najbolj podaljšamo, ne da bi vplivali na celotno trajanje izvedbe.
- (d) Določi variance trajanj opravil in oceni verjetnost, da bo izvedba trajala manj kot 55 dni.

	Opravo	Pogoji	Minimalno trajanje	Najbolj verjetno trajanje	Maksimalno trajanje
<i>a</i>	Izbira lokacije	/	10 dni	13 dni	22 dni
<i>b</i>	Rezervacija sob za goste	<i>f</i>	13 dni	22 dni	25 dni
<i>c</i>	Dogovarjanje za cene hotelskih sob	<i>a</i>	3 dni	6 dni	9 dni
<i>d</i>	Naročilo hrane in pijače	<i>a</i>	6 dni	15 dni	21 dni
<i>e</i>	Priprava letakov	<i>c, j</i>	5 dni	8 dni	11 dni
<i>f</i>	Pošiljanje letakov	<i>e</i>	4 dni	4 dni	4 dni
<i>g</i>	Priprava zbornika s povzetki	<i>d, j</i>	22 dni	28 dni	31 dni
<i>h</i>	Določitev glavnega govorca	/	5 dni	8 dni	14 dni
<i>i</i>	Planiranje poti za glavnega govorca	<i>a, h</i>	11 dni	14 dni	17 dni
<i>j</i>	Določitev ostalih govorcev	<i>h</i>	12 dni	15 dni	21 dni
<i>k</i>	Planiranje poti za ostale govorce	<i>a, j</i>	9 dni	12 dni	18 dni

Tabela 7: Podatki za nalogo 6.8.

opravilo	opis	trajanje	pogoji	najmanjše trajanje	cena za dan manj
<i>a</i>	izgradnja nosu	40 dni	/	36 dni	1 000
<i>b</i>	izgradnja trupa s krili	50 dni	/	48 dni	1 500
<i>c</i>	izgradnja repa	35 dni	/	31 dni	800
<i>d</i>	vgraditev pilotske kabine	30 dni	<i>a</i>	28 dni	1 200
<i>e</i>	opremljanje potniške kabine	18 dni	<i>f, g</i>	16 dni	500
<i>f</i>	povezovanje nosu s trupom	10 dni	<i>a, b</i>	9 dni	1 100
<i>g</i>	povezovanje repa s trupom	12 dni	<i>b, c</i>	10 dni	1 300
<i>h</i>	vgraditev motorjev	20 dni	<i>b</i>	18 dni	1 400

Tabela 8: Podatki za nalogo 6.9.

### Naloga 6.9.

Vir: Izpit OR 28.8.2018

Pri izdelavi letala imamo faze, opisane v tabeli 8.

- S pomočjo topološke ureditve ustreznega grafa določi kritična opravila in čas izdelave. Uporabi podatke iz stolpca "trajanje".
- Katero opravilo je najmanj kritično? Najmanj kritično jo opravilo, katerega trajanje lahko najbolj podaljšamo, ne da bi vplivali na trajanje izdelave.
- Naročnik bi rad, da letalo izdelamo v 75 dneh. Posamezna opravila lahko skrajšamo tako, da zanje zadolžimo več delavcev. Seveda prinese tako krajšanje dodatne stroške, poleg tega pa za vsako opravilo poznamo najmanjše možno trajanje (glej zadnja dva stolpca v zgornji tabeli). Zapiši linearni program, s katerim modeliramo iskanje razporeda opravil, ki nam prinese čim manjše stroške. Linearne programa ne rešuj.



	aktivnost	trajanje	predhodna opravila
<i>a</i>	postavitev stebra 1	4 dni	/
<i>b</i>	postavitev stebra 2	3 dni	<i>d</i>
<i>c</i>	postavitev stebra 3	5 dni	<i>h</i>
<i>d</i>	postavitev stebra 4	6 dni	<i>a, e</i>
<i>e</i>	postavitev stene 1	4 dni	<i>a, g</i>
<i>f</i>	postavitev stene 2	2 dni	<i>h, i</i>
<i>g</i>	postavitev stene 3	2 dni	/
<i>h</i>	postavitev stene 4	1 dan	/
<i>i</i>	postavitev stene 5	6 dni	<i>c</i>
<i>j</i>	betoniranje plošče	2 dni	<i>c, e, f</i>

Tabela 9: Podatki za nalogo 6.10.

**Naloga 6.10.**

Vir: Izpit OR 28.8.2015

Gradimo objekt in želimo narediti opravila iz tabele 9, za katera vemo tudi, katera druga opravila morajo biti predhodno izvedena.

- Opravila *a–j* določajo usmerjen acikličen graf, kjer so povezave določene s predhodnimi opravili. Nariši ta graf in ga topološko uredi (lahko ga že narišeš topološko urejenega).
- Določi kritična opravila, kritično pot in trajanje gradnje.
- Katero opravilo je najmanj kritično?

**Naloga 6.11.**

Vir: Kolokvij OR 3.6.2019

Izdelati želimo terminski plan za izgradnjo manjšega stanovanjskega naselja. V tabeli 10 so zbrana opravila pri gradnji.

- Topološko uredi ustrezni graf in ga nariši. Za trajanja opravil vzemi pričakovana trajanja po modelu PERT.
- Določi pričakovano kritično pot in čas izdelave.
- Katero opravilo je (ob zgornjih predpostavkah) najmanj kritično? Najmanj kritično je opravilo, katerega trajanje lahko najbolj podaljšamo, ne da bi vplivali na celotno trajanje izvedbe.
- Določi variance trajanj opravil in oceni verjetnost, da bo gradnja trajala manj kot 180 dni.

	Opravo	Pogoji	Minimalno trajanje	Najbolj verjetno trajanje	Maksimalno trajanje
<i>a</i>	pridobitev gradbenega dovoljenja	/	14 dni	14 dni	23 dni
<i>b</i>	pridobitev uporabnega dovoljenja	<i>f, g</i>	18 dni	21 dni	30 dni
<i>c</i>	izgradnja komunalne infrastrukture	<i>a</i>	40 dni	45 dni	59 dni
<i>d</i>	postavitev temeljev	<i>a, i</i>	28 dni	37 dni	46 dni
<i>e</i>	izgradnja podpornega zidu	<i>a</i>	7 dni	8 dni	12 dni
<i>f</i>	izgradnja severne stolpnice	<i>c, d</i>	63 dni	78 dni	99 dni
<i>g</i>	izgradnja južne stolpnice	<i>d, e</i>	75 dni	84 dni	99 dni
<i>h</i>	izgradnja otroškega igrišča	<i>a, i</i>	20 dni	29 dni	29 dni
<i>i</i>	odstranitev rastja	/	19 dni	22 dni	25 dni

Tabela 10: Podatki za nalogo 6.11.

opravilo	opis	trajanje	pogoji	najmanjše trajanje	cena za dan manj
<i>a</i>	izgradnja južnega priključka	45 dni	/	40 dni	300
<i>b</i>	izgradnja severnega priključka	20 dni	/	15 dni	200
<i>c</i>	vrtanje tunelske cevi	90 dni	<i>a, b</i>	70 dni	700
<i>d</i>	postavitev zahodnega stebra viadukta	25 dni	<i>a</i>	22 dni	1 100
<i>e</i>	postavitev vzhodnega stebra viadukta	30 dni	<i>c</i>	26 dni	1 500
<i>f</i>	gradnja cestišča na viaduktu	35 dni	<i>d, e</i>	28 dni	900
<i>g</i>	ureditev napeljave v tunelu	80 dni	<i>c</i>	65 dni	400
<i>h</i>	asfaltiranje viadukta	10 dni	<i>f</i>	8 dni	150

Tabela 11: Podatki za nalogo 6.12.

### Naloga 6.12.

Vir: Izpit OR 19.6.2019

Gradbeno podjetje gradi avtocestni odsek na težavnem terenu. Identificirali so faze gradnje, ki so zbrane v tabeli 11.

- S pomočjo topološke ureditve ustreznega grafa določi kritična opravila in čas izdelave. Uporabi podatke iz stolpca "trajanje".
- Katero opravilo je najmanj kritično? Najmanj kritično jo opravilo, katerega trajanje lahko najbolj podaljšamo, ne da bi vplivali na trajanje izdelave.
- Naročnik bi rad, da podjetje odsek zgradi v 200 dneh. Posamezna opravila lahko skrajšajo tako, da zanje zadolžijo več delavcev. Seveda prinese tako krajšanje dodatne stroške, poleg tega pa za vsako opravilo poznamo najmanjše možno trajanje (glej zadnja dva stolpca v tabeli 11). Zapiši linearni program, s katerim modeliramo iskanje razporeda opravil, ki nam prinese čim manjše stroške. Linearne programa ne rešuj.

	Opravilo	Pogoji	Minimalno trajanje	Najbolj verjetno trajanje	Maksimalno trajanje
<i>a</i>	določitev trase	/	24 dni	30 dni	36 dni
<i>b</i>	priprava trase za kronometer	<i>a</i>	5 dni	6 dni	10 dni
<i>c</i>	priprava trase za gorsko etapo	<i>a</i>	20 dni	20 dni	26 dni
<i>d</i>	priprava trase za zaključno etapo	<i>a</i>	6 dni	7 dni	8 dni
<i>e</i>	pridobivanje tujih udeležencev	<i>a</i>	16 dni	17 dni	21 dni
<i>f</i>	izvedba regionalnih kvalifikacij	/	15 dni	18 dni	24 dni
<i>g</i>	izvedba kronometra	<i>b, e, f</i>	1 dan	1 dan	1 dan
<i>h</i>	izvedba gorske etape	<i>c, g</i>	1 dan	1 dan	1 dan
<i>i</i>	izvedba zaključne etape	<i>d, h</i>	1 dan	1 dan	1 dan
<i>j</i>	priprava zaključne prireditve	<i>e, f</i>	4 dni	5 dni	6 dni

Tabela 12: Podatki za nalogi 6.13 in 6.14.

**Naloga 6.13.**

Vir: Izpit OR 27.8.2019

Izdelati želimo terminski plan za pripravo in izvedbo kolesarske dirke po Sloveniji. V tabeli 12 so zbrana opravila pri projektu.

- Topološko uredi ustrezni graf in ga nariši. Za trajanja opravil vzemi pričakovana trajanja po modelu PERT.
- Določi pričakovano kritično pot in čas izdelave.
- Katero opravilo je (ob zgornjih predpostavkah) najmanj kritično? Najmanj kritično je opravilo, katerega trajanje lahko najbolj podaljšamo, ne da bi vplivali na celotno trajanje izvedbe.

**Naloga 6.14.**

Vir: Kolokvij OR 3.6.2021

Določi variance trajanj opravil in oceni verjetnost, da bo projekt iz naloge 6.13 trajal manj kot 55 dni.



## 1.7 Upravljanje zalog

### Naloga 7.1.

Vir: [3, Problem 19.3-2]

V trgovini vsak teden prodajo 600 škatel toaletnega papirja. Vsako naročilo stane 25€, za posamezno škatlo pa trgovina plača 3€. Cena skladiščenja posamezne škatle je 0.05€ na teden.

- (a) Denimo, da primanjkljaj ni dovoljen. Kako pogosto naj trgovina naroča toaletni papir? Kako velika naj bodo naročila?
- (b) Kaj pa, če dovolimo primanjkljaj, ki nas stane 2€ na teden za vsako škatlo?

### Naloga 7.2.

Vir: [1, Naloga 10.7]

Tovarna Mitsuzuki načrtuje, da bo naslednje leto izdelala 10 000 stereo televizorjev. Odločijo se, da zvočnikov ne bodo izdelovali sami, ampak jih bodo (po dva za vsak televizor) naročili pri ponudniku najkvalitetnejših zvočnikov, ki je predložil naslednji cenik:

število zvočnikov v enem naročilu	cena zvočnika
1 – 1 999	150€
2 000 – 4 999	135€
5 000 – 7 999	125€
8 000 – 19 999	120€
20 000 ali več	115€

Poleg tega naročilo pošiljke stane Mitsuzuki 500€, letni stroški skladiščenja vsakega zvočnika pa znašajo 20% njegove cene. Koliko zvočnikov naj vsakič naročijo in kakšni so skupni stroški naročil za naslednje leto?

### Naloga 7.3.

Vir: [5, §15, Example 5]

Tovarna avtomobilov za svoje potrebe izdeluje akumulatorje. Vsako leto proizvedejo 10 000 avtomobilov, izdelajo pa lahko 25 000 akumulatorjev letno. Stroški zagona proizvodnje so 200€, stroški zaloge pa 0.25€ letno za vsak akumulator. Primanjkljaja ne dovolimo. Kolikokrat na leto naj zaženejo proizvodnjo akumulatorjev? Koliko časa naj traja proizvodnja? Koliko akumulatorjev naj takrat izdelajo? Kako veliko skladišče potrebujejo?

### Naloga 7.4.

Vir: Vaje OR 6.6.2018

Marta izdeluje nakit iz školjk v delavnici, ki jo najema v bližini ljubljanske tržnice, in ga prodaja po vnaprej dogovorjeni ceni. Povpraševanje je 10 kosov na teden, stroški skladiščenja so 0.2€ za kos na teden. Zagonski stroški izdelovanja nakita so 150€. Marta lahko na teden izdela 12.5 kosa nakita. Dovolj si, da pride do primanjkljaja, pri čemer jo ta stane 0.8€ za kos nakita na teden. Kako naj Marta organizira proizvodnjo in skladiščenje, da bo imela čim manj stroškov?

**Naloga 7.5.**

Vir: Kolokvij OR 11.6.2018

Vulkanizer v svoji delavnici izdeluje in prodaja avtomobilske pnevmatike. Glede na trenutno povpraševanje vsak teden proda 60 pnevmatik, v istem času pa jih lahko izdelata 90. Cena zagona proizvodnje je 180€, cena skladiščenja posamezne pnevmatike pa je 0.5€ na teden.

- (a) Denimo, da primanjkljaja ne dovolimo. Izračunaj dolžino cikla proizvodnje in prodaje pnevmatik, pri kateri so stroški najmanjši. Kako veliko skladišče mora vulkanizer imeti? Izračunaj tudi enotske stroške.
- (b) Kako dolgo naj pri zgornji rešitvi traja proizvodnja? Koliko pnevmatik naj vulkanizer naredi v vsakem ciklu?
- (c) Vulkanizer je ocenil, da bi ga primanjkljaj ene pnevmatike stal 2€ na teden. Kakšna naj bosta dolžina cikla in velikost skladišča, da bodo stroški čim manjši? Izračunaj tudi enotske stroške in določi največji primanjkljaj.

**Naloga 7.6.**

Vir: Izpit OR 5.7.2018

V trgovini s pohištvom vsak mesec prodajo 20 sedežnih garnitur. Z vsakim naročilom imajo 750€ stroškov, pri tem pa vse naročeno blago dobijo hkrati. Skladiščenje posamezne sedežne garniture jih stane 10€ na mesec, primanjkljaj za en kos pa jih stane 50€ na mesec.

- (a) Kako pogosto naj v trgovini naročajo sedežne garniture, da bodo stroški čim manjši? Kako veliko skladišče morajo imeti? Izračunaj tudi enotske stroške.
- (b) Izračunaj največji dovoljeni primanjkljaj. Koliko sedežnih garnitur naj vsakič naročijo?
- (c) V trgovini dobijo ponudbo za uporabo drugega skladišča, v katerem imajo za posamezno sedežno garnituro le 6€ stroškov na mesec, vendar lahko hrani največ 40 sedežnih garnitur. Kako naj organizirajo naročanje, če namesto prvotnega uporabljajo to skladišče? Ali se jim ga splača uporabiti?  
**Namig:** izpelji formulo za enotske stroške in poišči optimalen interval naročanja pri fiksni velikosti skladišča.

**Naloga 7.7.**

Vir: Izpit OR 22.6.2020

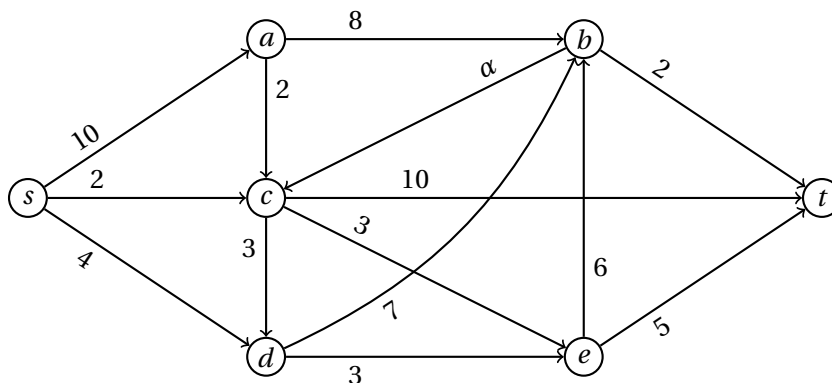
V šiviljski delavnici so začeli proizvajati zaščitne maske. Vsak dan lahko izdelajo 400 mask, prodajo pa jih 300. Cena zagona proizvodnje je 240€, cena skladiščenja ene maske za en dan pa je 0.1€.

- (a) Denimo, da primanjkljaj ni dovoljen. Kako pogosto naj zaženejo proizvodnjo in koliko časa naj ta teče? Koliko mask naj izdelajo v posameznem ciklu ter kako veliko skladišče potrebujejo?

- (b) Obravnavajmo še primer, ko dovolimo primanjkljaj, ki nas stane 0.4€ na dan za eno masko. Izračunaj podatke iz prejšnje točke še za ta primer. Kolikšen je največji dovoljeni primanjkljaj?







Slika 39: Omrežje za nalogo 8.1.

## 1.8 Pretoki in prerezi

### Naloga 8.1.

Vir: Kolokvij OR 19.11.2009

Poišči minimalno vrednost parametra  $\alpha$ , da bo pretok skozi omrežje s slike 39 maksimalen. Utemelji izbiro.

### Naloga 8.2.

Vir: Kolokvij OR 26.1.2010

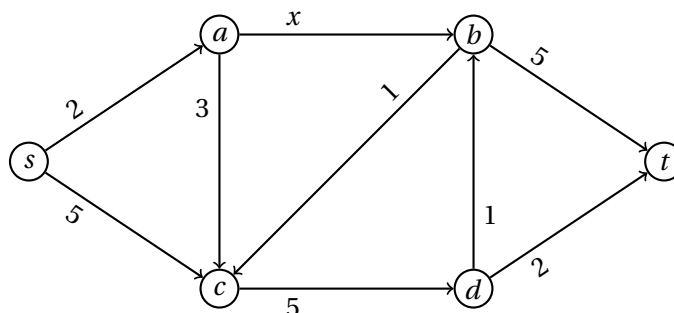
Imamo podatke za problem maksimalnega pretoka. Ali je povezava z najmanjšo kapaciteto (predpostavimo, da je natanko ena taka) vedno vsebovana v kakem minimalnem prerezu? Ali je morda vsebovana celo v vsakem minimalnem prerezu?

### Naloga 8.3.

Vir: Izpit OR 5.2.2010

Na univerzi v mestu Lenority so asistenti res nekoliko razvajeni in imajo močen sindikat. (To ni nič nenavadnega, če pa univerza zahteva, da se izpiti začnejo že ob sedmih zjutraj!) Bliža se izpitno obdobje in potrebno je paziti izpite. Za vsak dan so izpiti dani vnaprej. Znano je, da asistenti ne pazijo izpitov, če so vmes "luknje", saj si je to pravico izbral njihov sindikat. Vodstvu univerze tako ne preostane nič drugega, kot da to upošteva. Asistenti torej pridejo na faks ob neki uri, pazijo poljubno dolgo zaporedje izpitov brez odmorov, in potem gredo domov. Izpiti so podani v obliki časovnih intervalov, ki se začnejo in končajo ob polni uri. Upoštevajoč te pogoje mora vodstvo univerze venomer prositi asistente, naj pridejo pazit izpite. Toda tudi vodstvo si želi minimizirati "bolečino" ob moledovanju in hoče vsak dan prositi kar se da malo asistentov, naj vendarle pridejo pazit izpite.

- (a) Modeliraj problem s pomočjo usmerjenih acikličnih grafov, pri katerih so intervali predstavljeni z vozlišči. Identificiraj, kaj je tvoja optimizacijska naloga na takem grafu.



Slika 40: Omrežje za nalogo 8.4.

- (b) Opiši, kako problem prevedeš na problem maksimalnih pretokov. Natančno utemelji, zakaj so maksimalni pretoki v tvojem modelu v bijektivni korespondenci z optimalnimi rešitvami tvoje optimizacijske naloge.
- (c) S pomočjo algoritma iz prejšnje točke reši problem za petkove izpite, ki so podani z intervali (7, 8), (8, 9), (9, 10), (8, 12), (10, 13), (12, 15), (13, 15), (9, 12), (7, 10).

**Naloga 8.4.**

Vir: Izpit OR 28.6.2010

Na sliki 40 je podano enoparametrično omrežje za problem največjega pretoka. Parameter  $x$  je poljubno nenegativno število.

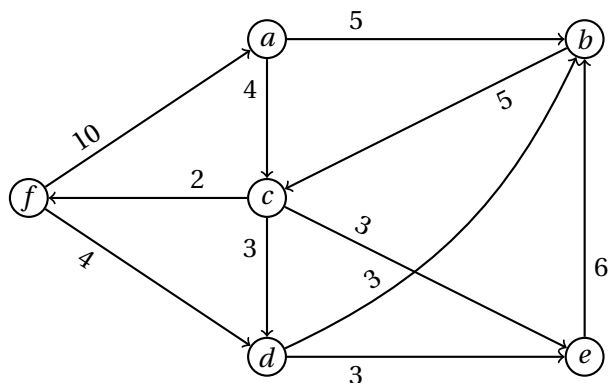
- (a) Kakšno je pridruženo omrežje, če začnemo z ničelnim  $(s, t)$ -tokom in ga najprej povečamo vzdolž povečujoče poti  $s \rightarrow a \rightarrow c \rightarrow d \rightarrow t$  ter nato še vzdolž povečujoče poti  $s \rightarrow c \rightarrow d \rightarrow b \rightarrow t$ ?
- (b) Poišči maksimalni  $(s, t)$ -tok in minimalni  $(s, t)$ -prerez.
- (c) Kapaciteto ene povezave vhodnega omrežja lahko povečamo za 1. Kateri povezavi naj povečamo kapaciteto, da bo vrednost maksimalnega  $(s, t)$ -toka v spremenjenem omrežju čim večja? Odgovor je načeloma odvisen od vrednosti parametra  $x$ .

**Naloga 8.5.**

Vir: Izpit OR 15.9.2010

Na sliki 41 je podano omrežje enosmernih prehodov med letališkimi terminali skupaj s prepuštnostmi (v 1000 ljudeh na uro).

- (a) Načrtovalce zanima, koliko največ potnikov na uro lahko preide od terminala  $f$  do terminala  $b$ . Predlagaj ustrezen algoritem in ga izvedi na grafu.



Slika 41: Omrežje za nalogo 8.5.

- (b) V izrednih razmerah dva terminala proglasijo za vstopna (na njih letala le pristajajo), ostale terminale pa za izstopne (letala le vzletajo). Tako morajo potniki nujno prehajati od vstopnih k izstopnim terminalom. Strateška odločitev je, da je vstopni terminal  $f$ , zaradi redundance pa želijo, da bi bila vstopna terminala dva. Ostali terminali so torej izstopni. Katerega izmed terminalov (poleg  $f$ ) se splača še proglasiti za vstopnega, da bo pretočnost med vstopnimi in izstopnimi terminali kar največja?

**Naloga 8.6.**

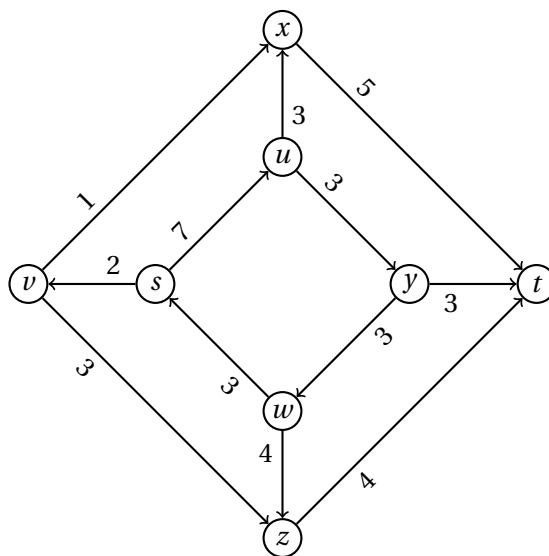
Vir: Kolokvij OR 25.11.2010

Nov in neizkušen predavatelj Operacijskih raziskav se odloči, da bo poenostavil Ford-Fulkersonov algoritem. Študentom pove, naj za iskanje maksimalnega pretoka na omrežju s končnimi kapacitetami uporabijo naslednji algoritem:

```

function GREEDYFLOW( $G = (V, E), C, s, t$ )
   $F \leftarrow$  slovar z vrednostjo 0 za vsako povezavo iz  $E$ 
  for  $e \in E$  do
     $F[e] \leftarrow 0$ 
  end for
  while obstaja pot od  $s$  do  $t$  do
     $P \leftarrow$  poljubna pot od  $s$  do  $t$ 
     $\gamma \leftarrow$  minimalna kapaciteta povezav poti  $P$ 
    for  $e \in P$  do
       $F[e] \leftarrow F[e] + \gamma$ 
      if  $C[e] = \gamma$  then
        odstrani  $e$  iz  $G$ 
      else
         $C[e] \leftarrow C[e] - \gamma$ 
      end if
    end for
  end while
  return  $F$ 
end function

```



Slika 42: Graf za nalogo 8.6(a).

- Dokaži, da lahko na grafu s slike 42 z uporabo algoritma GREEDYFLOW dobimo pravilno rešitev.
- Poišči primer omrežja in izbire poti, tako da algoritem GREEDYFLOW ne da pravilnega rezultata.
- Ali za vsako omrežje obstaja taka izbira poti, da bo algoritem GREEDYFLOW dal maksimalni pretok?
- Dokaži, da je razlika med maksimalnim pretokom in pretokom, ki ga vrne algoritem GREEDYFLOW, lahko poljubno velika.

### Naloga 8.7.

Vir: Izpit OR 9.2.2011

- Poišči usmerjen graf  $G$ , vozlišči  $s$ ,  $t$  in kapacitete povezav, da bo imel  $G$  natanko en minimalen  $(s, t)$ -prerez.
- Poišči usmerjen graf  $G'$ , vozlišči  $s$ ,  $t$  in kapacitete povezav, da bo imel  $G'$  več minimalnih  $(s, t)$ -prerezov.
- Poišči usmerjen graf  $G_n$ , vozlišči  $s$ ,  $t$  in kapacitete povezav, da bo imel  $G_n$  natanko  $n$  minimalnih  $(s, t)$ -prerezov.

**Naloga 8.8.**

Vir: Izpit OR 28.6.2011

Navdušenje nad koncem študijskega leta na ljubljanski univerzi je botrovalo temu, da v ambulantni Univerzitetne klinike 169 študentov išče nujno zdravniško pomoč. Vsak od študentov potrebuje eno enoto krvi, klinika pa ima na zalogi 170 enot krvi. V spodnji tabeli je prikazano število enot krvi posamezne krvne skupine, ki so na zalogi, in število študentov z dano krvno skupino.

Krvna skupina	A	B	O	AB
Zaloga	46	34	45	45
Povpraševanje	39	38	42	50

Osebe s krvno skupino A lahko prejmejo kri tipa A ali O. Osebe tipa B lahko prejmejo kri tipa B ali O. Osebe tipa O lahko prejmejo samo kri tipa O. Osebe tipa AB lahko prejmejo katerokoli krvno skupino. Naša naloga je razporediti enote krvi tako, da bomo z njo oskrbeli kar se da veliko število študentov. Iz podatkov sestavi omrežje in modeliraj nalogo kot problem maksimalnega pretoka. Največ koliko pacientov lahko oskrbimo? Utemelji.

**Naloga 8.9.**

Vir: Izpit OR 24.6.2014

Pri podatkih iz naloge 8.8 poišči še odgovora na naslednji vprašanji.

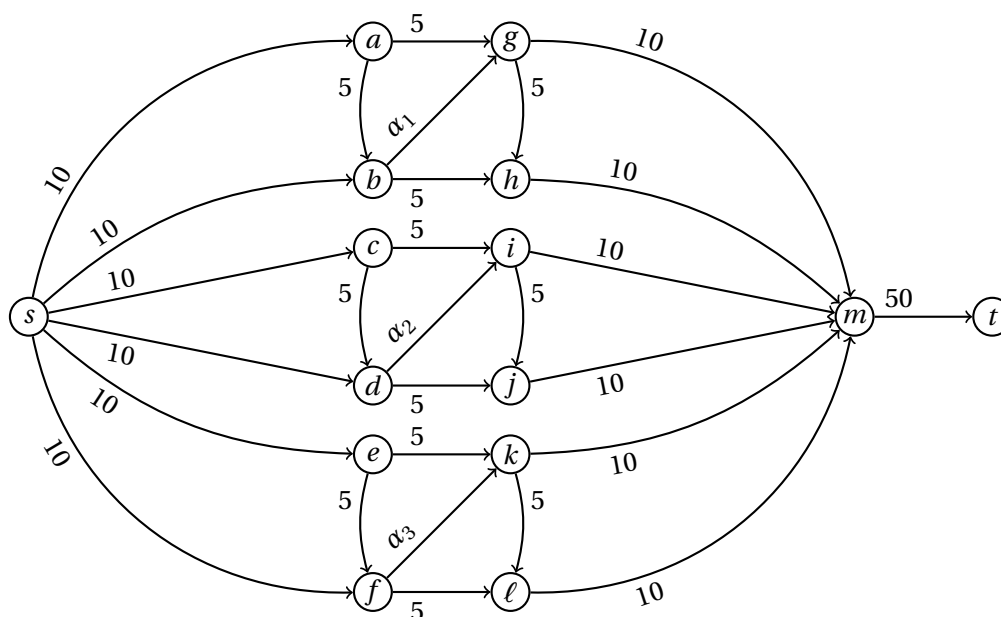
- (a) Koliko enot krvi vsake krvne skupine ostane neporabljenih?
- (b) Koliko enot ustrezne krvne skupine naj na kliniki še naročijo, da bodo pozdravili vse paciente?

**Naloga 8.10.**

Vir: Kolokvij OR 5.4.2012

Iz zajetja  $s$  želimo preko treh čistilnih postaj (vsako predstavlja 4 vozlišča) napeljati vodo do tovarne  $t$ . Opravka imamo z vodovodnim omrežjem, predstavljenim z grafom na sliki 43. Na povezavah so kapacitete cevi (v litrih na sekundo), cevi z oznakami  $\alpha_1$ ,  $\alpha_2$  in  $\alpha_3$  pa še niso zgrajene.

- (a) Koliko vode lahko po danem omrežju dobi tovarna na sekundo?
- (b) Trenutni dotok vode v tovarno za potrebe proizvodnje ni dovolj velik, zato razmišljajo o izgradnji novih cevi. Možne lokacije novih cevi so  $\alpha_1$ ,  $\alpha_2$  in  $\alpha_3$ . Uprava tovarne se je odločila, da bodo s postavitvijo novih cevi na nekaterih možnih lokacijah povečali pretok kar največ, kot je mogoče. Cev kapacitete  $K$  stane  $K \cdot 100$  €. Za inštalacijo ene cevi morajo plačati dodatnih 10000 €. Koliko najmanj jih bo stal projekt?



Slika 43: Omrežje za nalogo 8.10.

### Naloga 8.11.

Vir: Kolokvij OR 5.4.2012

Na divjem žuru matematikov se je zbralo  $n$  ljudi. Organizator je priskrbel  $p$  različnih vrst sendvičev, a ne v neomejenih količinah. Na voljo imajo  $u_i$  sendvičev sorte  $i$  ( $1 \leq i \leq p$ ). Prav tako imajo na voljo  $q$  različnih sokov;  $v_j$  predstavlja število sokov sorte  $j$  ( $1 \leq j \leq q$ ). Za vsakega udeleženca zabave je znano, katere sokove in sendviče ima rad.

- (a) Zanima nas, ali lahko hrano in pijačo razdelimo tako, da dobi vsak udeleženec zabave en sendvič, ki ga ima rad, in en sok, ki ga ima rad. Nalogo formuliraj kot problem maksimalnega pretoka v ustreznem grafu. Jasno napiši, kako iz dobljenega maksimalnega toka ugotovimo, ali želena razdelitev hrane in pijače obstaja.
- (b) S pomočjo formulacije iz prejšnje točke reši nalogo za naslednje konkretne podatke:

Vrsta soka	Količina	Vrsta sendviča	Količina
Ananas	1	Pohanček	1
Črni ribez	1	Šoferski	2
Rdeča pomaranča	2	Kraški	2
Jagoda	2	Vegi	1
Borovnica	2		

Ime	Seznam sokov	Seznam sendvičev
Damir	ananas, jagoda	pohanček, šoferski
Jasna	ananas, črni ribez, jagoda	kraški, vegi
Bernard	črni ribez, jagoda	vegi, pohanček, šoferski
Matjaž	ananas, rdeča pomaranča	kraški, vegi
Sanela	borovnica, črni ribez, jagoda	vegi
Patrik	borovnica, jagoda	kraški, vegi

### Naloga 8.12.

Vir: Izpit OR 26.6.2012

V nekem velikem mehiškem mestu je center mesta sestavljen iz ulic, ki so pravokotne ena na drugo in iz ptičje perspektive izgledajo kot pravokotna karirasta mreža. Vse banke se nahajajo v centru, in sicer tik ob križiščih (v vsakem križišču kvečjemu ena banka). Križišča so označena s svojimi koordinatami v mreži. V tem mestu so ropi bank zelo pogosti. Tako pogosti, da se včasih roparji na begu zaletijo med seboj (saj bežijo pred policijskimi vozili pri veliki hitrosti). Če ropar uspe pripeljati do ulice, ki vodi ven iz centra, je "rešen", saj center obdajajo slumi, kjer policija nima moči, saj jih nadzirajo mafijski botri. Roparji so se sestali na Veliki mafijski konferenci in sklenili temu kaosu narediti konec. Odločili so se, da bodo naročili informacijski sistem, kamor bodo vsak večer vnesli lokacije bank, ki jih nameravajo oropati, program pa jim bo sestavil načrt ropov za naslednji dan. Rop je mogoče izvesti, če:

- so vse lokacije bank različne (dve ekipi ne smeta na isti dan oropati iste banke), in
- za vsako lokacijo obstaja pot, ki pelje ven iz centra, tako da nobeni dve poti ne gresta vzdolž iste ulice ali čez isto križišče (s tem poskušamo preprečiti, da bi se roparji medsebojno zaleteli; smer vožnje pri tem ni pomembna).

Opiši algoritem, ki bo kot vhod dobil celi števili  $n$  in  $m$ , ki predstavljata število vodoravnih in navpičnih ulic, ter seznam lokacij bank  $L$ . Algoritem naj vrne seznam poti, po katerih bodo roparji bežali (za vsako lokacijo po eno pot). Če vseh ropov ni mogoče izvesti, naj algoritem javi napako.

Za  $n = 7$ ,  $m = 6$  in

$$L = \{(4, 5), (4, 4), (4, 3), (6, 4), (7, 4), (2, 5), (6, 5), (2, 4), (2, 3), (6, 3)\}$$

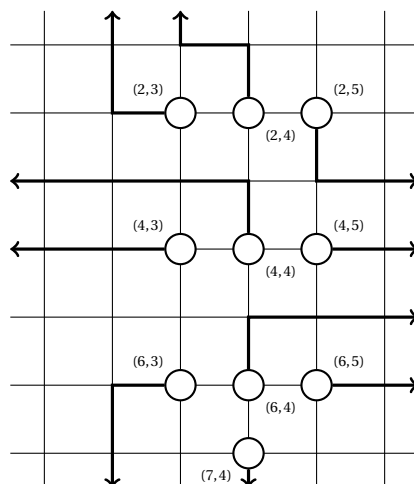
lahko dobimo na primer poti, podane na sliki 44.

### Naloga 8.13.

Vir: Izpit OR 14.6.2013

Pohorski škratje že 644. leto zapored organizirajo tradicionalni tek po rovih od  $a$  do  $z$ , ki letos poteka na trasi, prikazani na sliki 45.

Udeleženci začnejo v točki  $a$  in končajo v točki  $z$ , zmeraj pa tečejo le v smeri povezav. Vsakemu škrtu posebej pot določijo organizatorji, zaradi dobre pripravljenosti pa lahko pričakujemo, da bodo vsi prišli do cilja. Ker so škratje zelo težki,



Slika 44: Shema ulic in bank ter primer izhoda za nalogo 8.12.

je med tekom po rovu nevarnost, da se le-ta poruši. Na povezavah je napisano število udeležencev, ki jih vsak rov prenese. Organizatorji takoj za tem, ko rov preči en tekmovalec manj, kot je nosilnost rova, rov zaprejo (npr. rov  $x$  se zapre, ko ga preči  $(x - 1)$ -ti tekmovalec –  $x$ -ti tekmovalec ne stopi v rov).

- Trenutno stanje trase je takšno, da je  $x = 1$  in  $y = 2$ . Največ koliko udeležencev lahko preteče od  $a$  do  $z$ ? Odgovor utemelji. Napiši, kako naj organizatorji pripravijo poti za skrate (tj., koliko škratov so pripravljene spustiti po kateri poti).
- Rova  $x$  in  $y$  lahko škratje dodatno utrdijo, ostalih ni moč preurejati. Da lahko povečajo nosilnost rova za enega udeleženca, morajo delati 10 dni. Najmanj koliko dni morajo delati in na katerem rovu, da bo na tradicionalnem teku od  $a$  do  $z$  lahko sodelovalo kar največ udeležencev? Odgovor utemelji. Napiši, kako naj organizatorji v tem primeru pripravijo poti za skrate (tj., koliko škratov so pripravljene spustiti po kateri poti).

**Naloga 8.14.**

Vir: Izpit OR 25.8.2014

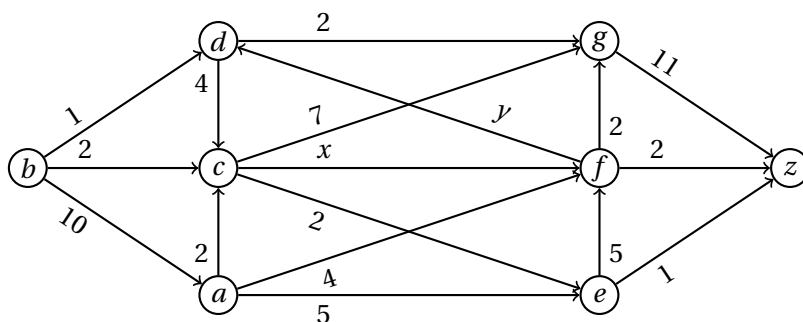
Dana sta računalnika  $s$  in  $t$  ter omrežje strežnikov in usmerjenih povezav. S pomočjo algoritma za maksimalni pretok v grafu opiši in utemelji splošen algoritem, ki določi minimalno število strežnikov, ki jih moramo onemogočiti, da prekinemo komunikacijo od  $s$  do  $t$ , in ga uporabi na grafu s slike 46.

**Naloga 8.15.**

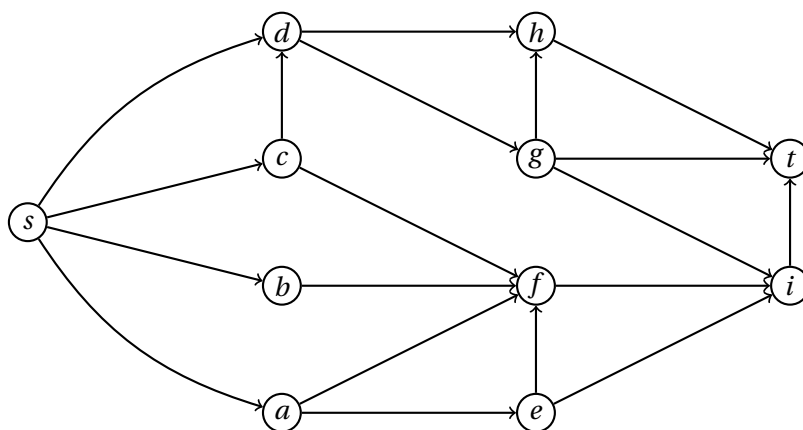
Vir: Vaje OM 23.4.2014

Poišči maksimalni pretok in minimalni prerez na grafu s slike 47.





Slika 45: Graf za nalogo 8.13.



Slika 46: Graf za nalogo 8.14.

**Naloga 8.16.**

Vir: Vaje OM 23.4.2014

Poišči maksimalni pretok in minimalni prerez na grafu s slike 48.

**Naloga 8.17.**

Vir: Vaje OM 23.4.2014

Poišči maksimalni pretok in minimalni prerez na grafu s slike 49 v odvisnosti od parametra  $\alpha$ .

**Naloga 8.18.**

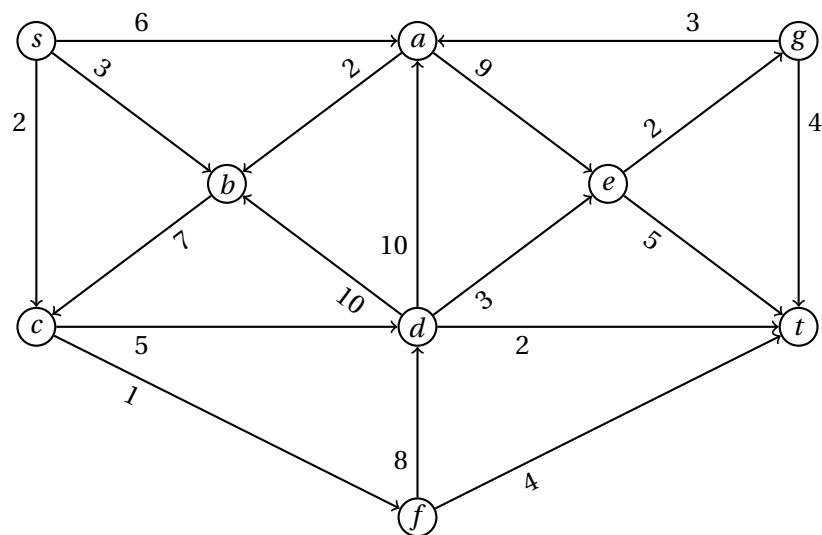
Vir: Vaje OM 23.4.2014

V grafu na sliki 50 lahko kapaciteto ene povezave povečamo za 1. Katera naj bo ta povezava, da bomo povečali maksimalni pretok?

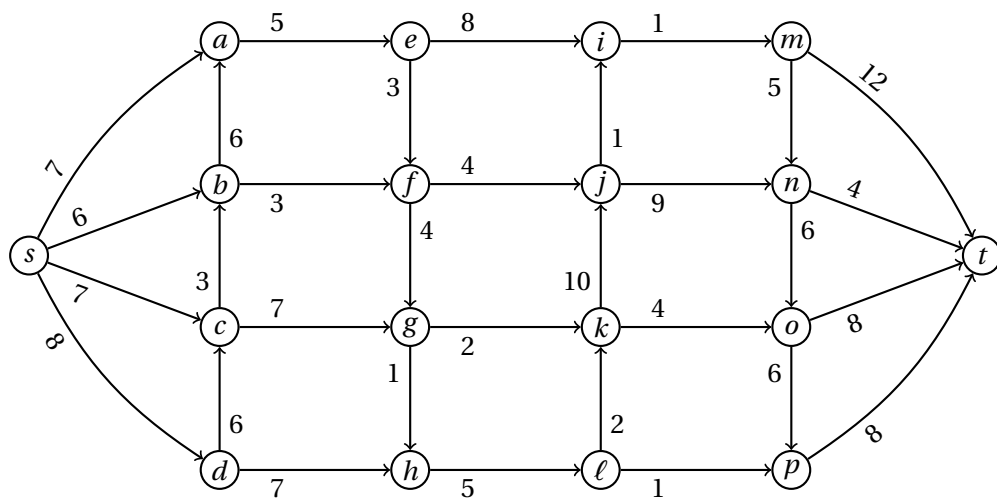
**Naloga 8.19.**

Vir: Vaje OM 23.4.2014

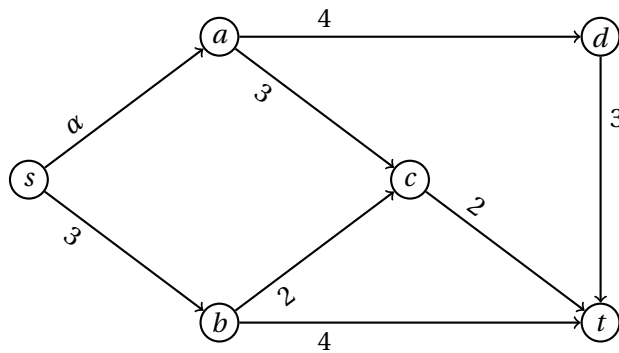
Poišči maksimalni pretok na grafu s slike 51, če ta obstaja!



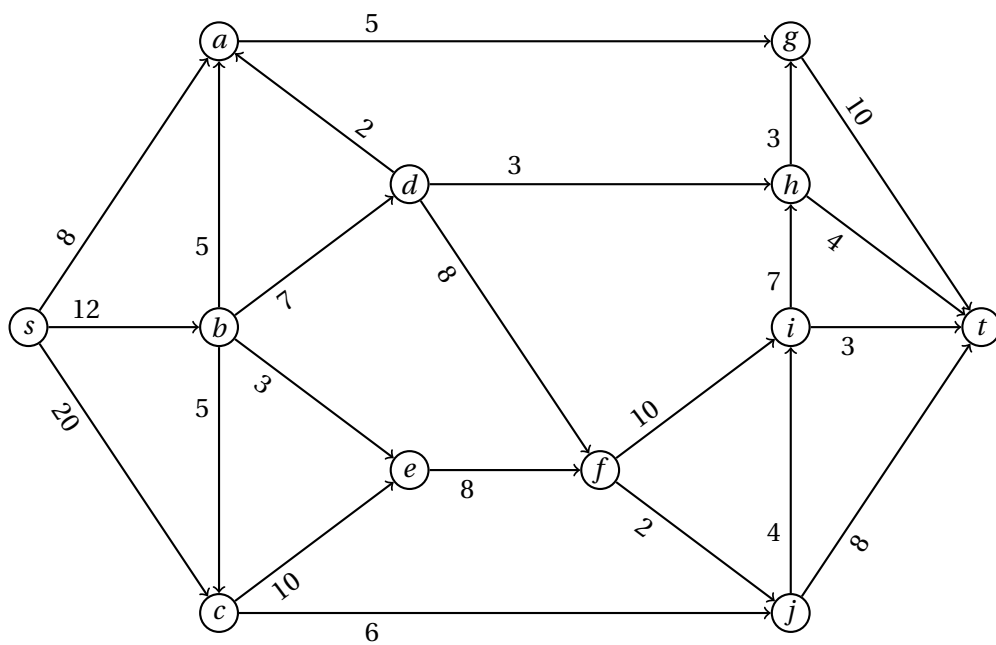
Slika 47: Graf za nalogo 8.15.



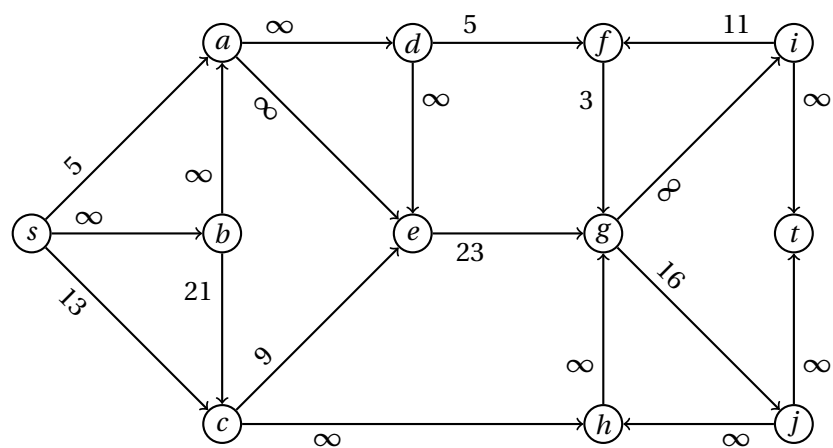
Slika 48: Graf za nalogo 8.16.



Slika 49: Graf za nalogo 8.17.



Slika 50: Graf za nalogo 8.18.



Slika 51: Graf za nalogo 8.19.

## 1.9 Razdelitve

### Naloga 9.1.

Vir: Kolokvij OR 26.1.2010

S pomočjo Austinovega postopka opiši postopek, ki prvemu igralcu zagotovi točno četrtno celote in drugemu igralcu točno tri četrtine celote.

### Naloga 9.2.

Vir: Izpit OR 5.2.2010

Naša naloga je, da med pet ljudi z uporabo denarnih nadomestil razdelimo pet predmetov, tako da bo vsak dobil pošten delež in da bo delitev brez zavisti. V spodnji matriki je podano vrednotenje vsakega izmed 5 predmetov s strani petih ljudi.

11	9	19	14	11
4	10	15	20	12
6	5	5	17	9
20	23	6	3	18
10	18	23	9	10

Seveda želimo pri tem kot svetovalec zaslužiti čimveč. Ker ni zunanjega financiranja, lahko naš zaslužek pridobimo le iz nadomestil. Uporabi kak znan algoritem za delitev pod temi pogoji in razloži, kako si pridobiš kar največji dobiček.

### Naloga 9.3.

Vir: Kolokvij OR 9.5.2013

V turistični agenciji Zobček že vrsto let organizirajo izlete po Sloveniji. Zaradi velikega povpraševanja so se odločili, da bodo ponudbo razširili tudi na Madžarsko. Izbrali so destinacije in vodiče, sedaj pa želijo slednje razporediti po destinacijah tako, da bi vsako destinacijo pokrivala natanko dva vodiča in bi vsak vodič pokrival natanko eno destinacijo. Vsak vodič je podal seznam destinacij, urejenih po priljubljenosti, začnši s tisto, po kateri bi najraje vodil turiste. Agencija je za vsako destinacijo naredila urejen seznam vodičev, tako da je na prvem mestu tisti vodič, ki bi predvidoma najbolje vodil turiste po destinaciji, na zadnjem pa tisti, ki bi to predvidoma počel najslabše.

Če bi bilo destinacij in vodičev malo, bi agencija sama nekako razdelila vodiče med destinacije, a temu ni tako. Še več, seznam destinacij in vodičev nameravajo z leti spreminjati, zato nujno potrebujejo postopek, ki bi jim pomagal razdeliti vodiče po destinacijah.

- (a) Danih je  $n$  destinacij in  $2n$  vodičev, pri čemer za vsako destinacijo poznamo preference agencije glede vodičev in za vsakega vodiča poznamo njegove preference glede destinacij. Opiši algoritem časovne zahtevnosti  $O(n^2)$ , ki izračuna *stabilno razdelitev* vodičev med destinacije oz. nam pove, da stabilna razdelitev ne obstaja.

Razdelitev  $2n$  vodičev med  $n$  destinacij je *nestabilna*, če obstajata vodič  $v$  in destinacija  $d$ , tako da bi  $v$  raje vodil po destinaciji  $d$  kakor po trenutni destinaciji in bi agencija raje imela vodiča  $v$  na destinaciji  $d$  namesto enega

izmed trenutnih vodičev na tej destinaciji. Razdelitev  $2n$  vodičev med  $n$  destinacij je *stabilna*, če sta vsaki destinaciji dodeljena natanko dva vodiča in če ni nestabilna.

(b) Reši nalogo za naslednje podatke ( $n = 3$ ):

Destinacija	preference agencije glede vodičev
Budimpešta	Anisa, Bela, Črt, Ema, Dana, Corina
Győr	Črt, Bela, Corina, Dana, Ema, Anisa
Szeged	Ema, Anisa, Bela, Dana, Črt, Corina

Vodič	preference vodičev
Anisa	Szeged, Budimpešta, Győr
Bela	Budimpešta, Szeged, Győr
Corina	Győr, Szeged, Budimpešta
Črt	Budimpešta, Győr, Szeged
Dana	Budimpešta, Szeged, Győr
Ema	Győr, Budimpešta, Szeged

#### Naloga 9.4.

Vir: Izpit OR 24.6.2013

V turistični agenciji Zobček že vrsto let organizirajo izlete po Sloveniji in na Madžarskem. Zaradi velikega povpraševanja so se odločili, da bodo ponudbo razširili tudi na države območja bivše Jugoslavije. Ker bodo tako nudili izlete po veliko novih državah, želijo postaviti štiri najboljše dosedanje vodiče za nove šefe, tako da bo vsak odgovoren za nekaj držav. Naša naloga je šefe stabilno razporediti po območjih. Tabeli preferenc sta naslednji:

Šefi	preference šefov
Črt	Srbija in Črna gora, Slovenija in Madžarska, Hrvaška in BiH, Makedonija in Kosovo
Žan	Slovenija in Madžarska, Srbija in Črna gora, Hrvaška in BiH, Makedonija in Kosovo
Šime	Slovenija in Madžarska, Makedonija in Kosovo, Hrvaška in BiH, Srbija in Črna gora
Mujo	Hrvaška in BiH, Slovenija in Madžarska, Makedonija in Kosovo, Srbija in Črna gora

Območje	preference agencije glede šefov
Slovenija in Madžarska	Črt, Mujo, Žan, Šime
Hrvaška in BiH	Šime, Žan, Črt
Srbija in Črna gora	Šime, Žan, Mujo, Črt
Makedonija in Kosovo	Šime, Mujo, Črt, Žan

Opazimo, da agencija za šefa območja Hrvaške in BiH ne želi postaviti Muja.

Poišči stabilno prirejanje, v katerem Mujo ni šef območja Hrvaške in BiH, ali utemelji, da le-ta ne obstaja<sup>6</sup>.

<sup>6</sup>Slovenija in Madžarska je eno nedeljivo območje. Enako velja za ostale pare držav.

## 2 Rešitve

### 2.1 Zahtevnost algoritmov

#### Naloga 1.1.

- (a) Program prepíše vnose v matriki  $A$  nad diagonalo na ustrezno mesto pod diagonalo tako, da je po izvedbi programa matrika  $A$  simetrična.
- (b) Kot korak bomo upoštevali posamezno izvedbo notranje zanke **for**, kjer kopiramo vrednost v matriki na drugo mesto – ob predpostavki, da je velikost vnosov omejena (npr. 32-bitna cela števila), bo trajanje take operacije omejeno s konstanto. Preštejmo število takih korakov:

$$\sum_{i=1}^n \sum_{j=i+1}^n 1 = \sum_{i=1}^n (n-i) = \sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$$

Lahko bi seveda upoštevali še korake, ki so potrebni za vzdrževanje števec zank (inicializacija števca, povečevanje števca, preverjanje konca zanke), a bi spet dobili kvadratni polinom v  $n$ . Tako lahko rečemo, da je število korakov omejeno z  $O(n^2)$ .

#### Naloga 1.2.

- (a) V vsakem obhodu zanke **while** se vrednost  $\ell[i]$  spremeni iz 0 v 1 ali obratno. Če se vrednost spremeni na 1, se  $i$  nastavi na 1, sicer se pa poveča za 1.

Izpišimo si vrednosti v seznamu  $\ell$  in spremenljivke  $i$  ob koncu vsakega obhoda zanke **while** tekom izvajanja algoritma, recimo za  $n = 4$ :

obhod	$\ell[4 \dots 1]$	$i$	obhod	$\ell[4 \dots 1]$	$i$
1	0001	1	16	1001	1
2	0000	2	17	1000	2
3	0010	1	18	1010	1
4	0011	1	19	1011	1
5	0010	2	20	1010	2
6	0000	3	21	1000	3
7	0100	1	22	1100	1
8	0101	1	23	1101	1
9	0100	2	24	1100	2
10	0110	1	25	1110	1
11	0111	1	26	1111	1
12	0110	2	27	1110	2
13	0100	3	28	1100	3
14	0000	4	29	1000	4
15	1000	1	30	0000	5

Če pogledamo samo tiste obhode, na koncu katerih velja  $i = 1$ , opazimo, da vrednosti v seznamu  $\ell$  predstavljajo dvojiške zapise števil od 1 do  $2^n - 1$  v ostalih pa se najmanj pomembna 1 zamenja z 0. Algoritem torej simulira dvojiški števec z  $n$  mesti.

- (b) Algoritem obišče vseh  $2^n - 1$  števil, poleg tega pa mora vsakič poskrbeti za zamenjavo vseh enic za najmanj pomembno ničlo. Ob upoštevanju, da obstaja  $2^{n-i}$  števil z najmanj pomembno ničlo na  $i$ -tem mestu, za vrednost  $2^n - 1$  pa je potrebno nadomestiti vseh  $n$  mest, je skupno število korakov enako

$$\begin{aligned} n + \sum_{i=1}^n (i \cdot 2^{n-i}) &= \sum_{j=1}^n \left( 1 + \sum_{i=j}^n 2^{n-i} \right) = \\ &= \sum_{j=1}^n \left( 1 + \sum_{i=0}^{n-j} 2^i \right) = \sum_{j=1}^n 2^{n-j+1} = 2^{n+1} - 2. \end{aligned}$$

Časovna zahtevnost algoritma je torej  $O(2^n)$ .

### Naloga 1.3.

- (a) Izpišimo vrednosti spremenljivk ob koncu vsakega obhoda zanke **for** oziroma **while**, ko se prejšnja konča.

obhod <b>while</b>	$i$	$y$	$z$	$\ell[1 \dots 5]$
1	2	1	5	[7, 11, 16, 7, 5]
1	3	1	5	[7, 11, 16, 7, 5]
1	4	4	5	[7, 11, 7, 16, 5]
1	5	5	5	[7, 11, 7, 5, 16]
1		5	4	[7, 11, 7, 5, 16]
2	2	1	4	[7, 11, 7, 5, 16]
2	3	3	4	[7, 7, 11, 5, 16]
2	4	4	4	[7, 7, 5, 11, 16]
2		4	3	[7, 7, 5, 11, 16]
3	2	1	3	[7, 7, 5, 11, 16]
3	3	3	3	[7, 5, 7, 11, 16]
3		3	2	[7, 5, 7, 11, 16]
4	2	2	2	[5, 7, 7, 11, 16]
4		2	1	[5, 7, 7, 11, 16]

- (b) Naj bodo  $z_1, z_2, \dots, z_k$  vrednosti, ki jih zavzame spremenljivka  $z$  ob vsakem vstopu v zanko **while**. Očitno velja  $z_i - 1 \geq z_{i+1}$  za vsak  $i$ , tako da velja  $k \leq n - 1$ . Največje število korakov je torej

$$\sum_{z=2}^n (z - 1) = \frac{n(n-1)}{2}.$$



Tako število korakov dosežemo, če je seznam  $\ell$  na začetku urejen padajoče – tako vsakič pride do zamenjave v zadnjem koraku zanke **for**, zato se  $z$  vsakič zmanjša za 1. Časovna zahtevnost algoritma je torej  $O(n^2)$ .

#### Naloga 1.4.

```
(a) function MERGESORT( $\ell[1 \dots n]$ )
    if  $n \leq 1$  then
        return  $\ell$ 
    end if
     $m \leftarrow \lceil \frac{n}{2} \rceil$ 
     $\ell_1 \leftarrow \text{MERGESORT}(\ell[1 \dots m])$ 
     $\ell_2 \leftarrow \text{MERGESORT}(\ell[m + 1 \dots n])$ 
     $i, j \leftarrow 1, 1$ 
     $\ell' \leftarrow []$ 
    while  $i \leq m \wedge j \leq n - m$  do
        if  $\ell_1[i] \leq \ell_2[j]$  then
             $\ell'.\text{append}(\ell_1[i])$ 
             $i \leftarrow i + 1$ 
        else
             $\ell'.\text{append}(\ell_2[j])$ 
             $j \leftarrow j + 1$ 
        end if
    end while
    pripni  $\ell_1[i \dots m]$  na konec  $\ell'$ 
    pripni  $\ell_2[j \dots n - m]$  na konec  $\ell'$ 
    return  $\ell'$ 
end function
```

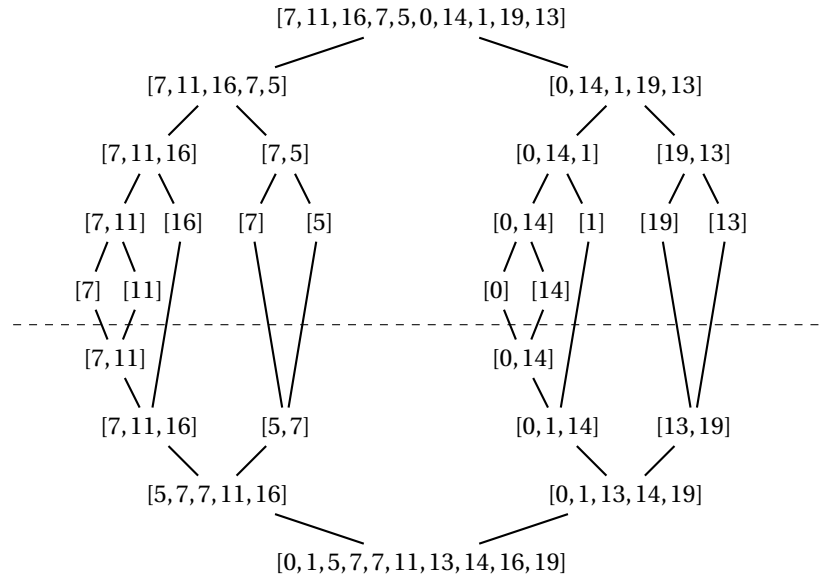
- (b) Izvajanje algoritma je prikazano na sliki 52. Nad črtkano črto je prikazano rekurzivno razbijanje seznamov, pod njo pa zlivanje dobljenih urejenih podseznamov.
- (c) Funkcija obsega dva rekurzivna klica na seznamih polovične dolžine ter združevanje obeh dobljenih seznamov v enega. Naj bo  $T(n)$  čas izvajanja algoritma pri vhodnem seznamu dolžine  $n$ . Ker združevanje poteka v linearnem času, velja rekurzivna zveza

$$T(n) = O(n) + 2T\left(\frac{n}{2}\right).$$

Po krovnem izreku lahko izpeljemo, da je časovna zahtevnost algoritma  $O(n \log n)$ .

#### Naloga 1.5.

Algoritem teče v času  $O(\sqrt{n})$ . Ker je vhod algoritma število  $n$ , ki je zapisano kot zaporedje  $\ell = O(\log n)$  bitov, vidimo, da algoritem teče v času  $O(2^{\ell/2})$  in torej ni polinomski v dolžini vhoda.



Slika 52: Diagram izvajanja algoritma za nalogo 1.4(b).

### Naloga 1.6.

Po krovnem izreku ima časovno zahtevnost  $O(\sqrt{n})$  algoritem, katerega čas izvajanja  $T(n)$  je opisan z rekurzivno zvezo

$$T(n) = O(1) + 2T\left(\frac{n}{4}\right).$$

Zapišimo tak algoritem:

```

function KORENSKI( $\ell[1 \dots n]$ )
  if  $n \geq 4$  then
     $m \leftarrow \lfloor \frac{n}{4} \rfloor$ 
    KORENSKI( $\ell[1 \dots m]$ )
    KORENSKI( $\ell[n - m + 1 \dots n]$ )
  end if
end function

```

### Naloga 1.7.

Algoritem poišče  $k$ -ti najmanjši element v množici  $S$ . Množico razdeli na dva dela glede na naključno izbrani element  $x$ . Če ima množica  $S^-$  elementov, manjših od  $x$ , natanko  $k - 1$  elementov, potem je  $x$  iskani element. Če ima  $S^-$  manj kot  $k - 1$  elementov, algoritem rekurzivno poišče  $(|S^-| - k - 1)$ -ti element v množici  $S^+$  elementov, večjih od  $x$ , sicer pa rekurzivno poišče  $k$ -ti element v množici  $|S^-|$ .

V najslabšem primeru je velikost množice, ki jo algoritem rekurzivno preišče, enaka  $n - 1$ . Ker algoritem porabi  $O(n)$  korakov za razporejanje elementov po

množicah, v najslabšem primeru za čas izvajanja  $T(n)$  velja

$$T(n) = O(n) + T(n-1) = O\left(\sum_{i=0}^n (n-i)\right) = O(n^2).$$

Poglejmo si sedaj povprečni čas izvajanja  $\hat{T}(n)$ . Denimo, da je izbrani element  $x$   $j$ -ti po vrsti. Če velja  $j = k$ , potem algoritem konča. Če velja  $j < k$ , potem algoritem rekurzivno pregleda množico  $S^+$  z  $n - j$  elementi, če pa velja  $j > k$ , pa pregleda množico  $S^-$  z  $j - 1$  elementi. Zapišimo rekurzivno zvezo za  $\hat{T}(n)$ :

$$\hat{T}(n) = O(n) + \frac{1}{n} \left( \sum_{j=1}^{k-1} \hat{T}(n-j) + \sum_{j=k+1}^n \hat{T}(j-1) \right)$$

Denimo, da je čas izvajanja nerekurzivnega dela funkcije omejen s  $c \cdot n$  koraki za neko konstanto  $c > 0$ . Z indukcijo bomo pokazali, da velja  $\hat{T}(n) \leq C \cdot n$  za neko konstanto  $C > 0$ .

Po zgornji predpostavki velja  $\hat{T}(1) \leq cn$ . Denimo, da za vse  $m < n$  velja  $\hat{T}(m) \leq Cm$ . Potem velja

$$\begin{aligned} \hat{T}(n) &\leq cn + \frac{1}{n} \left( \sum_{j=1}^{k-1} C(n-j) + \sum_{j=k+1}^n C(j-1) \right) \\ &= cn + \frac{C}{2n} ((2n-k)(k-1) + (n+k-1)(n-k)) \\ &= cn + \frac{C}{2n} (n^2 + 2nk - 3n - 2k^2 + 2k) \\ &= \left( c + \frac{C}{2} \right) n + \frac{C(2k-3)}{2} - \frac{Ck(k-1)}{n} \end{aligned}$$

Označimo sedaj  $\alpha = k/n$  – velja torej  $0 < \alpha \leq 1$ .

$$\begin{aligned} \hat{T}(n) &\leq \left( c + C \left( \frac{1}{2} + \alpha - \alpha^2 \right) \right) n + C \left( \alpha - \frac{3}{2} \right) \\ &\leq \left( c + C \left( \frac{1}{2} + \alpha(1-\alpha) \right) \right) n \\ &\leq \left( c + \frac{3C}{4} \right) n \end{aligned}$$

Če vzamemo  $C \geq 4c$ , potem po indukciji velja  $\hat{T}(n) \leq Cn$  za vse  $n$ . Algoritem torej v povprečnem primeru opravi  $O(n)$  korakov.

### Naloga 1.8.

(a) Dokažimo, da velja  $(n+a)^b = \Theta(n^b)$ . Dokazati moramo torej, da obstajata konstanti  $c, C \in \mathbb{R}_+$  ter konstanta  $n_0 \in \mathbb{N}$ , tako da velja

$$c \cdot n^b \leq (n+a)^b \leq C \cdot n^b$$

za vsak  $n > n_0$ . Ker je  $b > 0$ , funkcija  $x \mapsto x^{\frac{1}{b}}$  monotonno narašča pri  $x > 0$ , torej jo lahko brez škode za splošnost uporabimo na zgornji neenakosti. Tako dobimo

$$c^{\frac{1}{b}} \cdot n \leq n + a \leq C^{\frac{1}{b}} \cdot n.$$

Naj bosta  $c = \frac{1}{2^b}$  in  $C = 2^b$ . Dobimo pogoja

$$\frac{n}{2} \leq n + a \leq 2n.$$

Ker moramo paziti, da so vsi členi znotraj neenakosti pozitivni, dodamo še pogoj  $n + a > 0$ . Če izberemo  $n_0 = \max(-2a, a)$ , bodo ti pogoji veljali pri vseh  $n > n_0$ .

- (b) Po premisleku hitro ugotovimo, da za poljubno funkcijo  $f$  to ne velja, saj lahko ta konkretno spremeni hitrost naraščanja funkcije  $g$ .

Za protiprimer vzemimo  $g(n) = n$  in  $f(n) = \log(n)$  ter pogledajmo  $\lim_{n \rightarrow \infty} (c \cdot \log(n)/n)$  pri poljubni izbiri  $c > 0$ . Z razširitvijo funkcij na  $\mathbb{R}$  in L'Hôpitalovim pravilom ugotovimo, da bo limita enaka 0 ne glede na izbrano konstanto  $c$ . Tako  $n$  ne moremo asimptotično omejiti z  $O(\log(n))$ .

- (c) Rešitev naloge je klasična uporaba krovnega izreka, a jo enostavno rešimo tudi brez tega.

Ob upoštevanju tega, da velja  $T(n) < 2T(\lfloor (n+1)/2 \rfloor)$ , razvijajmo rekurzivno zvezo do  $k$ -tega koraka.

$$T(n) < 2T\left(\left\lfloor \frac{n+1}{2} \right\rfloor\right) < 4T\left(\left\lfloor \frac{n+3}{4} \right\rfloor\right) < \dots < 2^k T\left(\left\lfloor \frac{n+2^k-1}{2^k} \right\rfloor\right)$$

Ker velja  $T(1) = O(1)$ , naj bo  $k$  tak, da bo veljalo  $(n-1)/2^k < 1$ , torej  $k = \log_2(n)$ . Tako dobimo

$$T(n) = 2^{\log_2(n)} \cdot O(1) = n \cdot O(1) = O(n).$$

### Naloga 1.9.

- (a) Algoritem deluje pravilno, saj za vsako urejeno četverico vozlišč preveri, ali so ta zaporedoma povezana ter ali je začetno vozlišče enako končnemu, in se ustavi, ko naleti na četverico, ki ustreza tem pogojem – torej na trikotnik. Časovna zahtevnost je  $O(n^4)$ .

- (b) Zapišimo učinkovitejši algoritem.

```

for  $i = 1, \dots, n$  do
  for  $j = 1, \dots, n$  do
    if  $a_{ij} = 1$  then
      for  $k = 1, \dots, n$  do
        if  $a_{jk} = a_{ki} = 1$  then

```

```

        return TRUE
    end if
end for
end if
end for
end for
return FALSE

```

Zgornji algoritem reši problem v času  $O(n^3)$ .

V najboljšem primeru, torej ko obstaja usmerjen trikotnik (1, 2, 3) (ignorirali bomo primere, ko ima graf največ dve vozlišči), algoritem enkrat vstopi v zunanjo zanko in dvakrat v srednjo – pri drugem vstopu v srednjo zanko nato še trikrat vstopi v notranjo zanko. Ocenimo torej, da v tem primeru algoritem opravi 6 korakov.

V najslabšem primeru usmerjenega trikotnika ni in se tako vse zanke iztečejo. Algoritem bo  $n$ -krat vstopil v zunanjo zanko,  $n^2$ -krat v srednjo zanko, za vsako povezavo pa bo še  $n$ -krat vstopil v notranjo zanko. V tem primeru torej algoritem opravi  $n + n^2 + mn$  korakov, kjer je  $m$  število povezav v grafu.

### Naloga 1.10.

(a) Zapišimo popravljen algoritem.

```

n ← A.length()
for i = 2, ..., n do
    j ← i
    while j > 1 ∧ A[j - 1] > A[i] do
        j ← j - 1
    end while
    A.reverse(j, i)
    if j + 1 < i then
        A.reverse(j + 1, i)
    end if
end for

```

popravek:  $j - 1$  namesto  $i - 1$

ponovno obrnemo že urejen del

(b) Izpišimo stanje “tabele” po vsakem klicu ukaza  $A.reverse$ .

$i$	$j$	$A.reverse(j, i)$	$A.reverse(j + 1, i)$
2	1	[3, 5, 12, 9, 1, 15]	
3	3	[3, 5, 12, 9, 1, 15]	
4	3	[3, 5, 9, 12, 1, 15]	
5	1	[1, 12, 9, 5, 3, 15]	[1, 3, 5, 9, 12, 15]
6	6	[1, 3, 5, 9, 12, 15]	

**Naloga 1.11.**

- (a) Algoritem vrne TRUE natanko tedaj, ko v grafu  $G$  obstaja Hamiltonova pot med vozliščema  $u$  in  $v$ . Trditev bomo dokazali z indukcijo.

Za graf z dvema vozliščema algoritem vrne TRUE natanko tedaj, ko sta vozlišči povezani, torej, ko obstaja Hamiltonova pot. Denimo, da je zgornja trditev pravilna za grafe z  $n \geq 2$  vozlišči. Če algoritem na vhod dobi graf z  $n + 1$  vozlišči, potem vrne TRUE, če obstaja vozlišče  $w$ , ki je sosednje vozlišču  $u$  in različno od vozlišča  $v$ , tako da v grafu  $G - u$  (z  $n$  vozlišči) obstaja Hamiltonova pot od  $w$  do  $v$ . Če je to res, potem v grafu  $G$  obstaja Hamiltonova pot od  $u$  do  $v$ . Po indukciji trditev velja za vse grafe.

- (b) Spet bomo uporabili indukcijo. Pri  $n = 2$  prvi pogoj ni resničen, v zanko pa ne vstopimo, tako da se zadnja vrstica izvede enkrat. Denimo, da se pri grafih z  $n - 1$  vozlišči ( $n \geq 3$ ) zadnja vrstica izvede  $\Theta((n - 3)!)$ -krat, in obravnavajmo primer, ko ima graf  $G$  natanko  $n$  vozlišč. Ker ima vozlišče  $u$  natanko  $n - 2$  sosedov in ker vozlišče  $v$  ni krajišče nobene Hamiltonove poti, se v zanki izvede  $n - 2$  rekurzivnih klicev na grafu  $G - u$  z  $n - 1$  vozlišči in isto lastnostjo kot  $G$ . Po izteku zanke se nato izvede še zadnja vrstica. Skupno število izvedb zadnje vrstice je tako

$$1 + (n - 2)\Theta((n - 3)!) = \Theta((n - 2)!).$$

**Naloga 1.12.**

- (a) Algoritem kliče ukaz  $A.\text{reverseStart}$   $O(n^2)$ -krat. Ker se ta izvede v konstantnem času, je torej časovna zahtevnost algoritma  $O(n^2)$ .
- (b) V tabeli 13 je izpisano stanje "tabele" po vsakem klicu ukaza  $A.\text{reverseStart}$ . Vidimo, da po koncu algoritma "tabela" ni urejena, torej algoritem ne deluje pravilno.
- (c) Da bomo dobili algoritem, ki bo pravilno urejal, bomo poskrbeli, da po obhodu notranje zanke pri indeksih  $i$  in  $j$  velja  $A[k] \leq A[i]$  za vse  $k \leq j$ . Tako bomo dosegli, da po obhodu zunanje zanke vrednost  $A[i]$  ni manjša od svojih predhodnikov, kar bo privedlo do urejenosti ob izteku zanke. V notranji zanki bo torej treba poskrbeti, da vrednost  $A[j]$  pride na  $i$ -to mesto, elementi med  $j$ -tim in  $i$ -tim pa ostanejo v tem intervalu (ne nujno na istem mestu).

```

n ← A.length()
for i = n, ..., 2 do
  for j = 1, ..., i - 1 do
    if A[j] > A[i] then
      A.reverseStart(i - 1)
      A.reverseStart(i - j)
      A.reverseStart(i)

```

$i$	$j$	$A[i]$	$A[j]$	$A.reverseStart(j)$	$A.reverseStart(i)$
5	1	15	5		
	2		9		
	3		12		
	4		7		
4	1	7	5		
	2		9	[9, 5, 12, 7, 15]	[7, 12, 5, 9, 15]
	3	9	5		
3	1	5	7	[7, 12, 5, 9, 15]	[5, 12, 7, 9, 15]
	2	7	12	[12, 7, 5, 9, 15]	[5, 7, 12, 9, 15]
2	1	7	5		

Tabela 13: Potek izvajanja algoritma za nalogo 1.12(b).

**end if**  
**end for**  
**end for**

Algoritem še vedno teče v času  $O(n^2)$ .

Da se prepričamo, da algoritem deluje pravilno, dokažimo, da velja zgoraj omenjena invarianta za notranjo zanko. Predpostavimo torej, da velja  $A[k] \leq A[i]$  za vse  $k < j$ . Če velja tudi  $A[j] \leq A[i]$ , potem se ne zgodi nič in invarianta drži. V nasprotnem primeru označimo z  $a_h$  ( $1 \leq h \leq n$ ) vrednost  $A[i]$  ob vstopu v obhod zanke, in z  $b_h$  ( $1 \leq h \leq n$ ) vrednost  $A[i]$  ob izstopu iz obhoda zanke. Po predpostavki velja  $a_k \leq a_i$  ( $1 \leq k \leq j$ ) in  $a_i > a_j$ . Poglejmo, kaj se dogaja ob klicih ukazov  $A.reverseStart$ .

$$\begin{aligned}
& a_1, \quad \dots, \quad a_{j-1}, \quad a_j, \quad a_{j+1}, \quad \dots, \quad a_{i-1}, \quad a_i, \quad a_{i+1}, \quad \dots \\
& A.reverseStart(i-1): \\
& \quad a_{i-1}, \quad \dots, \quad a_{j+1}, \quad a_j, \quad a_{j-1}, \quad \dots, \quad a_1, \quad a_i, \quad a_{i+1}, \quad \dots \\
& A.reverseStart(i-j): \\
& \quad a_j, \quad a_{j+1}, \quad \dots, \quad a_{i-1}, \quad a_{j-1}, \quad \dots, \quad a_1, \quad a_i, \quad a_{i+1}, \quad \dots \\
& A.reverseStart(i): \\
& \quad a_i, \quad a_1, \quad \dots, \quad a_{j-1}, \quad a_{i-1}, \quad \dots, \quad a_{j+1}, \quad a_j, \quad a_{i+1}, \quad \dots \\
& = \quad b_1, \quad b_2, \quad \dots, \quad b_j, \quad b_{j+1}, \quad \dots, \quad b_{i-1}, \quad b_i, \quad b_{i+1}, \quad \dots
\end{aligned}$$

Opazimo, da velja  $b_i = a_j > a_i = b_1$  in  $b_i = a_j \geq a_{k-1} = b_k$  ( $2 \leq k \leq j$ ). Zanjča invarianta torej drži.

Preizkusimo algoritem še na primeru iz točke (b). Potek algoritma je zabeležen v tabeli 14, iz katere je razvidno, da algoritem na tem primeru res deluje pravilno.

$i$	$j$	$A[i]$	$A[j]$	$A.reverseStart(i-1)$	$A.reverseStart(i-j)$	$A.reverseStart(i)$
5	1	15	5			
	2		9			
	3		12			
	4		7			
4	1	7	5			
	2		9	[12, 9, 5, 7, 15]	[9, 12, 5, 7, 15]	[7, 5, 12, 9, 15]
	3	9	12	[12, 5, 7, 9, 15]	[12, 5, 7, 9, 15]	[9, 7, 5, 12, 15]
3	1	5	9	[7, 9, 5, 12, 15]	[9, 7, 5, 12, 15]	[5, 7, 9, 12, 15]
	2	9	7			
2	1	7	5			

Tabela 14: Potek izvajanja algoritma za nalogo 1.12(c).

### Naloga 1.13.

- (a) Opazimo, da ima algoritem tri dele, ki se razlikujejo le po tem, katera številka v številih se obravnava. Vsakič se števila iz tabele  $s$  dodajo v ustrezno vrsto, nato pa se vsebina vrst zlepi (po vrsti po indeksih tabele  $T$ ) v novo vsebino tabele  $T$ . Zapisali bomo torej vsebino vrst po vsaki od zunanjih zank, vsebino tabele  $s$  po sledeči zanki pa lahko razberemo.

zadnje: [100, 200, 980, 100], [], [], [], [145], [146, 486], [367, 257], [], [359, 359, 909]

srednje: [100, 200, 100, 909], [], [], [145, 146], [257, 359, 359], [367], [], [980, 486], []

prve: [], [100, 100, 145, 146], [200, 257], [359, 359, 367], [486], [], [], [], [909, 980]

Algoritem torej vrne tabelo

[100, 100, 145, 146, 200, 257, 359, 359, 367, 486, 909, 980].

- (b) Časovna zahtevnost algoritma je  $O(n)$  – notranje zanke namreč naredijo konstantno število korakov.
- (c) Algoritem uredi vhodno tabelo po velikosti in jo vrne. Pri tem upošteva dejstvo, da je mogoče trimestna števila predstaviti s končnim številom simbolov iz končne abecede in tako doseže časovno zahtevnost, ki je boljša od  $\Omega(n \log n)$  (t.i. *bucket sort*).
- (d) Ker algoritem teče v linearnem času, bo čas izvajanja premo sorazmeren z velikostjo vhoda. Pri vhodu velikosti  $n = 50\,000$  torej pričakujemo, da bo algoritem potreboval približno

$$\frac{50\,000}{10\,000} \cdot 0.25 \text{ s} = 1.25 \text{ s}.$$



**Naloga 1.14.**

Zapišimo algoritem.

```

function ZLIVANJE( $S[0 \dots n-1]$  seznam števil)
  if  $S.length() \leq 1$  then
    return  $S$ 
  end if
   $k \leftarrow \lfloor \frac{n}{3} \rfloor$ 
   $\ell \leftarrow \lfloor \frac{2n}{3} \rfloor$ 
   $A \leftarrow \text{ZLIVANJE}(S[0 \dots k-1])$ 
   $B \leftarrow \text{ZLIVANJE}(S[k \dots \ell-1])$ 
   $C \leftarrow \text{ZLIJ}(A, B)$ 
   $D \leftarrow \text{ZLIVANJE}(S[\ell \dots n-1])$ 
   $E \leftarrow \text{ZLIJ}(C, D)$ 
  return  $E$ 
end function

```

Časovno zahtevnost algoritma lahko opišemo z rekurzivno zvezo

$$T(n) = 3T\left(\frac{n}{3}\right) + O(n).$$

Po krovnem izreku je časovna zahtevnost algoritma  $T(n) = O(n \log n)$ .

**Naloga 1.15.**

Napišemo si matriko  $A$ , jo pomnožimo z  $x$ , in v  $i$ -ti vrstici izrazimo  $x_i$  iz enačb  $Ax = b$ . Opazimo, da so vse vrednosti, ki jih rabimo za izračun  $x_i$ , že poračunane. To je natanko reševanje sistema enačb z Gaussovo eliminacijo, ki je bila za nas že vnaprej narejena, ko smo kot vhod dobili zgornje trikotno matriko.

**Vhod:**  $A \in \mathbb{Z}^{n \times n}$  zgornje trikotna matrika z neničelnimi vrednostmi na diagonalni,  $b \in \mathbb{Z}^n$

```

for  $i = n, n-1, \dots, 1$  do
   $x_i \leftarrow (b_i - \sum_{j=i+1}^n A_{ij}x_j) / A_{ii}$ 
end for
return  $(x_i)_{i=1}^n$ 

```

Časovna zahtevnost algoritma je  $O(n^2)$ .

**Naloga 1.16.**

- (a) Primer vhoda, ki ga algoritem ne uredi pravilno, je  $[2, 3, 1]$ . V prvem obhodu algoritem najprej primerja 2 in 3 ter ju ne zamenja, nato pa še zamenja 3 in 1. Ker je v obhodu opravil samo eno zamenjavo, se algoritem ustavi in vrne seznam  $[2, 1, 3]$ .

Primer lahko posplošimo na sezname  $A[1 \dots n]$  poljubne dolžine, za katere velja  $A[i-1] \leq A[i]$  ( $2 \leq i \leq n-1$ ) in  $A[n] < A[n-2]$ . Tako kot v prejšnjem primeru algoritem zamenja le zadnja dva elementa, kar pa ne zadostuje za pravilno ureditev.

- (b) Zapišimo opisani algoritem.

**Vhod:**  $A[1 \dots n]$  seznam števil

```
 $k \leftarrow 2$ 
while  $k \geq 2$  do
   $k \leftarrow 0$ 
  for  $i = 2, \dots, n$  do
    if  $A[i-1] > A[i]$  then
       $A[i-1], A[i] \leftarrow A[i], A[i-1]$ 
       $k \leftarrow k + 1$ 
    end if
  end for
end while
return  $A$ 
```

Prepričamo se lahko, da po  $j$ -tem obhodu zunanje zanke zadnjih  $j$  elementov na mestu. V najslabšem primeru algoritem torej opravi  $O(n^2)$  korakov, kar tudi dosežemo, če na vhod dobi padajoče urejen seznam. Časovna zahtevnost algoritma je torej  $O(n^2)$ .

- (c) Število primerjav je odvisno od števila obhodov zanke **while**. En sam obhod dosežemo pri že urejenih seznamih in pri primerih iz točke (a). V tem primeru algoritem opravi  $n$  primerjav elementov seznama  $A$  in dve primerjavi pri vstopu v zanko **while** (pri zadnji se zanka konča).

## 2.2 Celoštevilsko linearno programiranje

### Naloga 2.1.

Za vsak projekt  $p_i \in \mathcal{P}$  bomo uvedli spremenljivko  $x_i$ , katere vrednost interpretiramo kot

$$x_i = \begin{cases} 1, & \text{če naj občina sponzorira projekt } p_i, \text{ in} \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\begin{aligned} \max \quad & \sum_{i=1}^n c_i x_i \quad \text{p.p.} \\ \forall i \in \{1, \dots, n\}: \quad & 0 \leq x_i \leq 1, \quad x_i \in \mathbb{Z} \\ & \sum_{i=1}^n d_i x_i \leq M \\ & \sum_{i=1}^n a_i x_i \geq N \end{aligned}$$

### Naloga 2.2.

Celoštevilskemu linearnemu programu iz rešitve naloge 2.1 bomo dodali nove omejitve.

(a)  $\forall (p_i, p_j) \in V: x_i \leq x_j$

(b)  $\forall H \in S: \sum_{p_i \in H} x_i \leq 1$

### Naloga 2.3.

Za vsako skladišče, ki ga uporabimo, želimo vedeti, kateri stranki bomo dostavili dobrino. Za  $i$ -to skladišče ( $1 \leq i \leq n$ ) in  $j$ -to stranko ( $1 \leq j \leq m$ ) bomo uvedli spremenljivko  $x_{ij}$ , katere vrednost interpretiramo kot

$$x_{ij} = \begin{cases} 1, & \text{če bomo iz } i\text{-tega skladišča dostavili } j\text{-ti stranki, in} \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\begin{aligned} \min \quad & \sum_{i=1}^n \sum_{j=1}^m (f_i + c_{ij}) x_{ij} \quad \text{p.p.} \\ \forall i \in \{1, \dots, n\} \forall j \in \{1, \dots, m\}: \quad & 0 \leq x_{ij} \leq 1, \quad x_{ij} \in \mathbb{Z} \end{aligned}$$

Vsako skladišče uporabimo največ enkrat:

$$\forall i \in \{1, \dots, n\}: \sum_{j=1}^m x_{ij} \leq 1$$

Zadostimo potrebe vsake stranke:

$$\forall j \in \{1, \dots, m\}: \sum_{i=1}^n x_{ij} \geq d_j$$

**Naloga 2.4.**

Naj bo  $G = (V, E)$  neusmerjen graf. Množica  $D \subseteq V$  *dominira* graf  $G$  če je vsako vozlišče iz  $V$  bodisi v  $D$ , bodisi ima soseda v  $D$ . Pri problemu dominacijske množice iščemo najmanjšo množico  $D_{\min}$ , ki dominira graf  $G$ .

Za vsako vozlišče  $v \in V$  bomo uvedli spremenljivko  $x_v$ , katere vrednost interpretiramo kot

$$x_v = \begin{cases} 1, & \text{vozlišče } v \text{ je v množici } D_{\min}, \text{ in} \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \quad \text{p.p.} \\ \forall v \in V: \quad & 0 \leq x_v \leq 1, \quad x_v \in \mathbb{Z} \\ \forall v \in V: \quad & x_v + \sum_{uv \in E} x_u \geq 1 \end{aligned}$$

**Naloga 2.5.**

Naj bo  $G = (V, E)$  dvodelen graf. Lahko torej zapišemo  $V = A \cup B$  tako, da sta množici  $A$  in  $B$  disjunktni ter za vsako povezavo  $uv \in E$  velja  $u \in A, v \in B$ . Množica  $M \subseteq E$  je *prirejanje*, če nobeni dve povezavi iz  $M$  nimata skupnega krajišča. Iščemo največje prirejanje  $M_{\max}$ .

Za vsako povezavo  $uv \in E$  bomo uvedli spremenljivko  $x_{uv}$ , katere vrednost interpretiramo kot

$$x_{uv} = \begin{cases} 1, & \text{povezava } uv \text{ je v množici } M_{\max}, \text{ in} \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\begin{aligned} \max \quad & \sum_{uv \in E} x_{uv} \quad \text{p.p.} \\ \forall uv \in E: \quad & 0 \leq x_{uv} \leq 1, \quad x_{uv} \in \mathbb{Z} \\ \forall u \in A: \quad & \sum_{uv \in E} x_{uv} \leq 1 \\ \forall v \in B: \quad & \sum_{uv \in E} x_{uv} \leq 1 \end{aligned}$$

**Naloga 2.6.**

Naj bo  $G = (V, E)$  usmerjen graf in  $u, v \in V$  njegovi vozlišči. Poiskati želimo najkrajšo pot  $P_{uv}$  od vozlišča  $u$  do vozlišča  $v$ .

Za vsako povezavo  $uv \in E$  bomo uvedli spremenljivko  $x_{uv}$ , katere vrednost interpretiramo kot

$$x_{uv} = \begin{cases} 1, & \text{povezava } uv \text{ je v poti } P_{uv}, \text{ in} \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\begin{aligned} \min \quad & \sum_{uv \in E} x_{uv} \quad \text{p.p.} \\ \forall uv \in E: \quad & 0 \leq x_{uv} \leq 1, \quad x_{uv} \in \mathbb{Z} \end{aligned}$$

Iz vozlišča  $u$  izstopimo enkrat več kakor vstopimo:

$$\sum_{yu \in E} x_{yu} - \sum_{uz \in E} x_{uz} = -1$$

V vozlišče  $v$  vstopimo enkrat več kakor izstopimo:

$$\sum_{yv \in E} x_{yv} - \sum_{vz \in E} x_{vz} = 1$$

V ostala vozlišča vstopimo natanko tolikokrat kakor vstopimo:

$$\forall w \in V \setminus \{u, v\}: \sum_{yw \in E} x_{yw} - \sum_{wz \in E} x_{wz} = 0$$

Zgornje omejitve sicer dovolijo, da posamezno vozlišče obiščemo večkrat, a se je mogoče teh zank znebiti, pri čemer se vrednost ciljne funkcije zmanjša. Tako bo optimalna rešitev zagotovo pot od vozlišča  $u$  do vozlišča  $v$ .

**Naloga 2.7.**

Oštevilčimo gnezda s števili od 1 do 12: gnezdi taščic naj imata števili 1 in 2, gnezda vrtnih penic naj imajo števila 3 in 4, 5 in 6, gnezda travniških cip naj imajo števila 7 in 8, in 9, gnezdi belih pastiric naj imata števili 10 in 11, sovino gnezdo pa naj ima število 12. Za  $i$ -to gnezdo ( $1 \leq i \leq 12$ ) in  $j \in \{1, 2, 3\}$  bomo uvedli spremenljivko  $x_{ij}$ , katere vrednost interpretiramo kot

$$x_{ij} = \begin{cases} 1, & \text{če naj kukavica v } i\text{-to gnezdo izleže natanko } j \text{ jajc, in} \\ 0 & \text{sicer.} \end{cases}$$

Naj bo  $V = \{\{1, 2\}, \{3, 4, 5, 6\}, \{7, 8, 9\}, \{10, 11\}, \{12\}\}$  particija indeksov gnezd glede na vrsto ptice. Zapišimo celoštevilski linearni program.

$$\begin{aligned} \max \quad & \sum_{i=1}^{12} \sum_{j=1}^3 j p_{ij} x_{ij} \quad \text{p.p.} \\ \forall i \in \{1, \dots, 12\} \forall j \in \{1, 2, 3\}: \quad & 0 \leq x_{ij} \leq 1, \quad x_{ij} \in \mathbb{Z} \end{aligned}$$

V vsakem gnezdu je določeno število jajc:

$$\forall i \in \{1, \dots, 12\} : \sum_{j=1}^3 x_{ij} \leq 1$$

Skupaj izleže 16 jajc:

$$\sum_{i=1}^{12} \sum_{j=1}^3 j x_{ij} = 16$$

K vsaki vrsti ptice izleže vsaj eno jajce:

$$\forall S \in V : \sum_{i \in S} \sum_{j=1}^3 x_{ij} \geq 1$$

Pri taščicah izleže strogo več jajc kot pri belih pastiricah:

$$\sum_{i=1}^2 \sum_{j=1}^3 j x_{ij} - \sum_{i=10}^{11} \sum_{j=1}^3 j x_{ij} \geq 1$$

Pri drugi beli pastirici ne bo odložila jajca, če bo pri prvi taščici odložila dve jajci ali več:

$$\sum_{j=2}^3 x_{1j} + \sum_{j=1}^3 x_{11,j} \leq 1$$

### Naloga 2.8.

Za vsako sorto  $i \in I = \{RM, LR, RR\}$  (rumeni muškata, laški rizling, renski rizling) in kupca  $j \in J = \{K, L, Z, O\}$  (bar Kocka, bar Luka, župnišče Sv. Martin, občina Duplek) bomo uvedli spremenljivko  $x_{ij}$ , katere vrednost interpretiramo kot

$$x_{ij} = \begin{cases} 1, & \text{če naj Janez proda sorto } i \text{ kupcu } j, \text{ in} \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\begin{aligned} \max & 2000x_{RM,K} + 2200x_{RM,L} + 750x_{RM,Z} + 1800x_{RM,O} \\ & + 10000x_{LR,K} + 5500x_{LR,L} + 750x_{LR,Z} + 1800x_{LR,O} \\ & + 5000x_{RR,K} + 5500x_{RR,L} + 750x_{RR,Z} + 1800x_{RR,O} \quad \text{p.p.} \end{aligned}$$

$$\forall i \in I \forall j \in J : 0 \leq x_{ij} \leq 1, \quad x_{ij} \in \mathbb{Z}$$

Vsako sorto proda največ enemu kupcu:

$$\forall i \in I : \sum_{j \in J} x_{ij} \leq 1$$

Vsak kupec dobi največ eno sorto:

$$\forall j \in J : \sum_{i \in I} x_{ij} \leq 1$$

Občina ne bo kupila rumenega muškata ali laškega rizlinga:

$$x_{RM,O} + x_{LR,O} = 0$$

Če Kocka dobi laški rizling, potem Luka dobi rumeni muškati:

$$x_{LR,K} \leq x_{RM,L}$$

Če občina in župnišče ne kupita ničesar, tudi Kocka ne kupi ničesar:

$$\sum_{i \in I} (x_{i,O} + x_{i,Z} - x_{i,K}) \geq 0$$

Ženin pogoj:

$$x_{s_A,A} + x_{s_B,B} + \sum_{j \in J \setminus \{C\}} x_{s_C,j} \leq 2$$

### Naloga 2.9.

Graf  $G = (V, E)$  smatramo kot usmerjen graf (če je dani graf neusmerjen, vsako povezavo nadomestimo s parom nasprotno usmerjenih povezav). Naj bo  $n = |V|$  število vozlišč grafa  $G$ . Rešitev celoštevilkega linearnega programa predstavlja usmerjeno drevo najkrajših poti s korenem v vozlišču  $u$ .

Za vsako povezavo  $vw \in E$  bomo uvedli spremenljivko  $x_{vw}$ , katere vrednost interpretiramo kot

$$x_{vw} = \begin{cases} 1, & \text{povezava } vw \text{ je v drevesu najkrajših poti, in} \\ 0 & \text{sicer.} \end{cases}$$

Poleg tega bomo za vsako vozlišče  $v \in V$  uvedli še spremenljivko  $y_v$ , ki nam pove razdaljo od vozlišča  $u$  do vozlišča  $v$ . Minimizarati želimo vse vrednosti  $y_v$ , zato bomo minimizirali njihovo vsoto.

Zapišimo celoštevilski linearni program.

$$\begin{aligned} & \min \sum_{v \in V} y_v \quad \text{p.p.} \\ & \forall vw \in E: \quad 0 \leq x_{vw} \leq 1, \quad x_{vw} \in \mathbb{Z} \end{aligned}$$

Vozlišče  $u$  nima vstopne povezave v drevesu:

$$\sum_{vu \in E} x_{vu} = 0$$

Ostala vozlišča imajo natanko eno vstopno povezavo v drevesu:

$$\forall vw \in E, v \neq u: \quad \sum_{vw \in E} x_{vw} = 1$$

Vozlišče  $u$  je na razdalji 0 od sebe:

$$y_u = 0$$

Če je povezava  $vw$  v drevesu, je razdalja do  $w$  vsaj za 1 večja od razdalje do  $v$ , sicer pa je razlika največ  $n - 1$ :

$$\forall vw \in E: \quad y_v + nx_{vw} \leq y_w + n - 1$$

V drevesu je natanko  $n - 1$  povezav:

$$\sum_{vw \in E} x_{vw} = n - 1$$

**Naloga 2.10.**

*Pravilno barvanje* (neusmerjenega) grafa  $G = (V, E)$  je taka dodelitev barv vozliščem grafa  $G$ , da imata krajišči vsake povezave različno barvo. *Kromatično število* grafa  $G$  je najmanjše število barv, ki jih potrebujemo za pravilno barvanje grafa  $G$ .

Naj bo  $n = |V|$  število vozlišč grafa  $G$  – to je tudi največje število barv, ki jih lahko uporabimo. Za vsako vozlišče  $v \in V$  in vsako barvo  $i$  ( $1 \leq i \leq n$ ) bomo uvedli spremenljivko  $x_{vi}$ , katere vrednost interpretiramo kot

$$x_{vi} = \begin{cases} 1, & \text{vozlišče } v \text{ pobarvamo z barvo } i, \text{ in} \\ 0 & \text{sicer.} \end{cases}$$

Poleg tega bomo uvedli še spremenljivko  $t$ , ki šteje uporabljene barve.

Zapišimo celoštevilski linearni program.

$$\min t \quad \text{p.p.}$$

$$\forall v \in V \forall i \in \{1, \dots, n\}: \quad 0 \leq x_{vi} \leq 1, \quad x_{vi} \in \mathbb{Z}$$

Vsako vozlišče je natanko ene barve:

$$\forall v \in V: \quad \sum_{i=1}^n x_{vi} = 1$$

Krajišči povezav sta različnih barv:

$$\forall uv \in E \forall i \in \{1, \dots, n\}: \quad x_{ui} + x_{vi} \leq 1$$

Omejimo  $t$  z indeksi uporabljenih barv:

$$\forall uv \in E \forall i \in \{1, \dots, n\}: \quad i x_{vi} \leq t$$

**Naloga 2.11.**

Pot bomo opisali z množico povezav v polnem grafu z  $n$  vozlišči (mesti), ki tvorijo cikel. Za vsak par mest  $(i, j)$  ( $1 \leq i, j \leq n$ ) bomo uvedli spremenljivko  $x_{ij}$ , katere vrednost interpretiramo kot

$$x_{ij} = \begin{cases} 1, & \text{iz } i\text{-tega mesta odpotujemo v } j\text{-to mesto} \\ 0 & \text{sicer.} \end{cases}$$

Poskrbeti bomo morali še, da bo naša rešitev res sestavljena iz enega samega cikla. V ta namen bomo za  $i$ -to mesto ( $1 \leq i \leq n$ ) uvedli spremenljivko  $y_i$ , ki pove, katero po vrsti smo to mesto obiskali. Ker je rešitev cikel, si lahko poljubno izberemo, katero je prvo obiskano mesto (npr. mesto z indeksom 1).

Zapišimo celoštevilski linearni program.

$$\min \sum_{i=1}^n \sum_{j=1}^n x_{ij} c_{ij} \quad \text{p.p.}$$

$$\forall i, j \in \{1, \dots, n\}: \quad 0 \leq x_{ij} \leq 1, \quad x_{ij} \in \mathbb{Z}$$



V vsako mesto prispemo natanko enkrat:

$$\forall j \in \{1, \dots, n\}: \sum_{i=1}^n x_{ij} = 1$$

Vsako mesto zapustimo natanko enkrat:

$$\forall i \in \{1, \dots, n\}: \sum_{j=1}^n x_{ij} = 1$$

Če iz mesta  $i$  potujemo v mesto  $j > 1$ , mu to sledi v vrstnem redu:

$$\forall i \in \{1, \dots, n\} \forall j \in \{2, \dots, n\}: y_i + nx_{ij} \leq y_j + n - 1$$

Zadnji pogoj poskrbi, da rešitev nima cikla brez mesta z indeksom 1. Skupaj s pogojema, da vsako mesto obiščemo in zapustimo natanko enkrat, si tako zagotovimo, da bo rešitev obsegala natanko en cikel. Vrednost spremenljivk  $y_i$  ( $1 \leq i \leq n$ ) ni omejena – zadnji pogoj poskrbi le, da je največja razlika med njimi enaka  $n - 1$ .

### Naloga 2.12.

Denimo, da imamo usmerjen graf  $G = (V, E)$ , kjer ima vsaka povezava  $uv \in E$  svojo ceno  $c_{uv}$  (če je dani graf neusmerjen, vsako povezavo nadomestimo s parom nasprotno usmerjenih povezav z isto ceno). Minimalno vpeto drevo je drevo  $T = (V, E')$  z  $E' \subseteq E$ , za katerega je vsota cen povezav iz  $E'$  najmanjša.

Naj bo  $n = |V|$  število vozlišč grafa  $G$ . Za vsako povezavo  $uv \in E$  bomo uvedli spremenljivko  $x_{uv}$ , katere vrednost interpretiramo kot

$$x_{uv} = \begin{cases} 1, & \text{povezava } uv \text{ je v množici } E', \text{ in} \\ 0 & \text{sicer.} \end{cases}$$

Poskrbeti moramo še, da bo naša rešitev res drevo. Izberimo si vozlišče  $r \in V$  kot koren drevesa in za vsako vozlišče  $v \in V$  uvedimo še spremenljivko  $y_v$ , ki se povečuje z globino vozlišča v drevesu.

Zapišimo celoštevilski linearni program.

$$\begin{aligned} \min \quad & \sum_{uv \in E} x_{uv} c_{uv} \quad \text{p.p.} \\ \forall uv \in E: \quad & 0 \leq x_{uv} \leq 1, \quad x_{uv} \in \mathbb{Z} \end{aligned}$$

V vozlišče  $r$  ne vstopi nobena povezava v drevesu:

$$\sum_{ur \in E} x_{ur} = 0$$

V ostala vozlišča vstopi natanko ena povezava:

$$\forall u \in V \setminus \{r\}: \sum_{uv \in E} x_{uv} = 1$$

Vrednost  $y_v$  ( $v \in V$ ) se povečuje po povezavah iz drevesa:

$$\forall uv \in E: y_u + nx_{uv} \leq y_v + n - 1$$

Zadnji pogoj poskrbi, da so vsa vozlišča v drevesu in ne v nepovezanem ciklu. Podobno kot pri nalogi 2.11 vrednost spremenljivk  $y_v$  ( $v \in V$ ) ni omejena – pogoj poskrbi le, da je največja razlika med njimi največ  $n - 1$ .

**Naloga 2.13.**

Za  $i$ -tega asistenta ( $1 \leq i \leq n$ ) in  $j$ -ti predmet ( $1 \leq j \leq m$ ) bomo uvedli spremenljivki  $x_{ij}$  in  $y_{ij}$ , kjer je  $x_{ij}$  število skupin, ki jih  $i$ -temu asistentu dodelimo pri  $j$ -tem predmetu, vrednost  $y_{ij}$  pa interpretiramo kot

$$y_{ij} = \begin{cases} 1, & i\text{-ti asistent bo izvajal vaje pri } j\text{-tem predmetu, in} \\ 0 & \text{sicer.} \end{cases}$$

Poleg tega bomo uvedli še spremenljivko  $t$ , s katero omejimo največje število različnih predmetov pri posameznem asistentu.

Zapišimo celoštevilski linearni program.

$$\begin{array}{ll} \min t & \text{p.p.} \\ \forall i \in \{1, \dots, n\} \forall j \in \{1, \dots, m\} : & x_{ij} \geq 0, \quad x_{ij} \in \mathbb{Z} \\ \forall i \in \{1, \dots, n\} \forall j \in \{1, \dots, m\} : & 0 \leq y_{ij} \leq 1, \quad y_{ij} \in \mathbb{Z} \end{array}$$

$x_{ij} = 0$  natanko tedaj, ko  $y_{ij} = 0$ :

$$\forall i \in \{1, \dots, n\} \forall j \in \{1, \dots, m\} : \quad y_{ij} \leq x_{ij} \leq c_j y_{ij}$$

Omejimo število ur za vsakega asistenta:

$$\forall i \in \{1, \dots, n\} : a_i \leq \sum_{j=1}^m u_j x_{ij} \leq b_i$$

Neželenih predmetov ne dodelimo:

$$\forall i \in \{1, \dots, n\} : \sum_{j \in N_i} y_{ij} = 0$$

Za vsak predmet dodelimo potrebno število skupin:

$$\forall j \in \{1, \dots, m\} : \sum_{i=1}^n x_{ij} = c_j$$

Nobenega predmeta ne dodelimo asistentoma  $p$  in  $q$ :

$$\forall j \in \{1, \dots, m\} : \quad y_{pj} + y_{qj} \leq 1$$

Število različnih predmetov na asistenta omejimo s  $t$ :

$$\forall i \in \{1, \dots, n\} : \sum_{j=1}^m y_{ij} \leq t$$

**Naloga 2.14.**

Za stroj  $i$  ( $1 \leq i \leq n$ ) bomo uvedli spremenljivko  $x_i$ , dodatno za opravilo  $j$  in časovno enoto  $h$  ( $1 \leq h, j \leq m$ ) pa še spremenljivko  $y_{ijh}$ . Njihove vrednosti interpretiramo kot

$$x_i = \begin{cases} 1, & \text{če stroj } i \text{ izvede vsaj eno opravilo, in} \\ 0 & \text{sicer;} \end{cases}$$

ter  $y_{ijh} = \begin{cases} 1, & \text{če stroj } i \text{ izvede opravilo } j \text{ v časovni enoti } h, \text{ in} \\ 0 & \text{sicer.} \end{cases}$

Dodatno bomo uvedli še spremenljivko  $t$ , katere vrednost bo enaka časovni enoti dokončanja zadnjega stroja.

Zapišimo celoštevilski linearni program.

$$\begin{array}{ll} \min t & \text{p.p.} \\ \forall i \in \{1, \dots, n\} : & 0 \leq x_i \leq 1, \quad x_i \in \mathbb{Z} \\ \forall i \in \{1, \dots, n\} \forall j, h \in \{1, \dots, m\} : & 0 \leq y_{ijh} \leq 1, \quad y_{ijh} \in \mathbb{Z} \end{array}$$

Vsako opravilo se izvede natanko enkrat:

$$\forall j \in \{1, \dots, m\} : \quad \sum_{i=1}^n \sum_{h=1}^m y_{ijh} = 1$$

Vsako stroj lahko hkrati opravlja največ eno opravilo:

$$\forall i \in \{1, \dots, n\} \forall h \in \{1, \dots, m\} : \quad \sum_{j=1}^m y_{ijh} \leq 1$$

Omejitev stroškov obratovanja:

$$\begin{aligned} \forall i \in \{1, \dots, n\} : \quad & \sum_{j=1}^m \sum_{h=1}^m y_{ijh} \leq m x_i \\ & \sum_{i=1}^n c_i x_i \leq C \end{aligned}$$

Vrstni red opravil:

$$\forall (j, k) \in P : \quad \sum_{i=1}^n \sum_{h=1}^m h(y_{ikh} - y_{ijh}) \geq 1$$

Hkratnost opravil:

$$\forall (j, k) \in S \forall h \in \{1, \dots, m\} : \quad \sum_{i=1}^n (y_{ijh} + y_{ikh}) \leq 1$$

Omejitev hkrati aktivnih strojev:

$$\forall h \in \{1, \dots, m\} : \quad \sum_{i=1}^n \sum_{j=1}^m y_{ijh} \leq A$$

Omejitev časa dokončanja:

$$\forall i \in \{1, \dots, n\} \forall j, h \in \{1, \dots, m\} : \quad h y_{ijh} \leq t$$

**Naloga 2.15.**

- (a) Očitno je, da bomo uporabili največ  $n$  košev, saj lahko vsak predmet postavimo v svoj koš. Zato bomo za  $i$ -ti predmet in  $j$ -ti koš ( $1 \leq i, j \leq n$ ) uvedli spremenljivko  $x_{ij}$ , katere vrednost interpretiramo kot

$$x_{ij} = \begin{cases} 1, & i\text{-ti predmet postavimo v } j\text{-ti koš, in} \\ 0 & \text{sicer.} \end{cases}$$

Poleg tega bomo uvedli še spremenljivko  $t$ , ki bo štela število uporabljenih košev.

Zapišimo celoštevilski linearni program.

$$\begin{array}{ll} \min t & \text{p.p.} \\ \forall i, j \in \{1, \dots, n\}: & 0 \leq x_{ij} \leq 1 \quad x_{ij} \in \mathbb{Z} \end{array}$$

Vsak predmet gre v en koš:

$$\forall i \in \{1, \dots, n\}: \quad \sum_{j=1}^n x_{ij} = 1$$

Omejitev velikosti košev:

$$\forall j \in \{1, \dots, n\}: \quad \sum_{i=1}^n s_i x_{ij} \leq V$$

Štetje košev:

$$\forall i, j \in \{1, \dots, n\}: \quad j x_{ij} \leq t$$

- (b) Dodajmo še zahtevano omejitev.

$$\forall (h, i) \in L \quad \forall j \in \{1, \dots, n\}: x_{hj} + x_{ij} \leq 1$$

**Naloga 2.16.**

Za vsak strežnik  $h$  ( $0 \leq h \leq k$ ), ponudnika  $i$  ( $1 \leq i \leq m$ ) in posnetek  $j$  ( $1 \leq j \leq n$ ) bomo uvedli spremenljivko  $x_{hij}$ , za strežnik  $h$  ( $1 \leq h \leq k$ ) in posnetek  $j$  ( $1 \leq j \leq n$ ) pa dodatno še spremenljivko  $y_{hj}$ . Njihove vrednosti interpretiramo kot

$$x_{hij} = \begin{cases} 1, & \text{ponudniku } i \text{ pošljamo posnetek } j \text{ iz strežnika } h, \text{ in} \\ 0 & \text{sicer;} \end{cases}$$

ter

$$y_{hj} = \begin{cases} 1, & \text{posnetek } j \text{ naložimo na strežnik } h, \text{ in} \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\begin{aligned} \min \quad & \sum_{h=0}^k \sum_{i=1}^m \sum_{j=1}^n r_{ij} \ell_{hi} x_{hij} \quad \text{p.p.} \\ \forall h \in \{0, \dots, k\} \quad \forall i \in \{1, \dots, m\} \quad \forall j \in \{1, \dots, n\} : & \quad 0 \leq x_{hij} \leq 1, \quad x_{hij} \in \mathbb{Z} \\ \forall h \in \{1, \dots, k\} \quad \forall j \in \{1, \dots, n\} : & \quad 0 \leq y_{hj} \leq 1, \quad y_{hj} \in \mathbb{Z} \end{aligned}$$

Omejitev prostora:

$$\forall h \in \{1, \dots, k\} : \quad \sum_{j=1}^n s_j y_{hj} \leq c_h$$

Za dostavo mora biti posnetek prisoten na strežniku:

$$\forall h \in \{0, \dots, k\} \quad \forall i \in \{1, \dots, m\} \quad \forall j \in \{1, \dots, n\} : \quad x_{hij} \leq y_{hj}$$

Strežnika za posnetek ne uporabimo, če se nahaja na strežniku z manjšo latenco:

$$\forall h, t \in \{0, \dots, k\} \quad \forall i \in \{1, \dots, m\} \quad \forall j \in \{1, \dots, n\} : \quad (\ell_{hi} - \ell_{ti})(x_{hij} + y_{tj}) \leq 1$$

Vsak posnetek pošljamo ponudniku iz natanko enega strežnika:

$$\forall h \in \{0, \dots, k\} \quad \forall j \in \{1, \dots, n\} : \quad \sum_{i=1}^m x_{hij} = 1$$

### Naloga 2.17.

Za  $i$ -tega trgovca ( $1 \leq i \leq n$ ) bomo uvedli spremenljivke  $x_i$ ,  $y_i$  in  $z_i$ , pri čemer sta  $x_i$  in  $y_i$  število zabojev avokadov oziroma banan, ki jih distributer proda  $i$ -temu trgovcu, in  $z_i$  število voženj tovornjakov, ki bodo sadje dostavili do  $i$ -tega trgovca. Poleg tega bomo uvedli še spremenljivki  $u$  in  $v$ , ki ju interpretiramo kot

$$\begin{aligned} u &= \begin{cases} 1, & \text{trgovec } p \text{ kupi avokade, in} \\ 0 & \text{sicer;} \end{cases} \\ \text{ter } v &= \begin{cases} 1, & \text{trgovec } p \text{ kupi banane, in} \\ 0 & \text{sicer.} \end{cases} \end{aligned}$$

Zapišimo celoštevilski linearni program.

$$\begin{aligned} \max \quad & \sum_{i=1}^n (a_i x_i + b_i y_i - d_i z_i) \quad \text{p.p.} \\ \forall i \in \{1, \dots, n\} : & \quad x_i \geq 0, \quad x_i \in \mathbb{Z} \\ \forall i \in \{1, \dots, n\} : & \quad y_i \geq 0, \quad y_i \in \mathbb{Z} \\ \forall i \in \{1, \dots, n\} : & \quad z_i \geq 0, \quad z_i \in \mathbb{Z} \\ & \quad 0 \leq u \leq 1, \quad u \in \mathbb{Z} \\ & \quad 0 \leq v \leq 1, \quad v \in \mathbb{Z} \end{aligned}$$

Skupno število zabojev:

$$\sum_{i=1}^n x_i \leq A$$

$$\sum_{i=1}^n y_i \leq B$$

Omejitev porabe trgovca:

$$\forall i \in \{1, \dots, n\} : a_i x_i + b_i y_i \leq c_i$$

Število tovornjakov:

$$\forall i \in \{1, \dots, n\} : x_i + y_i \leq K z_i$$

Trgovec  $p$  ne kupi obojega:

$$x_p \leq A u$$

$$y_p \leq B v$$

$$u + v \leq 1$$

Trgovec  $q$  kupi avokade, če jih tudi  $r$ :

$$x_r \leq A x_q$$

### Naloga 2.18.

Očitno bomo potrebovali največ  $n$  delavcev, zato bomo za  $h$ -tega delavca ( $1 \leq h \leq n$ ),  $i$ -to nalogo ( $1 \leq i \leq n$ ) in  $k$ -to časovno enoto ( $1 \leq k \leq T$ ) uvedli spremenljivko  $x_{hik}$ , katere vrednost interpretiramo kot

$$x_{hik} = \begin{cases} 1; & h\text{-ti delavec začne izvajati } i\text{-to nalogo v } k\text{-ti časovni enoti, in} \\ 0 & \text{sicer.} \end{cases}$$

Poleg tega bomo uvedli še spremenljivko  $r$ , ki šteje število delavcev.

Zapišimo celoštevilski linearni program, pri čemer uporabimo oznako  $T_i = T - t_i + 1$  za zadnji možen čas začetka  $i$ -te naloge.

$$\begin{array}{ll} \min r & \text{p.p.} \\ \forall h, i \in \{1, \dots, n\} \forall k \in \{1, \dots, T_i\} : & 0 \leq x_{hik} \leq 1, \quad x_{hik} \in \mathbb{Z} \\ & r \geq 0, \quad r \in \mathbb{Z} \end{array}$$

Vsako nalogo opravi natanko en delavec in jo pravočasno konča:

$$\forall i \in \{1, \dots, n\} : \sum_{h=1}^n \sum_{k=1}^{T_i} x_{hik} = 1$$

Vsak delavec lahko hkrati izvaja samo eno nalogo:

$$\forall h, i \in \{1, \dots, n\} \forall k \in \{1, \dots, T_i\} : (t_i - 2)x_{hik} + \sum_{j=1}^n \sum_{\ell=k}^{k+t_i-1} x_{hj\ell} \leq t_i - 1$$

Vrstni red opravljanja nalog:

$$\forall (i, j) \in S \forall k \in \{1, \dots, T_i\}: \quad \sum_{h=1}^n \left( x_{hik} - \sum_{\ell=k+t_i}^{T_j} x_{hj\ell} \right) \leq 0$$

Štetje delavcev:

$$\forall h, i \in \{1, \dots, n\} \forall k \in \{1, \dots, T\}: \quad hx_{hik} \leq r$$

### Naloga 2.19.

Naj bo  $m$  število vozlišč grafa  $G$ . Očitno bo linija obiskala največ  $m$  postaj, zato bomo za postajališče  $u \in V$  in  $i \in \{1, \dots, m\}$  uvedli spremenljivko  $x_{ui}$ , za vsako povezavo  $uv \in E$  pa še spremenljivko  $y_{uv}$ . Njihove vrednosti interpretiramo kot

$$x_{ui} = \begin{cases} 1; & \text{postajališče } u \text{ obiščemo } i\text{-to po vrsti, in} \\ 0 & \text{sicer;} \end{cases}$$

ter  $y_{uv} = \begin{cases} 1; & \text{postajališči } u \text{ in } v \text{ obiščemo zaporedoma, in} \\ 0 & \text{sicer.} \end{cases}$

Zapišimo celoštevilski linearni program.

$$\begin{array}{ll} \max & \sum_{u \in V} \sum_{i=1}^n x_{ui} \quad \text{p.p.} \\ \forall u \in V \forall i \in \{1, \dots, m\}: & 0 \leq x_{ui} \leq 1, \quad x_{ui} = \mathbb{Z} \\ \forall uv \in E: & 0 \leq y_{uv} \leq 1, \quad y_{uv} = \mathbb{Z} \end{array}$$

Vsako postajališče obiščemo največ enkrat:

$$\forall u \in V: \quad \sum_{i=1}^m x_{ui} \leq 1$$

Naenkrat obiščemo največ eno postajališče:

$$\forall i \in \{1, \dots, m\}: \quad \sum_{u \in V} x_{ui} \leq 1$$

Indeksov ne izpuščamo:

$$\forall i \in \{2, \dots, m\}: \quad \sum_{u \in V} x_{ui} \leq \sum_{u \in V} x_{u,i-1}$$

Nesosedna postajališča niso zaporedna:

$$\forall uv \notin E \forall i \in \{2, \dots, m\}: \quad x_{u,i-1} + x_{ui} + x_{v,i-1} + x_{vi} \leq 1$$

Začnemo v postajališču  $p_1$ :

$$x_{p_1,1} = 1$$

Postajališča iz seznama si sledijo v podanem vrstnem redu:

$$\forall i \in \{1, \dots, m\} \forall j \in \{2, \dots, n\}: \quad x_{p_{j-1},i} \leq \sum_{h=i+1}^m x_{p_j,h}$$

Končamo v postajališču  $p_n$ :

$$\forall i \in \{1, \dots, m\} : (m-i)x_{p_n, i} + \sum_{u \in V} \sum_{h=i+1}^m x_{uh} \leq m-i$$

Časovna omejitev:

$$\forall uv \in E \forall i \in \{2, \dots, m\} : x_{u, i-1} + x_{ui} + x_{v, i-1} + x_{vi} \leq y_{uv} + 1$$

$$\sum_{uv \in E} c_{uv} y_{uv} \leq T$$

### Naloga 2.20.

Za igralca  $i \in A$  in  $j$ -ti klub ( $1 \leq j \leq n$ ) bomo uvedli spremenljivko  $x_{ij}$ , za igralca  $i \in B$  pa spremenljivko  $y_i$ . Njihove vrednosti interpretiramo kot

$$x_{ij} = \begin{cases} 1; & \text{igralca } i \text{ prodamo } j\text{-temu klubu, in} \\ 0 & \text{sicer;} \end{cases}$$

$$\text{ter } y_i = \begin{cases} 1; & \text{odkupimo igralca } i \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\max \quad \sum_{i \in B} q_i y_i - \sum_{i \in A} \sum_{j=1}^n q_i x_{ij} \quad \text{p.p.}$$

$$\forall i \in A \forall j \in \{1, \dots, n\} : \quad 0 \leq x_{ij} \leq 1, \quad x_{ij} \in \mathbb{Z}$$

$$\forall i \in B : \quad 0 \leq y_i \leq 1, \quad y_i \in \mathbb{Z}$$

Vsakega igralca prodamo največ enemu klubu:

$$\forall i \in A : \quad \sum_{j=1}^n x_{ij} \leq 1$$

Omejitev stroškov:

$$\sum_{i \in B} r_i y_i - \sum_{i \in A} \sum_{j=1}^n p_{ij} x_{ij} \leq S$$

Skupno število igralcev:

$$\sum_{i \in B} y_i - \sum_{i \in A} \sum_{j=1}^n x_{ij} = 0$$

Število igralcev za vsako pozicijo:

$$-1 \leq \sum_{i \in B \cap G} y_i - \sum_{i \in A \cap G} \sum_{j=1}^n x_{ij} \leq 1$$

$$-1 \leq \sum_{i \in B \cap D} y_i - \sum_{i \in A \cap D} \sum_{j=1}^n x_{ij} \leq 1$$

$$-1 \leq \sum_{i \in B \cap M} y_i - \sum_{i \in A \cap M} \sum_{j=1}^n x_{ij} \leq 1$$



$$-1 \leq \sum_{i \in B \cap F} y_i - \sum_{i \in A \cap F} \sum_{j=1}^n x_{ij} \leq 1$$

Igralca  $a$  in  $b$  gresta v isti klub:

$$\forall j \in \{1, \dots, n\} : \quad x_{aj} = x_{bj}$$

### Naloga 2.21.

Za del Evrope  $i \in D$ , kjer je  $D = \{S, Z, V, J\}$  (severna, zahodna, vzhodna, južna Evropa), in kraj  $j \in K$ , kjer je  $K = \{\text{Lon, Ber, Bud, Mad}\}$  (London, Berlin, Budimpešta, Madrid) bomo uvedli spremenljivko  $x_{ij}$ , za kraj  $j \in K$  pa še spremenljivko  $y_j$ . Njihove vrednosti interpretiramo kot

$$x_{ij} = \begin{cases} 1; & \text{iz dela } i \text{ pošljamo čeke v mesto } j, \text{ in} \\ 0 & \text{sicer;} \end{cases}$$

ter  $y_j = \begin{cases} 1; & \text{odpremo podružnico v mestu } j \\ 0 & \text{sicer.} \end{cases}$

Zapišimo celoštevilski linearni program, pri čemer bomo stroške računali v enotah po 1 000€.

$$\begin{aligned} \min \quad & 50 \sum_{j \in K} y_j + 28x_{S,\text{Lon}} + 84x_{S,\text{Ber}} + 112x_{S,\text{Bud}} + 84x_{S,\text{Mad}} \\ & + 60x_{Z,\text{Lon}} + 20x_{Z,\text{Ber}} + 50x_{Z,\text{Bud}} + 50x_{Z,\text{Mad}} \\ & + 96x_{V,\text{Lon}} + 60x_{V,\text{Ber}} + 24x_{V,\text{Bud}} + 60x_{V,\text{Mad}} \\ & + 64x_{J,\text{Lon}} + 40x_{J,\text{Ber}} + 40x_{J,\text{Bud}} + 16x_{J,\text{Mad}} \quad \text{p.p.} \end{aligned}$$

$$\forall i \in D \quad \forall j \in K : 0 \leq x_{ij} \leq 1, \quad x_{ij} \in \mathbb{Z}$$

$$\forall j \in K : 0 \leq y_j \leq 1, \quad y_j \in \mathbb{Z}$$

Iz posameznega dela pošljamo v eno podružnico:

$$\forall i \in D : \sum_{j \in K} x_{ij} = 1$$

Če nekam pošljamo, odpremo podružnico:

$$\forall j \in K : \sum_{i \in D} x_{ij} \leq 4y_j$$

### Naloga 2.22.

Za vsakega igralca želimo vedeti, ali bo član začetne postave. Igralcem po vrsti dodelimo števila od 1 do 8. Za  $i$ -tega igralca ( $1 \leq i \leq 8$ ) bomo uvedli spremenljivko  $x_i$ , katere vrednost interpretiramo kot

$$x_i = \begin{cases} 1, & \text{če bo na } i\text{-ti igralec v začetni postavi, in} \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\begin{aligned} \max \quad & 213x_1 + 208x_2 + 203x_3 + 192x_4 \\ & + 196x_5 + 197x_6 + 200x_7 + 195x_8 \quad \text{p.p.} \\ \forall i \in \{1, \dots, 8\}: \quad & 0 \leq x_i \leq 1, \quad x_i \in \mathbb{Z} \end{aligned}$$

V postavi je pet članov:

$$\sum_{i=1}^8 x_i = 5$$

V postavi so zastopane vse pozicije, od tega natanko en center:

$$\begin{aligned} x_1 + x_2 &= 1 \\ x_3 + x_4 + x_5 &\geq 1 \\ x_6 + x_7 + x_8 &\geq 1 \end{aligned}$$

V postavi je ali Ciril ali Filip:

$$x_3 + x_6 = 1$$

Če začne Borut ali David, Hugo ostane v rezervi:

$$x_2 + x_4 + 2x_8 \leq 2$$

### Naloga 2.23.

Glede na to, da imamo v vsaki vrstici in stolpcu samo po dva otoka, bomo v vsakem od njih zgradili največ dva mostova med otokoma. Za  $i = 1, 2, 3$  bomo torej definirali spremenljivki  $x_i$  in  $y_i$ , ki povesta, koliko mostov bomo zgradili v  $i$ -ti vrstici oziroma stolpcu. Zapišimo linearne omejitve.

$$\begin{aligned} \forall i \in \{1, 2, 3\}: \quad & 0 \leq x_i \leq 2, \quad x_i \in \mathbb{Z} \\ \forall i \in \{1, 2, 3\}: \quad & 0 \leq y_i \leq 2, \quad y_i \in \mathbb{Z} \\ x_1 + y_1 &= 2 \\ x_1 + y_2 &= 1 \\ x_2 + y_2 &= 3 \\ x_2 + y_3 &= 3 \\ x_3 + y_1 &= 3 \\ x_3 + y_3 &= 2 \end{aligned}$$

Ciljna funkcija tukaj ni pomembna – zanima nas samo dopustna rešitev.

**Naloga 2.24.**

Uporabili bomo oznake sort in kupcev iz naloge 2.8, poleg tega pa naj  $n_A$  pomeni število flaškonov sorte  $A$ , ki jih imamo na voljo. Za vsako sorto  $i \in I$  in kupca  $j \in J$  bomo uvedli spremenljivko  $x_{ij}$ , ki pove, koliko flaškonov sorte  $i$  prodamo kupcu  $j$ . Zapišimo celoštevilski linearni program.

$$\max \sum_{j \in J} (12x_{RM,j} + 8x_{LR,j} + 15x_{RR,j}) \quad \text{p.p.}$$

$$\forall i \in I \forall j \in J: x_{ij} \geq 0, \quad x_{ij} \in \mathbb{Z}$$

Omejitve za sorte:

$$\sum_{j \in J} x_{RM,j} \leq 200$$

$$\sum_{j \in J} x_{LR,j} \leq 1000$$

$$\sum_{j \in J} x_{RR,j} \leq 5000$$

Omejitve za stranke:

$$\sum_{i \in I} x_{i,K} \leq 1500$$

$$\sum_{i \in I} x_{i,L} \leq 500$$

$$\sum_{i \in I} x_{i,O} \leq 100$$

$$\sum_{i \in I} x_{i,Z} \leq 50$$

Omejitvi barov:

$$12x_{RM,K} + 8x_{LR,K} + 15x_{RR,K} \leq \sum_{j \in \{O,Z\}} (12x_{RM,j} + 8x_{LR,j} + 15x_{RR,j})$$

$$12x_{RM,L} + 8x_{LR,L} + 15x_{RR,L} \leq \sum_{j \in \{O,Z\}} (12x_{RM,j} + 8x_{LR,j} + 15x_{RR,j})$$

Želje glede sort:

$$x_{RM,Z} + x_{RR,Z} \geq 50$$

$$x_{RM,O} + x_{LR,O} = 0$$

$$x_{RR,O} \geq 50$$

Ženin pogoj:

$$n_A - \sum_{j \in J} x_{Aj} \geq x_{AB} + x_{AC}$$

**Naloga 2.25.**

Za vsako odprto celico  $s \in \mathcal{O}$  naj  $a_s$  označuje številko v celici  $s$ , tj., število min v celicah iz množice  $N(s)$ . Za vsako zaprto celico  $t \in \mathcal{Z}$  bomo uvedli spremenljivko  $x_t$ , katere vrednost interpretiramo kot

$$x_t = \begin{cases} 1; & \text{v celici } t \text{ je mina, in} \\ 0 & \text{sicer.} \end{cases}$$

Najprej zapišimo omejitve, ki jih bomo vseskozi upoštevali.

$$\begin{aligned} \forall t \in \mathcal{Z} : \quad & 0 \leq x_t \leq 1, \quad x_t \in \mathbb{Z} \\ \forall s \in \mathcal{O} : \quad & \sum_{t \in N(s) \setminus \mathcal{O}} x_t = a_s \end{aligned}$$

(a) Začetnim omejitvam dodamo še

$$\sum_{t \in \mathcal{Z}} x_t = p.$$

Zanima nas le obstoj dopustne rešitve, zato ciljna funkcija ni pomembna.

(b) Pri začetnih omejitvah uporabimo ciljno funkcijo

$$\max \sum_{t \in \mathcal{Z}} x_t.$$

(c) Za vsako zaprto celico  $r \in \mathcal{Z}$  poiščemo dopustno rešitev celoštevilskega linearnega programa z začetnimi omejitvami (in še omejitvijo iz točke (a), če poznamo skupno število min) in ciljno funkcijo

$$\min x_r.$$

Če je celoštevilski linearni program dopusten z vrednostjo ciljne funkcije enako 1, potem celica  $r$  nujno vsebuje mino.

**Naloga 2.26.**

Za škatli  $i$  in  $j$  ( $1 \leq i, j \leq n$ ) bomo uvedli spremenljivko  $x_{ij}$ , katere vrednost interpretiramo kot

$$x_{ij} = \begin{cases} 1, & \text{če } i\text{-to škatlo postavimo v } j\text{-to škatlo, in} \\ 0 & \text{sicer.} \end{cases}$$

Ker želimo minimizirati volumen škatel, ki jih ne zložimo v večje škatle, lahko maksimiziramo volumen škatel, ki jih zložimo v večje škatle. Zapišimo celoštevilski linearni program.

$$\max \sum_{i=1}^n \sum_{j=1}^n v_i x_{ij} \quad \text{p.p.}$$

$$\forall i, j \in \{1, \dots, n\}: \quad 0 \leq x_{ij} \leq 1, \quad x_{ij} \in \mathbb{Z}$$

Škatle ne moremo postaviti vase:

$$\forall i \in \{1, \dots, n\}: \quad x_{ii} = 0$$

Vsako škatlo lahko postavimo v največ eno večjo škatlo:

$$\forall i \in \{1, \dots, n\}: \quad \sum_{j=1}^n x_{ij} \leq 1$$

Omejitev na volumnu škatle:

$$\forall j \in \{1, \dots, n\}: \quad \sum_{i=1}^n v_i x_{ij} \leq v_j$$

Škatle, ki vsebuje drugo škatlo, ne moremo postaviti v večjo škatlo:

$$\forall i, j, k \in \{1, \dots, n\}: \quad x_{ij} + x_{jk} \leq 1$$

### Naloga 2.27.

- (a) Za  $i$ -to ponudbo ( $1 \leq i \leq k$ ) bomo uvedli spremenljivko  $x_i$ , katere vrednost interpretiramo kot

$$x_i = \begin{cases} 1, & \text{če naj dražitelj sprejme } i\text{-to ponudbo, in} \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\max \sum_{i=1}^n c_i x_i \quad \text{p.p.}$$

$$\forall i \in \{1, \dots, k\}: \quad 0 \leq x_i \leq 1, \quad x_i \in \mathbb{Z}$$

$$\forall p \in A: \quad \sum_{\substack{i=1 \\ p \in B_i}}^k x_i \leq 1$$

- (b) Celostevilskemu linearnemu programu dodamo še dva pogoja.

- Če je ponudba 1 uspešna, potem mora biti uspešna tudi ponudba 2 ali ponudba 3:

$$x_1 \leq x_2 + x_3$$

- Če so uspešne vse ponudbe s sodim indeksom, potem morajo biti ne-uspešne vsaj tri ponudbe z lihim indeksom:

$$\sum_{i=1}^k (2 + (-1)^i) x_i \leq k + 2 \left\lfloor \frac{k}{2} \right\rfloor - 3$$

Razlaga: na levi strani ponudbe z lihim indeksom štejemo enkrat, ponudbe s sodim indeksom (teh je  $\lfloor \frac{k}{2} \rfloor$ ) pa trikrat. Če sprejmemo vse ponudbe s sodim indeksom, potem desna stran omejuje število sprejetih ponudb na  $k - 3$  – vsaj treh ponudb z lihim indeksom torej nismo sprejeli. Če pa ne sprejmemo vseh ponudb s sodim indeksom, je vsota na levi za vsaj 3 manjša od maksimalne, tako da lahko sprejmemo poljubno število ponudb z lihim indeksom.

### Naloga 2.28.

Za  $i \in \{1, \dots, n\}$  bomo uvedli spremenljivko  $x_i$ , katere vrednost interpretiramo kot

$$x_i = \begin{cases} 1; & a_i \in L, \\ 0; & a_i \notin L. \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\begin{aligned} \max \quad & \sum_{i=1}^n x_i (a_i)^2 \quad \text{p.p.} \\ \forall i \in \{1, \dots, n\}: \quad & 0 \leq x_i \leq 1, \quad x_i \in \mathbb{Z} \\ & x_1 + x_2 \leq 1 \\ & x_3 + x_5 - x_7 \leq 1 \\ & (n-4)x_4 + \sum_{i=1}^n x_i \leq n \\ & \sum_{\substack{i=1 \\ a_i > 0}}^n x_i \geq \sum_{\substack{i=1 \\ a_i < 0}}^n x_i + 1 \end{aligned}$$

### Naloga 2.29.

Za  $i$ -ti predmet ( $1 \leq i \leq n$ ) in  $j$ -ti tovornjak ( $1 \leq j \leq 5$ ) bomo uvedli spremenljivki  $x_{ij}$  in  $y_j$ , katerih vrednosti interpretiramo kot

$$\begin{aligned} x_{ij} &= \begin{cases} 1; & i\text{-ti predmet naložimo na } j\text{-ti tovornjak, in} \\ 0 & \text{sicer;} \end{cases} \\ \text{ter } y_j &= \begin{cases} 1; & \text{tovor na } j\text{-tem tovornjaku ne presega mase } M, \text{ in} \\ 0 & \text{sicer.} \end{cases} \end{aligned}$$

Poleg tega bomo uvedli še spremenljivko  $t$ , ki bo predstavljala obremenitev najbolj obremenjenega tovornjaka.

Zapišimo celoštevilski linearni program.

$$\begin{array}{lll} \min t & \text{p.p.} \\ \forall i \in \{1, \dots, n\} \forall j \in \{1, \dots, 5\} : & 0 \leq x_{ij} \leq 1, & x_{ij} \in \mathbb{Z} \\ \forall j \in \{1, \dots, 5\} : & 0 \leq y_j \leq 1, & y_j \in \mathbb{Z} \end{array}$$

Vsak predmet naložimo na en tovornjak:

$$\forall i \in \{1, \dots, n\} : \sum_{j=1}^5 x_{ij} = 1$$

Največja obremenitev:

$$\forall j \in \{1, \dots, 5\} : \sum_{i=1}^n m_i x_{ij} \leq t$$

Zapišimo še dodatna pogoja.

(a) Tovor na prvem tovornjaku ne presega mase  $M$ :

$$\sum_{i=1}^n m_i x_{1j} \leq M$$

(b) Na vsaj enem tovornjaku tovor ne presega mase  $M$ :

$$\forall j \in \{1, \dots, 5\} : \sum_{i=1}^n m_i (x_{ij} + y_j - 1) \leq M y_j$$

$$\sum_{j=1}^5 y_j \geq 1$$

### Naloga 2.30.

(a) Za  $i, j \in \{1, \dots, n\}$  bomo uvedli spremenljivko  $x_{ij}$ , katere vrednost interpretiramo kot

$$x_{ij} = \begin{cases} 1; & \text{na } i\text{-to mesto postavimo število } j, \text{ in} \\ 0 & \text{sicer.} \end{cases}$$

Poleg tega bomo za  $i \in \{1, \dots, n\}$  uvedli še spremenljivko  $y_i$ , ki bo predstavljala manjšo od vsot števil strogo levo in strogo desno od  $i$ -tega mesta.

Zapišimo celoštevilski linearni program.

$$\begin{array}{lll} \max \sum_{i=1}^n y_i & \text{p.p.} \\ \forall i, j \in \{1, \dots, n\} : & 0 \leq x_{ij} \leq 1, & x_{ij} \in \mathbb{Z} \\ \forall i \in \{1, \dots, n\} : & y_i \geq 0, & y_i \in \mathbb{Z} \end{array}$$

Vsako število postavimo na eno mesto:

$$\forall j \in \{1, \dots, n\} : \sum_{i=1}^n x_{ij} = 1$$

Na vsako mesto postavimo eno število:

$$\forall i \in \{1, \dots, n\} : \sum_{j=1}^n x_{ij} = 1$$

Dobitek ob izbiri  $i$ -tega mesta:

$$\forall i \in \{1, \dots, n\} : \sum_{h=1}^{i-1} \sum_{j=1}^n j x_{hj} \geq y_i$$

$$\forall i \in \{1, \dots, n\} : \sum_{h=i+1}^n \sum_{j=1}^n j x_{hj} \geq y_i$$

(b) Dodajmo še pogoj, da sta števili 1 in 2 sosednji.

$$-1 \leq \sum_{i=1}^n i(x_{i1} - x_{i2}) \leq 1$$

### Naloga 2.31.

Naj bo  $d_{ij}$  dobiček, ki ga tovarna na stopnji  $j$  prinese v državi  $i$ , ter  $n_{ij}$  trenutno število tovarn na stopnji  $j$  v državi  $i$ . Definirajmo še množice

$$\begin{aligned} EU &= \{AT, SI, IT, HU, HR\}, \\ \overline{EU} &= \{IR, IQ, AF, TM, PK, AM, AZ\} \quad \text{in} \\ D &= EU \cup \overline{EU} \end{aligned}$$

(tj., države označujemo s kodami ISO 3166-1 alpha-2). Za državo  $i \in D$  in stopnjo razvoja  $j$  ( $1 \leq j \leq 3$ ) bomo uvedli spremenljivko  $x_{ij}$ , katere vrednost bo enaka številu tovarn na stopnji  $j$  v državi  $i$ , ki jih nadgradimo do stopnje  $j+1$ .

Zapišimo celoštevilski linearni program.

$$\max \sum_{i \in D} \sum_{j=1}^3 (d_{i,j+1} - d_{ij}) x_{ij} \quad \text{p.p.}$$

$$\forall i \in D \quad \forall j \in \{1, 2, 3\} : \quad 0 \leq x_{ij} \leq n_{ij}, \quad x_{ij} \in \mathbb{Z}$$

Nadgradimo največ 15 tovarn:

$$\sum_{i \in D} \sum_{j=1}^3 x_{ij} \leq 15$$

V vsaki državi nadgradimo največ 3 tovarne:

$$\forall i \in D : \quad \sum_{j=1}^3 x_{ij} \leq 3$$



Slovenija ima več tovarn stopnje 4 kot Hrvaška:

$$n_{SI,4} + x_{SI,3} \geq n_{HR,4} + x_{HR,3} + 1$$

Slovenija ima več tovarn stopnje 3 kot Italija:

$$n_{SI,3} + x_{SI,2} - x_{SI,3} \geq n_{IT,3} + x_{IT,2} - x_{IT,3} + 1$$

Nadgradenj v EU vsaj toliko kot izven EU:

$$\sum_{i \in EU} \sum_{j=1}^3 x_{ij} \geq \sum_{i \in \overline{EU}} \sum_{j=1}^3 x_{ij}$$

Pogoj za Azerbajdžan in Armenijo:

$$x_{AM,1} \geq n_{AM,1}(x_{AZ,2} - x_{AM,3})$$

Iran ima vsaj toliko tovarn stopnje 4 kot EU:

$$\sum_{i \in EU} (n_{i,4} + x_{i,3}) \leq n_{IR,4} + x_{IR,3}$$

### Naloga 2.32.

Za vsako križišče oziroma vozlišče grafa  $G = (V, E)$  želimo vedeti, ali bo na njem plinska maska. Za vsako vozlišče  $v \in V$  bomo uvedli spremenljivko  $x_v$ , katere vrednost interpretiramo kot

$$x_v = \begin{cases} 1, & \text{če bo na } v\text{-tem vozlišču plinska maska, in} \\ 0 & \text{sicer.} \end{cases}$$

(a) Zapišimo celoštevilski linearni program.

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v && \text{p.p.} \\ \forall v \in V: \quad & 0 \leq x_v \leq 1, && x_v \in \mathbb{Z} \end{aligned}$$

Vsako križišče ima masko ali pa jo ima eno izmed njegovih sosedov:

$$\forall v \in V: x_v + \sum_{uv \in E} x_u \geq 1$$

(b) Dodajmo še omejitve za podane pogoje.

V križišču  $s$  je maska, v križišču  $t$  pa ne:

$$\begin{aligned} x_s &= 1 \\ x_t &= 0 \end{aligned}$$

Največ 2 izmed križišč  $s_1, \dots, s_k$  imata masko:

$$\sum_{i=1}^k x_{s_i} \leq 2$$

Če je v križišču  $v_1$  maska, v križišču  $v_2$  pa ne, je maska vsaj v enem izmed križišč  $v_3$  in  $v_4$ :

$$x_{v_1} - x_{v_2} \leq x_{v_3} + x_{v_4}$$

**Naloga 2.33.**

Za vsak bazen želimo vedeti, v kakšnem stanju bo po koncu petletnega obdobja. Za  $i$ -ti bazen ( $1 \leq i \leq 20$ ) bomo uvedli spremenljivki  $x_i$  in  $y_i$ , katerih vrednosti interpretiramo kot

$$x_i = \begin{cases} 1, & \text{če se } i\text{-ti bazen nadgradi, in} \\ 0 & \text{sicer,} \end{cases}$$

ter  $y_i = \begin{cases} 1, & \text{če se } i\text{-ti bazen proda, in} \\ 0 & \text{sicer.} \end{cases}$

Zapišimo celoštevilski linearni program.

$$\begin{aligned} \max \quad & \sum_{i=1}^{20} x_i \quad \text{p.p.} \\ \forall i \in \{1, \dots, 20\}: \quad & 0 \leq x_i \leq 1, \quad x_i \in \mathbb{Z} \\ \forall i \in \{1, \dots, 20\}: \quad & 0 \leq y_i \leq 1, \quad y_i \in \mathbb{Z} \end{aligned}$$

Že nadgrajeni bazeni:

$$\forall i \in \{1, 2, 3\}: \quad x_i = 1$$

Nadgrajeni bazeni se ne prodajajo:

$$\forall i \in \{1, \dots, 20\}: \quad x_i + y_i \leq 1$$

Nadgradi se največ dvakrat toliko bazenov, kot se jih proda:

$$\sum_{i=4}^{20} x_i \leq 2 \sum_{i=1}^{20} y_i$$

Če se nadgradi bazen 8, se tudi 11 in 14, bazen 15 pa ostane v Kopalčevi lasti:

$$3x_8 + y_{15} \leq x_{14} + x_{11} + 1$$

Če bazeni od 15 do 20 ostanejo v Kopalčevi lasti, se bazeni od 4 do 7 ne nadgradijo:

$$\sum_{i=4}^7 x_i \leq 4 \sum_{i=15}^{20} y_i$$

Bazen  $a$  se proda, bazen  $b$  pa nadgradi:

$$x_a = 1$$

$$y_b = 1$$

Bazen  $c$  ostane v nespremenjenem stanju:

$$x_c = \begin{cases} 1, & \text{če } c \in \{1, 2, 3\}, \text{ in} \\ 0 & \text{sicer} \end{cases}$$

$$y_c = 0$$

Ker je  $c$  konstanta, je tudi predzadnja omejitev linearna.

**Naloga 2.34.**

Za povezavo  $uv \in E$  bomo uvedli spremenljivko  $x_{uv}$ , katere vrednost interpretiramo kot

$$x_{uv} = \begin{cases} 1; & uv \in M, \\ 0; & uv \notin M. \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\begin{aligned} \min \quad & \sum_{uv \in E} w(uv)x_{uv} \quad \text{p.p.} \\ \forall uv \in E: \quad & 0 \leq x_{uv} \leq 1, \quad x_{uv} \in \mathbb{Z} \end{aligned}$$

Vsako vozlišče je krajišče največ ene povezave iz  $M$ :

$$\forall u \in V: \quad \sum_{uv \in E} x_{uv} \leq 1$$

Za vsako povezavo obstaja povezava v  $M$  s skupnim krajiščem:

$$\forall uv \in E: \quad \sum_{uw \in E} x_{uw} + \sum_{vw \in E} x_{vw} \geq 1$$

**Naloga 2.35.**

Naj bo  $n$  število vozlišč grafa  $G$ . Očitno bomo lahko množico njegovih vozlišč razbili na največ  $n$  podmnožic, zato bomo za vozlišče  $u \in V$  in  $i \in \{1, \dots, n\}$  uvedli spremenljivko  $x_{ui}$ , katere vrednost interpretiramo kot

$$x_{ui} = \begin{cases} 1; & u \in V_i, \\ 0; & u \notin V_i. \end{cases}$$

Poleg tega bomo za vozlišči  $u, v \in V$  uvedli še spremenljivko  $y_{uv}$ , ki bo predstavljala razdaljo med njima. Nazadnje bomo uvedli še spremenljivko  $t$ , ki bo štela, koliko podmnožic potrebujemo.

Zapišimo celoštevilski linearni program.

$$\begin{aligned} \min \quad & t \quad \text{p.p.} \\ \forall u \in V \forall i \in \{1, \dots, n\}: \quad & 0 \leq x_{ui} \leq 1, \quad x_{ui} \in \mathbb{Z} \\ \forall u, v \in V: \quad & y_{uv} \geq 0, \quad y_{uv} \in \mathbb{Z} \\ & t \geq 0, \quad t \in \mathbb{Z} \end{aligned}$$

Vsako vozlišče postavimo v eno podmnožico:

$$\forall u \in V: \quad \sum_{i=1}^n x_{ui} = 1$$

Štetje podmnožic:

$$\forall u \in V \forall i \in \{1, \dots, n\}: \quad i x_{ui} \leq t$$

Razdalje v podmnožicah:

$$\forall u, v \in V, u \neq v: (i+1)(x_{ui} + x_{vi} - 1) \leq y_{uv}$$

Razdalje med vozlišči:

$$\begin{aligned}\forall u \in V : & \quad y_{uu} = 0 \\ \forall u \in V \forall vw \in E : & \quad -1 \leq y_{uv} - y_{uw} \leq 1\end{aligned}$$

Mavrično pakirno število  $\chi_p(G)$  dobimo kot vrednost ciljne funkcije zgornjega celoštevilskega linearnega programa.

Opomnimo, da vrednosti  $y_{uv}$  v dopustnih rešitvah zgornjega programa ne predstavljajo nujno dejanskih razdalj med vozlišči – velja le  $y_{uv} \leq d_G(u, v)$ . Ker pa za  $u, v \in V_i$  velja  $y_{uv} \geq i + 1$ , sledi  $d_G(u, v) \geq i + 1$ , kar ustreza pogojem naloge.

### Naloga 2.36.

Za vozlišče  $u \in V$  in  $i$ -to barvo ( $1 \leq i \leq k$ ) bomo uvedli spremenljivko  $x_{ui}$ , katere vrednost interpretiramo kot

$$x_{ui} = \begin{cases} 1; & \text{vozlišče } u \text{ pobarvamo z barvo } i, \text{ in} \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo omejitve za celoštevilski linearni program. Ker nas zanima le obstoj dopustne rešitve, ciljna funkcija ni pomembna.

$$\forall u \in V \forall i \in \{1, \dots, k\} : \quad 0 \leq x_{ui} \leq 1, \quad x_{ui} \in \mathbb{Z}$$

Vsako vozlišče pobarvamo z eno barvo:

$$\forall u \in V : \quad \sum_{i=1}^k x_{ui} = 1$$

Sosedni vozlišči nimata iste barve:

$$\forall uv \in E \forall i \in \{1, \dots, k\} : \quad x_{ui} + x_{vi} \leq 1$$

Število vozlišč dveh različnih barv se razlikuje za največ 1:

$$\forall i, j \in \{1, \dots, k\}, i < j : -1 \leq \sum_{u \in V} (x_{ui} - x_{uj}) \leq 1$$

Naj bo  $n$  število vozlišč in  $m$  število povezav grafa  $G$ . Potem ima zgornji celoštevilski linearni program  $nk$  celoštevilskih spremenljivk in  $2nk + n + mk + k(k-1)$  pogojev.

### Naloga 2.37.

Opazimo, da želimo imeti čim nižje vrednosti  $f(x), g(y), h(z)$ , zato bo zadostovalo, da za ustrezne spremenljivke  $f, g, h$  podamo spodnje meje. Prav tako opazimo, da bo za optimalno rešitev zagotovo veljalo  $x, y, z \leq 100$ . Tako bomo uvedli še spremenljivke  $t, u, v, w$ , katerih vrednosti interpretiramo kot

$$t = \begin{cases} 1; & z > 0, \\ 0; & z = 0; \end{cases} \quad u = \begin{cases} 1; & x \geq 5, \\ 0; & x < 5; \end{cases}$$

$$v = \begin{cases} 1; & y \geq 5, \\ 0; & y < 5; \end{cases} \quad \text{ter} \quad w = \begin{cases} 1; & z \geq 5, \\ 0; & z < 5. \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\begin{array}{ll} \min f + g + h & \text{p.p.} \\ f, g, h, x, y, z \geq 0, & f, g, h, x, y, z \in \mathbb{Z} \\ 0 \leq t, u, v, w \leq 1, & t, u, v, w \in \mathbb{Z} \\ f = 10 + 5x & \\ g \geq 10 & \\ g \geq 5 + 5y & \\ z \leq 100t & \\ h \geq 8 + 2t + 4z & \\ x + y + z \geq 100 & \\ x \geq 5u & \\ y \geq 5v & \\ z \geq 5w & \\ u + v + w \geq 2 & \end{array}$$

**Naloga 2.38.**

Opazimo, da želimo imeti čim nižji vrednosti  $f_1(x_1)$  in  $f_2(x_2)$ , zato bo zadostovalo, da za ustrezni spremenljivki  $f_1$  in  $f_2$  podamo spodnje meje. Prav tako opazimo, da bo za optimalno rešitev zagotovo veljalo  $x_1 \leq 100$  in  $x_2 \leq \frac{200}{3}$ . Tako bomo uvedli še spremenljivki  $y_i$  ( $i = 1, 2$ ), katerih vrednosti interpretiramo kot

$$y_i = \begin{cases} 1; & x_i > 0, \\ 0; & x_i = 0. \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\begin{array}{ll} \min f_1 + f_2 & \text{p.p.} \\ 0 \leq y_1, y_2 \leq 1, & y_1, y_2 \in \mathbb{Z} \\ f_1, f_2, x_1, x_2 \geq 0 & \\ x_1 \leq 100y_1 & \\ 3x_2 \leq 200y_2 & \\ f_1 \geq 5x_1 + 10y_1 & \\ f_2 \geq 4x_2 + 20y_2 & \\ 100x_1 + 150x_2 \geq 10000 & \end{array}$$

**Naloga 2.39.**

Za  $i, j, k \in \{1, 2, 3, 4\}$  bomo uvedli spremenljivko  $x_{ijk}$ , katere vrednost interpretiramo kot

$$x_{ijk} = \begin{cases} 1; & \text{na polje } (i, j) \text{ vpišemo število } k, \text{ in} \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo omejitve za celoštevilski linearni program. Zanima nas samo njegova dopustnost, tako da ciljna funkcija ni pomembna.

$$\forall i, j, k \in \{1, 2, 3, 4\} : \quad 0 \leq x_{ijk} \leq 1, \quad x_{ijk} \in \mathbb{Z}$$

V vsako polje vpišemo eno število:

$$\forall i, j \in \{1, 2, 3, 4\} : \quad \sum_{k=1}^4 x_{ijk} = 1$$

Vsako število se pojavi enkrat v vsaki vrstici in stolpcu:

$$\forall i, k \in \{1, 2, 3, 4\} : \quad \sum_{j=1}^4 x_{ijk} = 1$$

$$\forall j, k \in \{1, 2, 3, 4\} : \quad \sum_{i=1}^4 x_{ijk} = 1$$

Vsako število se pojavi enkrat v vsakem kvadratu  $2 \times 2$ :

$$\forall i, j \in \{0, 2\} \quad \forall k \in \{1, 2, 3, 4\} : \quad \sum_{i'=1}^2 \sum_{j'=1}^2 x_{i+i', j+j', k} = 1$$

Že vpisana števila:

$$x_{111} = x_{122} = x_{232} = x_{241} = x_{324} = x_{413} = 1$$

Opisani celoštevilski linearni program ima  $4^3 = 64$  celoštevilskih spremenljivk in  $2 \cdot 4^3 + 4 \cdot 4^2 + 6 = 198$  pogojev.

**Naloga 2.40.**

- (a) Naj bo  $x_i$  število žetonov, ki jih vložimo v dvoboj z  $i$ -tim nasprotnikom ( $1 \leq i \leq n$ ). Poleg tega bomo uvedli še spremenljivke  $y_i$  in  $z_i$  ( $1 \leq i \leq n$ ), katerih vrednosti interpretiramo kot

$$y_i = \begin{cases} 1, & \text{če ne izgubimo proti } i\text{-temu nasprotniku, in} \\ 0 & \text{sicer;} \end{cases}$$

ter

$$z_i = \begin{cases} 1, & \text{če premagamo } i\text{-tega nasprotnika, in} \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\begin{aligned} & \max \sum_{i=1}^n (y_i + 2z_i) \quad \text{p.p.} \\ \forall i \in \{1, \dots, n\} : & \quad x_i \geq 0, \quad x_i \in \mathbb{Z} \\ \forall i \in \{1, \dots, n\} : & \quad 0 \leq y_i \leq 1, \quad y_i \in \mathbb{Z} \\ \forall i \in \{1, \dots, n\} : & \quad 0 \leq z_i \leq 1, \quad z_i \in \mathbb{Z} \end{aligned}$$

Porabimo  $M$  žetonov:

$$\sum_{i=1}^n x_i = M$$

$y_i = 1$  natanko tedaj, ko  $x_i \geq c_i$ :

$$\forall i \in \{1, \dots, n\} : \quad c_i y_i \leq x_i \leq c_i - 1 + (M + 1)y_i$$

$z_i = 1$  natanko tedaj, ko  $x_i > c_i$ :

$$\forall i \in \{1, \dots, n\} : (c_i + 1)z_i \leq x_i \leq c_i + Mz_i$$

(b) Zapišimo še dodatne omejitve.

Proti nasprotnikom iz  $A$  ne igramo izenačeno:

$$\forall i \in A : \quad y_i \leq z_i$$

Največ dva poraza proti nasprotnikom iz  $B$ :

$$\sum_{i \in B} y_i \geq |B| - 2$$

Če izgubimo proti  $u$ , dosežemo vsaj 4 točke proti  $v$  in  $w$ :

$$4y_u + y_v + y_w + 2z_v + 2z_w \geq 4$$

Največ  $k$  dvobojev z več kot  $t$  žetoni:

$$\begin{aligned} \forall i \in \{1, \dots, n\} : & \quad 0 \leq r_i \leq 1, \quad r_i \in \mathbb{Z} \\ \forall i \in \{1, \dots, n\} : & \quad x_i \leq t + Mr_i \\ & \quad \sum_{i=1}^n r_i \leq k \end{aligned}$$

Vsaj  $\ell$  porazov:

$$\sum_{i=1}^n y_i \leq n - \ell$$

**Naloga 2.41.**

Za  $i$ -to kolekcijo ( $1 \leq i \leq n$ ) bomo uvedli spremenljivki  $x_i$  in  $y_i$ , kjer je  $x_i$  število kupljenih izvodov  $i$ -te kolekcije, vrednost  $y_i$  pa interpretiramo kot

$$y_i = \begin{cases} 1, & \text{kupimo vsaj en izvod } i\text{-te kolekcije, in} \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\max \sum_{i=1}^n ((d_i - c_i)x_i + e_i y_i) \quad \text{p.p.}$$

Omejitev števila izvodov:

$$\forall i \in \{1, \dots, n\}: \quad 0 \leq x_i \leq k_i, \quad x_i \in \mathbb{Z}$$

$$\forall i \in \{1, \dots, n\}: \quad 0 \leq y_i \leq 1, \quad y_i \in \mathbb{Z}$$

Povezava med  $x_i$  in  $y_i$ :

$$\forall i \in \{1, \dots, n\}: \quad y_i \leq x_i \leq k_i y_i$$

Omejitev kapitala za nakup:

$$\sum_{i=1}^n c_i x_i \leq C$$

Omejitev števila proizvajalcev:

$$\sum_{i=1}^n y_i \leq K$$

Omejitev proizvajalca  $p$ :

$$x_q + x_r - x_p \leq (k_q + k_r)(1 - y_p)$$

**Naloga 2.42.**

Za  $h$ -ti dan ( $1 \leq h \leq m$ ),  $i$ -to skupino ( $1 \leq i \leq n$ ) in  $j$ -ti termin ( $1 \leq j \leq k$ ) bomo uvedli spremenljivko  $x_{hij}$ , katere vrednost interpretiramo kot

$$x_{hij} = \begin{cases} 1, & i\text{-ta skupina nastopi } j\text{-ta v } h\text{-tem dnevu festivala, in} \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo celoštevilski linearni program.

$$\max \sum_{h=1}^m \sum_{i=1}^n \sum_{j=1}^k a_{ij} x_{hij} \quad \text{p.p.}$$

$$\forall h \in \{1, \dots, m\} \quad \forall i \in \{1, \dots, n\} \quad \forall j \in \{1, \dots, k\}: \quad 0 \leq x_{hij} \leq 1, \quad x_{hij} \in \mathbb{Z}$$



Vsaka skupina nastopi največ enkrat:

$$\forall i \in \{1, \dots, n\} : \\ \sum_{h=1}^m \sum_{j=1}^k x_{hij} \leq 1$$

Naenkrat nastopa ena skupina:

$$\forall h \in \{1, \dots, m\} \forall j \in \{1, \dots, k\} : \\ \sum_{i=1}^n x_{hij} = 1$$

Omejitev plačila:

$$\sum_{h=1}^m \sum_{i=1}^n \sum_{j=1}^k m_i x_{hij} \leq M$$

Vrstni red v dnevu:

$$\forall h \in \{1, \dots, m\} \forall j \in \{2, \dots, k\} : \\ \sum_{i=1}^n m_i (x_{hij} - x_{h,i,j-1}) \geq 0$$

Razporeditev po dnevih:

$$\forall h \in \{2, \dots, n\} : \\ \sum_{i=1}^n \sum_{j=1}^k a_{ij} (x_{hij} - x_{h-1,i,j}) \geq 0$$

$r$  ne nastopa, če nastopa  $s$ :

$$\sum_{h=1}^m \sum_{j=1}^k (x_{hrj} + x_{hsj}) \leq 1$$

$t$  ne nastopa, če ni zadnja ali pred  $u$ :

$$\forall h \in \{1, \dots, m\} : \\ \sum_{j=1}^{k-1} x_{htj} \leq \sum_{j=2}^k j x_{huj} - \sum_{j=1}^{k-1} j x_{htj} \leq k - (k-1) \sum_{j=1}^{k-1} x_{htj}$$

$v$  in  $w$  ne nastopata na isti dan:

$$\forall h \in \{1, \dots, m\} : \\ \sum_{j=1}^k (x_{hvj} + x_{hwj}) \leq 1$$

### Naloga 2.43.

Za  $h$ -ti center ( $1 \leq h \leq m$ ) ter  $i$ -tega in  $j$ -tega strokovnjaka ( $1 \leq i, j \leq n$ ) bomo uvedli spremenljivko  $x_{hij}$ , katere vrednost interpretiramo kot

$$x_{hij} = \begin{cases} 1, & i\text{-ti in } j\text{-ti strokovnjak sta oba nastanjena} \\ & \text{v } h\text{-tem karantenskem centru, in} \\ 0 & \text{sicer.} \end{cases}$$

Informacija o tem, ali je  $i$ -ti strokovnjak nastanjen v  $h$ -tem centru, hrani spremenljivka  $x_{hii}$  ( $1 \leq h \leq m$ ,  $1 \leq i \leq n$ ).

Zapišimo celoštevilski linearni program.

$$\min \sum_{h=1}^m \sum_{i=1}^n \sum_{j=1}^n p_{ij} x_{hij} \quad \text{p.p.}$$

$$\forall h \in \{1, \dots, m\} \forall i, j \in \{1, \dots, n\}: \quad 0 \leq x_{hij} \leq 1, \quad x_{hij} \in \mathbb{Z}$$

Vsak strokovnjak je nastanjen v natanko enem centru:

$$\forall i \in \{1, \dots, n\}: \quad \sum_{h=1}^m x_{hii} = 1$$

Kapacitete centrov:

$$\forall h \in \{1, \dots, m\}: \quad \sum_{i=1}^n x_{hii} \leq k_h$$

Povezava med  $x_{hii}$  in  $x_{hij}$ :

$$\forall h \in \{1, \dots, m\} \forall i, j \in \{1, \dots, n\}: 2x_{hij} \leq x_{hii} + x_{hjj} \leq x_{hij} + 1$$

Območja z mutacijo:

$$\forall i \in A \forall j \notin A: \quad \sum_{h=1}^m x_{hij} = 0$$

Sorodniki:

$$\forall (i, j) \in B: \quad \sum_{h=1}^m x_{hij} = 1$$

### Naloga 2.44.

Za zaposlenega  $z \in Z$  in  $1 \leq i \leq n$ ,  $1 \leq j \leq m$  bomo uvedli spremenljivko  $x_{zij}$ , katere vrednost interpretiramo kot

$$x_{zij} = \begin{cases} 1 & \text{zaposlenemu } z \text{ kupijo licenco za različico } i \text{ po ceni } c_{ij}, \text{ in} \\ 0 & \text{sicer.} \end{cases}$$

(a) Zapišimo celoštevilski linearni program.

$$\min \sum_{z \in Z} \sum_{i=1}^n \sum_{j=1}^m c_{ij} x_{zij} \quad \text{p.p.}$$

$$\forall z \in Z \forall i \in \{1, \dots, n\} \forall j \in \{1, \dots, m\}: \quad 0 \leq x_{zij} \leq 1, \quad x_{zij} \in \mathbb{Z}$$

Vsak zaposleni dobi natanko eno licenco:

$$\forall z \in Z: \quad \sum_{i=1}^n \sum_{j=1}^m x_{zij} = 1$$

Vsak zaposleni ima na voljo vse funkcionalnosti, ki jih potrebuje:

$$\forall z \in Z \forall f \in F \forall i \in \{1, \dots, n\}: \quad \sum_{j=1}^m x_{zij} \leq q_{fi} - p_{zf} + 1$$

Ustrezna cena glede na število licenc:

$$\forall z \in Z \forall i \in \{1, \dots, n\} \forall j \in \{1, \dots, m\}: \sum_{y \in Z} x_{yij} \geq r_j x_{zij}$$

(b) Zapišimo še dodatne omejitve.

$a$  in  $b$  dobita enako licenco:

$$\forall i \in \{1, \dots, n\}: \sum_{j=1}^m (x_{aij} - x_{bij}) = 0$$

Direktor ima na voljo vsako funkcionalnost, ki jo imajo na voljo ostali zaposleni:

$$\forall z \in Z \forall f \in F \forall h, i \in \{1, \dots, n\} \forall j, k \in \{1, \dots, m\}: x_{zij} + x_{dhk} \leq q_{fh} - q_{fi} + 2$$

Če ima kdorkoli funkcionalnost  $e$ , potem jo imajo vsi:

$$\forall y, z \in Z \forall h, i \in \{1, \dots, n\} \forall j, k \in \{1, \dots, m\}: x_{zij} + x_{yhk} \leq q_{eh} - q_{ei} + 2$$

### Naloga 2.45.

Za artikel  $i \in A$  in  $j$ -ti kupon ( $1 \leq j \leq k$ ) bomo uvedli spremenljivke  $x_i$ ,  $y_j$  in  $z_{ij}$ , kjer je  $x_i$  število kosov artikla  $i$ , ki naj jih Igor kupi,  $y_j$  predstavlja "pokritost"  $j$ -tega kupona (tj., koliko naborov artiklov iz  $K_j$  kupi), vrednost  $z_{ij}$  pa interpretiramo kot

$$z_{ij} = \begin{cases} 1, & \text{Igor uporabi } j\text{-ti kupon za artikel } i \in A, \text{ in} \\ 0 & \text{sicer.} \end{cases}$$

Poleg tega definirajmo še konstanto  $N = \max\{n_h \mid 1 \leq h \leq m\}$ . Zapišimo celoštevilski linearni program.

$$\begin{aligned} \min \quad & \sum_{i \in A} c_i x_i - \sum_{j=1}^k p_j y_j \quad \text{p.p.} \\ \forall i \in A: \quad & x_i \geq 0, \quad x_i \in \mathbb{Z} \\ \forall j \in \{1, \dots, k\}: \quad & y_j \geq 0, \quad y_j \in \mathbb{Z} \\ \forall i \in A \forall j \in \{1, \dots, k\}: \quad & 0 \leq z_{ij} \leq 1, \quad z_{ij} \in \mathbb{Z} \end{aligned}$$

Zadostimo potrebam:

$$\forall h \in \{1, \dots, m\}: \sum_{i \in B_h} x_i \geq n_h$$

Kupon  $j$  pokrijemo največ tolikokrat, kolikor kupimo posameznega artikla iz  $K_j$ :

$$\forall j \in \{1, \dots, k\} \forall i \in K_j: y_j \leq x_i$$

Če uporabimo  $j$ -ti kupon za  $i \in K_j$ , naj bo  $z_{ij} = 1$ :

$$\forall j \in \{1, \dots, k\} \forall i \in K_j: y_j \leq N z_{ij}$$

Za vsak izdelek uporabimo največ en kupon:

$$\forall i \in A: \sum_{j=1}^k z_{ij} \leq 1$$

**Naloga 2.46.**

Za  $i$ -tega vlagatelja in  $j$ -to zagonsko podjetje ( $1 \leq i \leq m$ ,  $1 \leq j \leq n$ ) bomo uvedli spremenljivko  $x_{ij}$ , katere vrednost interpretiramo kot

$$x_{ij} = \begin{cases} 1 & i\text{-ti vlagatelj kupi delež v } j\text{-tem podjetju, in} \\ 0 & \text{sicer.} \end{cases}$$

(a) Zapišimo celoštevilski linearni program.

$$\begin{array}{ll} \max & t \\ \forall i \in \{1, \dots, m\} \forall j \in \{1, \dots, n\} : & 0 \leq x_{ij} \leq 1, \quad x_{ij} \in \mathbb{Z} \end{array} \quad \text{p.p.}$$

Omejitev kapitala za vsakega vlagatelja:

$$\forall i \in \{1, \dots, m\} : \sum_{j=1}^n c_{ij} x_{ij} \leq k_i$$

Omejitev deleža, ki ga proda vsak lastnik:

$$\forall j \in \{1, \dots, n\} : \sum_{i=1}^m p_{ij} x_{ij} \leq q_j$$

Spodnja meja za število podjetij, v katera vложи vsak vlagatelj:

$$\forall i \in \{1, \dots, m\} : \sum_{j=1}^n x_{ij} \geq t$$

(b) Za  $j$ -to podjetje ( $1 \leq j \leq n$ ) bomo uvedli še spremenljivko  $y_j$ , katere vrednost interpretiramo kot

$$y_j = \begin{cases} 1 & \text{vsaj en vlagatelj kupi delež v } j\text{-tem podjetju, in} \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo še dodatne omejitve.

Nobeno podjetje ne proda deleža pod ceno:

$$\forall j \in \{1, \dots, n\} : v_j \sum_{i=1}^m p_{ij} x_{ij} \leq \sum_{i=1}^m c_{ij} x_{ij}$$

Lastnik vsakega podjetja prejme želeni znesek, ali pa ne proda ničesar:

$$\begin{array}{ll} \forall j \in \{1, \dots, n\} : & \sum_{i=1}^m c_{ij} x_{ij} \geq u_j y_j \\ \forall i \in \{1, \dots, m\} \forall j \in \{1, \dots, n\} : & y_j \geq x_{ij} \end{array}$$

## 2.3 Teorija odločanja

### Naloga 3.1.

Če predpostavimo, da je kovanec pošten, je potem pričakovani dobiček ob metu kovanca enak

$$\frac{1}{2} \cdot 250\,000\text{€} - \frac{1}{2} \cdot 100\,000\text{€} = 75\,000\text{€}.$$

Ta vrednost je večja od ničelnega dobička, če ponudbe ne sprejmemo, zato se nam izplača ponudbo sprejeti. Seveda gre pri tej odločitvi tudi za oceno, kolikšno tveganje smo pripravljeni sprejeti – ali si lahko privoščimo izgubiti 100 000 €, če ne bomo imeli sreče?

### Naloga 3.2.

Naj bo  $x$  število kupljenih žemelj,  $k$  pa število prodanih žemelj. Če je  $k \leq x$ , imamo dobiček

$$k \cdot 0.2\text{€} - (x - k) \cdot 0.05\text{€} = k \cdot 0.25\text{€} - x \cdot 0.05\text{€}.$$

Če pa je  $k \geq x$ , je dobiček enak

$$x \cdot 0.2\text{€} + (k - x) \cdot 0.1\text{€} = k \cdot 0.1\text{€} + x \cdot 0.1\text{€}.$$

Pri  $x = k$  obe formuli dasta dobiček  $k \cdot 0.2\text{€}$ . Izračunajmo pričakovani dobiček pri naročilu  $x$  žemelj za različne intervale:

$$\begin{aligned} x \leq 50 : & x \cdot 0.1\text{€} + (0.1 \cdot 50 + 0.15 \cdot 60 + 0.3 \cdot 70 + \\ & 0.2 \cdot 80 + 0.15 \cdot 90 + 0.1 \cdot 100) \cdot 0.1\text{€} \\ & = x \cdot 0.1\text{€} + 7.45\text{€} \end{aligned}$$

$$\begin{aligned} 50 \leq x \leq 60 : & x \cdot (0.9 \cdot 0.1\text{€} - 0.1 \cdot 0.05\text{€}) + 0.1 \cdot 50 \cdot 0.25\text{€} + \\ & (0.15 \cdot 60 + 0.3 \cdot 70 + 0.2 \cdot 80 + 0.15 \cdot 90 + 0.1 \cdot 100) \cdot 0.1\text{€} \\ & = x \cdot 0.085\text{€} + 8.2\text{€} \end{aligned}$$

$$\begin{aligned} 60 \leq x \leq 70 : & x \cdot (0.75 \cdot 0.1\text{€} - 0.25 \cdot 0.05\text{€}) + (0.1 \cdot 50 + 0.15 \cdot 60) \cdot 0.25\text{€} + \\ & (0.3 \cdot 70 + 0.2 \cdot 80 + 0.15 \cdot 90 + 0.1 \cdot 100) \cdot 0.1\text{€} \\ & = x \cdot 0.0625\text{€} + 9.55\text{€} \end{aligned}$$

$$\begin{aligned} 70 \leq x \leq 80 : & x \cdot (0.45 \cdot 0.1\text{€} - 0.55 \cdot 0.05\text{€}) + (0.1 \cdot 50 + 0.15 \cdot 60 + \\ & 0.3 \cdot 70) \cdot 0.25\text{€} + (0.2 \cdot 80 + 0.15 \cdot 90 + 0.1 \cdot 100) \cdot 0.1\text{€} \\ & = x \cdot 0.0175\text{€} + 12.7\text{€} \end{aligned}$$

$$\begin{aligned} 80 \leq x \leq 90 : & x \cdot (0.25 \cdot 0.1\text{€} - 0.75 \cdot 0.05\text{€}) + (0.1 \cdot 50 + 0.15 \cdot 60 + 0.3 \cdot 70 + \\ & 0.2 \cdot 80) \cdot 0.25\text{€} + (0.15 \cdot 90 + 0.1 \cdot 100) \cdot 0.1\text{€} \\ & = -x \cdot 0.0125\text{€} + 15.1\text{€} \end{aligned}$$

$$\begin{aligned} 90 \leq x \leq 100 : & x \cdot (0.1 \cdot 0.1\text{€} - 0.9 \cdot 0.05\text{€}) + (0.1 \cdot 50 + 0.15 \cdot 60 + 0.3 \cdot 70 + \\ & 0.2 \cdot 80 + 0.15 \cdot 90) \cdot 0.25\text{€} + 0.1 \cdot 100 \cdot 0.1\text{€} \end{aligned}$$

$$\begin{aligned}
&= -x \cdot 0.035\text{€} + 17.125\text{€} \\
x \geq 100: &-x \cdot 0.05\text{€} + (0.1 \cdot 50 + 0.15 \cdot 60 + 0.3 \cdot 70 + \\
&0.2 \cdot 80 + 0.15 \cdot 90 + 0.1 \cdot 100) \cdot 0.25\text{€} \\
&= -x \cdot 0.05\text{€} + 18.625\text{€}
\end{aligned}$$

Funkcija pričakovanega dobička je torej zvezna in odsekoma linearna v  $x$ , pri čemer je smerni koeficient pozitiven pri  $x < 80$  in negativen pri  $x > 80$ . Pričakovan dobiček torej maksimiziramo pri naročilu 80 žemelj – tedaj ta znaša

$$80 \cdot 0.0175\text{€} + 12.7\text{€} = 14.1\text{€}.$$

### Naloga 3.3.

- (a) Ob opravljeni operaciji je pričakovana življenjska doba (v mesecih) enaka

$$0.7 \cdot 12 + 0.3 \cdot 0 = 8.4.$$

Ker je to več od 3 mesecev brez operacije, se zanjo pacient odloči.

- (b) Zanesljivost 0.9 razumemo tako, da test z verjetnostjo 0.9 pravilno napove vsak izid operacije, torej

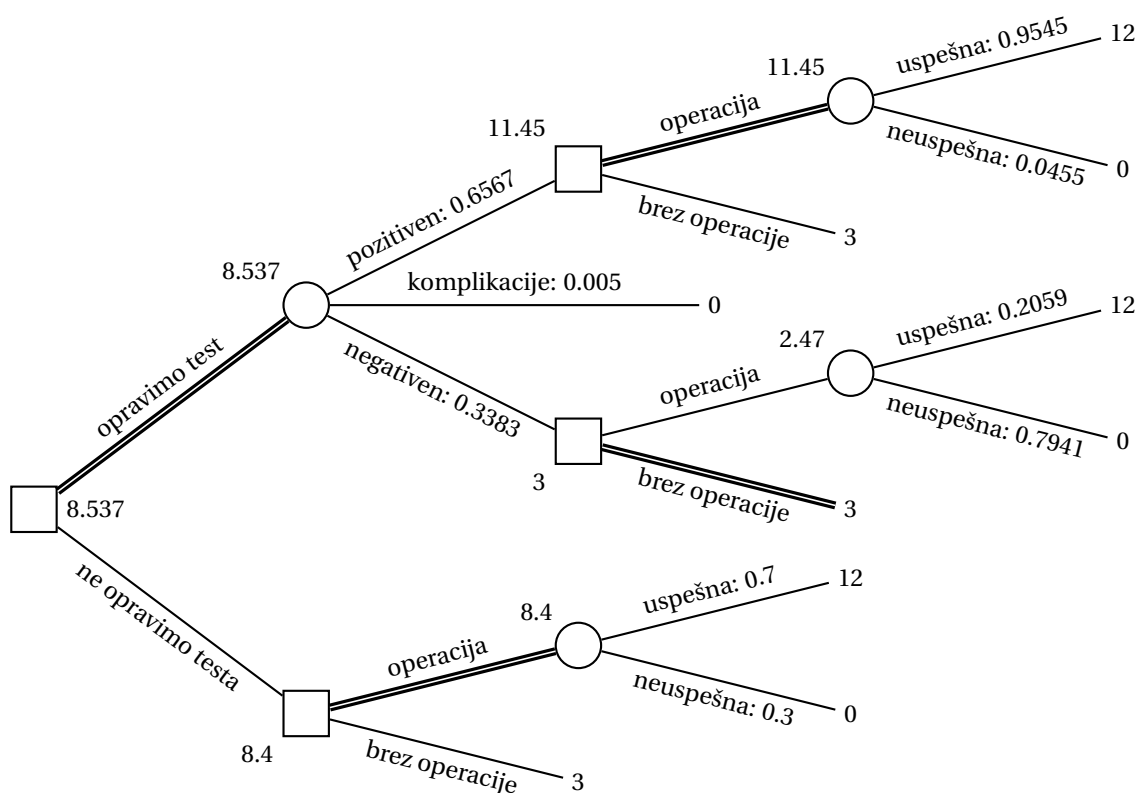
$$\begin{aligned}
P(\text{test napove uspeh} \mid \text{operacija je uspešna}) &= 0.9 \quad \text{in} \\
P(\text{test napove uspeh} \mid \text{operacija je neuspešna}) &= 0.1.
\end{aligned}$$

Izračunajmo še verjetnosti napovedi testa ter uspešnosti operacije ob vsaki napovedi.

$$\begin{aligned}
P(\text{test napove uspeh}) &= 0.7 \cdot 0.9 + 0.3 \cdot 0.1 = 0.66 \\
P(\text{test napove neuspeh}) &= 0.7 \cdot 0.1 + 0.3 \cdot 0.9 = 0.34 \\
P(\text{operacija je uspešna} \mid \text{test napove uspeh}) &= \frac{0.7 \cdot 0.9}{0.66} = \frac{21}{22} \approx 0.9545 \\
P(\text{operacija je neuspešna} \mid \text{test napove uspeh}) &= \frac{0.3 \cdot 0.1}{0.66} = \frac{1}{22} \approx 0.0455 \\
P(\text{operacija je uspešna} \mid \text{test napove neuspeh}) &= \frac{0.7 \cdot 0.1}{0.34} = \frac{7}{34} \approx 0.2059 \\
P(\text{operacija je uspešna} \mid \text{test napove neuspeh}) &= \frac{0.3 \cdot 0.9}{0.34} = \frac{27}{34} \approx 0.7941
\end{aligned}$$

Ob upoštevanju, da se bo test končal brez komplikacij z verjetnostjo 0.995, sta verjetnosti pozitivnega in negativnega izida testa brez komplikacij enaki 0.6567 oziroma 0.3383.

S pomočjo zgoraj izračunanih verjetnosti lahko narišemo odločitveno drevo s slike 53. Izračunamo lahko, da je ob opravljanju testa pričakovana življenjska doba 8.537 mesecev, kar je več kot 8.4 mesecev brez testa, zato se odločimo za test. Če test napove uspeh, se odločimo za operacijo, sicer pa ne.



Slika 53: Odločitveno drevo za nalogo 3.3.

### Naloga 3.4.

Če se podjetje odloči za samostojno prodajo, bo ob uspehu imelo  $100\,000 \cdot 600\text{€} - 6\text{M€} = 54\text{M€}$  dobička, ob neuspehu pa  $10\,000 \cdot 600\text{€} - 6\text{M€} = 0\text{M€}$ . Ker je pričakovani dobiček enak

$$\frac{1}{2} \cdot 54\text{M€} + \frac{1}{2} \cdot 0\text{M€} = 27\text{M€},$$

kar je več kot 15 M€, kolikor bi zaslužili ob prodaji konkurenci, se podjetju (v odsotnosti raziskave) bolj izplača samostojna prodaja.

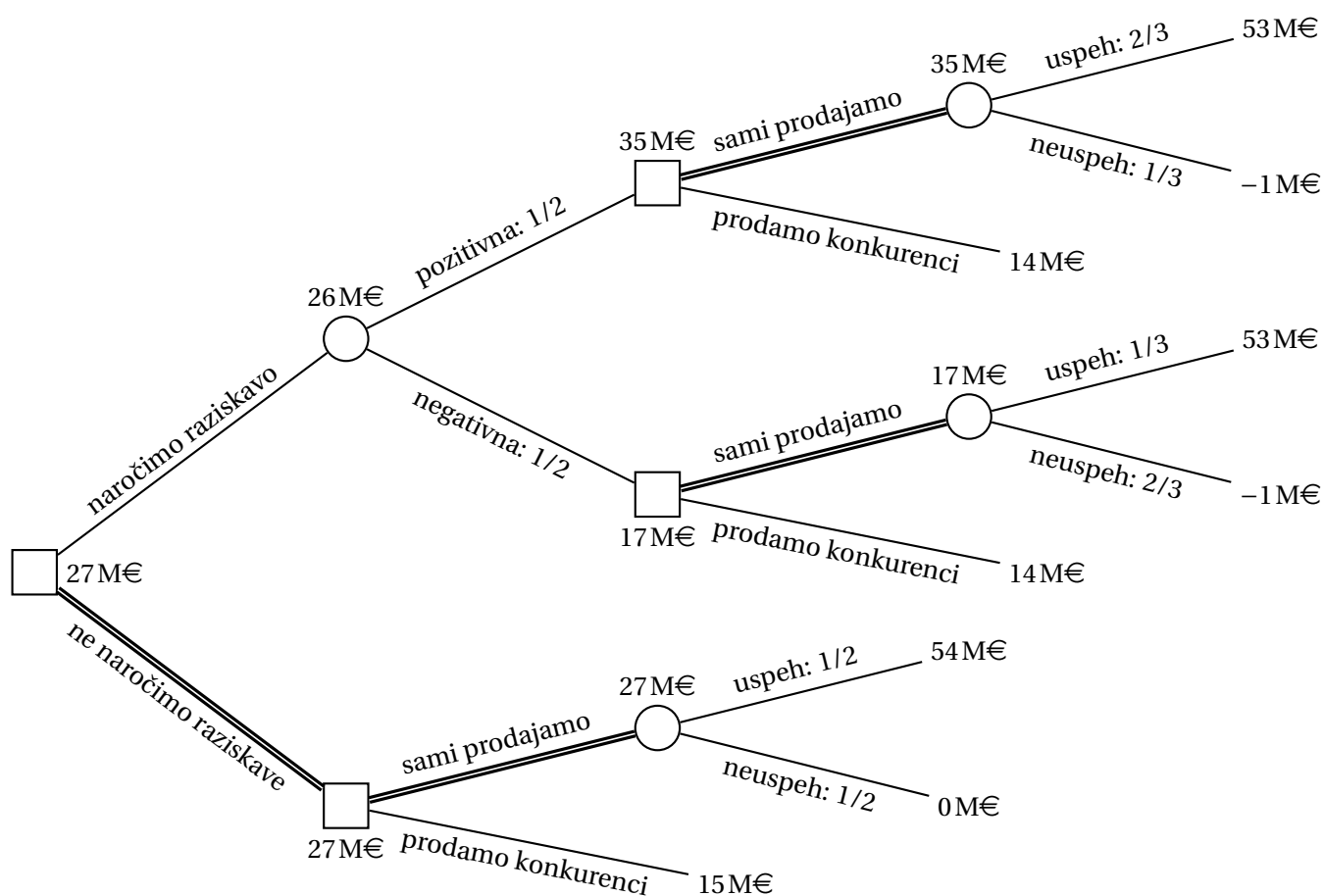
Zanesljivost raziskave je tako v primeru uspeha kot neuspeha enaka 2/3, torej

$$P(\text{raziskava napove uspeh} \mid \text{investicija uspe}) = \frac{2}{3}$$

$$P(\text{raziskava napove uspeh} \mid \text{investicija ne uspe}) = \frac{1}{3}$$

Izračunajmo še verjetnosti napovedi raziskave ter uspešnosti investicije ob vsaki napovedi.

$$P(\text{raziskava napove uspeh}) = \frac{2}{3} \cdot \frac{1}{2} + \frac{1}{3} \cdot \frac{1}{2} = \frac{1}{2}$$



Slika 54: Odločitveno drevo za nalogo 3.4.

$$\begin{aligned}
 P(\text{raziskava napove neuspeh}) &= \frac{1}{3} \cdot \frac{1}{2} + \frac{2}{3} \cdot \frac{1}{2} = \frac{1}{2} \\
 P(\text{investicija uspe} \mid \text{raziskava napove uspeh}) &= \frac{\frac{2}{3} \cdot \frac{1}{2}}{\frac{1}{2}} = \frac{2}{3} \\
 P(\text{investicija ne uspe} \mid \text{raziskava napove uspeh}) &= \frac{\frac{1}{3} \cdot \frac{1}{2}}{\frac{1}{2}} = \frac{1}{3} \\
 P(\text{investicija uspe} \mid \text{raziskava napove neuspeh}) &= \frac{\frac{1}{3} \cdot \frac{1}{2}}{\frac{1}{2}} = \frac{1}{3} \\
 P(\text{investicija ne uspe} \mid \text{raziskava napove neuspeh}) &= \frac{\frac{2}{3} \cdot \frac{1}{2}}{\frac{1}{2}} = \frac{2}{3}
 \end{aligned}$$

S pomočjo zgoraj izračunanih verjetnosti lahko narišemo odločitveno drevo s slike 54. Opazimo, da rezultat raziskave ne vpliva na našo odločitev o prodaji, zato se za raziskavo ne odločimo in tako pričakujemo dobiček 27M€.



**Naloga 3.5.**

Naj bo  $x_n$  pričakovani dobiček pri prodanih  $n$  kartah. Izračunajmo  $x_n$  pri  $n \in \{100, 101, 102, 103\}$ :

$$x_{100} = 100 \cdot 10\text{€} = 1\,000\text{€}$$

$$x_{101} = 101 \cdot 10\text{€} - 0.2 \cdot 30\text{€} = 1\,004\text{€}$$

$$x_{102} = 102 \cdot 10\text{€} - 0.2 \cdot 60\text{€} - 0.3 \cdot 30\text{€} = 999\text{€}$$

$$x_{103} = 103 \cdot 10\text{€} - 0.2 \cdot 90\text{€} - 0.3 \cdot 60\text{€} - 0.4 \cdot 30\text{€} = 982\text{€}$$

Organizator bo torej maksimiziral pričakovani dobiček, če proda 101 karto.

**Naloga 3.6.**

- (a) Če se Rexhep odloči za najem, bo ob uspehu imel  $12\,000\text{€} - 6\,000\text{€} = 6\,000\text{€}$  dobička, ob neuspehu pa bo ta enak  $3\,000\text{€} - 6\,000\text{€} = -3\,000\text{€}$  (tj.,  $3\,000\text{€}$  izgube). Ker je pričakovani dobiček enak

$$\frac{2}{3} \cdot 6\,000\text{€} - \frac{1}{3} \cdot 3\,000\text{€} = 3\,000\text{€},$$

se torej odloči za najem. Proces odločanja je prikazan v spodnji veji odločitvenega drevesa na sliki 55.

- (b) Izračunajmo verjetnosti za Seadovo mnenje ter kakovost parcele v odvisnosti od njega.

$$P(\text{Sead priporoča}) = \frac{2}{3} \cdot \frac{2}{3} + \frac{1}{3} \cdot \frac{1}{2} = \frac{11}{18}$$

$$P(\text{Sead odsvetuje}) = \frac{2}{3} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{1}{2} = \frac{7}{18}$$

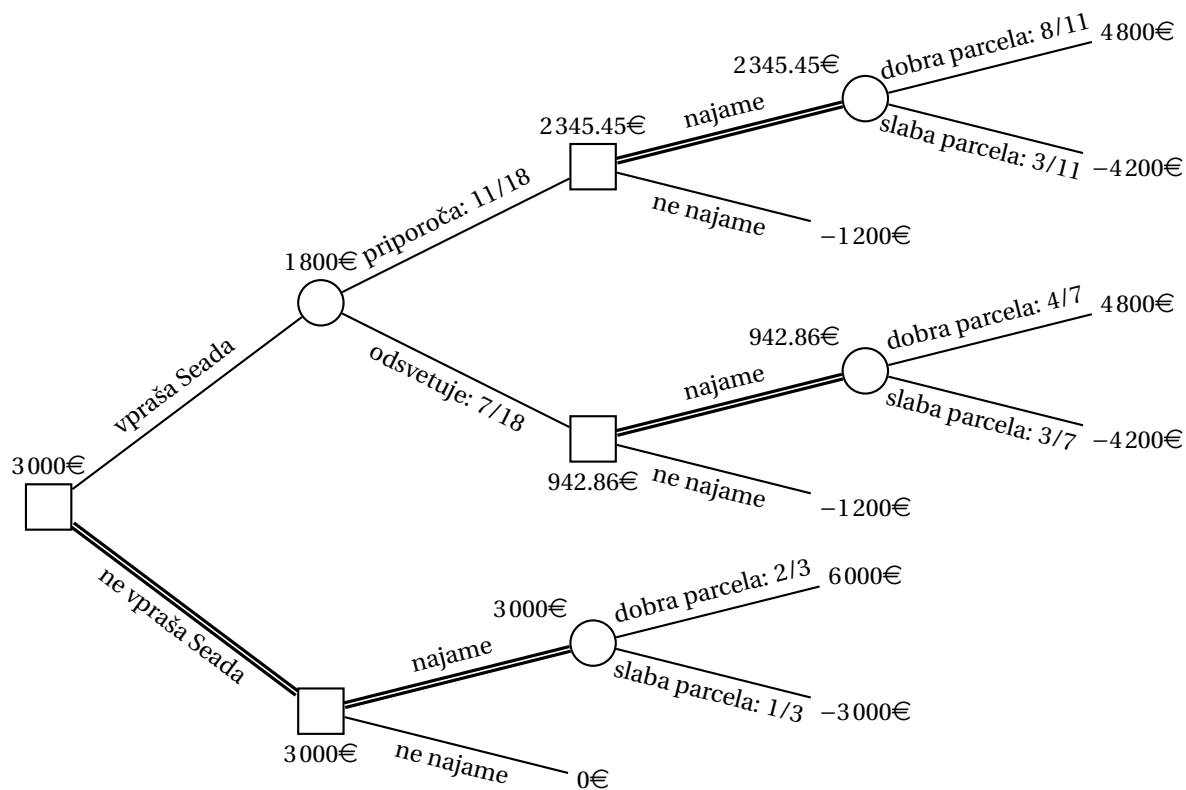
$$P(\text{dobra parcela} \mid \text{Sead priporoča}) = \frac{2/3 \cdot 2/3}{11/18} = \frac{8}{11}$$

$$P(\text{slaba parcela} \mid \text{Sead priporoča}) = \frac{1/3 \cdot 1/2}{11/18} = \frac{3}{11}$$

$$P(\text{dobra parcela} \mid \text{Sead odsvetuje}) = \frac{2/3 \cdot 1/3}{7/18} = \frac{4}{7}$$

$$P(\text{slaba parcela} \mid \text{Sead odsvetuje}) = \frac{1/3 \cdot 1/2}{7/18} = \frac{3}{7}$$

S pomočjo zgoraj izračunanih verjetnosti lahko narišemo odločitveno drevo s slike 55. Opazimo, da Seadovo mnenje ne vpliva na našo odločitev o prodaji, zato naj ga Rexhep ne vpraša in tako pričakuje dobiček  $3\,000\text{€}$ .



Slika 55: Odločitveno drevo za nalogo 3.6.

### Naloga 3.7.

Izračunajmo najprej pričakovane vrednosti v vozliščih  $C$ ,  $D$  in  $E$  odločitvenega drevesa s slike 4.

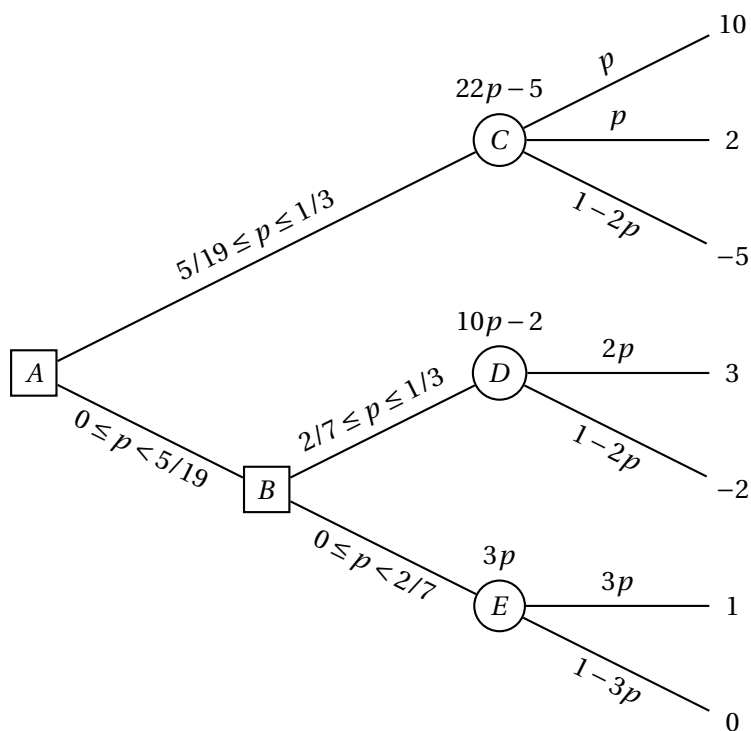
$$C = 10 \cdot p + 2 \cdot p - 5 \cdot (1 - 2p) = 22p - 5$$

$$D = 3 \cdot 2p - 2 \cdot (1 - 2p) = 10p - 2$$

$$E = 1 \cdot 3p + 0 \cdot (1 - 3p) = 3p$$

Obravnavajmo najprej odločitev v vozlišču  $B$ . Za pot k vozlišču  $D$  se odločimo, če velja  $10p - 2 \geq 3p$  oziroma  $p \geq 2/7$ , sicer se odločimo za pot k vozlišču  $E$ . Obravnavajmo sedaj še odločitev v vozlišču  $A$  – za pot k vozlišču  $C$  se odločimo, če velja:

$$\begin{aligned} p < \frac{2}{7}: \quad 22p - 5 \geq 3p & \Rightarrow \frac{5}{19} \leq p < \frac{2}{7} \\ p \geq \frac{2}{7}: \quad 22p - 5 \geq 10p - 2 & \Rightarrow p \geq \frac{2}{7} \geq \frac{1}{4} \end{aligned}$$



Slika 56: Odločitveno drevo za nalogo 3.7 v odvisnosti od vrednosti parametra  $p$ .

Odločamo se torej tako:

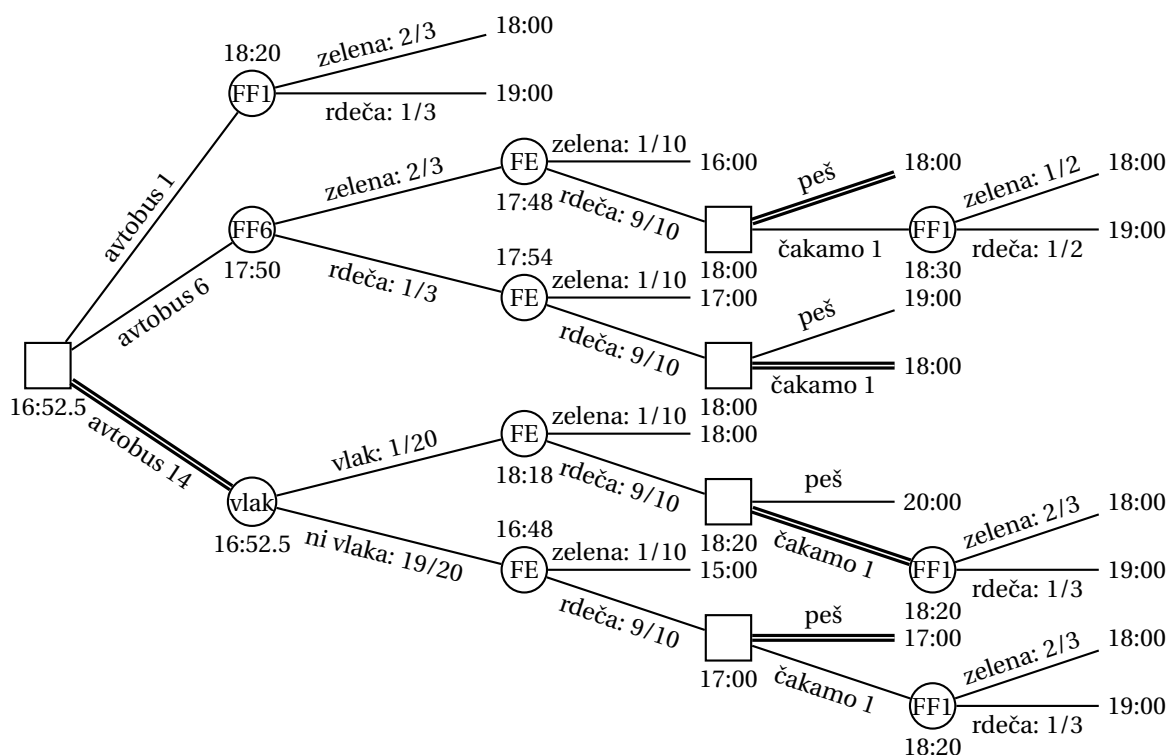
- če je  $0 \leq p < 5/19$ , se odločimo za pot preko vozlišča  $B$  k vozlišču  $E$ , pričakovana vrednost je  $3p \in [0, 15/19]$ ; in
- če je  $5/19 \leq p \leq 1/3$ , se odločimo za pot k vozlišču  $C$ , pričakovana vrednost je  $22p - 5 \in [15/19, 7/3]$ .

Proces odločanja je prikazan na sliki 56.

### Naloga 3.8.

Izračunamo lahko, da pot z avtobusom 1 traja 18 minut oziroma minuto dlje ob rdečem semaforju pri FF, pot z avtobusom 6 in naknadno pešačenje traja 16 minut ter minuto dlje ob rdečem semaforju pri FF in še dve minuti dlje ob rdečem semaforju pri FE, pot z avtobusom 14 in naknadno pešačenje pa traja 15 minut ter tri minute dlje ob čakanju na vlak in še dve minuti dlje ob rdečem semaforju pri FE. Če gremo na avtobus 6 in pri FF naletimo na zeleno luč, potem je verjetnost, da tudi avtobus 1 tam naleti na zeleno luč, enaka

$$P(\text{avtobus 1 ima zeleno pri FF} \mid \text{avtobus 6 ima zeleno pri FF}) = \frac{1/3}{2/3} = \frac{1}{2}.$$



Slika 57: Odločitveno drevo za nalogo 3.8.

S temi podatki lahko narišemo odločitveno drevo s slike 57, s katerega je razvidno, da pričakovani čas trajanja minimiziramo, če gremo na avtobus 14, potem pa v primeru čakanja na vlak in rdeče luči pri FE počakamo na avtobus 1, sicer pa pot nadaljujemo peš. Pričakovani čas trajanja poti je tako 16 minut in 52.5 sekund.

### Naloga 3.9.

Naj bo  $A$  dogodek, da izvedenec pripiše večje možnosti prvemu podjetju, ter  $B_1$ ,  $B_2$  in  $B_0$  dogodki, da uspe prvo, drugo oziroma nobeno podjetje. Izračunajmo verjetnosti za mnenje izvedenca ter uspešnosti podjetij v odvisnosti od njega.

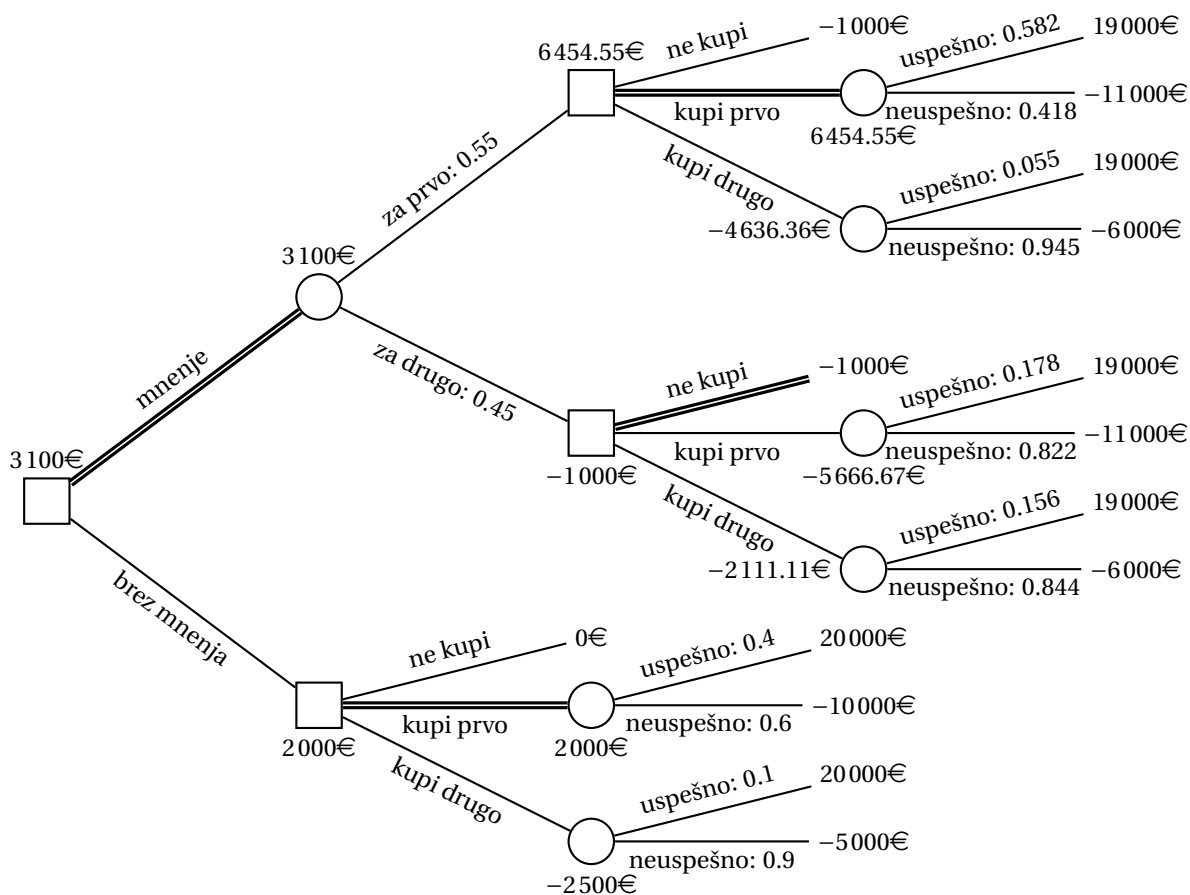
$$P(A) = 0.4 \cdot 0.8 + 0.1 \cdot 0.3 + 0.5 \cdot 0.4 = 0.55$$

$$P(\bar{A}) = 0.4 \cdot 0.2 + 0.1 \cdot 0.7 + 0.5 \cdot 0.6 = 0.45$$

$$P(B_0 | A) = \frac{0.5 \cdot 0.4}{0.55} = \frac{4}{11}$$

$$P(B_1 | A) = \frac{0.4 \cdot 0.8}{0.55} = \frac{32}{55}$$

$$P(B_2 | A) = \frac{0.1 \cdot 0.3}{0.55} = \frac{3}{55}$$



Slika 58: Odločitveno drevo za nalogo 3.9.

$$\begin{aligned}
 P(B_0 | \bar{A}) &= \frac{0.5 \cdot 0.6}{0.45} = \frac{2}{3} \\
 P(B_1 | \bar{A}) &= \frac{0.4 \cdot 0.2}{0.45} = \frac{8}{45} \\
 P(B_2 | \bar{A}) &= \frac{0.1 \cdot 0.7}{0.45} = \frac{7}{45}
 \end{aligned}$$

S pomočjo zgoraj izračunanih verjetnosti lahko narišemo odločitveno drevo s slike 58. Odločimo se torej za mnenje izvedenca – če je to naklonjeno prvemu podjetju, kupimo njegove delnice, sicer pa ne kupimo ničesar. Pričakovan dobiček je 3100€.

### Naloga 3.10.

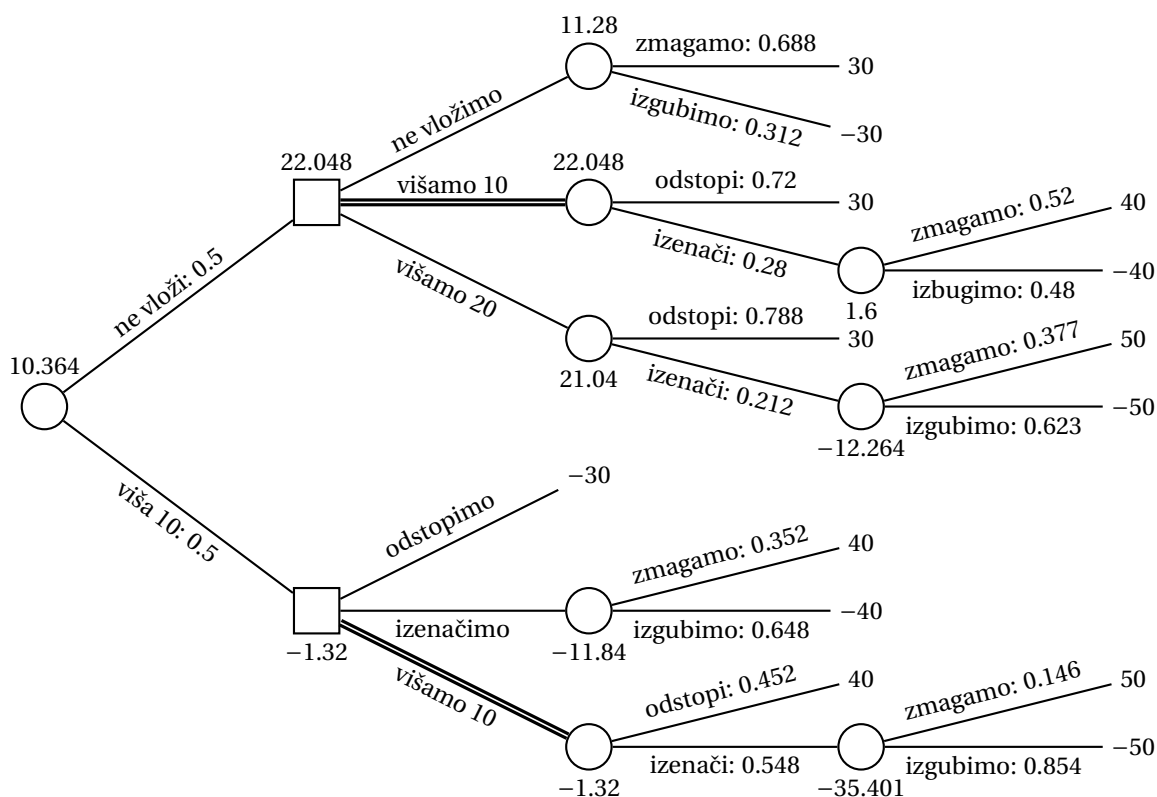
Naj bo  $A$  dogodek, da nasprotnik na začetku vloži 10 žetonov,  $B_n$  dogodek, da odgovorimo z  $n$  žetoni ( $n \in \{0, 10, 20\}$ ),  $C$  dogodek, da nasprotnik za tem izenači, in  $D$  dogodek, da dobimo igro. Izračunajmo ustrezne pogojne verjetnosti.

$$\begin{aligned}
P(A) &= 0.6 \cdot 0.3 + 0.4 \cdot 0.8 &= 0.5 \\
P(\bar{A}) &= 0.6 \cdot 0.7 + 0.4 \cdot 0.2 &= 0.5 \\
P(C | A \cap B_{20}) &= \frac{0.6 \cdot 0.3 \cdot 0.1 + 0.4 \cdot 0.8 \cdot 0.8}{0.5} &= 0.548 \\
P(\bar{C} | A \cap B_{20}) &= \frac{0.6 \cdot 0.3 \cdot 0.9 + 0.4 \cdot 0.8 \cdot 0.2}{0.5} &= 0.452 \\
P(C | \bar{A} \cap B_{10}) &= \frac{0.6 \cdot 0.7 \cdot 0.2 + 0.4 \cdot 0.2 \cdot 0.7}{0.5} &= 0.28 \\
P(\bar{C} | \bar{A} \cap B_{10}) &= \frac{0.6 \cdot 0.7 \cdot 0.8 + 0.4 \cdot 0.2 \cdot 0.3}{0.5} &= 0.72 \\
P(C | \bar{A} \cap B_{20}) &= \frac{0.6 \cdot 0.7 \cdot 0.1 + 0.4 \cdot 0.2 \cdot 0.8}{0.5} &= 0.212 \\
P(\bar{C} | \bar{A} \cap B_{20}) &= \frac{0.6 \cdot 0.7 \cdot 0.9 + 0.4 \cdot 0.2 \cdot 0.2}{0.5} &= 0.788 \\
P(D | A \cap B_{10}) &= \frac{0.6 \cdot 0.3 \cdot 0.8 + 0.4 \cdot 0.8 \cdot 0.1}{0.5} &= 0.352 \\
P(\bar{D} | A \cap B_{10}) &= \frac{0.6 \cdot 0.3 \cdot 0.2 + 0.4 \cdot 0.8 \cdot 0.9}{0.5} &= 0.648 \\
P(D | A \cap B_{20} \cap C) &= \frac{0.6 \cdot 0.3 \cdot 0.1 \cdot 0.8 + 0.4 \cdot 0.8 \cdot 0.8 \cdot 0.1}{0.5 \cdot 0.548} &\approx 0.146 \\
P(\bar{D} | A \cap B_{20} \cap C) &= \frac{0.6 \cdot 0.3 \cdot 0.1 \cdot 0.2 + 0.4 \cdot 0.8 \cdot 0.8 \cdot 0.9}{0.5 \cdot 0.548} &\approx 0.854 \\
P(D | \bar{A} \cap B_0) &= \frac{0.6 \cdot 0.7 \cdot 0.8 + 0.4 \cdot 0.2 \cdot 0.1}{0.5} &= 0.688 \\
P(\bar{D} | \bar{A} \cap B_0) &= \frac{0.6 \cdot 0.7 \cdot 0.2 + 0.4 \cdot 0.2 \cdot 0.9}{0.5} &= 0.312 \\
P(D | \bar{A} \cap B_{10} \cap C) &= \frac{0.6 \cdot 0.7 \cdot 0.2 \cdot 0.8 + 0.4 \cdot 0.2 \cdot 0.7 \cdot 0.1}{0.5 \cdot 0.28} &= 0.52 \\
P(\bar{D} | \bar{A} \cap B_{10} \cap C) &= \frac{0.6 \cdot 0.7 \cdot 0.2 \cdot 0.2 + 0.4 \cdot 0.2 \cdot 0.7 \cdot 0.9}{0.5 \cdot 0.28} &= 0.48 \\
P(D | \bar{A} \cap B_{20} \cap C) &= \frac{0.6 \cdot 0.7 \cdot 0.1 \cdot 0.8 + 0.4 \cdot 0.2 \cdot 0.8 \cdot 0.1}{0.5 \cdot 0.212} &\approx 0.377 \\
P(\bar{D} | \bar{A} \cap B_{20} \cap C) &= \frac{0.6 \cdot 0.7 \cdot 0.1 \cdot 0.2 + 0.4 \cdot 0.2 \cdot 0.8 \cdot 0.9}{0.5 \cdot 0.212} &\approx 0.623
\end{aligned}$$

S pomočjo zgoraj izračunanih verjetnosti lahko narišemo odločitveno drevo s slike 59. Opazimo, da se nam v vsakem primeru izplača višati za 10 žetonov (tj., če nasprotnik na začetku vloži 10 žetonov, odgovorimo z 20 žetoni). Pričakovani dobiček je 10.364 žetonov.

### Naloga 3.11.

Izračunajmo najprej pričakovane vrednosti v vozliščih  $F$ ,  $G$  in  $H$  odločitvenega drevesa s slike 6.



Slika 59: Odločitveno drevo za nalogo 3.10.

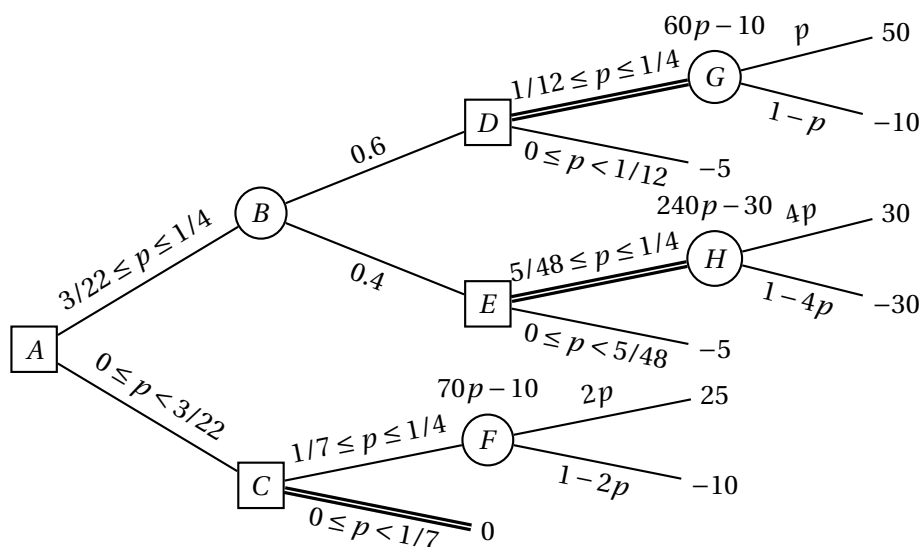
$$F = 25 \cdot 2p - 10 \cdot (1 - 2p) = 70p - 10$$

$$G = 50 \cdot p - 10 \cdot (1 - p) = 60p - 10$$

$$H = 30 \cdot 4p - 30 \cdot (1 - 4p) = 240p - 30$$

Obravnavajmo najprej odločitve v vozliščih  $C$ ,  $D$  in  $E$ . Za pot od vozlišča  $C$  k vozlišču  $F$  se odločimo, če velja  $70p - 10 \geq 0$  oziroma  $p \geq 1/7$ . Za pot od vozlišča  $D$  k vozlišču  $G$  se odločimo, če velja  $60p - 10 \geq -5$  oziroma  $p \geq 1/12$ . Za pot od vozlišča  $E$  k vozlišču  $H$  se odločimo, če velja  $240p - 30 \geq -5$  oziroma  $p \geq 5/48$ . Sedaj lahko izračunamo pričakovano vrednost v vozlišču  $B$  v odvisnosti od parametra  $p$ .

$$\begin{aligned} 0 \leq p < \frac{1}{12} &\Rightarrow B = 0.6 \cdot (-5) + 0.4 \cdot (-5) = -5 \\ \frac{1}{12} \leq p < \frac{5}{48} &\Rightarrow B = 0.6 \cdot (60p - 10) + 0.4 \cdot (-5) = 36p - 8 \\ \frac{5}{48} \leq p \leq \frac{1}{4} &\Rightarrow B = 0.6 \cdot (60p - 10) + 0.4 \cdot (240p - 30) = 132p - 18 \end{aligned}$$



Slika 60: Odločitveno drevo za nalogo 3.11 v odvisnosti od vrednosti parametra  $p$ .

Obravnavajmo sedaj še odločitev v vozlišču A – za pot k vozlišču B se odločimo, če velja:

$$\begin{array}{llll}
 0 \leq p < \frac{1}{12}: & -5 \geq 0 & \Rightarrow & \perp \\
 \frac{1}{12} \leq p < \frac{5}{48}: & 36p - 8 \geq 0 & \Rightarrow & \perp \\
 \frac{5}{48} \leq p < \frac{1}{7}: & 132p - 18 \geq 0 & \Rightarrow & \frac{3}{22} \leq p < \frac{1}{7} \\
 \frac{1}{7} \leq p \leq \frac{1}{4}: & 132p - 18 \geq 70p - 10 & \Rightarrow & \frac{1}{7} \leq p \leq \frac{1}{4}
 \end{array}$$

Odločamo se torej po sledečem pravilu.

- Če je  $0 \leq p < 3/22$ , se odločimo za pot preko vozlišča C v list z vrednostjo 0.
- Če je  $3/22 \leq p \leq 1/4$ , se odločimo za pot preko vozlišča B.
  - Če pridemo v vozlišče D, se odločimo za pot preko vozlišča G.
  - Če pridemo v vozlišče E, se odločimo za pot preko vozlišča H.

Pričakovana vrednost je  $132p - 18 \in [0, 15]$ .

Proces odločanja je prikazan na sliki 60.



### Naloga 3.12.

(a) Izračunajmo pričakovane dobičke pri različnih pogojih:

Izdela 500, prodaja po 50€, tržno zanimiv:

$$-10\,000\text{€} + 500 \cdot 50\text{€} = 15\,000\text{€}$$

Izdela 500, prodaja po 50€, tržno nezanimiv:

$$-10\,000\text{€} + 250 \cdot 50\text{€} = 2\,500\text{€}$$

Izdela 500, prodaja po 60€, tržno zanimiv:

$$-10\,000\text{€} + 500 \cdot 60\text{€} = 20\,000\text{€}$$

Izdela 500, prodaja po 60€, tržno nezanimiv:

$$-10\,000\text{€} + 100 \cdot 60\text{€} = -4\,000\text{€}$$

Izdela 1 000, prodaja po 50€, tržno zanimiv:

$$-18\,000\text{€} + 650 \cdot 50\text{€} = 14\,500\text{€}$$

Izdela 1 000, prodaja po 50€, tržno nezanimiv:

$$-18\,000\text{€} + 250 \cdot 50\text{€} = -5\,500\text{€}$$

Izdela 1 000, prodaja po 60€, tržno zanimiv:

$$-18\,000\text{€} + 550 \cdot 60\text{€} = 15\,000\text{€}$$

Izdela 1 000, prodaja po 60€, tržno nezanimiv:

$$-18\,000\text{€} + 100 \cdot 60\text{€} = -12\,000\text{€}$$

Odločitveno drevo je prikazano na sliki 61. Podjetnik naj se odloči, da naroči izdelavo 500 izdelkov, te pa prodaja po ceni 60€. Pričakovani dobiček je tedaj 15200€.

(b) Najprej določimo verjetnosti, ki jih bomo potrebovali za odločitev.

$$P(\text{zmaga}) = 0.6 \cdot 0.8 + 0.1 \cdot 0.2 = 0.5$$

$$P(\text{ne zmaga}) = 0.4 \cdot 0.8 + 0.9 \cdot 0.2 = 0.5$$

$$P(\text{zanimiv} \mid \text{zmaga}) = \frac{0.6 \cdot 0.8}{0.5} = 0.96$$

$$P(\text{ni zanimiv} \mid \text{zmaga}) = \frac{0.1 \cdot 0.2}{0.5} = 0.04$$

$$P(\text{zanimiv} \mid \text{ne zmaga}) = \frac{0.4 \cdot 0.8}{0.5} = 0.64$$

$$P(\text{ni zanimiv} \mid \text{ne zmaga}) = \frac{0.9 \cdot 0.2}{0.5} = 0.36$$

Sedaj izračunajmo pričakovane dobičke pri različnih pogojih po zmagi na razpisu. Opazimo, da v nobenem pogoju pričakovano število prodanih izdelkov ne bo preseglo količine 980 izdelkov, kolikor jih ostane po kontroli kvalitete.

Če ne zmaga na razpisu, so pričakovani dobički enaki tistim iz točke (a) pri izdelavi 1 000 kosov.

Zmaga, prodaja po 50€, tržno zanimiv:

$$-18\,000\text{€} + 650 \cdot 50\text{€} \cdot 1.2 + k = 21\,000\text{€} + k$$

Zmaga, prodaja po 50€, tržno nezanimiv:

$$-18\,000\text{€} + 250 \cdot 50\text{€} \cdot 1.2 + k = -3\,000\text{€} + k$$

Zmaga, prodaja po 60€, tržno zanimiv:

$$-18\,000\text{€} + 550 \cdot 60\text{€} \cdot 1.2 + k = 21\,600\text{€} + k$$

Zmaga, prodaja po 60€, tržno nezanimiv:

$$-18\,000\text{€} + 100 \cdot 60\text{€} \cdot 1.2 + k = -10\,800\text{€} + k$$

Z zgornjimi podatki lahko narišemo odločitveno drevo s slike 62. Izračunajmo pričakovane dobičke v vozliščih  $E, F, G, H$ .

$$E = 0.96 \cdot (21\,000\text{€} + k) + 0.04 \cdot (-3\,000\text{€} + k) = 20\,040\text{€} + k$$

$$F = 0.96 \cdot (21\,600\text{€} + k) + 0.04 \cdot (-10\,800\text{€} + k) = 20\,304\text{€} + k$$

$$G = 0.64 \cdot 14\,500\text{€} - 0.36 \cdot 5\,500\text{€} = 7\,300\text{€}$$

$$H = 0.64 \cdot 15\,000\text{€} - 0.36 \cdot 12\,000\text{€} = 5\,280\text{€}$$

Očitno je v vozlišču  $C$  ugodnejša pot k vozlišču  $F$ , v vozlišču  $D$  pa je ugodnejša pot k vozlišču  $G$  – velja torej  $C = 20\,304\text{€} + k$  in  $D = 7\,300\text{€}$ . Sedaj lahko izračunamo še pričakovano vrednost v vozlišču  $B$ .

$$B = 0.5 \cdot (20\,304\text{€} + k) + 0.5 \cdot 7\,300\text{€} = 13\,802\text{€} + \frac{k}{2}$$

Podjetnik naj se torej prijav na razpis, če velja  $13\,802\text{€} + k/2 \geq 15\,200\text{€}$  oziroma  $k \geq 2\,796\text{€}$ . Če nato zmaga na razpisu, naj izdelke prodaja po ceni 60€, v nasprotnem primeru pa naj jih prodaja po ceni 50€. Pričakovani dobiček je tedaj  $13\,802\text{€} + k/2 \in [15\,200\text{€}, 16\,302\text{€}]$ .

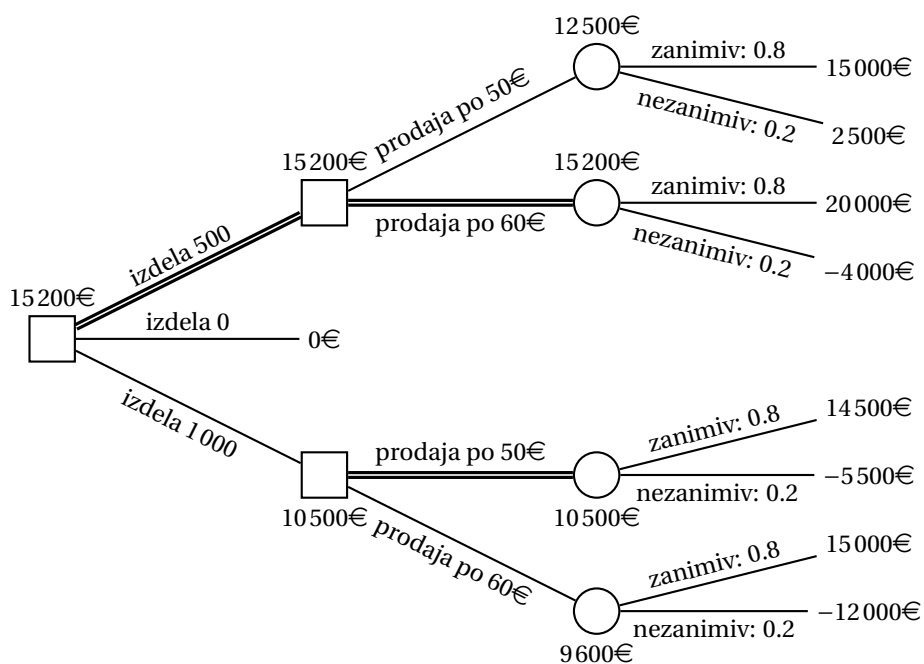
Če velja  $k < 2\,796\text{€}$ , naj se podjetnik ne prijavi na razpis. Tako kot v točki (a) naj naroči izdelavo 500 izdelkov, te pa prodaja po ceni 60€. Pričakovani dobiček je tedaj 15 200€.

### Naloga 3.13.

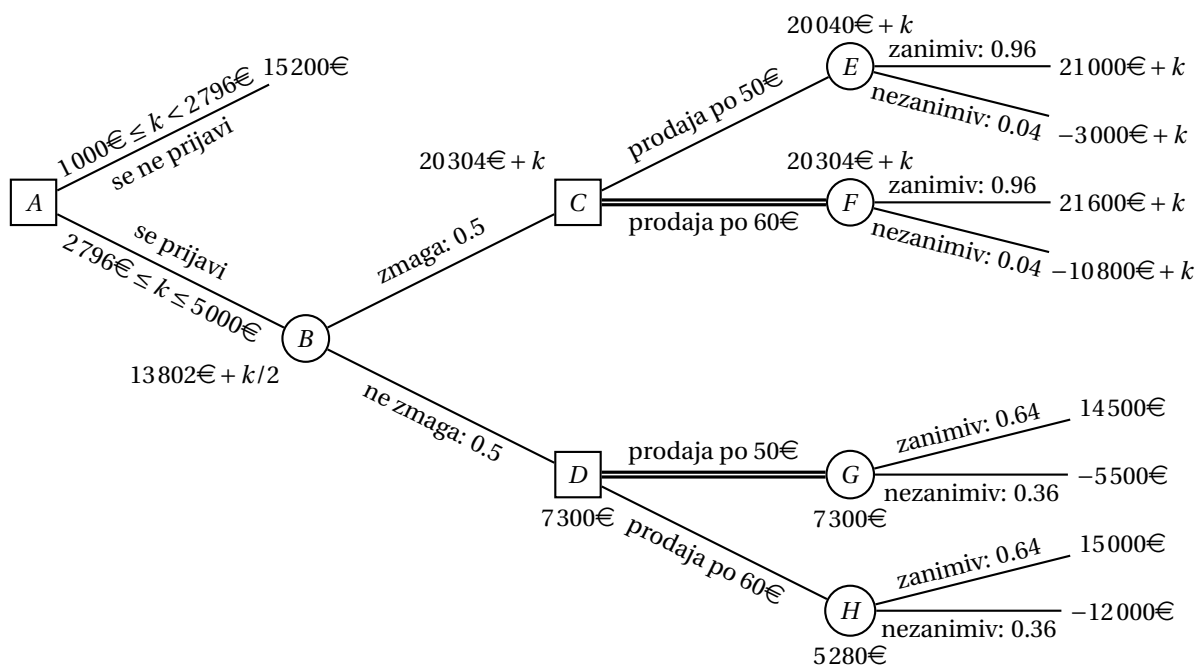
Če se odločimo za pot okoli gorovja, bomo porabili 12 ur, če se pa odločimo za pot čez prelaz, bomo porabili 6, 10 oziroma 14 ur, če je ta čist, delno zasnežen oziroma neprevozen. Če se pred tem odločimo za obisk vrača, se čas potovanja podaljša za eno uro.

Pričakovano število ur potovanja, če ne obiščemo vrača in se odločimo za pot čez prelaz, je enako

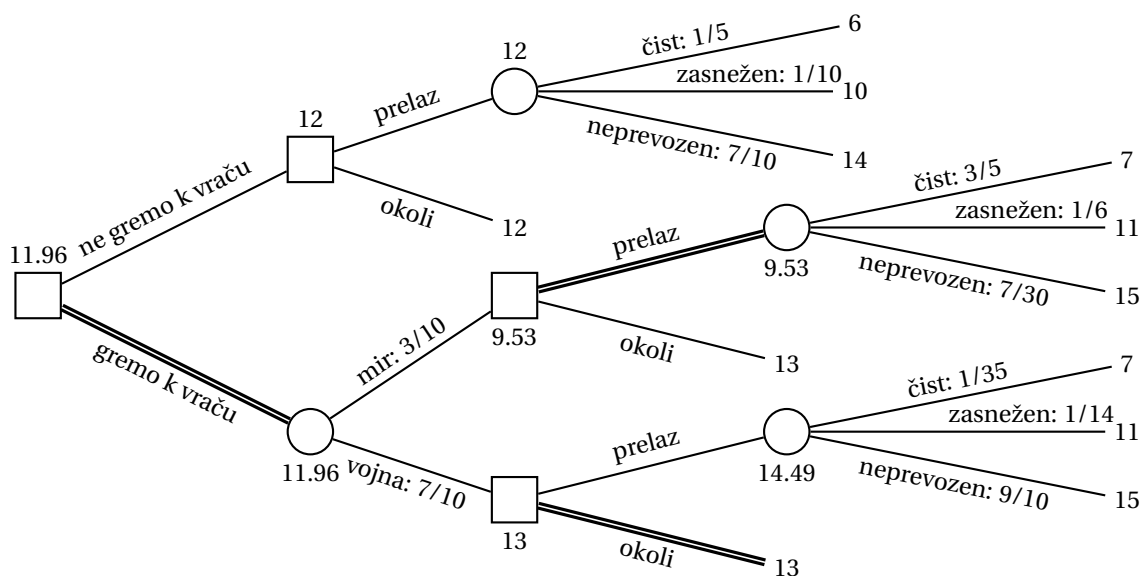
$$0.2 \cdot 6 + 0.1 \cdot 10 + 0.7 \cdot 14 = 12,$$



Slika 61: Odločitveno drevo za nalogo 3.12(a).



Slika 62: Odločitveno drevo za nalogo 3.12(b) v odvisnosti od vrednosti parametra  $k$ .



Slika 63: Odločitveno drevo za nalogo 3.13.

kar je ravno enako kot pot okoli gorovja.

Izračunajmo še verjetnosti za primer, da se odločimo za obisk vrača.

$$\begin{aligned}
 P(\text{mir}) &= 0.9 \cdot 0.2 + 0.5 \cdot 0.1 + 0.1 \cdot 0.7 = \frac{3}{10} \\
 P(\text{vojna}) &= 0.1 \cdot 0.2 + 0.5 \cdot 0.1 + 0.9 \cdot 0.7 = \frac{7}{10} \\
 P(\text{čist} \mid \text{mir}) &= \frac{0.9 \cdot 0.2}{3/10} = \frac{3}{5} \\
 P(\text{delno zasnežen} \mid \text{mir}) &= \frac{0.5 \cdot 0.1}{3/10} = \frac{1}{6} \\
 P(\text{neprevozen} \mid \text{mir}) &= \frac{0.1 \cdot 0.7}{3/10} = \frac{7}{30} \\
 P(\text{čist} \mid \text{vojna}) &= \frac{0.1 \cdot 0.2}{7/10} = \frac{1}{35} \\
 P(\text{delno zasnežen} \mid \text{vojna}) &= \frac{0.5 \cdot 0.1}{7/10} = \frac{1}{14} \\
 P(\text{neprevozen} \mid \text{vojna}) &= \frac{0.9 \cdot 0.7}{7/10} = \frac{9}{10}
 \end{aligned}$$

S pomočjo zgoraj izračunanih verjetnosti lahko narišemo odločitveno drevo s slike 63. Opazimo, da se nam izplača iti k vraču ter se odločiti za pot čez prelaz, če nam pove, da na gori vlada mir, in za pot okoli gorovja v nasprotnem primeru.

**Naloga 3.14.**

Izračunajmo najprej pričakovane vrednosti v vozliščih  $C$ ,  $F$  in  $G$  odločitvenega drevesa s slike 8.

$$\begin{aligned}C &= 0.5 \cdot (40p + 20) &= 20p + 10 \\F &= 50 \cdot p - 10 \cdot (1 - p) &= 60p - 10 \\G &= -30 \cdot p + 30 \cdot (1 - p) &= -60p + 30\end{aligned}$$

Obravnavajmo najprej odločitev v vozlišču  $D$ . Za pot k vozlišču  $F$  se odločimo, če velja  $60p - 10 \geq -5$  oziroma  $p \geq 1/12$ .

Obravnavajmo sedaj še odločitev v vozlišču  $E$ . Za pot k vozlišču  $G$  se odločimo, če velja  $-60p + 30 \geq -5$  oziroma  $p \leq 7/12$ .

Pričakovana vrednost v vozlišču  $B$  bo sledeča.

$$\begin{aligned}p < \frac{1}{12} : & \quad 0.9 \cdot (-5) + (-60p + 30) \cdot 0.1 = -1.5 - 6p \\ \frac{1}{12} \leq p \leq \frac{7}{12} : & \quad 0.9 \cdot (60p - 10) + 0.1 \cdot (-60p + 30) = -6 + 48p \\ p > \frac{7}{12} : & \quad 0.9 \cdot (60p - 10) + 0.1 \cdot (-5) = -9.5 + 54p\end{aligned}$$

Obravnavajmo sedaj še odločitev v vozlišču  $A$ .

$$\begin{aligned}0 \leq p < \frac{1}{12} : & \quad \max\{-1.5 - 6p, 20p + 10\} = 20p + 10 \\ \frac{1}{12} \leq p \leq \frac{7}{12} : & \quad \max\{-6 + 48p, 20p + 10\} = \begin{cases} 20p + 10 & p < \frac{4}{7} \\ -6 + 48p & p \geq \frac{4}{7} \end{cases} \\ p > \frac{7}{12} : & \quad \max\{-9.5 + 54p, 20p + 10\} = -9.5 + 54p\end{aligned}$$

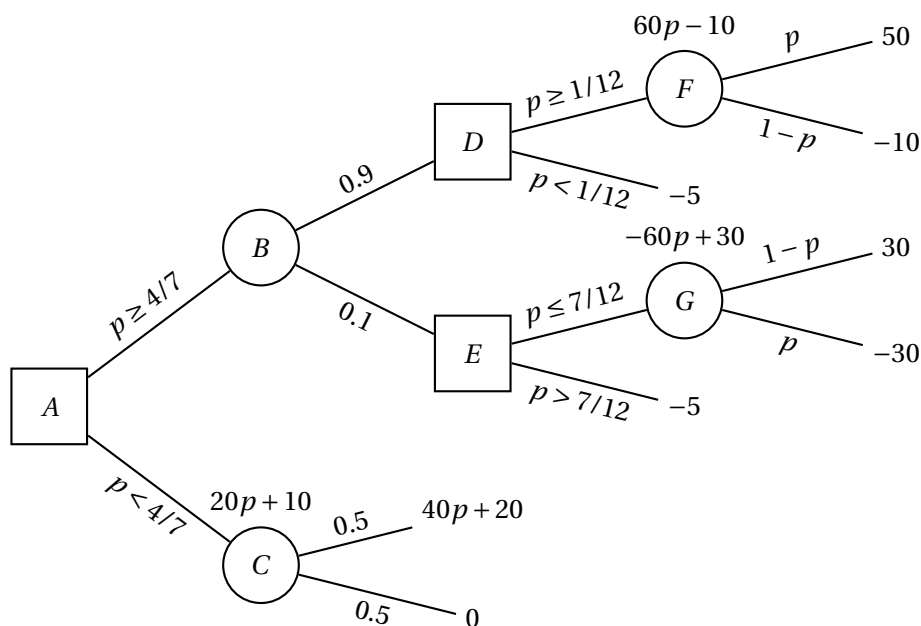
Proces odločanja je prikazan na sliki 64.

**Naloga 3.15.**

- (a) Imamo eno odločitev – ali naj zaposlimo g. Ajkalaja. Če ga ne zaposlimo, je dobiček 0€, če pa ga zaposlimo, pa pridemo v stanje, v katerem imamo z verjetnostjo  $p$  dobiček 2 100 000€, z verjetnostjo  $1 - p$  pa -800 000€. V tem stanju je torej pričakovan dobiček enak

$$p \cdot 2\,100\,000\text{€} - (1 - p) \cdot 800\,000\text{€} = p \cdot 2\,900\,000\text{€} - 800\,000\text{€}.$$

Za  $p < 8/29$  torej pričakujemo izgubo in se nam ga ne izplača zaposliti, sicer pa imamo dobiček in se ga nam tako izplača zaposliti. Graf pričakovanega dobička je prikazan na sliki 65. Pri podani vrednosti  $p = 3/5$  se nam torej izplača zaposliti g. Ajkalaja, saj tedaj pričakujemo dobiček 940 000€.



Slika 64: Odločitveno drevo za nalogo 3.14 v odvisnosti od vrednosti parametra  $p$ .

- (b) EVPI izračunamo tako, da od pričakovanega dobička pri znanem izidu odšte-  
jemo pričakovani dobiček pri neznanem izidu:

$$\text{EVPI} = 3/5 \cdot 2\,100\,000\text{€} + 2/5 \cdot 0\text{€} - 940\,000\text{€} = 320\,000\text{€}$$

Ker je EVPI večji od cene dodatnega testiranja, izračunajmo še EVE. Najprej  
določimo verjetnosti:

$$P(\text{test pozitiven}) = 7/8 \cdot 3/5 + 1/4 \cdot 2/5 = 5/8$$

$$P(\text{test negativen}) = 1/8 \cdot 3/5 + 3/4 \cdot 2/5 = 3/8$$

$$P(\text{uspešen rezultat} \mid \text{test pozitiven}) = \frac{7/8 \cdot 3/5}{5/8} = 21/25$$

$$P(\text{neuspešen rezultat} \mid \text{test pozitiven}) = \frac{1/4 \cdot 2/5}{5/8} = 4/25$$

$$P(\text{uspešen rezultat} \mid \text{test negativen}) = \frac{1/8 \cdot 3/5}{3/8} = 1/5$$

$$P(\text{neuspešen rezultat} \mid \text{test negativen}) = \frac{3/4 \cdot 2/5}{3/8} = 4/5$$

Sedaj lahko izračunamo EVE:

$$\begin{aligned} \text{EVE} &= (21/25 \cdot 2\,100\,000\text{€} - 4/25 \cdot 800\,000\text{€}) \cdot 5/8 + \\ &\quad (1/5 \cdot 0\text{€} + 4/5 \cdot 0\text{€}) \cdot 3/8 - 940\,000\text{€} = 82\,500\text{€} \end{aligned}$$

Ker je EVE večji od cene testiranja, se torej odločimo za testiranje.

- (c) Odločitveno drevo je prikazano na sliki 66. Odločimo se za test – če je ta pozitiven, zaposlimo g. Ajkalaja, sicer pa ne.

### Naloga 3.16.

- (a) Imamo eno samo odločitev – ali naj izvedemo mašenje po metodi dr. Totalkova. Če ga ne izvedemo, je dobiček 0\$, če pa ga, pa preidemo v stanje, v katerem imamo z verjetnostjo  $p$  dobiček 2800M\$, z verjetnostjo  $1 - p$  pa  $-700$ M\$. V tem primeru je pričakovani dobiček enak

$$p \cdot 2800\text{M\$} - (1 - p) \cdot 700\text{M\$} = p \cdot 3500\text{M\$} - 700\text{M\$}.$$

Za  $p < 1/5$  torej pričakujemo izgubo in se podjetju mašenja ne izplača izvesti, sicer pa podjetje pričakuje dobiček, tako da se izvedba mašenja izplača. Graf pričakovanega dobička je prikazan na sliki 67. Pri podani vrednosti  $p = 3/7$  se podjetju torej izplača izvesti mašenje, pri čemer pričakuje dobiček 800M\$.

- (b) EVPI izračunamo tako, da od pričakovanega dobička pri znanem izidu odštejemo pričakovani dobiček pri neznanem izidu:

$$\text{EVPI} = 3/7 \cdot 2800\text{M\$} + 4/7 \cdot 0\text{M\$} - 800\text{M\$} = 400\text{M\$}$$

Ker je EVPI večji od cene dodatnega testiranja, izračunajmo še EVE. Najprej določimo verjetnosti:

$$P(\text{raziskava pozitivna}) = \frac{7}{9} \cdot \frac{3}{7} + \frac{1}{3} \cdot \frac{4}{7} = \frac{11}{21}$$

$$P(\text{raziskava negativna}) = \frac{2}{9} \cdot \frac{3}{7} + \frac{2}{3} \cdot \frac{4}{7} = \frac{10}{21}$$

$$P(\text{uspešen projekt} \mid \text{raziskava pozitivna}) = \frac{7/9 \cdot 3/7}{11/21} = \frac{7}{11}$$

$$P(\text{neuspešen projekt} \mid \text{raziskava pozitivna}) = \frac{1/3 \cdot 4/7}{11/21} = \frac{4}{11}$$

$$P(\text{uspešen projekt} \mid \text{raziskava negativna}) = \frac{2/9 \cdot 3/7}{10/21} = \frac{1}{5}$$

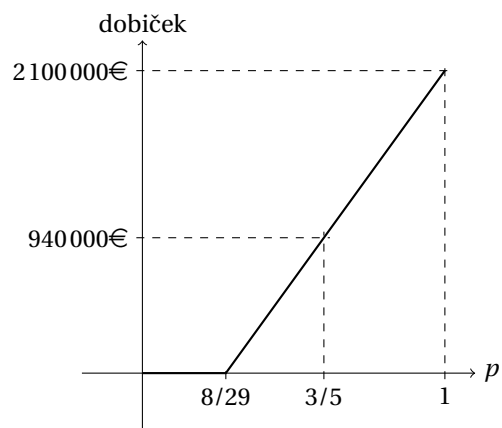
$$P(\text{neuspešen projekt} \mid \text{raziskava negativna}) = \frac{2/3 \cdot 4/7}{10/21} = \frac{4}{5}$$

Sedaj lahko izračunamo EVE:

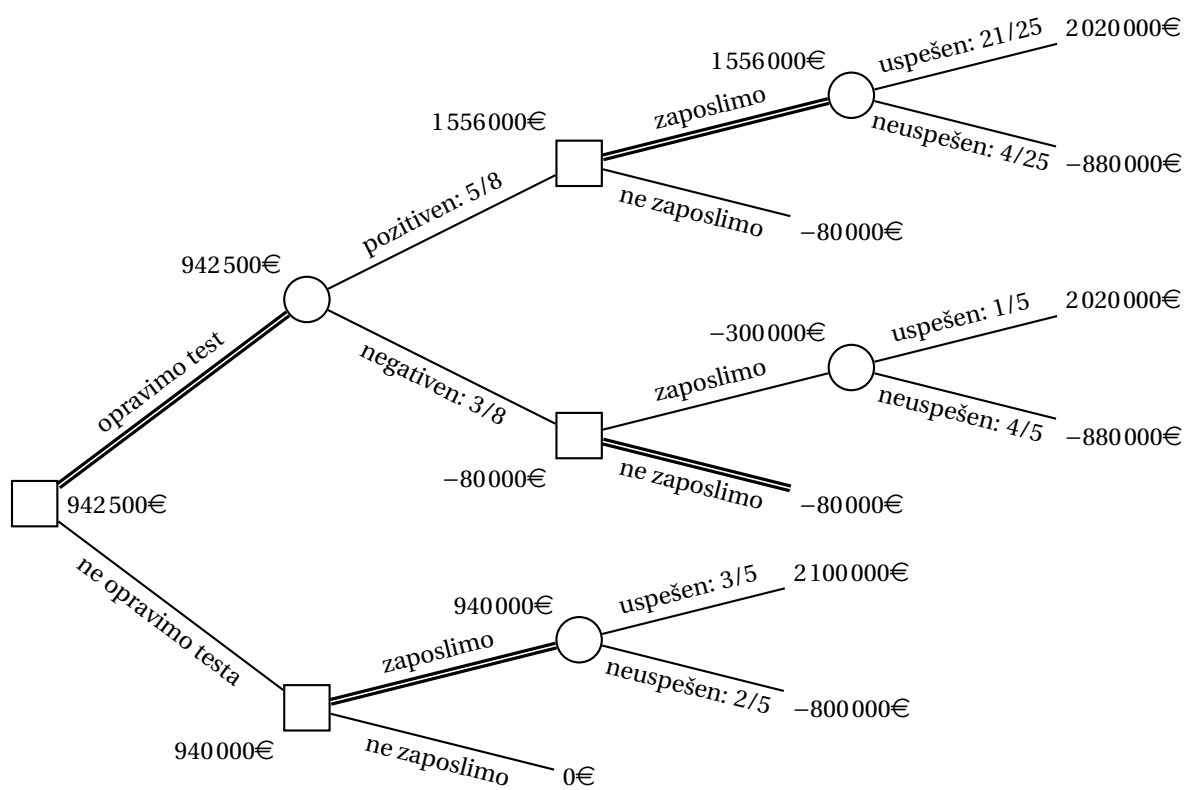
$$\begin{aligned} \text{EVE} &= (7/11 \cdot 2800\text{M\$} - 4/11 \cdot 700\text{M\$}) \cdot 11/21 + \\ &\quad (1/5 \cdot 0\text{M\$} + 4/5 \cdot 0\text{M\$}) \cdot 10/21 - 800\text{M\$} = 0\text{M\$} \end{aligned}$$

Ker je EVE manjši od cene raziskave, se zanjo ne odločimo.

- (c) Odločitveno drevo je prikazano na sliki 68. Za raziskavo se ne bomo odločili, pač pa bomo poskusili zamašiti vrtino po metodi dr. Totalkova.

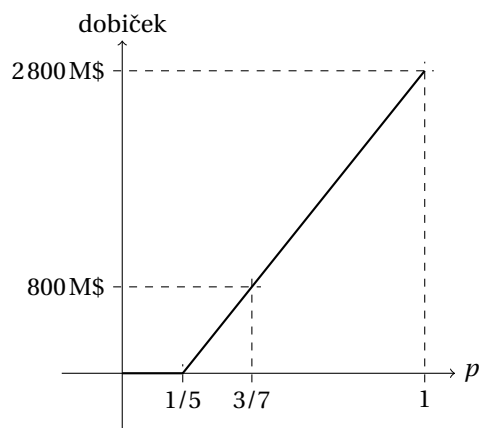


Slika 65: Graf pričakovanega dobička za nalogo 3.15(a).

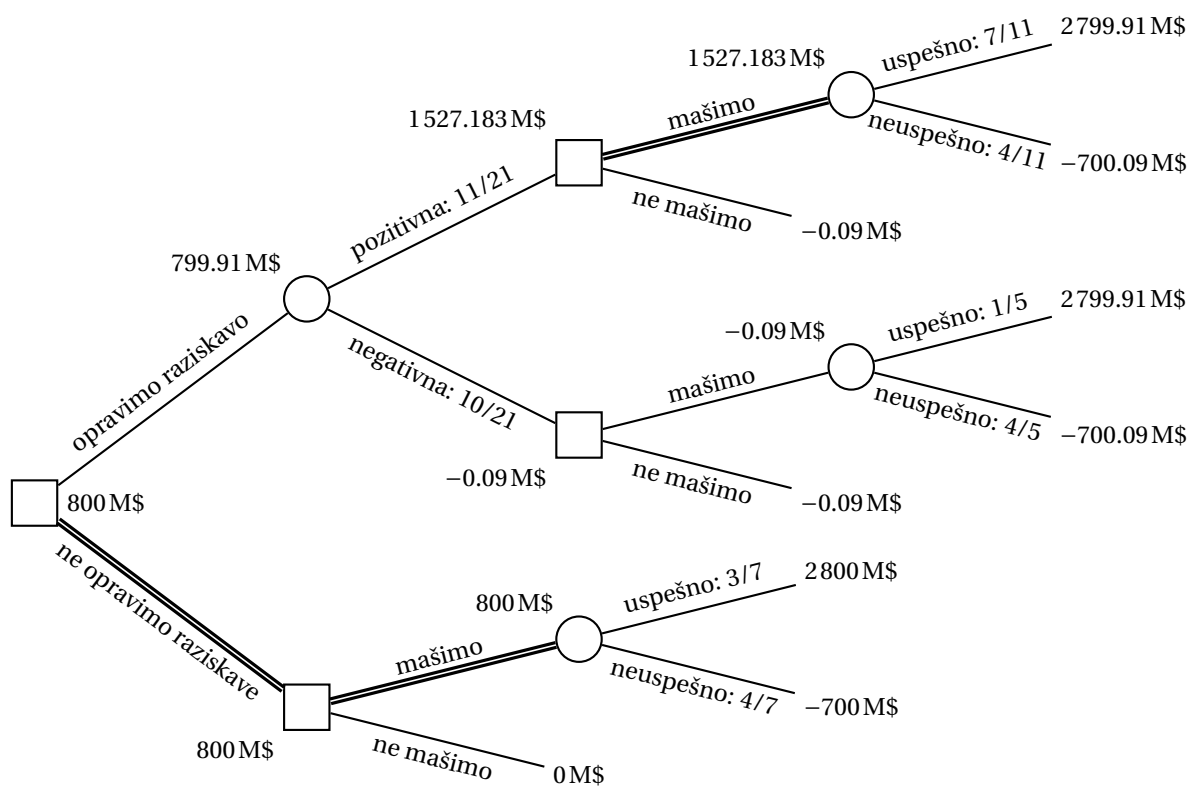


Slika 66: Odločitveno drevo za nalogo 3.15(c).





Slika 67: Graf pričakovanega dobička za nalogo 3.16(a).



Slika 68: Odločitveno drevo za nalogo 3.16(c).

**Naloga 3.17.**

- (a) Imamo eno samo odločitev – ali naj nabavimo letalo. Če ga ne nabavimo, imamo 0€ dobička, če pa ga, pa preidemo v stanje, v katerem imamo z verjetnostjo 0.2 dobiček  $1\,000\,000\text{€} - 170\,000\text{€} = 830\,000\text{€}$ , z verjetnostjo 0.3 dobiček  $340\,000\text{€} - 170\,000\text{€} = 170\,000\text{€}$ , z verjetnostjo 0.5 pa dobiček  $10\,000\text{€} - 170\,000\text{€} = -160\,000\text{€}$ . V tem primeru je pričakovani dobiček enak

$$0.2 \cdot 830\,000\text{€} + 0.3 \cdot 170\,000\text{€} - 0.5 \cdot 160\,000\text{€} = 137\,000\text{€}.$$

Letalski družbi se torej nabava letala izplača, pri tem pa pričakuje dobiček 137 000€.

- (b) Izračunajmo verjetnosti za primer, da naročimo izvedensko mnenje.

$$P(\text{ugodno}) = 0.9 \cdot 0.2 + 0.6 \cdot 0.3 + 0.1 \cdot 0.5 = 0.41$$

$$P(\text{neugodno}) = 0.1 \cdot 0.2 + 0.4 \cdot 0.3 + 0.9 \cdot 0.5 = 0.59$$

$$P(\text{odlično} \mid \text{ugodno}) = \frac{0.9 \cdot 0.2}{0.41} \approx 0.439$$

$$P(\text{zadovoljivo} \mid \text{ugodno}) = \frac{0.6 \cdot 0.3}{0.41} \approx 0.439$$

$$P(\text{slabo} \mid \text{ugodno}) = \frac{0.1 \cdot 0.5}{0.41} \approx 0.122$$

$$P(\text{odlično} \mid \text{neugodno}) = \frac{0.1 \cdot 0.2}{0.59} \approx 0.034$$

$$P(\text{zadovoljivo} \mid \text{neugodno}) = \frac{0.4 \cdot 0.3}{0.59} \approx 0.203$$

$$P(\text{slabo} \mid \text{neugodno}) = \frac{0.9 \cdot 0.5}{0.59} \approx 0.763$$

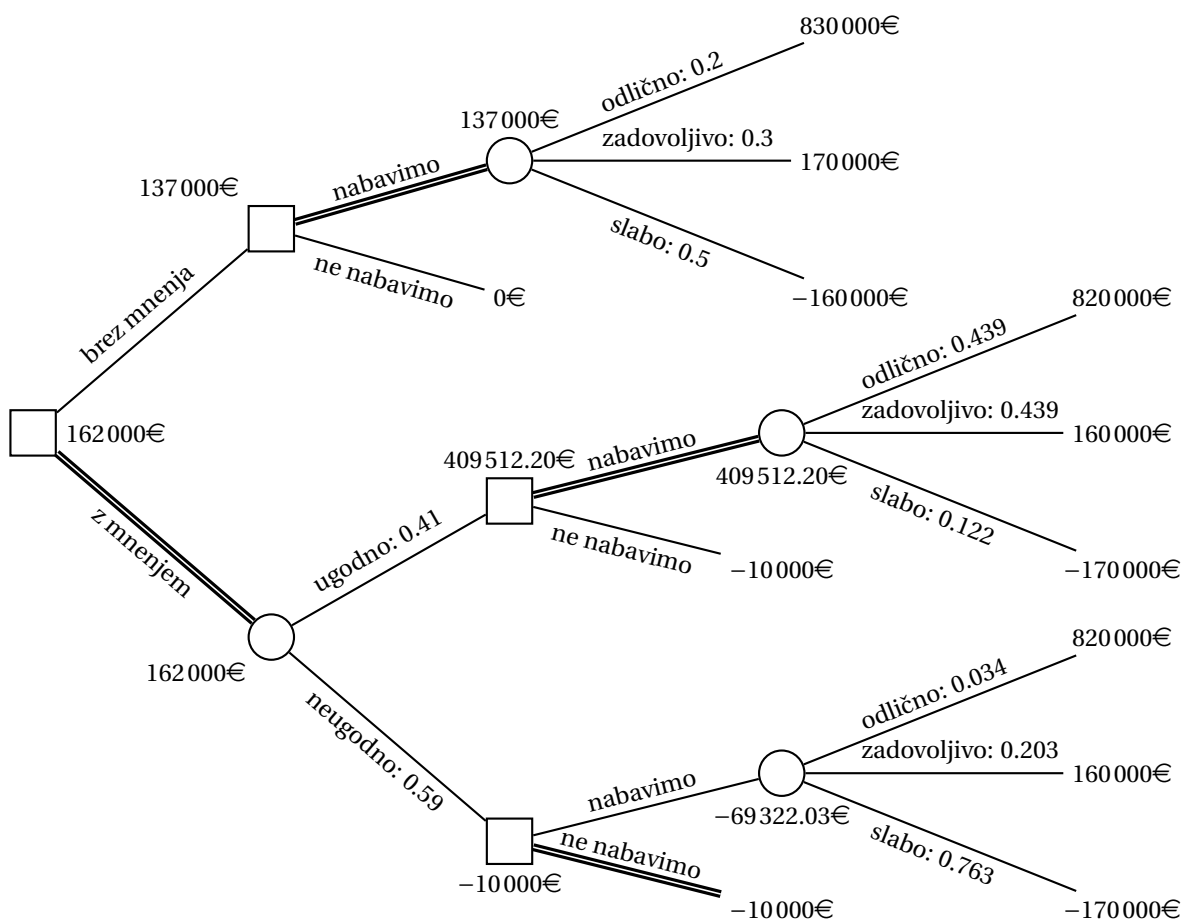
S pomočjo zgoraj izračunanih verjetnosti lahko narišemo odločitveno drevo s slike 69. Opazimo, da se nam izplača naročiti izvedensko mnenje, ter letalo nabaviti, če je to ugodno, in ga ne nabaviti, če mnenje ni ugodno.

**Naloga 3.18.**

- (a) Študent ima eno samo odločitev – ali naj nadaljuje študij. Če ga ne nadaljuje, ima 0€ dobička, če pa ga, pa preide v stanje, v katerem ima z verjetnostjo 0.8 dobiček 200 000€, z verjetnostjo 0.2 pa –40 000€. V tem primeru je pričakovani dobiček enak

$$0.8 \cdot 200\,000\text{€} - 0.2 \cdot 40\,000\text{€} = 152\,000\text{€}.$$

Študentu se torej nadaljevanje študija izplača, pri tem pa pričakuje dobiček 152 000€.



Slika 69: Odločitveno drevo za nalogo 3.17.

(b) Izračunajmo verjetnosti za primer, da se študent odloči za testiranje.

$$P(\text{test pozitiven}) = \frac{19}{20} \cdot 0.8 + \frac{1}{10} \cdot 0.2 = 0.78$$

$$P(\text{test negativen}) = \frac{1}{20} \cdot 0.8 + \frac{9}{10} \cdot 0.2 = 0.22$$

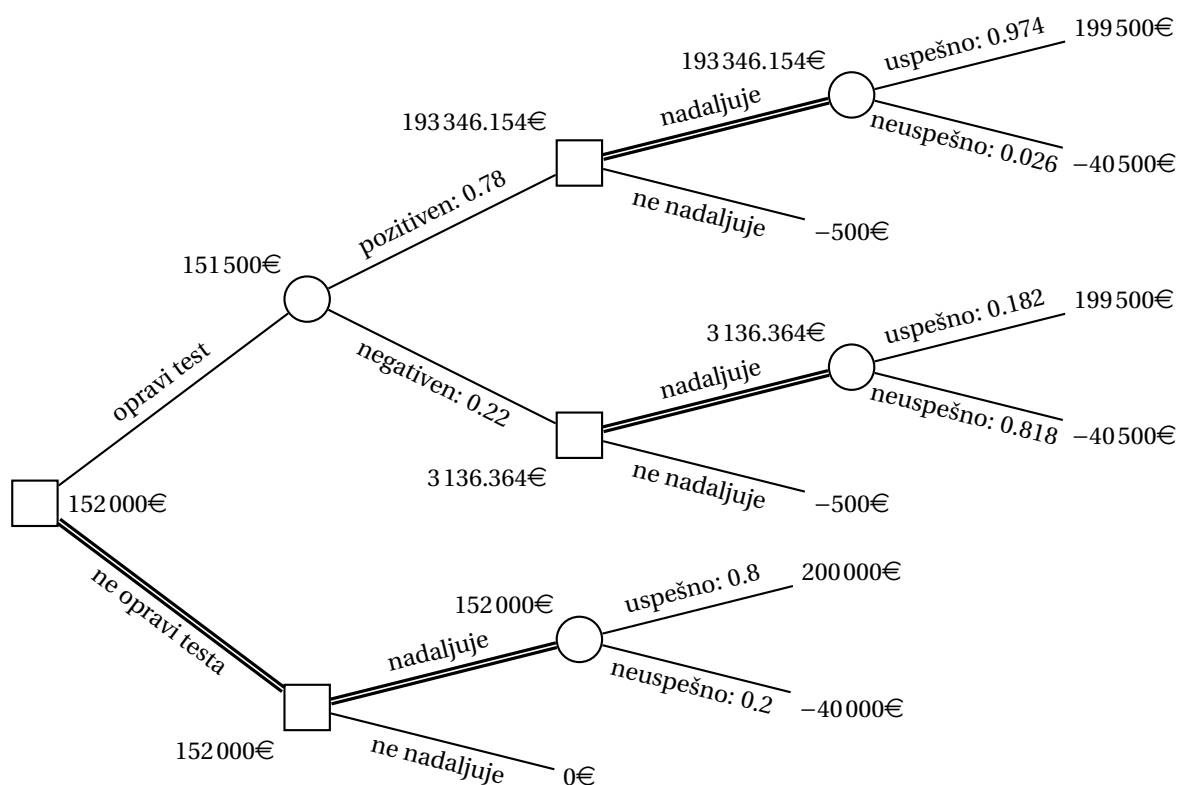
$$P(\text{uspešen študij} \mid \text{test pozitiven}) = \frac{19/20 \cdot 0.8}{0.78} \approx 0.974$$

$$P(\text{neuspešen študij} \mid \text{test pozitiven}) = \frac{1/10 \cdot 0.2}{0.78} \approx 0.026$$

$$P(\text{uspešen študij} \mid \text{test negativen}) = \frac{1/20 \cdot 0.8}{0.22} \approx 0.182$$

$$P(\text{neuspešen študij} \mid \text{test negativen}) = \frac{9/10 \cdot 0.2}{0.22} \approx 0.818$$

S pomočjo zgoraj izračunanih verjetnosti lahko narišemo odločitveno drevo s slike 70. Opazimo, da se študentu ne izplača prijaviti na testiranje, saj mu to



Slika 70: Odločitveno drevo za nalogo 3.18.

ne da zadostne informacije za morebitno spremembo odločitve.

### Naloga 3.19.

- (a) Imamo eno samo odločitev – ali naj podjetju odobrimo posojilo. Če ga ne odobrimo, imamo 0€ dobička, če pa ga, preidemo v stanje, v katerem imamo z verjetnostjo 0.3 dobiček v višini 20 000€, z verjetnostjo 0.5 dobiček v višini 10 000€ in z verjetnostjo 0.2 dobiček v višini –15 000€. V tem primeru je pričakovani dobiček enak

$$0.3 \cdot 20\,000\text{€} + 0.5 \cdot 10\,000\text{€} - 0.2 \cdot 15\,000\text{€} = 8\,000\text{€}.$$

Direktorici torej svetujemo, naj odobri posojilo, saj bo tedaj pričakovala dobiček 8 000€.

Izračunajmo še EVPI. Tukaj upoštevamo, da posojilo odobrimo, če bo to prineslo dobiček.

$$\text{EVPI} = 0.3 \cdot 20\,000\text{€} + 0.5 \cdot 10\,000\text{€} + 0.2 \cdot 0\text{€} = 11\,000\text{€}$$

(b) Izračunajmo verjetnosti za primer, da direktorica naroči test.

$$P(\text{test pozitiven}) = \frac{2}{5} \cdot 0.3 + \frac{1}{10} \cdot 0.5 + \frac{1}{10} \cdot 0.2 = 0.19$$

$$P(\text{test nevtralen}) = \frac{2}{5} \cdot 0.3 + \frac{1}{2} \cdot 0.5 + \frac{2}{5} \cdot 0.2 = 0.45$$

$$P(\text{test negativen}) = \frac{1}{5} \cdot 0.3 + \frac{2}{5} \cdot 0.5 + \frac{1}{2} \cdot 0.2 = 0.36$$

$$P(\text{podjetje uspešno} \mid \text{test pozitiven}) = \frac{2/5 \cdot 0.3}{0.19} \approx 0.632$$

$$P(\text{podjetje povprečno uspešno} \mid \text{test pozitiven}) = \frac{1/10 \cdot 0.5}{0.19} \approx 0.263$$

$$P(\text{podjetje neuspešno} \mid \text{test pozitiven}) = \frac{1/10 \cdot 0.2}{0.19} \approx 0.105$$

$$P(\text{podjetje uspešno} \mid \text{test nevtralen}) = \frac{2/5 \cdot 0.3}{0.45} \approx 0.267$$

$$P(\text{podjetje povprečno uspešno} \mid \text{test nevtralen}) = \frac{1/2 \cdot 0.5}{0.45} \approx 0.556$$

$$P(\text{podjetje neuspešno} \mid \text{test nevtralen}) = \frac{2/5 \cdot 0.2}{0.45} \approx 0.178$$

$$P(\text{podjetje uspešno} \mid \text{test negativen}) = \frac{1/5 \cdot 0.3}{0.36} \approx 0.167$$

$$P(\text{podjetje povprečno uspešno} \mid \text{test negativen}) = \frac{2/5 \cdot 0.5}{0.36} \approx 0.556$$

$$P(\text{podjetje neuspešno} \mid \text{test negativen}) = \frac{1/2 \cdot 0.2}{0.36} \approx 0.278$$

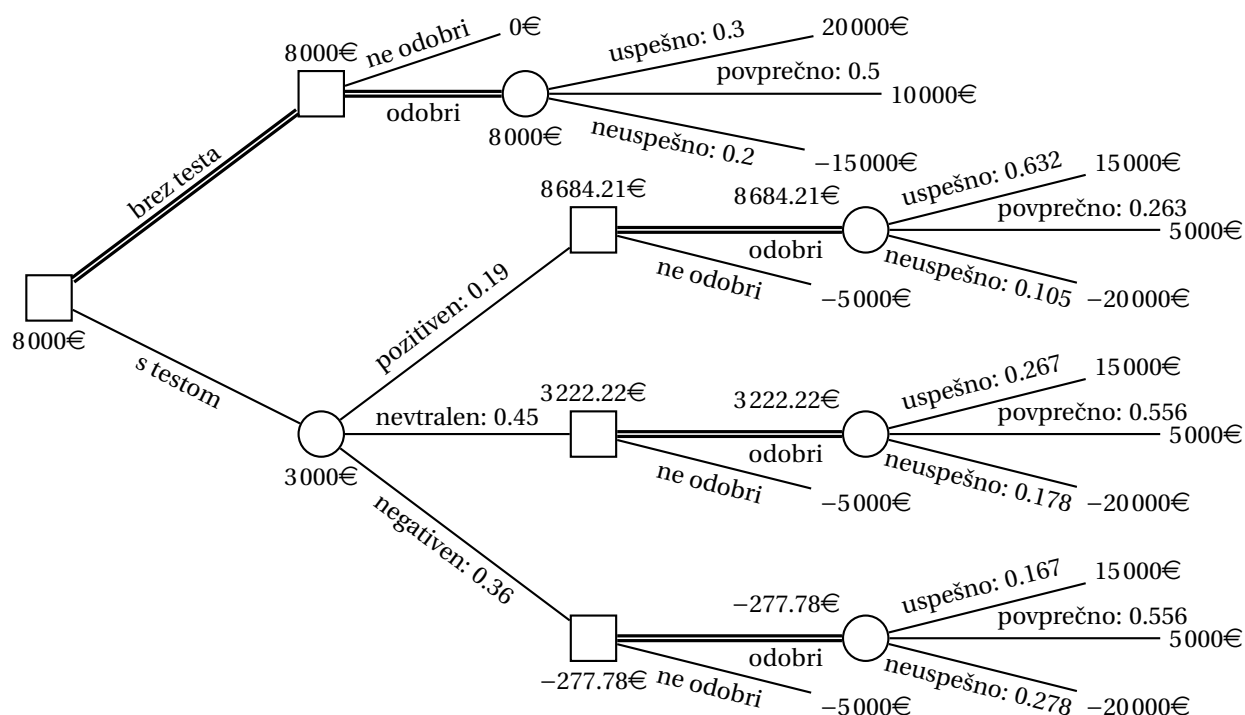
S pomočjo zgoraj izračunanih verjetnosti lahko narišemo odločitveno drevo s slike 71. Opazimo, da se direktorici ne izplača naročiti testa, saj ji ta ne da zadostne informacije za morebitno spremembo odločitve.

### Naloga 3.20.

Zaradi enostavnosti predpostavimo, da se obrestuje samo glavnica (enkrat na leto), pri čemer pri petletnem varčevanju vsakič velja začetna obrestna mera. Izračunajmo obresti po petih letih pri različnih shemah vezave glede na dogajanje po treh letih:

shema	nespremenjena	povečana	zmanjšana	prekinitev
5 let	250	250	250	–
3 + 2 · 1 leto	170	194	154	–
3 leta	90	90	90	90
5 · 1 leto	200	224	184	–
3 · 1 leto	120	120	120	120

Drugih možnosti ne bomo obravnavali, saj se očitno 5-letna vezava ne izplača, če jo bomo predčasno prekinili. Iz zgornje tabele je torej jasno, da se bomo



Slika 71: Odločitveno drevo za nalogo 3.19.

odločili bodisi za 5-letno vezavo ali pa vsakoletno vezavo, z morebitno prekinitvijo oziroma prenehanjem po 3 letih brez nadaljnje vezave. Dobimo torej sledečo odločitveno tabelo:

začetna shema	nespremenjena	povečana	zmanjšana	prekinitev
5 let	250	250	250	90
5 · 1 leto	200	224	184	120

Pri optimistovem kriteriju se ravnamo glede na najboljši možen izid. Pri 5-letni vezavi je to 250€, pri letni vezavi pa 224€, zato se odločimo za 5-letno vezavo.

Pri pesimistovem (Waldovem) kriteriju se ravnamo glede na najslabši možen izid. Pri 5-letni vezavi je to 90€, pri letni vezavi pa 120€, zato se odločimo za letno vezavo.

Pri Laplaceovem kriteriju so vse možnosti enako verjetne. Pri 5-letni vezavi je torej pričakovani dobiček enak  $(250€ + 250€ + 250€ + 90€)/4 = 210€$ , pri letni vezavi pa  $(200€ + 224€ + 184€ + 120€)/4 = 182€$ , zato se odločimo za 5-letno vezavo.

Pri Savageovem kriteriju želimo minimizirati največje možno obžalovanje. Pri 5-letni vezavi je največje obžalovanje enako  $120€ - 90€ = 30€$ , pri letni vezavi pa  $250€ - \min\{200€, 224€, 184€\} = 66€$ , zato se odločimo za 5-letno vezavo.

Pri Hurwiczevem kriteriju upoštevamo, da se najboljši izid zgodi z verjetnostjo  $\alpha$ , najslabši pa z verjetnostjo  $1 - \alpha$ . Pri 5-letni vezavi je torej pričakovan dobiček enak  $\alpha \cdot 250\text{€} + (1 - \alpha) \cdot 90\text{€} = \alpha \cdot 160\text{€} + 90\text{€}$ , pri letni vezavi pa  $\alpha \cdot 224\text{€} + (1 - \alpha) \cdot 120\text{€} = \alpha \cdot 104\text{€} + 120\text{€}$ . Pri vrednosti  $\alpha = 15/28$  sta obe možnosti enakovredni. Pri večjih vrednostih  $\alpha$  se odločimo za 5-letno vezavo, pri manjših pa za letno vezavo.

### Naloga 3.21.

- (a) Naj bo  $X_k$  slučajna spremenljivka, ki označuje dobiček pri prodaji  $k$  kart ( $k \in \{100, 101, 102\}$ ). Izračunajmo pričakovane dobičke

$$E(X_{100}) = 100 \cdot 100\text{€} = 10\,000\text{€}$$

$$E(X_{101}) = 101 \cdot 100\text{€} - 0.25 \cdot 200\text{€} = 10\,050\text{€}$$

$$E(X_{102}) = 102 \cdot 100\text{€} - (0.25 \cdot 2 + 0.5) \cdot 200\text{€} = 10\,000\text{€}$$

Letalska družba naj torej proda 101 karto.

- (b) Naj bo  $A_i$  dogodek, da analiza napove odpoved  $i$  potnikov, in  $B_j$  dogodek, da imamo dejansko  $j$  dopovedi ( $i, j = 0, 1, 2$ ). Poznamo pogojne verjetnosti  $P(A_i | B_j) = p_{ij}$ . Izračunajmo najprej verjetnosti  $P(A_j)$ .

$$P(A_0) = 0.7 \cdot 0.25 + 0.1 \cdot 0.5 + 0 \cdot 0.25 = 0.225$$

$$P(A_1) = 0.2 \cdot 0.25 + 0.8 \cdot 0.5 + 0.1 \cdot 0.25 = 0.475$$

$$P(A_2) = 0.1 \cdot 0.25 + 0.1 \cdot 0.5 + 0.9 \cdot 0.25 = 0.3$$

Izračunajmo še verjetnosti posameznega števila odpovedi ob različnih izidih analize.

$$P(B_0 | A_0) = \frac{0.7 \cdot 0.25}{0.225} = \frac{7}{9}$$

$$P(B_1 | A_0) = \frac{0.1 \cdot 0.5}{0.225} = \frac{2}{9}$$

$$P(B_2 | A_0) = \frac{0 \cdot 0.25}{0.225} = 0$$

$$P(B_0 | A_1) = \frac{0.2 \cdot 0.25}{0.475} = \frac{2}{19}$$

$$P(B_1 | A_1) = \frac{0.8 \cdot 0.5}{0.475} = \frac{16}{19}$$

$$P(B_2 | A_1) = \frac{0.1 \cdot 0.25}{0.475} = \frac{1}{19}$$

$$P(B_0 | A_2) = \frac{0.1 \cdot 0.25}{0.3} = \frac{1}{12}$$

$$P(B_1 | A_2) = \frac{0.1 \cdot 0.5}{0.3} = \frac{1}{6}$$

$$P(B_2 | A_2) = \frac{0.9 \cdot 0.25}{0.3} = \frac{3}{4}$$

Sedaj lahko izračunamo pričakovane dobičke ob različnih izidih analize.

$$\begin{aligned}
 E(X_{100} | A_0) &= 100 \cdot 100\text{€} && = \underline{10000\text{€}} \\
 E(X_{101} | A_0) &= 101 \cdot 100\text{€} - \frac{7}{9} \cdot 200\text{€} && \approx 9944.44\text{€} \\
 E(X_{102} | A_0) &= 102 \cdot 100\text{€} - \left(\frac{7}{9} \cdot 2 + \frac{2}{9}\right) \cdot 200\text{€} && \approx 9844.44\text{€} \\
 E(X_{100} | A_1) &= 100 \cdot 100\text{€} && = 10000\text{€} \\
 E(X_{101} | A_1) &= 101 \cdot 100\text{€} - \frac{2}{19} \cdot 200\text{€} && \approx \underline{10078.95\text{€}} \\
 E(X_{102} | A_1) &= 102 \cdot 100\text{€} - \left(\frac{2}{19} \cdot 2 + \frac{16}{19}\right) \cdot 200\text{€} && \approx 9989.47\text{€} \\
 E(X_{100} | A_2) &= 100 \cdot 100\text{€} && = 10000\text{€} \\
 E(X_{101} | A_2) &= 101 \cdot 100\text{€} - \frac{1}{12} \cdot 200\text{€} && \approx 10083.33\text{€} \\
 E(X_{102} | A_2) &= 102 \cdot 100\text{€} - \left(\frac{1}{12} \cdot 2 + \frac{1}{6}\right) \cdot 200\text{€} && \approx \underline{10133.33\text{€}}
 \end{aligned}$$

Najboljše izbire v vsakem primeru so podčrtane. Ker v nobenem primeru pričakovan dobiček ne preseže pričakovanega dobička brez analize za več kot 200€, se nam torej analize ne izplača izvesti.

### Naloga 3.22.

Najprej uvedemo slučajne spremenljivke  $X$ ,  $Y_i$  in  $Z_i$  ( $i \in \{1, 2\}$ ), s katerimi bomo računali v nadaljevanju.

$X$  ... število gulfov po drugi noči  
 $Y_i$  ... število gulfov pred  $i$ -to nočjo ( $i \in \{1, 2\}$ )  
 $Z_i$  ... kilogrami granodiorita pred  $i$ -to nočjo ( $i \in \{1, 2\}$ )

Odločitev pred drugo nočjo bomo obravnavali glede na število gulfov, ki jih imamo na voljo po prvi noči.

$$E(X | Y_2 = k - 2x, Z_2 = x) = 0.4 \cdot (k - 2x + 4x) + 0.6 \cdot (k - 2x + x) = k + 0.2x$$

Spremenljivka  $k$  tu predstavlja število gulfov po prvi noči,  $x$  pa število kilogramov kupljenega granodiorita pred drugo nočjo. Ker je  $k + 0.2x$  večji od  $k$ , Fenko vedno kupi maksimalno količino granodiorita. Ne glede na izid iz prve noči torej vemo, da po prvi noči kupi maksimalno število granodiorita.

Obravnavamo torej štiri primere. Če v prvi noči kupi 2 kg granodiorita ter izdelava uspe in proda zlato, ima torej  $k = 9$  gulfov, s katerimi kupi  $x = 4$  kg granodiorita, en guld pa mu ostane; če izdelava ne uspe, potem proda granodiorit in ima torej  $k = 3$  gulde, s katerimi kupi  $x = 1$  kg granodiorita, en guld pa mu ostane. Če pa v prvi noči kupi 1 kg granodiorita ter izdelava uspe in proda zlato,



ima torej  $k = 7$  guldo, s katerimi kupi  $x = 3$  kg granodiorita, en guld pa mu ostane; če izdelava ne uspe, potem proda granodiorit in ima torej  $k = 4$  gulde, s katerimi kupi  $x = 2$  kg granodiorita.

$$E(X | Y_2 = 1, Z_2 = 4) = 0.4 \cdot 17 + 0.6 \cdot 5 = 9.8$$

$$E(X | Y_2 = 1, Z_2 = 1) = 0.4 \cdot 5 + 0.6 \cdot 2 = 3.2$$

$$E(X | Y_2 = 1, Z_2 = 3) = 0.4 \cdot 13 + 0.6 \cdot 4 = 7.6$$

$$E(X | Y_2 = 0, Z_2 = 2) = 0.4 \cdot 8 + 0.6 \cdot 2 = 4.4$$

Izračunajmo pričakovne vrednosti še za prvo noč. Obravnavali bomo dva primera: v prvem kupi 2 kg granodiorita, v drugem pa 1 kg granodiorita.

$$E(X | Y_1 = 1, Z_1 = 2) = 0.4 \cdot 9.8 + 0.6 \cdot 3.2 = 5.84$$

$$E(X | Y_1 = 3, Z_1 = 1) = 0.4 \cdot 7.6 + 0.6 \cdot 4.4 = 5.68$$

Na začetku bo torej kupil 2 kg granodiorita, za kar bo porabil 4 gulde, ter oba nastavljal prvo noč. Če uspe, bo pridobil 20 g zlata, ki jih proda, in kupi 4 kg granodiorita, ki jih nastavi drugo noč. Če mu prvo noč ne uspe, potem proda granodiorit za 2 gulda ter kupi 1 kg svežega granodiorita, ki ga nastavi drugo noč. Celoten proces odločanja je prikazan na sliki 72.

### Naloga 3.23.

- (a) Naj slučajna spremenljivka  $X$  predstavlja dnevni zaslužek,  $n$  pa naj bo število naročenih časopisov. Izračunajmo pričakovane zaslužke v odvisnosti od  $n$ .

$$E(X | n \leq 6) = n \cdot 1\text{€} - n \cdot 0.8\text{€} \leq 1.2\text{€}$$

$$E(X | n = 7) = (0.1 \cdot 6 + 0.9 \cdot 7) \cdot 1\text{€} - 7 \cdot 0.8\text{€} = 1.3\text{€}$$

$$E(X | n = 8) = (0.1 \cdot 6 + 0.2 \cdot 7 + 0.7 \cdot 8) \cdot 1\text{€} - 8 \cdot 0.8\text{€} = 1.2\text{€}$$

$$E(X | n = 9) = (0.1 \cdot 6 + 0.2 \cdot 7 + 0.3 \cdot 8 + 0.4 \cdot 9) \cdot 1\text{€} - 9 \cdot 0.8\text{€} = 0.8\text{€}$$

$$E(X | n \geq 10) = (0.1 \cdot 6 + 0.2 \cdot 7 + 0.3 \cdot 8 + 0.3 \cdot 9 + 0.1 \cdot 10) \cdot 1\text{€} - n \cdot 0.8\text{€} \leq 0.1\text{€}$$

Ljubo bo torej imel največji pričakovani zaslužek 1.3€ na dan, če vsakič naroči 7 časopisov.

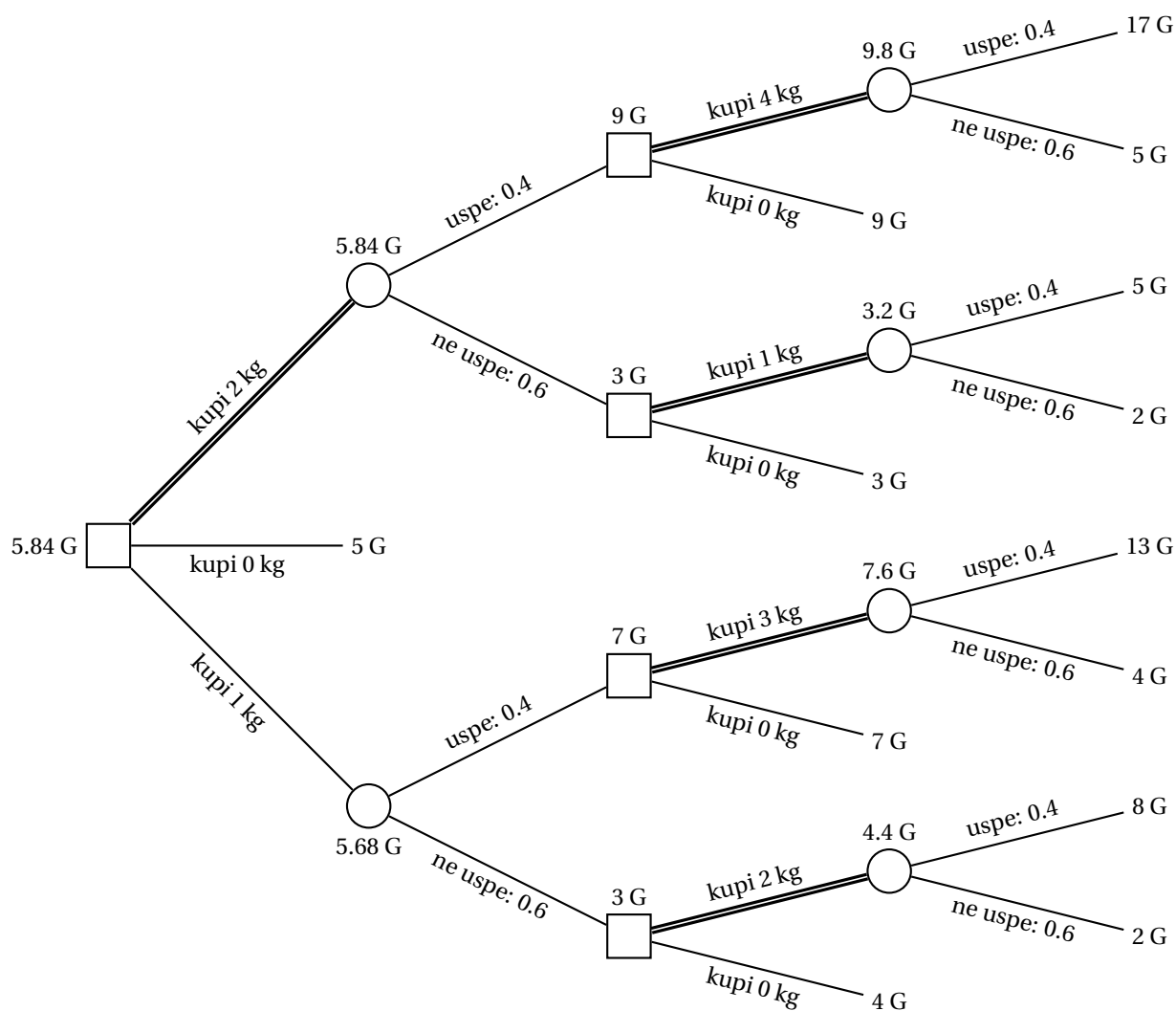
- (b) V mesecu lahko pri naročilu 7 časopisov vsak dan, ko ta izide, pričakuje zaslužek  $25 \cdot 1.3\text{€} = 32.5\text{€}$ .

### Naloga 3.24.

Definirajmo spremenljivke in izpišimo znane podatke.

$X$  ... strošek

$C$  ... izberemo pokvarjen izdelek



Slika 72: Odločitveno drevo za nalogo 3.22.

$D_i \dots$  imamo  $i$  pokvarjenih izdelkov ( $i \in \{1, 2, 3\}$ )

$E \dots$  tovarna pregleda vseh 10 izdelkov

$F \dots$  tovarna ne pregleda nobenega izdelka

$G \dots$  tovarna pregleda naključen izdelek

$$p_1 = 0.7$$

$$p_2 = 0.2$$

$$p_3 = 0.1$$

- (a) Izračunajmo, koliko tovarno stane, da pregleda vse izdelke in popravi pokvarjene. Vemo, da podjetje plača  $45\text{€} \cdot 10$  za 10 pregledov.

$$E(X | E \cap D_1) = 450\text{€} + 350\text{€} = 800\text{€}$$

$$E(X | E \cap D_2) = 450\text{€} + 350\text{€} \cdot 2 = 1\,150\text{€}$$

$$E(X | E \cap D_3) = 450\text{€} + 350\text{€} \cdot 3 = 1\,500\text{€}$$

Pričakovani strošek, če podjetje pregleda vseh 10 izdelkov, je torej

$$E(X | E) = 0.7 \cdot 800\text{€} + 0.2 \cdot 1\,150\text{€} + 0.1 \cdot 1\,500\text{€} = 940\text{€}$$

Sedaj si pogledajmo še, koliko znaša pričakovani strošek, če tovarna ne pregleda nobenega izdelka.

$$E(X | F) = 0.7 \cdot 350\text{€} + 0.2 \cdot 2 \cdot 350\text{€} + 0.1 \cdot 3 \cdot 350\text{€} = 490\text{€}$$

Če dobljena stroška primerjamo, lahko vidimo, da ima podjetje manjši strošek, če ne pregleda nobenega dobavljenega izdelka.

- (b) Strošek za pregled vseh komponent je enak kot v primeru (a), in sicer 940€. Izračunajmo še pričakovani strošek, če tovarna naključno pregleda eno komponento. Problem ločimo na tri dele: imamo en, dva ali tri pokvarjene izdelke. Vsakega od teh primerov ločimo še na dva dela: ali izberemo pokvarjen izdelek ali ne.

$$E(X | G \cap C \cap D_1) = 45\text{€}$$

$$E(X | G \cap \bar{C} \cap D_1) = 45\text{€} + 350\text{€} = 395\text{€}$$

$$E(X | G \cap D_1) = \frac{1}{10} \cdot 45\text{€} + \frac{9}{10} \cdot 395\text{€} = 360\text{€}$$

$$E(X | G \cap C \cap D_2) = 45\text{€} + 350\text{€} = 395\text{€}$$

$$E(X | G \cap \bar{C} \cap D_2) = 45\text{€} + 350\text{€} \cdot 2 = 745\text{€}$$

$$E(X | G \cap D_2) = \frac{2}{10} \cdot 395\text{€} + \frac{8}{10} \cdot 745\text{€} = 675\text{€}$$

$$E(X | G \cap C \cap D_3) = 45\text{€} + 350\text{€} \cdot 2 = 745\text{€}$$

$$E(X | G \cap \bar{C} \cap D_3) = 45\text{€} + 350\text{€} \cdot 3 = 1\,095\text{€}$$

$$E(X | G \cap D_3) = \frac{3}{10} \cdot 745\text{€} + \frac{7}{10} \cdot 1\,095\text{€} = 990\text{€}$$

Sedaj lahko izračunamo pričakovani strošek, če tovarna izbere naključen izdelek, ga pregleda in po potrebi popravi:

$$E(X | G) = 0.7 \cdot 360\text{€} + 0.2 \cdot 675\text{€} + 0.1 \cdot 990\text{€} = 486\text{€}$$

Vidimo, da velja  $E(X | G) < E(X | E)$ , zato je za tovarno boljše, da izbere eno naključno komponento za pregled, kot če pregleda vseh 10 komponent.

**Naloga 3.25.**

Očitno se nam po uspešnem poskusu ne izplača več poskušati. Tako naj slučajni spremenljivki  $X_i$  in  $Y_i$  ( $0 \leq i \leq 4$ ) predstavljata profit v primerih, ko se odločimo za  $(i+1)$ -ti poskus oziroma se odločimo končati igro po  $i$  poskusih, ob predpostavki, da so bili prejšnji poskusi neuspešni. Imamo torej

$$E(X_i) = \frac{1}{5-i} \cdot (70\text{€} - i \cdot 30\text{€}) + \frac{4-i}{5-i} \cdot \max\{E(X_{i+1}), E(Y_{i+1})\} \quad \text{in} \\ E(Y_i) = -i \cdot 30\text{€}.$$

Izračunajmo ustrezne vrednosti.

$$\begin{array}{ll} E(X_4) = -50\text{€} & E(Y_4) = -120\text{€} \\ E(X_3) = -35\text{€} & E(Y_3) = -90\text{€} \\ E(X_2) = -20\text{€} & E(Y_2) = -60\text{€} \\ E(X_1) = -5\text{€} & E(Y_1) = -30\text{€} \\ E(X_0) = 10\text{€} & E(Y_0) = 0\text{€} \end{array}$$

Vidimo, da za vse  $i$  ( $0 \leq i \leq 4$ ) velja  $E(X_i) > E(Y_i)$ , kar pomeni, da je optimalna strategija taka, da igramo, dokler ne dobimo zlatnika. Pričakovani profit je potem 10€.

**Naloga 3.26.**

Naj bo  $X_k$  ( $100 \leq k \leq 105$ ) dobiček v denotah, če družba LunaAirways proda  $k$  kart.

$$\begin{array}{ll} E(X_{100}) = 100 \cdot 300 & = 30\,000 \\ E(X_{101}) = 101 \cdot 300 - 0.2 \cdot 400 & = 30\,220 \\ E(X_{102}) = 102 \cdot 300 - 0.2 \cdot (2+1) \cdot 400 & = 30\,360 \\ E(X_{103}) = 103 \cdot 300 - 0.2 \cdot (3+2+1) \cdot 400 & = 30\,420 \\ E(X_{104}) = 104 \cdot 300 - 0.2 \cdot (4+3+2+1) \cdot 400 & = 30\,400 \\ E(X_{105}) = 105 \cdot 300 - (0.2 \cdot (5+4+3+2) + 0.1) \cdot 400 & = 30\,340 \end{array}$$

Podrobneje si pogledajmo primer, ko družba proda 103 karte. Najprej izračunamo, kolikšen dobiček ustvarimo s prodajo 103 kart, nato pa upoštevamo, da z verjetnostjo 0.2 nikogar ne zagrabi strah, torej pridejo na potovanje vsi 103 ljudje, in bodo morali sedaj trije ljudje zamenjati let. Podoben razmislek naredimo tudi za primere, če strah zagrabi enega ali dva človeka – takrat bosta morala let zamenjati dva oziroma en človek. Ker se vsi trije primeri zgodijo z isto verjetnostjo, smo to verjetnost lahko izpostavili.

Družba naj torej za maksimizacijo svojega dobička proda 103 karte.

**Naloga 3.27.**

Definirajmo spremenljivke, ki jih bomo potrebovali, in izpišimo znane podatke.

$X$  ... zaslužek

$A$  ... recept uspe

$B$  ... prosijo za nasvet

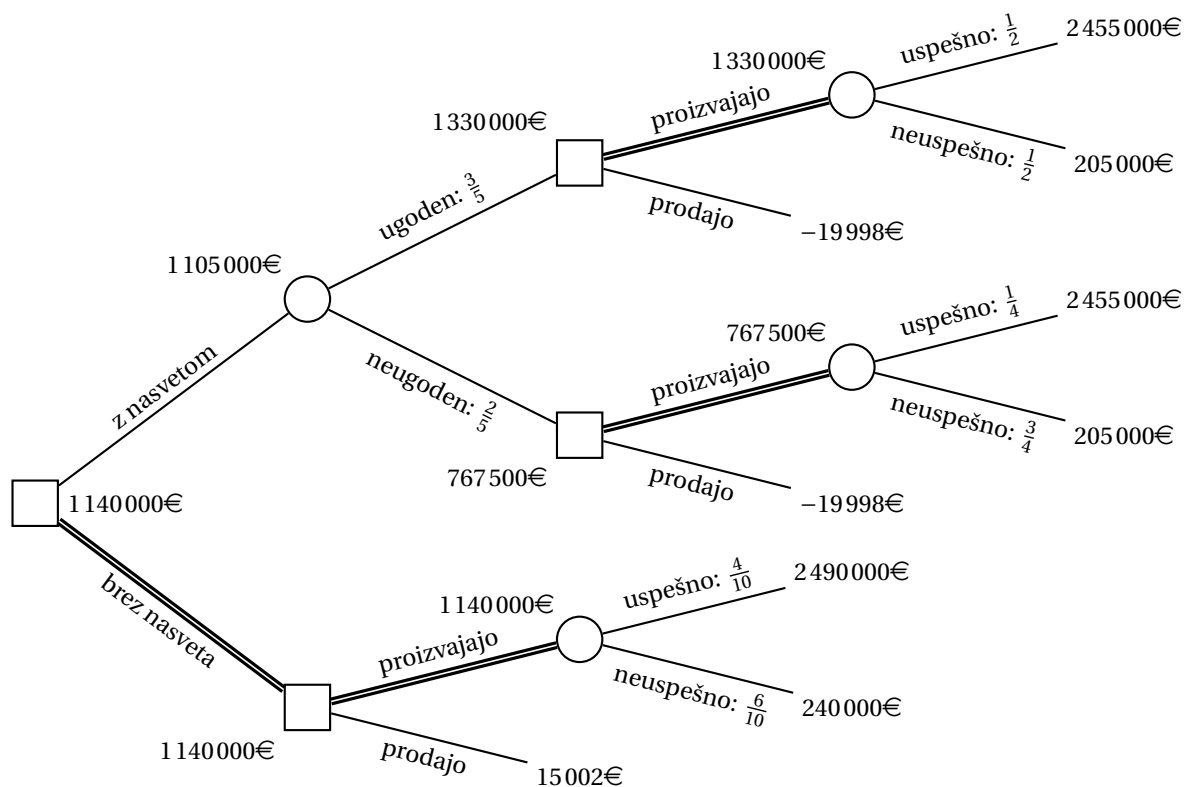
$C$  ... ugodno mnenje

$D$  ... sami proizvajajo

$$\begin{array}{ll} P(A) = \frac{2}{5} & P(\bar{A}) = \frac{3}{5} \\ P(C|A) = \frac{3}{4} & P(C|\bar{A}) = \frac{1}{2} \\ P(\bar{C}|A) = \frac{1}{4} & P(\bar{C}|\bar{A}) = \frac{1}{2} \end{array}$$

Izračunajmo verjetnosti in pričakovane vrednosti, ki jih nato vpišemo v odločitveno drevo.

$$\begin{aligned} P(C) &= P(C|A)P(A) + P(C|\bar{A})P(\bar{A}) = \frac{3}{4} \cdot \frac{2}{5} + \frac{1}{2} \cdot \frac{3}{5} = \frac{3}{5} \\ P(\bar{C}) &= P(\bar{C}|A)P(A) + P(\bar{C}|\bar{A})P(\bar{A}) = \frac{1}{4} \cdot \frac{2}{5} + \frac{1}{2} \cdot \frac{3}{5} = \frac{2}{5} \\ P(A|C) &= \frac{P(C|A)P(A)}{P(C)} = \frac{3/4 \cdot 2/5}{3/5} = \frac{1}{2} \\ P(\bar{A}|C) &= \frac{P(C|\bar{A})P(\bar{A})}{P(C)} = \frac{1/2 \cdot 3/5}{3/5} = \frac{1}{2} \\ P(A|\bar{C}) &= \frac{P(\bar{C}|A)P(A)}{P(\bar{C})} = \frac{1/4 \cdot 2/5}{2/5} = \frac{1}{4} \\ P(\bar{A}|\bar{C}) &= \frac{P(\bar{C}|\bar{A})P(\bar{A})}{P(\bar{C})} = \frac{1/2 \cdot 3/5}{2/5} = \frac{3}{4} \\ E(X|D \cap \bar{B} \cap A) &= 2490000\text{€} & P(A|\bar{B}) &= 0.4 \\ E(X|D \cap \bar{B} \cap \bar{A}) &= 240000\text{€} & P(\bar{A}|\bar{B}) &= 0.6 \\ E(X|\bar{D} \cap \bar{B}) &= 15002\text{€} \\ E(X|B \cap C \cap A) &= 2455000\text{€} & P(A|B \cap C) &= 0.5 \\ E(X|B \cap C \cap \bar{A}) &= 205000\text{€} & P(\bar{A}|B \cap C) &= 0.5 \\ E(X|\bar{D} \cap B \cap C) &= -19998\text{€} \\ E(X|B \cap \bar{C} \cap A) &= 2455000\text{€} & P(A|B \cap \bar{C}) &= 0.25 \\ E(X|B \cap \bar{C} \cap \bar{A}) &= 205000\text{€} & P(\bar{A}|B \cap \bar{C}) &= 0.75 \\ E(X|\bar{D} \cap B \cap \bar{C}) &= -19998\text{€} \end{aligned}$$



Slika 73: Odločitveno drevo za nalogo 3.27.

Ko s pomočjo odločitvenega drevesa s slike 73 izračunamo preostale pričakovane vrednosti, lahko vidimo, da velja

$$E(X | B) < E(X | \bar{B}),$$

zato je za kavarno ugodneje, da ne prosi za nasvet in se odloči za proizvodnjo.

### Naloga 3.28.

Naj bo  $X_k$  ( $1000 \leq k \leq 1005$ ) dobiček v denotah, če organizatorji olimpijskih iger prodajo  $k$  kart.

$$E(X_{1000}) = 1000 \cdot 1000 = 1000000$$

$$E(X_{1001}) = 1001 \cdot 1000 - 0.01 \cdot 1400 = 1000986$$

$$E(X_{1002}) = 1002 \cdot 1000 - (0.01 \cdot 2 + 0.19) \cdot 1400 = 1001706$$

$$E(X_{1003}) = 1003 \cdot 1000 - (0.01 \cdot 3 + 0.19 \cdot 2 + 0.3) \cdot 1400 = 1002006$$

$$E(X_{1004}) = 1004 \cdot 1000 - (0.01 \cdot 4 + 0.19 \cdot 3 + 0.3 \cdot 2 + 0.4) \cdot 1400 = 1001746$$

$$E(X_{1005}) = 1005 \cdot 1000 - (0.01 \cdot 5 + 0.19 \cdot 4 + 0.3 \cdot 3 + 0.4 \cdot 2 + 0.05) \cdot 1400 = 1001416$$

Podrobneje si pogledajmo primer, ko organizatorji prodajo 1003 karte. Najprej izračunamo, kolikšen dobiček ustvarijo s prodajo 1003 kart, nato upoštevamo, da

z verjetnostjo 0.01 na tekmo pridejo vsi imetniki kart, naprej upoštevamo da z verjetnostjo 0.19 na tekmo prideta 1 002 imetnika kart, ter da z verjetnostjo 0.3 na tekmo pride 1 001 imetnik kart. V slednjih treh primerih bodo organizatorji torej morali zagotoviti sedež v VIP sekciji dvorane za tri, dva oziroma enega človeka, za vsakega pa bodo imeli 1 400 denot stroškov.

Organizatorji olimpijskih iger naj torej za maksimizacijo svojega dobička prodajo 1 003 karte.

### Naloga 3.29.

Če se podjetje odloči za samostojno prodajo (brez predhodne akcijske ponudbe), bo ob uspehu imelo  $100\,000 \cdot (3\text{€} - 0.5\text{€}) - 1\,000\text{€} = 249\,000\text{€}$  dobička, ob neuspehu pa  $10\,000 \cdot 3\text{€} - 100\,000 \cdot 0.5\text{€} - 1\,000\text{€} = -21\,000\text{€}$ , torej 21 000€ izgube. Ker je pričakovani dobiček enak

$$0.7 \cdot 249\,000\text{€} - 0.3 \cdot 21\,000\text{€} = 168\,000\text{€},$$

kar je več kot  $100\,000 \cdot 1\text{€} = 100\,000\text{€}$ , kolikor bi zaslužili ob prodaji multinacionalki, se podjetju v primeru, da se ne odločijo za akcijsko ponudbo, bolj izplača samostojna prodaja.

Izračunajmo še dobičke, če se odločijo za akcijsko ponudbo:

Akcija uspe, produkt uspe:

$$1\,000 \cdot 2.5\text{€} + 99\,000 \cdot 3\text{€} - 100\,000 \cdot 0.5\text{€} - 2 \cdot 1\,000\text{€} = 247\,500\text{€}$$

Akcija uspe, produkt ne uspe:

$$1\,000 \cdot 2.5\text{€} + 9\,000 \cdot 3\text{€} - 100\,000 \cdot 0.5\text{€} - 2 \cdot 1\,000\text{€} = -22\,500\text{€}$$

Akcija uspe, prodajo multinacionalki:

$$1\,000 \cdot (2.5\text{€} - 0.5\text{€}) + 99\,000 \cdot 1\text{€} - 1\,000\text{€} = 100\,000\text{€}$$

Akcija ne uspe, produkt uspe:

$$100 \cdot 2.5\text{€} + 99\,900 \cdot 3\text{€} - 100\,000 \cdot 0.5\text{€} - 2 \cdot 1\,000\text{€} = 247\,950\text{€}$$

Akcija ne uspe, produkt ne uspe:

$$100 \cdot 2.5\text{€} + 9\,900 \cdot 3\text{€} - 100\,000 \cdot 0.5\text{€} - 2 \cdot 1\,000\text{€} = -22\,050\text{€}$$

Akcija ne uspe, prodajo multinacionalki:

$$100 \cdot 2.5\text{€} - 1\,000 \cdot 0.5\text{€} - 1\,000\text{€} + 99\,000 \cdot 1\text{€} = 97\,750\text{€}$$

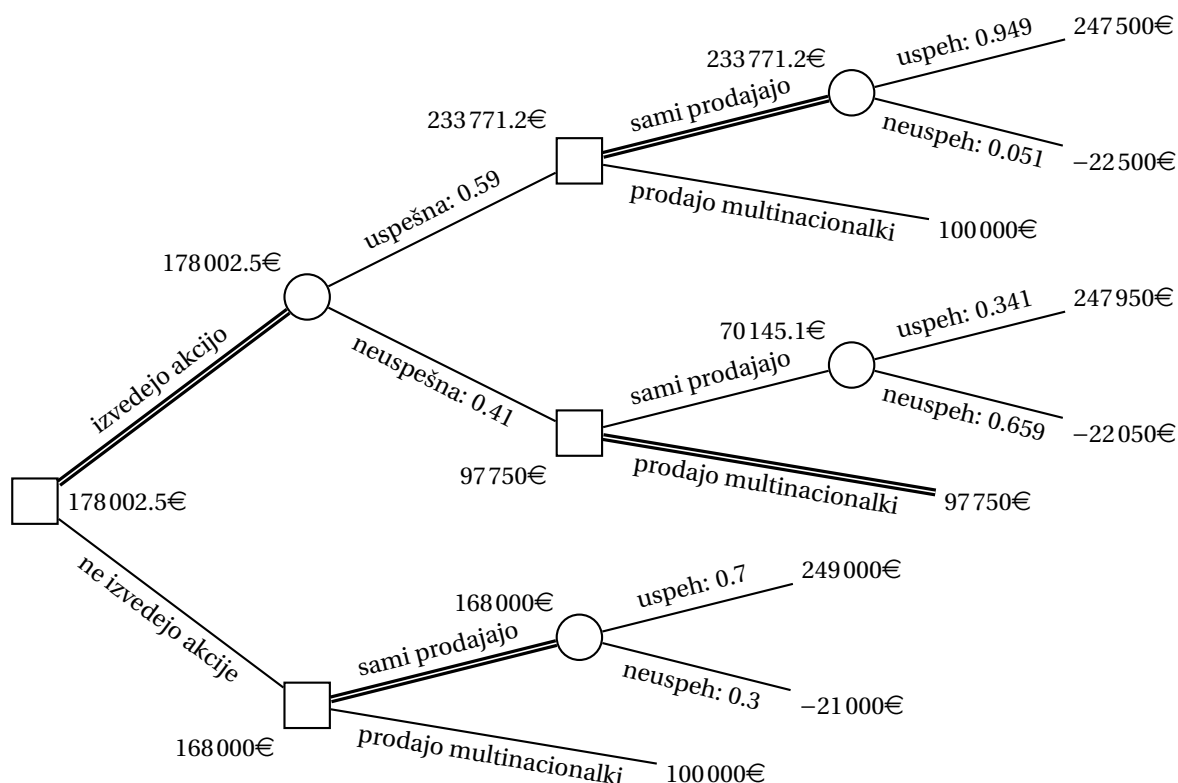
Izračunajmo še verjetnosti uspeha akcije ter uspešnosti produkta ob vsaki napovedi.

$$P(\text{akcija uspešna}) = 0.7 \cdot 0.8 + 0.3 \cdot 0.1 = 0.59$$

$$P(\text{akcija neuspešna}) = 0.7 \cdot 0.2 + 0.3 \cdot 0.9 = 0.41$$

$$P(\text{produkt uspe} \mid \text{akcija uspešna}) = \frac{0.7 \cdot 0.8}{0.59} = \frac{56}{59} \approx 0.949$$

$$P(\text{produkt ne uspe} \mid \text{akcija uspešna}) = \frac{0.3 \cdot 0.1}{0.59} = \frac{3}{59} \approx 0.051$$



Slika 74: Odločitveno drevo za nalogo 3.29.

$$P(\text{produkt uspe} \mid \text{akcija neuspešna}) = \frac{0.7 \cdot 0.2}{0.41} = \frac{14}{41} \approx 0.341$$

$$P(\text{produkt ne uspe} \mid \text{akcija neuspešna}) = \frac{0.3 \cdot 0.9}{0.41} = \frac{27}{41} \approx 0.659$$

S pomočjo zgoraj izračunanih verjetnosti lahko narišemo odločitveno drevo s slike 74. Opazimo, da se podjetju izplača izvesti akcijo, saj je pričakovani dobiček v tem primeru 178002.5€. Če akcija uspe, naj se odločijo za samostojno prodajo, če pa ne uspe, pa naj zaloge prodajo multinacionalki.

### Naloga 3.30.

Izračunajmo verjetnosti za potek servisa in stanje strojev po izvedenem servisu.

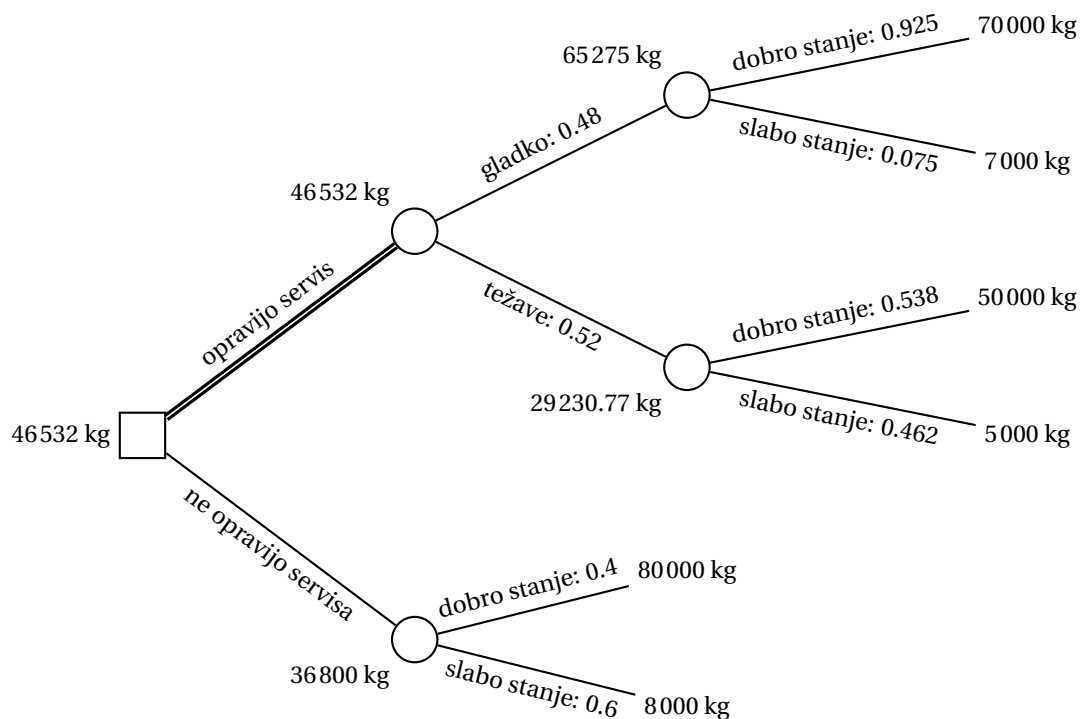
$$P(\text{servis poteka gladko}) = 0.9 \cdot 0.4 + 0.2 \cdot 0.6 = 0.48$$

$$P(\text{servis se zavleče}) = 0.1 \cdot 0.4 + 0.8 \cdot 0.6 = 0.52$$

$$P(\text{stroji v dobrem stanju} \mid \text{servis poteka gladko}) = \frac{0.9 \cdot 0.4 + 0.2 \cdot 0.6 \cdot 0.7}{0.48} = 0.925$$

$$P(\text{stroji v slabem stanju} \mid \text{servis poteka gladko}) = \frac{0.2 \cdot 0.6 \cdot 0.3}{0.48} = 0.075$$





Slika 75: Odločitveno drevo za nalogo 3.30.

$$P(\text{stroji v dobrem stanju} \mid \text{servis se zavleče}) = \frac{0.1 \cdot 0.4 + 0.8 \cdot 0.6 \cdot 0.5}{0.52} \approx 0.538$$

$$P(\text{stroji v slabem stanju} \mid \text{servis se zavleče}) = \frac{0.8 \cdot 0.6 \cdot 0.5}{0.52} \approx 0.462$$

S pomočjo zgoraj izračunanih verjetnosti lahko narišemo odločitveno drevo s slike 75. Opazimo, da se podjetju izplača opraviti servis, pričakovana količina proizvedene čokolade pa je tedaj 46532 kg.

### Naloga 3.31.

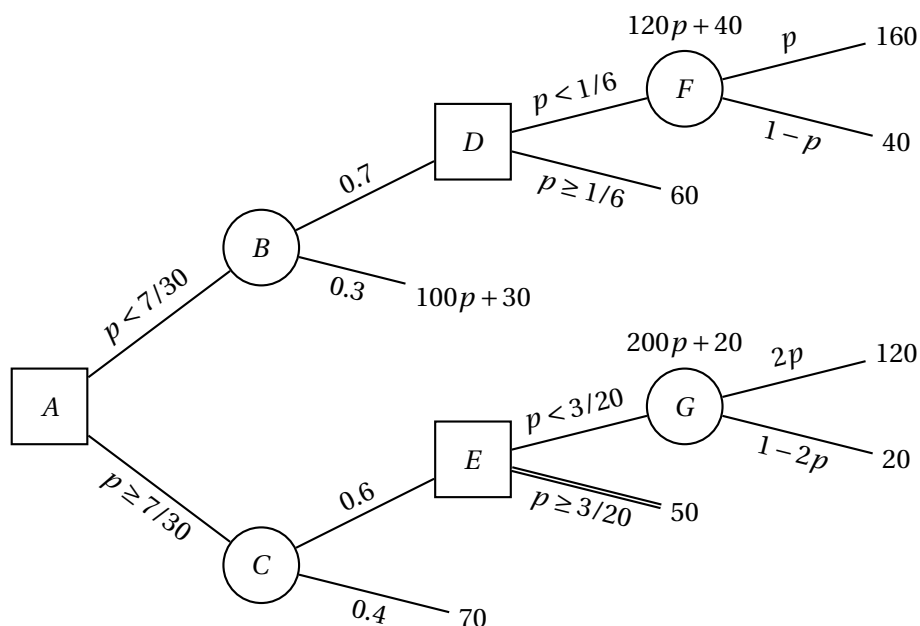
Izračunajmo najprej pričakovane vrednosti v vozliščih  $F$  in  $G$  odločitvenega drevesa s slike 9.

$$F = 160 \cdot p + 40 \cdot (1 - p) = 120p + 40$$

$$G = 120 \cdot 2p + 20 \cdot (1 - 2p) = 200p + 20$$

Obravnavajmo najprej odločitev v vozlišču  $D$ . Za pot k vozlišču  $F$  se odločimo, če velja  $120p + 40 < 60$  oziroma  $p < 1/6$ .

Obravnavajmo sedaj še odločitev v vozlišču  $E$ . Za pot k vozlišču  $G$  se odločimo, če velja  $200p + 20 < 50$  oziroma  $p < 3/20$ .



Slika 76: Odločitveno drevo za nalogo 3.31 v odvisnosti od vrednosti parametra  $p$ .

Pričakovana vrednost v vozlišču  $B$  bo sledeča.

$$\begin{aligned} p < \frac{1}{6}: & \quad 0.7 \cdot (120p + 40) + 0.3 \cdot (100p + 30) = 114p + 37 \\ p \geq \frac{1}{6}: & \quad 0.7 \cdot 60 + 0.3 \cdot (100p + 30) = 30p + 51 \end{aligned}$$

Izračunajmo še pričakovano vrednost v vozlišču  $C$ .

$$\begin{aligned} p < \frac{3}{20}: & \quad 0.6 \cdot (200p + 20) + 0.4 \cdot 70 = 120p + 40 \\ p \geq \frac{3}{20}: & \quad 0.6 \cdot 50 + 0.4 \cdot 70 = 58 \end{aligned}$$

Nazadnje obravnavajmo še odločitev v vozlišču  $A$ .

$$\begin{aligned} 0 \leq p < \frac{3}{20}: & \quad \min\{114p + 37, 120p + 40\} = 114p + 37 \\ \frac{3}{20} \leq p < \frac{1}{6}: & \quad \min\{114p + 37, 58\} = 114p + 37 \\ \frac{1}{6} \leq p \leq \frac{1}{2}: & \quad \min\{30p + 51, 58\} = \begin{cases} 30p + 51 & p < \frac{7}{30} \\ 58 & p \geq \frac{7}{30} \end{cases} \end{aligned}$$

Proces odločanja je prikazan na sliki 76. Ker velja  $\frac{7}{30} > \frac{3}{20}$ , opazimo, da v primeru  $p \geq \frac{7}{30}$  iz vozlišča  $E$  nikoli ne izberemo poti proti vozlišču  $G$ .

**Naloga 3.32.**

(a) Naj bo  $X_k$  ( $200 \leq k \leq 203$ ) dobiček, če družba proda  $k$  vozovnic.

$$E(X_{200}) = 200 \cdot 35\text{€} = 7000\text{€}$$

$$E(X_{201}) = 201 \cdot 35\text{€} - 0.3 \cdot 60\text{€} = 7017\text{€}$$

$$E(X_{202}) = 202 \cdot 35\text{€} - (0.3 \cdot 2 + 0.4) \cdot 60\text{€} = 7010\text{€}$$

$$E(X_{203}) = 203 \cdot 35\text{€} - (0.3 \cdot 3 + 0.4 \cdot 2 + 0.2) \cdot 60\text{€} = 6991\text{€}$$

Vidimo, da se družbi najbolj izplača prodati 201 vozovnico.

(b) Naj bo  $Y_k$  ( $200 \leq k \leq 203$ ) dobiček, če družba proda  $k$  vozovnic, pri čemer lahko uporabi tudi dodatno mesto.

$$E(Y_{200}) = 200 \cdot 35\text{€} = 7000\text{€}$$

$$E(Y_{201}) = 201 \cdot 35\text{€} - 0.3 \cdot 0.001 \cdot 40000\text{€} = 7023\text{€}$$

$$E(Y_{202}) = 202 \cdot 35\text{€} - 0.3 \cdot 60\text{€} - 0.7 \cdot 0.001 \cdot 40000\text{€} = 7024\text{€}$$

$$E(Y_{203}) = 203 \cdot 35\text{€} - (0.3 \cdot 2 + 0.4) \cdot 60\text{€} - 0.9 \cdot 0.001 \cdot 40000\text{€} = 7009\text{€}$$

Največji pričakovan dobiček je dosežen pri prodaji 202 vozovnic. Ker je ta večji kot v primeru, ko se dodatno mesto ne uporabi, se družbi torej to mesto izplača uporabiti.

**Naloga 3.33.**

Če se Janez odloči, da bo takoj zaprosil za gradbeno dovoljenje, bo pričakovano število dni do začetka gradnje enako

$$\frac{1}{2} \cdot 30 + \frac{1}{2} \cdot 55 = 42.5,$$

kar je manj kot 45 dni, ki bi jih potreboval, če bi soglasja pridobil pred vložitvijo prošnje. Brez Binetovega mnenja se Janezu torej izplača takoj zaprositi za gradbeno dovoljenje.

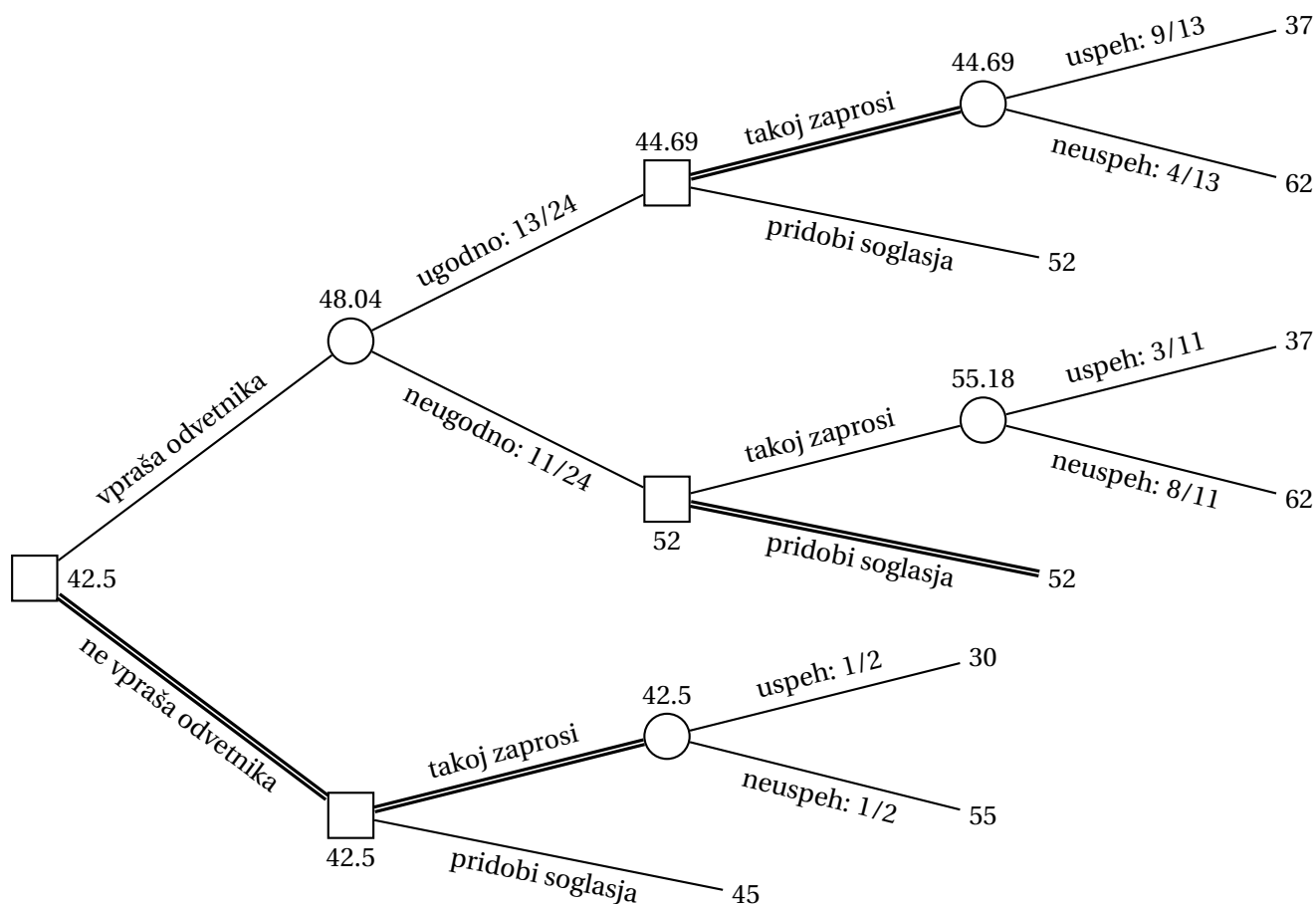
Izračunajmo še verjetnosti za Binetovo mnenje ter potrebe po soglasjih ob vsakem mnenju.

$$P(\text{ugodno mnenje}) = \frac{1}{3} \cdot \frac{1}{2} + \frac{3}{4} \cdot \frac{1}{2} = \frac{13}{24}$$

$$P(\text{neugodno mnenje}) = \frac{2}{3} \cdot \frac{1}{2} + \frac{1}{4} \cdot \frac{1}{2} = \frac{11}{24}$$

$$P(\text{soglasja so potrebna} \mid \text{ugodno mnenje}) = \frac{1/3 \cdot 1/2}{13/24} = \frac{4}{13}$$

$$P(\text{soglasja niso potrebna} \mid \text{ugodno mnenje}) = \frac{3/4 \cdot 1/2}{13/24} = \frac{9}{13}$$



Slika 77: Odločitveno drevo za nalogo 3.33.

$$P(\text{soglasja so potrebna} \mid \text{neugodno mnenje}) = \frac{2/3 \cdot 1/2}{11/24} = \frac{8}{11}$$

$$P(\text{soglasja niso potrebna} \mid \text{neugodno mnenje}) = \frac{1/4 \cdot 1/2}{11/24} = \frac{3}{11}$$

S pomočjo zgoraj izračunanih verjetnosti lahko narišemo odločitveno drevo s slike 77. Opazimo, da je pričakovani čas ob pridobitvi Binetovega mnenja večji kot 42.5 dni, zato naj Janez takoj zaprosi za gradbeno dovoljenje.

### Naloga 3.34.

Ker je  $t \leq 15$ , bo trajanje celotnega postopka v primerih, da se Janez pred sprožitvijo postopka odloči za pridobitev soglasij, ostalo nespremenjeno glede na trajanje iz naloge 3.33. Ker velja tudi  $t \geq 7$ , bo celoten postopek v primeru, da soglasja niso potrebna, trajal  $30 + t$  dni, v nasprotnem primeru pa  $55 + t$  dni – oboje neodvisno od tega, ali Janez vpraša Bineta za mnenje.

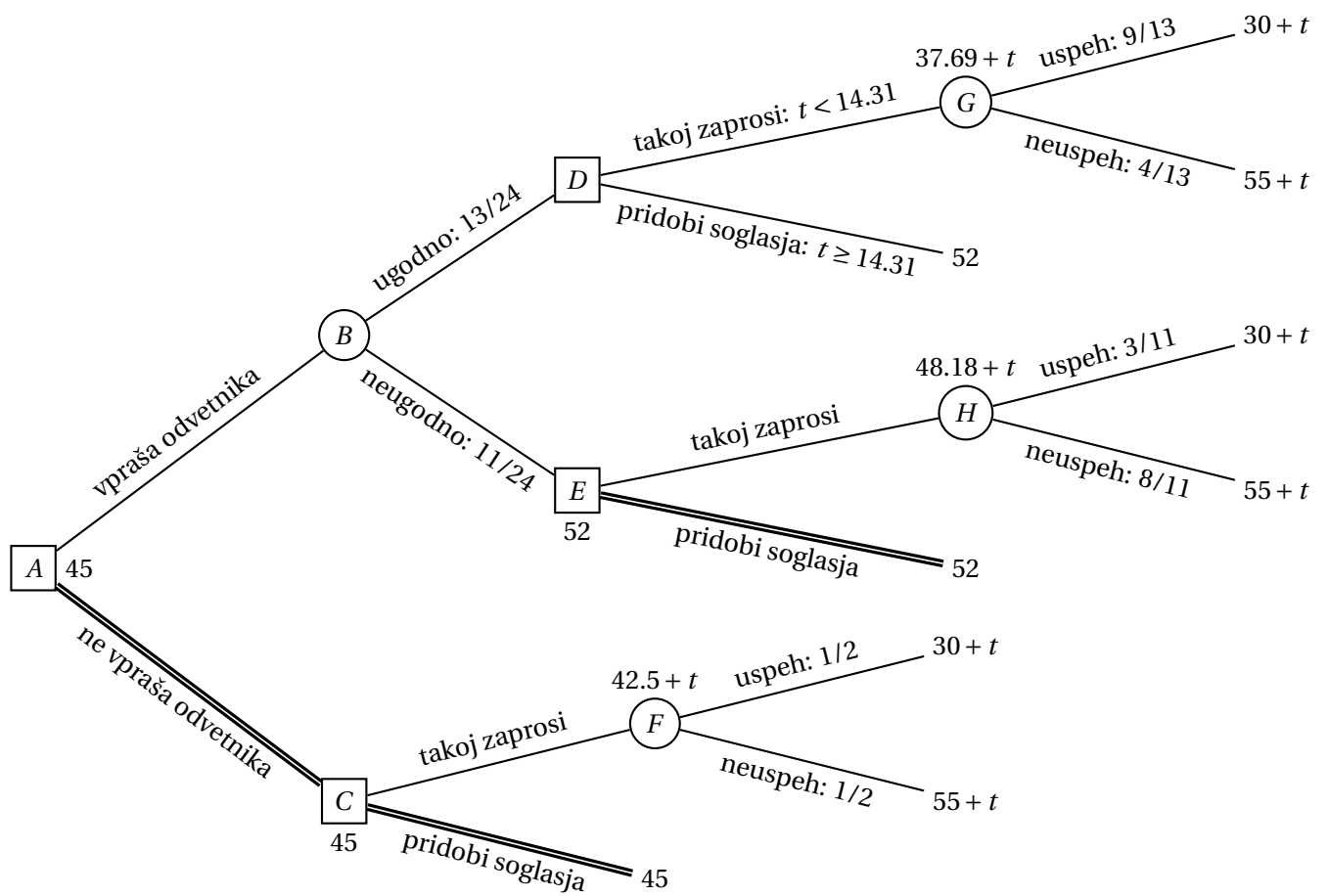
S pomočjo zgoraj izračunanih trajanj in verjetnosti z odločitvenega drevesa na sliki 77 lahko narišemo odločitveno drevo s slike 78. Opazimo, da je v primerih, če Janez ne pridobi Binetovega mnenja ali če je to neugodno, pričakovani čas trajanja celotnega postopka ob takojšnji prošnji za gradbeno dovoljenje večji kot v primeru, če pred tem pridobi soglasja. Če pa Janez od Bineta pridobi ugodno mnenje, se mu bolj izplača takoj zaprositi za gradbeno dovoljenje, natanko tedaj, ko velja  $t < 14.31$ . Izračunajmo torej pričakovano trajanje v vozlišču  $B$ :

$$E(B) = \frac{13}{24}E(D) + \frac{11}{24} \cdot 52 = \begin{cases} 44.25 + \frac{13}{24}t & \text{če } t < 14.31 \\ 52 & \text{če } t \geq 14.31 \end{cases}$$

Poglejmo si, za katere vrednosti  $t$  se Janezu izplača prositi Bineta za mnenje.

$$\begin{aligned} 44.25 + \frac{13}{24}t &< 45 \\ t &< \frac{18}{13} < 7 \end{aligned}$$

Vidimo, da zgornja neenakost ne velja za noben  $t \in [7, 15]$ , tako da se Janezu torej ne izplača pridobiti Binetovega mnenja, pač pa naj najprej pridobi soglasja sosedov. Celoten postopek bo tako trajal 45 dni.



Slika 78: Odločitveno drevo za nalogo 3.34.

## 2.4 Dinamično programiranje

### Naloga 4.1.

- (a) Naj bo  $p_i$  največja skupna profitabilnost, če obravnavamo le plakata mesta do  $x_i$ . Poleg tega naj bo  $y_i$  indeks zadnjega plakata mesta, ki je od trenutnega oddaljeno največ  $d$  kilometrov. Določimo začetne pogoje in rekurzivne enačbe.

$$y_0 = 0, \quad p_0 = 0$$

$$y_i = \max\{j \mid y_{i-1} \leq j \leq i, x_j \leq x_i - d\}, \quad p_i = \max\{p_{i-1}, v_i + p_{y_i}\} \quad (1 \leq i \leq n)$$

Vrednosti  $p_i$  računamo naraščajoče po indeksu  $i$  ( $1 \leq i \leq n$ ). Optimalno skupno profitabilnost dobimo kot  $p^* = p_n$ .

- (b) Po rekurzivni enačbi iz točke (a) izračunajmo vrednosti  $y_i$  in  $p_i$  ( $0 \leq i \leq n$ ).

$$\begin{array}{lll} y_0 = 0 & p_0 = 0 & \\ y_1 = 0 & p_1 = \max\{0, \underline{8} + 0\} & = 8 \\ y_2 = 0 & p_2 = \max\{\underline{8}, 8 + 0\} & = 8 \\ y_3 = 2 & p_3 = \max\{8, \underline{12} + 8\} & = 20 \\ y_4 = 2 & p_4 = \max\{\underline{20}, 10 + 8\} & = 20 \\ y_5 = 2 & p_5 = \max\{\underline{20}, 7 + 8\} & = 20 \\ y_6 = 3 & p_6 = \max\{20, \underline{5} + \underline{20}\} & = 25 \\ y_7 = 5 & p_7 = \max\{25, \underline{6} + \underline{20}\} & = 26 \\ y_8 = 6 & p_8 = \max\{26, \underline{10} + \underline{25}\} & = 35 \end{array}$$

Optimalna skupna profitabilnost je torej  $p^* = p_8 = 35$ , dobimo pa jo tako, da sledimo izbranim vrednostim (zgoraj so podčrtane). Na  $x_8$  postavimo plakat; ker je  $y_8 = 6$ , ugotovimo, da na  $x_6$  postavimo plakat; ker je  $y_6 = 3$ , ugotovimo, da na  $x_3$  postavimo plakat; ker je  $y_3 = 2$ , ugotovimo, da na  $x_2$  ne postavimo plakata in ga postavimo na  $x_1$ . Izbrane lokacije so torej  $x_1, x_3, x_6$  in  $x_8$ .

- (c) Ker je  $(x_i)_{i=1}^n$  naraščajoče zaporedje, opazimo, da lahko vrednosti  $y_i$  računamo tako, da povečujemo predhodni indeks, dokler ga še lahko. Ker vrednost  $y_i$  potrebujemo le še v  $(i+1)$ -tem koraku, bomo hranili samo trenutno vrednost  $y$ . Poleg  $p_i$  ( $1 \leq i \leq n$ ) bomo računali še vrednosti

$$j_i = \begin{cases} y_i & \text{če izberemo postavitev plakata na } i\text{-to mesto, in} \\ i-1 & \text{sicer;} \end{cases} \quad \text{ter}$$

$$b_i = \begin{cases} \text{TRUE} & \text{če izberemo postavitev plakata na } i\text{-to mesto, in} \\ \text{FALSE} & \text{sicer.} \end{cases}$$

Iz teh vrednosti bomo lahko rekonstruirali optimalno postavitev plakatov. Zapišimo algoritem, ki vrne optimalno skupno profitabilnost ter seznam lokacij, kamor naj postavimo plakate.

```

function PLAKATI( $((x_i)_{i=1}^n, (v_i)_{i=1}^n, d)$ )
   $p_0 \leftarrow 0$ 
   $y \leftarrow 0$ 
   $S \leftarrow$  prazen seznam
  for  $i = 1, \dots, n$  do
    while  $x_{y+1} \leq x_i - d$  do                                izračunamo  $y_i$ 
       $y \leftarrow y + 1$ 
    end while
     $p_i, j_i, b_i \leftarrow \max\{(p_{i-1}, i-1, \text{FALSE}), (v_i + p_y, y, \text{TRUE})\}$     rekurzivna enačba
  end for
   $P \leftarrow$  prazen seznam
   $i \leftarrow n$ 
  while  $i > 0$  do                                            rekonstruiramo rešitev
    if  $b_i$  then
       $P.append(x_i)$                                            postavimo plakat na mesto  $x_i$ 
    end if
     $i \leftarrow j_i$ 
  end while
   $P.reverse()$                                                 obrnemo, da bo seznam naraščajoč
  return  $(p_n, P)$                                            vrnemo profitabilnost in lokacije
end function

```

Opazimo, da obe zunanji zanki naredita  $O(n)$  korakov. Največ toliko korakov naredimo tudi v notranji zanki **while**, saj vrednost  $y$  samo povečujemo, ta pa nikdar ne preseže  $n$ . Časovna zahtevnost algoritma je torej  $O(n)$ .

#### Naloga 4.2.

- (a) Naj bosta  $I_j$  in  $c_j$  optimalna izbira predmetov ter njihova skupna vrednost, če imamo nahrbtnik z nosilnostjo  $j$  kilogramov. Določimo začetni pogoj in rekurzivno enačbo.

$$(c_0, I_0) = (0, \emptyset)$$

$$(c_j, I_j) = \max\left(\{(c_{j-1}, I_{j-1})\} \cup \{(c_{j-t_i} + v_i, I_{j-t_i} \cup \{i\}) \mid 1 \leq i \leq n, t_i \leq j, i \notin I_{j-t_i}\}\right)$$

Vrednosti  $c_j$  in  $I_j$  računamo naraščajoče po indeksu  $j$  ( $1 \leq j \leq M$ ). Optimalno skupno vrednost dobimo kot  $c^* = c_M$ , optimalno izbiro predmetov pa kot  $I^* = I_M$ .

- (b) Potek algoritma je prikazan v tabeli 15. V vsaki vrstici je podčrtana vrednost pri tistem predmetu, ki ga dodamo v ustreznem koraku. Če nobena vrednost ni podčrtana, do izboljšave ni prišlo in zato uporabimo predhodno rešitev. Preberemo lahko, da je optimalna izbira predmetov 3, 4 in 5 s skupno vrednostjo 29.



$j$	$c_j$	$v_i$	9	9	8	11	10	15	3	12
		$t_i$	3	5	1	4	3	8	2	7
		$I_j$	1	2	3	4	5	6	7	8
0	0	$\emptyset$								
1	8	{3}			<u>0+8</u>					
2	8	{3}							0+3	
3	11	{3,7}	0+9				0+10		<u>8+3</u>	
4	18	{3,5}	8+9			0+11	<u>8+10</u>		8+3	
5	19	{3,4}	8+9	0+9		<u>8+11</u>	8+10			
6	21	{3,5,7}	11+9	8+9		8+11	<u>11+10</u>		18+3	
7	27	{1,3,5}	<u>18+9</u>	8+9		11+11			19+3	0+12
8	29	{3,4,5}	19+9	11+9		<u>18+11</u>	19+10	0+15		8+12

Tabela 15: Potek reševanja za nalogo 4.2(b).

### Naloga 4.3.

- (a) Naj bo  $p_{ij}$  cena minimalne vsote poti od  $a_{11}$  do  $a_{ij}$ . Določimo začetni pogoj in rekurzivne enačbe.

$$p_{11} = a_{11}$$

$$p_{1j} = a_{1j} + p_{i,j-1} \quad (2 \leq j \leq n)$$

$$p_{i1} = a_{i1} + p_{i-1,1} \quad (2 \leq i \leq m)$$

$$p_{ij} = a_{ij} + \min\{p_{i-1,j}, p_{i,j-1}\} \quad (2 \leq i \leq m, 2 \leq j \leq n)$$

Vrednosti  $p_{ij}$  računamo v leksikografskem vrstnem redu indeksov (npr. najprej naraščajoče po  $i$ , nato pa naraščajoče po  $j$ ). Minimalno vsoto poti dobimo kot  $p^* = p_{mn}$ .

- (b) Vrednosti  $p_{ij}$  ( $1 \leq i \leq m$ ,  $1 \leq j \leq n$ ) zapišimo v matriki skupaj z informacijo, katero predhodno vrednost smo upoštevali.

$$(p_{ij})_{i,j=1}^5 = \begin{pmatrix} 131 & \leftarrow 804 & \leftarrow 1038 & \leftarrow 1141 & \leftarrow 1159 \\ \uparrow 332 & \leftarrow 428 & \leftarrow 770 & \leftarrow 1735 & \uparrow 1309 \\ \uparrow 962 & \uparrow 1231 & \uparrow 1516 & \leftarrow 1938 & \uparrow 1420 \\ \uparrow 1499 & \uparrow 1930 & \uparrow 2013 & \uparrow 2059 & \uparrow 2376 \\ \uparrow 2304 & \uparrow 2662 & \uparrow 2537 & \uparrow 2096 & \leftarrow 2428 \end{pmatrix}$$

Vidimo lahko, da je optimalna pot  $(a_{11}, a_{21}, a_{22}, a_{23}, a_{33}, a_{34}, a_{44}, a_{54}, a_{55})$  z vsoto 2428.

- (c) Da bomo lahko rekonstruirali pot, bomo hranili še vrednosti  $k_{ij}$  z indeksi predhodne lokacije.

$$\begin{aligned} k_{1j} &= (1, j-1) & (2 \leq j \leq n) \\ k_{i1} &= (i-1, 1) & (2 \leq i \leq m) \\ k_{ij} &= \begin{cases} (i-1, j) & \text{če } p_{i-1,j} \leq p_{i,j-1}, \text{ in} \\ (i, j-1) & \text{sicer} \end{cases} & (2 \leq i \leq m, 2 \leq j \leq n) \end{aligned}$$

Zapišimo algoritem, ki vrne minimalno vsoto in pot, ki jo doseže.

```

function MATRIXSUM( $(a_{ij})_{i,j=1}^{m,n}$ )
  for  $i = 1, \dots, m$  do
    for  $j = 1, \dots, n$  do
      if  $i = 1 \wedge j = 1$  then                                rekurzivna enačba
         $s \leftarrow 0$ 
      else if  $i = 1$  then
         $s \leftarrow p_{i,j-1}$ 
         $k_{ij} = (i, j-1)$ 
      else if  $j = 1$  then
         $s \leftarrow p_{i-1,j}$ 
         $k_{ij} = (i-1, j)$ 
      else
         $(s, k_{ij}) \leftarrow \min\{(p_{i,j-1}, (i, j-1)), (p_{i-1,j}, (i-1, j))\}$ 
      end if
       $p_{ij} = a_{ij} + s$ 
    end for
  end for
   $P \leftarrow$  prazen seznam
   $(i, j) \leftarrow (m, n)$ 
  while  $i > 1 \vee j > 1$  do                                rekonstruiramo rešitev
     $P.append((i, j))$                                        postavimo plakat na mesto  $x_i$ 
     $(i, j) \leftarrow k_{i,j}$ 
  end while
   $P.append((1, 1))$                                        dodamo še levi zgornji kot
   $P.reverse()$                                            obrnemo, da dobimo pot v pravem vrstnem redu
  return  $(p_{mn}, P)$                                        vrnemo vsoto in pot
end function

```

Dvojna zanka **for** opravi  $mn$  korakov, zanka **while** pa opravi  $m + n - 2$  korakov. Časovna zahtevnost algoritma je torej  $O(mn)$ .

#### Naloga 4.4.

Naj  $b_{ij}$  pove, ali je strnjen podniz  $a_i a_{i+1} \dots a_{j-2} a_{j-1}$  palindromski, torej, ali velja  $a_{i+k-1} = a_{j-k}$  za vsak  $k$  ( $1 \leq k \leq j-i$ ). Ker so vsi podnizi dolžin 0 in 1 palindromski, lahko določimo začetne pogoje in rekurzivno enačbo.

$$\begin{aligned}
b_{i,i} &= \top & (1 \leq i \leq n+1) \\
b_{i,i+1} &= \top & (1 \leq i \leq n) \\
b_{i,j} &= b_{i+1,j-1} \wedge (a_i = a_{j-1}) & (1 \leq i < j-1 \leq n)
\end{aligned}$$

Vrednosti  $b_{ij}$  računamo v leksikografskem vrstnem redu glede na  $(j-i, i)$ . Dolžino  $p^*$  najdaljšega palindromskega strnjenega podniza dobimo kot največjo razliko indeksov, za katere je  $b_{ij}$  resničen, torej

$$p^* = \max \{j-i \mid 1 \leq i \leq j \leq n+1 \wedge b_{ij}\}.$$

Hkrati lahko seveda dobimo tudi začetek in konec najdaljšega palindromskega strnjenega podniza.

Zapišimo algoritem, ki vrne dolžino najdaljšega palindromskega strnjenega podniza  $a_i a_{i+1} \dots a_{j-2} a_{j-1}$  skupaj z indeksoma  $i$  in  $j$ . Vrednost  $p^*$  ter pripadajoča indeksa bomo računali sproti.

```

function PALINDROM( $(a_i)_{i=1}^n$ )
  if  $n = 0$  then                                     če je niz prazen, imamo podniz dolžine 0
    return  $(0, 1, 1)$ 
  end if
   $(p^*, i^*, j^*) \leftarrow (1, n, n+1)$                 podniz dolžine 1
  for  $i = 1, \dots, n$  do                               začetni pogoji
     $b_{ii} \leftarrow \text{TRUE}$ 
     $b_{i,i+1} \leftarrow \text{TRUE}$ 
  end for
   $b_{n+1,n+1} \leftarrow \text{TRUE}$ 
  for  $h = 2, \dots, n$  do                               rekurzivna enačba
    for  $i = 1, \dots, n-h+1$  do
       $b_{i,i+h} \leftarrow b_{i+1,i+h-1} \wedge (a_i = a_{i+h-1})$ 
      if  $b_{i,i+h}$  then                                   če najdemo palindrom, posodobimo izhod
         $(p^*, i^*, j^*) \leftarrow (h, i, i+h)$ 
      end if
    end for
  end for
  return  $(p^*, i^*, j^*)$                                 vrnemo dolžino in meje
end function

```

V prvi zanki **for** naredimo  $n$  korakov, v dvojni zanki pa

$$\sum_{h=2}^n (n-h+1) = \sum_{j=1}^{n-1} j = \frac{n(n-1)}{2} = O(n^2)$$

korakov. Časovna zahtevnost algoritma je torej  $O(n^2)$ .

Problem je mogoče reševati tudi v linearnem času z **Manacherjevim algoritmom**.

**Naloga 4.5.**

- (a) Naj bo  $p_{hik}$  največja vsota komponent strnjene podmatrike z vrsticami od  $h$ -te do  $i$ -te vrstice matrike  $A$ , katere zadnji stolpec je  $k$ -ti stolpec matrike  $A$ . Poleg tega naj bo  $s_{ij}$  vsota prvih  $i$  komponent v  $j$ -tem stolpcu matrike  $A$ . Določimo začetne pogoje in rekurzivne enačbe.

$$s_{0j} = 0 \quad (1 \leq j \leq n)$$

$$s_{ij} = a_{ij} + s_{i-1,j} \quad (1 \leq i \leq m, 1 \leq j \leq n)$$

$$p_{hi0} = 0 \quad (1 \leq h \leq i \leq m)$$

$$p_{hik} = s_{ik} - s_{h-1,k} + \max\{0, p_{h,i,k-1}\} \quad (1 \leq h \leq i \leq m, 1 \leq k \leq n)$$

Najprej za vsak  $j$  ( $1 \leq j \leq n$ ) izračunamo vrednosti  $s_{ij}$  naraščajoče po indeksu  $i$ , nato pa za vsaka  $h$  in  $i$  ( $1 \leq h \leq i \leq m$ ) izračunamo vrednosti  $p_{hik}$  naraščajoče po indeksu  $k$ . Največjo vsoto komponent dobimo kot  $p^* = \max\{p_{hik} \mid 1 \leq h \leq i \leq m, 0 \leq k \leq n\}$ .

- (b) Najprej poračunajmo vrednosti  $s_{ij}$ .

$$(s_{ij})_{i,j=1}^4 = \begin{pmatrix} 1 & -1 & 2 & 4 \\ -2 & -3 & 10 & 6 \\ -5 & -1 & 8 & 10 \\ -4 & -6 & 7 & 8 \end{pmatrix}$$

Sedaj poračunajmo še vrednosti  $p_{hik}$ .

$p_{hik}$		$k$			
$h$	$i$	1	2	3	4
1	1	1	0	2	6
1	2	-2	-3	10	16
1	3	-5	-1	8	18
1	4	-4	-6	7	15
2	2	-3	-2	8	10
2	3	-6	0	6	12
2	4	-5	-5	5	9
3	3	-3	2	0	4
3	4	-2	-2	-3	2
4	4	1	-4	-1	-2

Opazimo, da je največja vsota komponent enaka  $p^* = p_{134} = 18$ . Indeks  $j$  prvega stolpca iskane podmatrike je najmanjša taka vrednost, da za vse  $\ell$  ( $j \leq \ell \leq k$ ) velja  $p_{13\ell} > 0$ . V našem primeru je torej  $j = 3$  – iskana podmatrika je torej

$$A_{1\dots 3,3\dots 4} = \begin{pmatrix} 2 & 4 \\ 8 & 2 \\ -2 & 4 \end{pmatrix}.$$

- (c) Zapišimo algoritem, ki bo vrnil največjo vsoto komponent strnjene podmatrike skupaj s paroma, ki določata prvo in zadnjo vrstico oziroma stolpec.

```

function MAXSTRNJENAPODMATRIKA( $(a_{ij})_{i,j=1}^{m,n}$ )
  for  $j = 1, \dots, n$  do                                računanje  $s_{ij}$ 
     $s_{0j} \leftarrow 0$                                        začetni pogoj
    for  $i = 1, \dots, m$  do
       $s_{ij} \leftarrow a_{ij} + s_{i-1,j}$                     rekurzivna enačba
    end for
  end for
   $p^*, h^*, i^*, j^*, k^* \leftarrow 0, 0, 0, 0, 0$           ničelna rešitev
  for  $h = 1, \dots, n$  do                                računanje  $p_{hik}$ 
    for  $i = h, \dots, n$  do
       $p_{hi0} \leftarrow 0$                                 začetni pogoj
       $j \leftarrow 1$                                     vsoto začnemo s prvim stolpcem
      for  $k = 1, \dots, m$  do
         $p_{hik} \leftarrow s_{ik} - s_{h-1,k} + \max\{0, p_{h,i,k-1}\}$     rekurzivna enačba
        if  $p_{hik} > p^*$  then                                preverimo, ali imamo boljšo rešitev
           $p^*, h^*, i^*, j^*, k^* \leftarrow p_{hik}, h, i, j, k$ 
        else if  $p_{hik} < 0$  then                            če je vsota negativna,
           $j \leftarrow k + 1$                                 bomo naslednjo vsoto začeli v naslednjem stolpcu
        end if
      end for
    end for
  end for
  return  $p^*, (h^*, i^*), (j^*, k^*)$ 
end function

```

Za računanje  $s_{ij}$  potrebujemo  $O(mn)$  korakov, za računanje  $p_{hik}$  pa  $O(mn^2)$  korakov. Skupna časovna zahtevnost algoritma je torej  $O(mn^2)$ . Opazimo, da se v primeru, ko velja  $n > m$ , matriko  $A$  izplača transponirati, saj tako dobimo časovno zahtevnost  $O(m^2n)$ .

#### Naloga 4.6.

- (a) Naj bo  $p_j$  število kovancev, ki jih potrebujemo za izplačilo vsote  $j$ . Določimo začetni pogoj in rekurzivne enačbe.

$$\begin{aligned}
 p_0 &= 0 \\
 p_j &= 1 + \min \{p_{j-v_i} \mid 1 \leq i \leq n, v_i \leq j\} \quad (1 \leq j \leq C)
 \end{aligned}$$

Vrednosti  $p_j$  ( $0 \leq j \leq C$ ) računamo naraščajoče po indeksu  $j$ . Najmanjše število kovancev dobimo s  $p^* = p_C$ .

(b) Izračunajmo vrednosti  $p_j$  pri danih podatkih.

$$\begin{array}{ll}
 p_0 = 0 & p_{13} = 1 + \min\{2, 3, 2, 2\} = 3 \\
 p_1 = 1 + \min\{0\} = 1 & p_{14} = 1 + \min\{3, 3, 2, 1\} = 2 \\
 p_2 = 1 + \min\{1, 0\} = 1 & p_{15} = 1 + \min\{2, 3, 2, 2\} = 3 \\
 p_3 = 1 + \min\{1, 1\} = 2 & p_{16} = 1 + \min\{3, 2, 3, 2\} = 3 \\
 p_4 = 1 + \min\{2, 1\} = 2 & p_{17} = 1 + \min\{3, 3, 2, 2\} = 3 \\
 p_5 = 1 + \min\{2, 2, 0\} = 1 & p_{18} = 1 + \min\{3, 3, 3, 3\} = 4 \\
 p_6 = 1 + \min\{1, 2, 1\} = 2 & p_{19} = 1 + \min\{4, 3, 2, 2\} = 3 \\
 p_7 = 1 + \min\{2, 1, 1, 0\} = 1 & p_{20} = 1 + \min\{3, 4, 3, 3\} = 4 \\
 p_8 = 1 + \min\{1, 2, 2, 1\} = 2 & p_{21} = 1 + \min\{4, 3, 3, 2\} = 3 \\
 p_9 = 1 + \min\{2, 1, 2, 1\} = 2 & p_{22} = 1 + \min\{3, 4, 3, 3\} = 4 \\
 p_{10} = 1 + \min\{2, 2, 1, 2\} = 2 & p_{23} = 1 + \min\{4, 3, 4, 3\} = 4 \\
 p_{11} = 1 + \min\{2, 2, 2, 2\} = 3 & p_{24} = 1 + \min\{4, 4, 3, 3\} = 4 \\
 p_{12} = 1 + \min\{3, 2, 1, 1\} = 2 & p_{25} = 1 + \min\{4, 4, 4, 4\} = 5
 \end{array}$$

Za izplačilo vrednosti 25 torej potrebujemo vsaj 5 kovancev. Iz zgornjega izračuna lahko izplačilo rekonstruiramo tako, da uporabimo tak kovanec, da je za izplačilo preostanka potrebnih najmanj kovancev. Opazimo, da lahko vsoto 25 s petimi kovanci izplačamo na tri različne načine:

$$25 = 1 \cdot 1 + 2 \cdot 5 + 2 \cdot 7 = 2 \cdot 2 + 3 \cdot 7 = 5 \cdot 5.$$

#### Naloga 4.7.

Naj bo  $p_i$  največja količina denarja, ki jo lahko tat pridobi, če oropa  $i$ -to hišo. Določimo začetna pogoja in rekurzivne enačbe.

$$\begin{aligned}
 p_0 &= 0, & p_1 &= c_1 \\
 p_i &= \max\{p_{i-1}, p_{i-2} + c_i\} & (2 \leq i \leq n)
 \end{aligned}$$

Vrednosti  $p_i$  ( $0 \leq i \leq n$ ) računamo naraščajoče po indeksu  $n$ . Največjo količino denarja dobimo s  $p^* = p_n$ . Da poiščemo, katere hiše se tatu izplača oropati, pregledamo vrednosti  $p_i$  v padajočem vrstnem redu ( $n \geq i \geq 1$ ): če velja  $p_i > p_{i-1}$ , naj  $i$ -to hišo oropa in pregled nadaljujemo pri  $p_{i-2}$ , sicer pa naj  $i$ -te hiše ne oropa in pregled nadaljujemo pri  $p_{i-1}$ .

#### Naloga 4.8.

(a) Naj bo  $x_0 = 0$  in  $x_{n+1} = \ell$ , ter naj bo  $p_{ij}$  cena rezanja dela hloda od  $x_i$  do  $x_{j+1}$ . Določimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned}
p_{ii} &= 0 & (0 \leq i \leq n) \\
p_{ij} &= x_{j+1} - x_i + \min \{p_{ih} + p_{h+1,j} \mid i \leq h < j\} & (0 \leq i < j \leq n)
\end{aligned}$$

Vrednosti  $p_{ij}$  ( $0 \leq i \leq j \leq n$ ) računamo v leksikografskem vrstnem redu glede na  $(j - i, i)$ . Najmanjšo ceno rezanja dobimo kot  $p^* = p_{0n}$ .

Za izračun spremenljivke  $p_{ij}$  potrebujemo  $j - i$  korakov. Skupno število korakov algoritma je torej

$$\begin{aligned}
\sum_{i=0}^n \sum_{j=i}^n (j - i) &= \sum_{i=0}^n \sum_{j=0}^{n-i} j = \sum_{i=0}^n \frac{(n-i)(n-i+1)}{2} = \\
&= \sum_{i=0}^n \frac{i(i+1)}{2} = \frac{n(n+1)(n+2)}{6} = O(n^3).
\end{aligned}$$

(b) Izračunajmo vrednosti  $p_{ij}$  ( $0 \leq i < j \leq n$ ).

$$\begin{aligned}
p_{01} &= 5 - 0 + \min\{0 + 0\} & = 5 \\
p_{12} &= 7 - 3 + \min\{0 + 0\} & = 4 \\
p_{23} &= 8 - 5 + \min\{0 + 0\} & = 3 \\
p_{34} &= 10 - 7 + \min\{0 + 0\} & = 3 \\
p_{02} &= 7 - 0 + \min\{0 + 4, 5 + 0\} & = 11 \\
p_{13} &= 8 - 3 + \min\{0 + 3, 4 + 0\} & = 8 \\
p_{24} &= 10 - 5 + \min\{0 + 3, 3 + 0\} & = 8 \\
p_{03} &= 8 - 0 + \min\{0 + 8, 5 + 3, 11 + 0\} & = 16 \\
p_{14} &= 10 - 3 + \min\{0 + 8, 4 + 3, 8 + 0\} & = 14 \\
p_{04} &= 10 - 0 + \min\{0 + 14, 5 + 8, 11 + 3, 16 + 0\} & = 23
\end{aligned}$$

Najmanjša cena rezanja je torej 23, dosežemo pa jo tako, da najprej razrežemo na mestu  $x_2 = 5$ , nato pa en del še na mestu  $x_1 = 3$ , drugi del pa še na mestih  $x_3 = 7$  in  $x_4 = 8$  (pri slednjih dveh vrstni red ni pomemben).

#### Naloga 4.9.

Naj bo  $z_{ij}$  največji zaslužek, ki ga lahko imamo, če  $i$  zabojev razporedimo v prvih  $j$  trgovin. Določimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned}
z_{i1} &= p_{i1} & (0 \leq i \leq m) \\
z_{ij} &= \max \{z_{ih,j-1} + p_{i-h,j} \mid 0 \leq h \leq i\} & (0 \leq i \leq m, 2 \leq j \leq n)
\end{aligned}$$

Vrednosti  $z_{ij}$  računamo v leksikografskem vrstnem redu glede na  $(j, i)$ . Največji skupni zaslužek dobimo kot  $z^* = z_{mn}$ .

(a) Izračunajmo vrednosti  $z_{ij}$  ( $0 \leq i \leq m, 2 \leq j \leq n$ ).

$$\begin{aligned}
z_{02} &= 0 + 0 &= 0 \\
z_{12} &= \max\{0 + \underline{6}, 5 + 0\} &= 6 \\
z_{22} &= \max\{0 + \underline{11}, 5 + 6, 9 + 0\} &= 11 \\
z_{32} &= \max\{0 + 15, \underline{5 + 11}, 9 + 6, 14 + 0\} &= 16 \\
z_{42} &= \max\{0 + 19, \underline{5 + 15}, 9 + 11, 14 + 6, 17 + 0\} &= 20 \\
z_{52} &= \max\{0 + 22, 5 + 19, 9 + 15, \underline{14 + 11}, 17 + 6, 21 + 0\} &= 25 \\
z_{03} &= 0 + 0 &= 0 \\
z_{13} &= \max\{0 + 4, \underline{6 + 0}\} &= 6 \\
z_{23} &= \max\{0 + 9, 6 + 4, \underline{11 + 0}\} &= 11 \\
z_{33} &= \max\{0 + 13, 6 + 9, 11 + 4, \underline{16 + 0}\} &= 16 \\
z_{43} &= \max\{0 + 18, 6 + 13, \underline{11 + 9}, 16 + 4, 20 + 0\} &= 20 \\
z_{53} &= \max\{0 + 20, 6 + 18, 11 + 13, \underline{16 + 9}, 20 + 4, 25 + 0\} &= 25
\end{aligned}$$

Največji zaslužek je torej  $z^* = z_{53} = 25$ . Poiščimo še optimalno razporeditev zabojev.

$$\begin{aligned}
z_{53} &= z_{32} + p_{23} && 2 \text{ zaboja v tretjo trgovino} \\
z_{32} &= z_{11} + p_{22} && 2 \text{ zaboja v drugo trgovino} \\
z_{11} &= p_{11} && 1 \text{ zaboje v prvo trgovino}
\end{aligned}$$

Druga optimalna razporeditev je, da gredo 3 zaboji v prvo trgovino, 2 v drugo trgovino in noben v tretjo.

(b) Da bomo lahko rekonstruirali rešitev, bomo hranili še vrednosti  $k_{ij}$  ( $0 \leq i \leq m, 2 \leq j \leq n$ ), tako da bo veljalo  $z_{ij} = z_{k_{ij}, j-1} + p_{i-k_{ij}, j}$ .

```

function JAGODE( $(p_{ij})_{i,j=0,1}^{m,n}$ )
  for  $i = 0, \dots, m$  do
     $z_{i1} \leftarrow p_{i1}$ 
     $k_{i1} \leftarrow 0$ 
  end for
  for  $j = 2, \dots, n$  do
    for  $i = 0, \dots, m$  do
       $z_{ij}, k_{ij} \leftarrow \max\{(z_{h,j-1} + p_{i-h,j}, h) \mid 0 \leq h \leq i\}$ 
    end for
  end for
   $resitev \leftarrow$  tabela z  $n$  polji
   $j \leftarrow m$ 
  for  $i = n, \dots, 1$  do
     $resitev[i] \leftarrow j - k_{ij}$ 
     $j \leftarrow k_{ij}$ 
  end for
  return  $(z_{mn}, resitev)$ 

```



**end function**

Algoritem teče v času  $O(m^2 n)$ .

**Naloga 4.10.**

Naj bodo  $m(x)$ ,  $f_2(x)$  in  $f_3(x)$  začetni tržni delež oziroma deleža, ohranjena po drugi in tretji fazi, ob vložku  $x$  milijonov evrov v posamezno fazo. Uporabili bomo sledeče rekurzivne enačbe.

$$\begin{aligned}p_3(x) &= f_3(x) \\p_2(x) &= \max\{f_2(x-y)p_3(y) \mid 0 \leq y \leq x\} \\p_1(x) &= \max\{m(x-y)p_2(y) \mid 0 \leq y \leq x\}\end{aligned}$$

Optimalni tržni delež (v milijonih evrov) pri vložku 4 M€ bomo dobili kot  $p^* = p_1(4)$ .

- (a) Najprej izračunajmo  $p_2(x)$  za  $x \in \{0, 1, 2, 3\}$ . Ker je  $m(0) = 0$ , nas namreč  $p_2(4)$  ne zanima.

$$\begin{aligned}p_2(0) &= 0.2 \cdot 0.3 &= 0.06 \\p_2(1) &= \max\{0.4 \cdot 0.3, 0.2 \cdot 0.5\} &= 0.12 \\p_2(2) &= \max\{0.5 \cdot 0.3, 0.4 \cdot 0.5, 0.2 \cdot 0.6\} &= 0.2 \\p_2(3) &= \max\{0.6 \cdot 0.3, 0.5 \cdot 0.5, 0.4 \cdot 0.6, 0.2 \cdot 0.7\} &= 0.25\end{aligned}$$

Izračunajmo še  $p^* = p_1(4)$ .

$$p_1(4) = \max\{50 \cdot 0.06, 40 \cdot 0.12, 30 \cdot 0.2, 20 \cdot 0.25\} = 6$$

Optimalni končni tržni delež 6% torej dosežemo tako, da v prvo vazo vložimo 2 M€, v vsako od naslednjih dveh faz pa po 1 M€.

- (b) Najprej izračunajmo  $p_2(x)$  za  $0 \leq x \leq 4$ .

$$\begin{aligned}p_2(x) &= \max\{(0.4 + 0.1x - 0.1y)(0.6 + 0.07y) \mid 0 \leq y \leq x\} \\&= \max\{-0.007y^2 + (0.007x - 0.032)y + 0.06x + 0.24 \mid 0 \leq y \leq x\}\end{aligned}$$

Odvajajmo zgornji izraz po  $y$ , da poiščemo maksimum na intervalu  $[0, x]$ .

$$\begin{aligned}0 &= -0.014y + 0.007x - 0.032 \\y &= \frac{1}{2}x - \frac{16}{7}\end{aligned}$$

Za  $x \in [0, 32/7]$  je ta vrednost manjša od 0 – ker imamo kvadraten polinom z negativnim vodilnim členom, je maksimum torej dosežen pri  $y = 0$ . Ker je  $4 < 32/7$ , torej velja

$$p_2(x) = 0.06x + 0.24 \quad (0 \leq x \leq 4)$$

– tj., v tretjo fazo se nam ne izplača vlagati. Sedaj izračunajmo še  $p^* = p_1(4)$ .

$$\begin{aligned} p_1(4) &= \max\{(4-z)(6+z)(0.06z+0.24) \mid 0 \leq z \leq 4\} \\ &= \max\{-0.06z^3 - 0.36z^2 + 0.96z + 5.76 \mid 0 \leq z \leq 4\} \end{aligned}$$

Poglejmo, kje ima zgornji izraz maksimum na intervalu  $[0, 4]$ .

$$\begin{aligned} 0 &= -0.18z^2 - 0.72z + 0.96 \\ z &= \frac{-12 \pm 4\sqrt{21}}{6} \end{aligned}$$

Ker imamo kubičen polinom z negativnim vodilnim členom, je maksimum dosežen pri večji od zgornjih rešitev, torej  $z \approx 1.055$ . Optimalni tržni delež je tako  $p^* \approx 6.302$ , dobimo pa ga tako, da 1.055 M€ vložimo v drugo fazo, preostanek pa v prvo fazo.

#### Naloga 4.11.

- (a) Naj bo  $p_{ij}$  največje število metrov za določitev nadstropja, če moramo določiti izmed  $i$  nadstropij in imamo na voljo  $j$  lončkov. Določimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned} p_{0j} &= 0 & (1 \leq j \leq k) \\ p_{i1} &= i & (0 \leq i \leq n) \\ p_{ij} &= 1 + \min\{\max\{p_{h-1,j-1}, p_{i-h,j}\} \mid 1 \leq h \leq i\} & (1 \leq i \leq n, 2 \leq j \leq k) \end{aligned}$$

Vrednosti  $p_{ij}$  ( $1 \leq i \leq n, 2 \leq j \leq k$ ) računamo v leksikografskem vrstnem redu indeksov. Največje število metrov pri optimalni strategiji dobimo kot  $p^* = p_{nk}$ .

- (b) Da bomo lahko rekonstruirali strategijo, bomo hranili še vrednosti  $r_{ij}$  ( $1 \leq i \leq n, 2 \leq j \leq k$ ), ki nam povedo, iz katerega od  $i$ -tih nadstropij, med katerimi določamo, naj vržemo lonček, če imamo na voljo še  $j$  lončkov. Velja torej

$$p_{ij} = 1 + \max\{p_{r_{ij}-1,j-1}, p_{i-r_{ij},j}\} \quad (1 \leq i \leq n, 2 \leq j \leq k).$$

Sedaj lahko zapišemo algoritem za določanje optimalne strategije metanja.

```
function LONČKI( $n, k$ )
  for  $j = 1, \dots, k$  do
     $p_{0j} \leftarrow 0$                                 robni pogoji za 0 nadstropij
  end for
  for  $i = 1, \dots, n$  do
     $p_{i1} \leftarrow i$                                 robni pogoji za 1 lonček
     $r_{i1} \leftarrow 1$                                 če imamo samo 1 lonček, ga vržemo iz najnižjega nadstropja
    for  $j = 2, \dots, k$  do                            rekurzivna enačba
       $p_{ij}, r_{ij} \leftarrow \min\{(1 + \max\{p_{h-1,j-1}, p_{i-h,j}\}, h) \mid 1 \leq h \leq i\}$ 
```

```

    end for
  end for
  function STRATEGIJA( $i, j, h$ )           pomožna funkcija za izračun strategije
    if  $j = 0$  then                       če smo ostali brez lončkov,
      return  $[h]$                        potem se ne razbijejo do  $h$ -tega nadstropja
    end if
     $\ell \leftarrow$  prazen seznam
    while  $i > 0$  do                       dodamo nadstropje za naslednji met in strategijo,
       $\ell.append((h + r_{ij}, STRATEGIJA(r_{ij} - 1, j - 1, h)))$    če se razbije
       $h \leftarrow h + r_{ij}$                če se ne razbije, nadaljujemo z višjimi nadstropji
       $i \leftarrow i - r_{ij}$ 
    end while
     $\ell.append(h)$                        če do konca ostane cel, imamo najvišje nadstropje
    return  $\ell$ 
  end function
  return ( $p_{nk}, STRATEGIJA(n, k, 0)$ )
end function

```

Funkcija LONČKI poleg največjega potrebnega števila metov vrne še strategijo metanja. Ta je podana kot seznam parov, kjer prvi element pove nadstropje, iz katerega naj vržemo lonček, drugi element pa nadaljnjo strategijo, če se lonček razbije. Če lonček preživi padec, nadaljujemo z naslednjim elementom seznama. Zadnji element seznama je številka najvišjega nadstropja, iz katerega lahko vržemo lonček, da se ta ne razbije, če je ta preživel vse predhodne mete v seznamu.

Algoritem najprej izračuna vrednosti spremenljivk z rekurzivnimi enačbami, pri čemer naredi  $O(nk)$  korakov. Strategijo potem sestavi z rekurzivnimi klici funkcije STRATEGIJA. Če bi strategijo zapisali kot drevo, bi opazili, da ima to drevo natanko  $n + 1$  listov (enega za vsak možen rezultat), saj vsak met razdeli množico možnih rezultatov na dva dela. Velikost strategije je torej  $O(n)$ , takšna pa je tudi časovna zahtevnost njenega sestavljanja, saj v vsakem obhodu zanke **while** dodamo en element. Skupna časovna zahtevnost algoritma je torej  $O(nk)$ .

#### Naloga 4.12.

Naj bo  $r_{ij}$  največja pričakovana porast v prodaji, če prvim  $j$  regijam dodelimo  $i$  trgovskih potnikov. Določimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned}
 r_{i1} &= p_{i1} & (1 \leq i \leq 4) \\
 r_{ij} &= \max\{r_{h,j-1} + p_{i-h,j} \mid 1 \leq h < i\} & (2 \leq i \leq 3, i \leq j \leq i+3)
 \end{aligned}$$

Vrednosti  $r_{ij}$  računamo v leksikografskem vrstnem redu glede na  $(j, i)$ . Največji pričakovani porast dobimo kot  $r^* = r_{63}$ .

Izračunajmo vrednosti  $r_{ij}$  ( $2 \leq i \leq 3$ ,  $i \leq j \leq i+3$ ).

$$\begin{aligned}
 r_{22} &= 35 + 21 &= 56 \\
 r_{32} &= \max\{35 + 42, 48 + 21\} &= 77 \\
 r_{42} &= \max\{35 + 56, 48 + 42, 70 + 21\} &= 91 \\
 r_{52} &= \max\{35 + 70, 48 + 56, 70 + 42, 89 + 21\} &= 112 \\
 r_{33} &= 56 + 28 &= 84 \\
 r_{43} &= \max\{56 + 41, 77 + 28\} &= 105 \\
 r_{53} &= \max\{56 + 63, 77 + 41, 91 + 28\} &= 119 \\
 r_{63} &= \max\{56 + 75, 77 + 63, 91 + 41, 112 + 28\} &= 140
 \end{aligned}$$

Največji pričakovani porast je torej  $r^* = r_{63} = 140$ . Poiščimo še optimalno razporeditev trgovskih potnikov.

$$\begin{aligned}
 r_{63} &= r_{32} + p_{33} && 3 \text{ trgovski potniki v tretjo regijo} \\
 r_{32} &= r_{11} + p_{22} && 2 \text{ trgovska potnika v drugo regijo} \\
 r_{11} &= p_{11} && 1 \text{ trgovski potnik v prvo regijo}
 \end{aligned}$$

Druga optimalna razporeditev je, da razporedimo 3 trgovske potnike v prvo regijo, 2 v drugo regijo in 1 v tretjo regijo.

#### Naloga 4.13.

Nelinearni program bomo rešili s pomočjo sledečih rekurzivnih enačb.

$$\begin{aligned}
 p_2(x) &= 2x \\
 p_1(x) &= \max \left\{ 2y^2 + p_2(x-2y) \mid 0 \leq y \leq \frac{x}{2} \right\} \\
 p_3(x) &= \max \left\{ p_1(x-z) + 4z - z^2 \mid 0 \leq z \leq x \right\}
 \end{aligned}$$

Optimalno vrednost ciljne funkcije bomo dobili kot  $p^* = p_3(4)$ .

Najprej izračunajmo  $p_1(x)$ , kjer je  $0 \leq x \leq 4$ .

$$p_1(x) = \max \left\{ 2y^2 + 2x - 4y \mid 0 \leq y \leq \frac{x}{2} \right\}$$

Zgornji izraz je kvadraten polinom v  $y$  s pozitivnim vodilnim členom, ki tako maksimum doseže na eni izmed robnih točk, torej z vrednostjo  $2x$  ali  $\frac{x^2}{2}$ . Ker za vse  $x \leq 4$  velja  $2x \geq \frac{x^2}{2}$ , imamo torej  $p_1(x) = 2x$  za vse  $x \in [0, 4]$ . V optimalni rešitvi nelinearnega programa bo torej veljalo  $x_1 = 0$ .

Sedaj izračunajmo še  $p^* = p_3(4)$ .

$$p_3(4) = \max \left\{ 2(4-z) + 4z - z^2 \mid 0 \leq z \leq 4 \right\}$$

Zgornji izraz je kvadraten polinom v  $z$  z negativnim vodilnim členom. Da poiščemo maksimum, pogledajmo, kje ima njegov odvod vrednost 0.

$$0 = 2 - 2z$$

$$z = 1$$

Optimalna rešitev nelinearnega programa je torej  $x_1 = 0$ ,  $x_2 = 3$ ,  $x_3 = 1$ , vrednost ciljne funkcije pa je tedaj enaka  $p^* = 9$ .

#### Naloga 4.14.

Naj bo  $p_i(x)$  največja verjetnost, da bo igralec na srečo na koncu imel natanko 100€, če bo v  $i$ -to igro vstopil z  $x$  evri. Opazimo, da če tretjo igro začne z manj kot 50€, je ne bo mogel končati s 100€. Če ima  $x$  evrov, kjer je  $50 \leq x < 100$ , potem bo stavil  $100 - x$  evrov in upal na zmago; pri  $x > 100$  pa bo stavil  $x - 100$  evrov in upal na poraz. Če ima natanko 100€, potem ne bo stavil in si tako zagotovil, da bo tudi na koncu imel natanko 100€. Določimo torej začetne pogoje in rekurzivne enačbe.

$$p_3(x) = \begin{cases} 0 & 0 \leq x < 50 \\ \frac{1}{2} & 50 \leq x < 100 \\ 1 & x = 100 \\ \frac{1}{2} & x > 100 \end{cases}$$

$$p_2(x) = \max \left\{ \frac{p_3(x-y) + p_3(x+y)}{2} \mid 0 \leq y \leq x \right\}$$

$$p_1(x) = \max \left\{ \frac{p_2(x-y) + p_2(x+y)}{2} \mid 0 \leq y \leq x \right\}$$

Maksimalno verjetnost dobimo kot  $p^* = p_1(75)$ .

Izračunajmo sedaj  $p_2(x)$  in  $p^* = p_1(75)$ .

$$p_2(x) = \frac{1}{2} \begin{cases} 0+0 & 0 \leq x < 25 \\ \max \left\{ 0+0, 0+\frac{1}{2} \right\} & 25 \leq x < 50 \\ \max \left\{ \frac{1}{2}+\frac{1}{2}, 0+\frac{1}{2}, 0+1 \right\} & x = 50 \\ \max \left\{ \frac{1}{2}+\frac{1}{2}, 0+\frac{1}{2}, 0+1, 0+\frac{1}{2} \right\} & 50 < x < 75 \\ \max \left\{ \frac{1}{2}+\frac{1}{2}, \frac{1}{2}+1, 0+\frac{1}{2} \right\} & x = 75 \\ \max \left\{ \frac{1}{2}+\frac{1}{2}, \frac{1}{2}+1, \frac{1}{2}+\frac{1}{2}, 0+\frac{1}{2} \right\} & 75 < x < 100 \\ \max \left\{ 1+1, \frac{1}{2}+\frac{1}{2}, 0+\frac{1}{2} \right\} & x = 100 \\ \max \left\{ \frac{1}{2}+\frac{1}{2}, 1+\frac{1}{2}, \frac{1}{2}+\frac{1}{2}, 0+\frac{1}{2} \right\} & x > 100 \end{cases} = \begin{cases} 0 & 0 \leq x < 25 \\ \frac{1}{4} & 25 \leq x < 50 \\ \frac{1}{2} & 50 \leq x < 75 \\ \frac{3}{4} & 75 \leq x < 100 \\ 1 & x = 100 \\ \frac{3}{4} & x \geq 100 \end{cases}$$

$$p_1(75) = \frac{1}{2} \max \left\{ \frac{3}{4} + \frac{3}{4}, \frac{1}{2} + \frac{3}{4}, \frac{1}{2} + 1, \frac{1}{4} + \frac{3}{4}, 0 + \frac{3}{4} \right\} = \frac{3}{4}$$

Maksimalna verjetnost je torej  $p^* = \frac{3}{4}$ . Poiščimo strategijo, pri kateri jo dobimo.

$$\begin{aligned} p_1(75) &= \frac{p_2(75-0) + p_2(75+0)}{2} && \text{v prvi igri ne stavimo} \\ p_2(75) &= \frac{p_3(75-25) + p_3(75+25)}{2} && \text{v drugi igri stavimo 25€} \\ p_3(50) &= \frac{1}{2} && \text{če izgubimo, v tretji stavimo 50€} \\ p_3(100) &= 1 && \text{če zmagamo, v tretji ne stavimo} \end{aligned}$$

Pri drugi optimalni strategiji v prvi igri stavimo 25€, v drugi igri pa ne glede na izid prve igre ne stavimo. V tretji igri potem ravnamo enako kot v zgornji strategiji.

#### Naloga 4.15.

Naj bo  $p_{ij}$  največji dobiček, ki ga lahko iztržimo z rezanjem kosa blaga dimenzij  $i \times j$  ( $1 \leq i \leq m$ ,  $1 \leq j \leq n$ ). Določimo rekurzivne enačbe.

$$\begin{aligned} p_{ij} = \max \big( & \{c_h \mid 1 \leq h \leq k \wedge (a_h, b_h) \in \{(i, j), (j, i)\}\} \\ & \cup \{p_{\ell j} + p_{i-\ell, j} \mid 1 \leq \ell \leq i/2\} \\ & \cup \{p_{i\ell} + p_{i, j-\ell} \mid 1 \leq \ell \leq j/2\} \cup \{0\} \big) \end{aligned}$$

Prva množica tukaj obravnava primere, ko imamo kos blaga želene velikosti, naslednji dve pa obravnavata vodoravne oziroma navpične reze. Zadnja množica poskrbi, da je vrednost  $p_{11}$  definirana tudi v primeru, ko nam kos blaga velikosti  $1 \times 1$  ne prinese dobička.

Vrednosti  $p_{ij}$  računamo v leksikografskem vrstnem redu indeksov (npr. najprej naraščajoče po  $i$ , nato pa naraščajoče po  $j$ ). Maksimalni dobiček dobimo kot  $p^* = p_{mn}$ .

#### Naloga 4.16.

Algoritem, ki izhaja iz rekurzivnih enačb, ima časovno zahtevnost  $O(mn(m + n + k))$ . Z ustrezno podatkovno strukturo (zgoščena tabela) je mogoče časovno zahtevnost izboljšati na  $O(mn(m + n) + k)$ .

Izračunajmo vrednosti  $p_{ij}$  ( $1 \leq i \leq m$ ,  $1 \leq j \leq n$ ) po rekurzivnih enačbah iz rešitve naloge 4.15. Pri tem upoštevamo, da velja  $p_{ij} = p_{ji}$ , tako da bomo izračunali samo primere z  $i \geq j$ .

$$\begin{aligned} p_{11} &= \max\{0\} && = 0 \\ p_{21} &= \max\{0 + 0, 0\} && = 0 \\ p_{22} &= \max\{\underline{6}, 0 + 0, 0\} && = 6 && \text{izdelek 1} \\ p_{31} &= \max\{\underline{3}, 0 + 0, 0\} && = 3 && \text{izdelek 2} \\ p_{32} &= \max\{\underline{7}, 0 + 6, 3 + 3, 0\} && = 7 && \text{izdelek 4} \end{aligned}$$

$$\begin{aligned}
p_{33} &= \max\{\underline{3+7}, 0\} &= 10 & \text{razrez na } 1 \times 3 \text{ in } 2 \times 3 \\
p_{41} &= \max\{\underline{5}, 0+3, 0+0, 0\} &= 5 & \text{izdelek 3} \\
p_{42} &= \max\{0+7, \underline{6+6}, 5+5, 0\} &= 12 & \text{razrez na } 2 \times 3 \text{ in } 2 \times 2 \\
p_{43} &= \max\{3+10, 7+7, \underline{5+12}, 0\} &= 17 & \text{razrez na } 4 \times 1 \text{ in } 4 \times 2 \\
p_{51} &= \max\{\underline{0+5}, 0+3, 0\} &= 5 & \text{razrez na } 1 \times 1 \text{ in } 4 \times 1 \\
p_{52} &= \max\{0+12, \underline{6+7}, 5+5, 0\} &= 13 & \text{razrez na } 2 \times 2 \text{ in } 3 \times 2 \\
p_{53} &= \max\{\underline{3+17}, 7+10, 5+13, 0\} &= 20 & \text{razrez na } 1 \times 3 \text{ in } 4 \times 3
\end{aligned}$$

Maksimalni dobiček je torej  $p^* = p_{53} = 20$ , dosežemo pa ga tako, da iz prvotnega kosa blaga dimenzij  $5 \times 3$  odrežemo kos dimenzij  $1 \times 3$  za izdelek s ceno 3, nato od preostanka dimenzij  $4 \times 3$  odrežemo kos dimenzij  $4 \times 1$  za izdelek s ceno 5, nazadnje pa preostanek dimenzij  $4 \times 2$  razrežemo na dva kosa dimenzij  $2 \times 2$  za izdelka s ceno 6.

#### Naloga 4.17.

- (a) Naj bo  $x_i$  največji produkt strnjenega podzaporedja, ki se konča pri številu  $a_i$ , in  $y_i$  najmanjši tak produkt. Določimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned}
x_1 &= y_1 = a_1 \\
x_i &= \max\{a_i, a_i x_{i-1}, a_i y_{i-1}\} & (2 \leq i \leq n) \\
y_i &= \min\{a_i, a_i x_{i-1}, a_i y_{i-1}\} & (2 \leq i \leq n)
\end{aligned}$$

Vrednosti  $x_i$  in  $y_i$  računamo naraščajoče po indeksu  $i$ . Največji produkt strnjenega podzaporedja dobimo kot  $x^* = \max\{x_i \mid 1 \leq i \leq n\}$ .

- (b) Za izračun vrednosti  $x_i$  in  $y_i$  pri vsakem  $i$  potrebujemo konstantno časa. Ker naredimo  $n$  takih korakov, je časovna zahtevnost algoritma  $O(n)$ .
- (c) Izračunajmo vrednosti  $x_i$  in  $y_i$  ( $2 \leq i \leq 10$ ).

$$\begin{aligned}
x_2 &= \max\{-2, \underline{-2 \cdot 0.9}, -2 \cdot 0.9\} &= -1.8 \\
y_2 &= \min\{\underline{-2}, -2 \cdot 0.9, -2 \cdot 0.9\} &= -2 \\
x_3 &= \max\{-0.6, -0.6 \cdot -1.8, \underline{-0.6 \cdot -2}\} &= 1.2 \\
y_3 &= \min\{\underline{-0.6}, -0.6 \cdot -1.8, -0.6 \cdot -2\} &= -0.6 \\
x_4 &= \max\{-0.5, -0.5 \cdot 1.2, \underline{-0.5 \cdot -0.6}\} &= 0.3 \\
y_4 &= \min\{-0.5, \underline{-0.5 \cdot 1.2}, -0.5 \cdot -0.6\} &= -0.6 \\
x_5 &= \max\{-2, -2 \cdot 0.3, \underline{-2 \cdot -0.6}\} &= 1.2 \\
y_5 &= \min\{\underline{-2}, -2 \cdot 0.3, -2 \cdot -0.6\} &= -2 \\
x_6 &= \max\{5, \underline{5 \cdot 1.2}, 5 \cdot -2\} &= 6 \\
y_6 &= \min\{5, 5 \cdot 1.2, \underline{5 \cdot -2}\} &= -10 \\
x_7 &= \max\{0.1, \underline{0.1 \cdot 6}, 0.1 \cdot -10\} &= 0.6
\end{aligned}$$

$$\begin{aligned}
y_7 &= \min\{0.1, 0.1 \cdot 6, \underline{0.1 \cdot -10}\} &= -1 \\
x_8 &= \max\{\underline{3}, 3 \cdot 0.6, 3 \cdot -1\} &= 3 \\
y_8 &= \min\{3, 3 \cdot 0.6, \underline{3 \cdot -1}\} &= -3 \\
x_9 &= \max\{0.5, \underline{0.5 \cdot 3}, 0.5 \cdot -3\} &= 1.5 \\
y_9 &= \min\{0.5, 0.5 \cdot 3, \underline{0.5 \cdot -3}\} &= -1.5 \\
x_{10} &= \max\{-3, -3 \cdot 1.5, \underline{-3 \cdot -1.5}\} &= 4.5 \\
y_{10} &= \min\{-3, \underline{-3 \cdot 1.5}, -3 \cdot -1.5\} &= -4.5
\end{aligned}$$

Največji produkt strnjene zaporedja je torej  $x^* = 6 = \prod_{i=2}^6 a_i$ .

#### Naloga 4.18.

Naj bosta  $p_i$  in  $q_i$  dolžini najdaljših oscilirajočih podzaporedij lihe oziroma sode dolžine zaporedja celih števil  $a_1, a_2, \dots, a_i$ , ki se končata z elementom  $a_i$ . Določimo rekurzivne enačbe.

$$\begin{aligned}
p_i &= 1 + \max(\{q_j \mid 1 \leq j \leq i-1 \wedge a_j < a_i\} \cup \{0\}) & (1 \leq i \leq n) \\
q_i &= \max(\{1 + p_j \mid 1 \leq j \leq i-1 \wedge a_j > a_i\} \cup \{0\}) & (1 \leq i \leq n)
\end{aligned}$$

Vrednosti  $p_i$  in  $q_i$  računamo naraščajoče glede na indeks  $i$ . Maksimalno dolžino oscilirajočega podzaporedja dobimo kot  $\ell^* = \max\{p_i, q_i \mid 1 \leq i \leq n\}$ . Algoritem, ki sledi iz zgornjih enačb, poišče rešitev v času  $O(n^2)$ .

Časovno zahtevnost je mogoče sicer nekoliko izboljšati. Opazimo namreč, da v  $i$ -tem koraku iščemo največji taki vrednosti  $q_j$  in  $p_j$  ( $j < i$ ), za kateri je vrednost  $a_j$  manjša oziroma večja od trenutne vrednosti  $a_i$ . Če torej za vsako možno vrednost  $p_i$  in  $q_i$  hranimo največjo oziroma najmanjšo vrednost  $a_i$ , pri kateri je ta dosežena, lahko omejimo število korakov pri izračunu posamezne vrednosti na dolžino najdaljšega oscilirajočega podzaporedja.

**function** OSCILIRAJOČEPODZAPOREDJE( $(a_i)_{i=1}^n$ )

```

 $r \leftarrow$  prazen slovar največjih in najmanjših koncev podzaporedij
 $r[-1] \leftarrow \infty$  pogoji za ustavitev iskanja
 $r[0] \leftarrow -\infty$  prejšnjega člena podzaporedja
 $r[1] \leftarrow a_1$  začetni pogoji
 $p^* \leftarrow p_1 \leftarrow 1$ 
 $q^* \leftarrow q_1 \leftarrow 0$ 
for  $i = 2, 3, \dots, n$  do
     $r[i] \leftarrow (-1)^i \cdot \infty$  začetna vrednost za največjo možno dolžino
     $h \leftarrow q^*$  računanje  $p_i$ 
    while  $r[h] \geq a_i$  do iskanje največjega  $q_j$ 
         $h \leftarrow h - 2$   $h$  ostane sod
    end while
     $p_i \leftarrow h + 1$ 
    if  $a_i > r[p_i]$  then posodobitev vrednosti za  $p_i$ 
         $r[p_i] \leftarrow a_i$ 
    end if
    if  $p_i > p^*$  then posodobitev največje lihe dolžine

```



```

     $p^* \leftarrow p_i$ 
  end if
   $h \leftarrow p^*$ 
  while  $r[h] \leq a_i$  do
     $h \leftarrow h - 2$ 
  end while
   $q_i \leftarrow h + 1$ 
  if  $a_i < r[q_i]$  then
     $r[q_i] \leftarrow a_i$ 
  end if
  if  $q_i > q^*$  then
     $q^* \leftarrow q_i$ 
  end if
end for
return  $\max\{p^*, q^*\}$ 
end function

```

računanje  $q_i$   
iskanje največjega  $p_j$   
 $h$  ostane lih

posodobitev vrednosti za  $q_i$

posodobitev največje sode dolžine

Tak algoritem ima časovno zahtevnost  $O(n\ell^*)$ , kjer je  $\ell^*$  dolžina najdaljšega oscilirajočega podzaporedja<sup>7</sup> (tj., vrnjena vrednost). Če bi želeli poiskati še samo oscilirajoče podzaporedje, bi si morali tekom algoritma za vsako vrednost  $p_i$  in  $q_i$  beležiti še indeks predhodnega elementa v podzaporedju, za vsako vrednost  $r[h]$  pa še indeks  $j$ , za katerega velja  $a_j = r[h]$  in  $p_j = h$  oziroma  $q_j = h$ .

#### Naloga 4.19.

- (a) Za izračun pričakovanega dobička bomo definirali funkcije  $v_i(z)$  ( $i = 1, 2, 3$ ). Zapišimo njihove definicije.

$$\begin{aligned}
 v_1(z) &= n_1 + k_1 z \\
 v_2(z) &= \max \{v_1(z - y) + n_2 + k_2 y \mid a_2 \leq y \leq z - a_1\} \\
 v_3(z) &= \max \{v_2(z - y) \cdot (n_3 + k_3 y) \mid a_3 \leq y \leq z - a_1 - a_2\}
 \end{aligned}$$

Pričakovani dobiček dobimo tako, da izračunamo  $v_3(m)$ .

- (b) Vstavimo podatke v zgornje formule.

$$\begin{aligned}
 v_1(z) &= 3 + 1.5z \\
 v_2(z) &= \max \{3 + 1.5(z - y) + 4 + 2y \mid 3 \leq y \leq z - 4\} \\
 &= \max \{7 + 1.5z + 0.5y \mid 3 \leq y \leq z - 4\} \\
 &= 5 + 2z \quad (z \geq 7) \\
 v_3(15) &= \max \{(5 + 2(15 - y)) \cdot (0.4 + 0.3y) \mid 2 \leq y \leq 8\} \\
 &= \max \{-0.6y^2 + 9.7y + 14 \mid 2 \leq y \leq 8\}
 \end{aligned}$$

<sup>7</sup>Algoritmom, pri katerih je časovna zahtevnost odvisna od izhoda, pravimo *output-sensitive* algoritmi. V tem primeru velja  $\ell^* = O(n)$ , tako da tudi v najslabšem primeru zahtevnost algoritma ne preseže  $O(n^2)$ .

Naj bo  $f(y) = -0.6y^2 + 9.7y + 14$  – gre za kvadraten polinom z negativnim vodilnim členom, tako da ima maksimum tam, kjer je odvod ničeln.

$$\begin{aligned} 0 &= f'(y) = -1.2y + 9.7 \\ y &= 9.7/1.2 > 8 \end{aligned}$$

Opazimo torej, da maksimum leži desno od intervala  $[2, 8]$ , kar pomeni, da velja  $v_3(15) = f(8) = 53.6$ . Podjetje bo torej maksimiziralo pričakovani dobiček na 53.8 milijonov evrov, če dodeli 8 milijonov evrov marketingu, 3 milijone evrov oblikovalcem in 4 milijone evrov razvijalcem.

#### Naloga 4.20.

- (a) Naj bo  $q_{ij}$  največja verjetnost, ki jo lahko dosežejo, da pridobijo podporo prvih  $j$  strank, če k njim pošljejo  $i$  lobistov. Določimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned} q_{i1} &= p_{i1} & (1 \leq i \leq m) \\ q_{ij} &= \max\{q_{h,j-1}p_{i-h,j} \mid 0 \leq h \leq i\} & (2 \leq j \leq n, 0 \leq i \leq m) \end{aligned}$$

Vrednosti  $q_{ij}$  računamo v leksikografskem vrstnem redu glede na  $(j, i)$ . Največjo verjetnost, da pridobijo podporo vseh  $n$  strank, dobimo kot  $q^* = q_{mn}$ .

- (b) Ker velja  $p_{0j} = 0$  za vsak  $j$ , bo veljalo  $q_{ij} = 0$  za vse  $i < j$ . Tako lahko v zgornji rekurzivni enačbi upoštevamo le primere z  $j-1 \leq h \leq i-1$ . Izračunajmo torej vrednosti  $q_{i2}$  ( $2 \leq i \leq 5$ ) in  $q^* = q_{63}$ .

$$\begin{aligned} q_{22} &= 0.2 \cdot 0.4 & = 0.08 \\ q_{32} &= \max\{0.2 \cdot 0.5, \underline{0.5 \cdot 0.4}\} & = 0.2 \\ q_{42} &= \max\{0.2 \cdot 0.5, 0.5 \cdot 0.5, \underline{0.7 \cdot 0.4}\} & = 0.28 \\ q_{52} &= \max\{0.2 \cdot 0.6, 0.5 \cdot 0.5, \underline{0.7 \cdot 0.5}, 0.8 \cdot 0.4\} & = 0.35 \\ q_{63} &= \max\{0.08 \cdot 0.9, \underline{0.2 \cdot 0.8}, 0.28 \cdot 0.4, 0.35 \cdot 0.3\} & = 0.16 \end{aligned}$$

Največja verjetnost je torej  $q^* = q_{63} = 0.16$ . Poiščimo še optimalno razporeditev lobistov.

$$\begin{aligned} q_{63} &= q_{32}p_{33} & 3 \text{ lobisti k tretji stranki} \\ q_{32} &= q_{21}p_{12} & 1 \text{ lobist k drugi stranki} \\ q_{21} &= p_{21} & 2 \text{ lobista k prvi stranki} \end{aligned}$$

### Naloga 4.21.

- (a) Naj bosta  $p_i$  in  $q_i$  največji vsoti za ustrezne postavitve domin do  $i$ -tega polja, pri čemer na  $i$ -tem polju dovolimo le del domine z znakom + oziroma – (tj., prepovemo – oziroma +, dovolimo pa, da  $i$ -to polje ni pokrito). Določimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned} p_0 = p_1 = 0, \quad p_i &= \max\{p_{i-1}, q_{i-1}, p_{i-2} - a_{i-1} + a_i\} & (2 \leq i \leq n) \\ q_0 = q_1 = 0, \quad q_i &= \max\{p_{i-1}, q_{i-1}, q_{i-2} + a_{i-1} - a_i\} & (2 \leq i \leq n) \end{aligned}$$

Vrednosti  $p_i$  in  $q_i$  ( $0 \leq i \leq n$ ) računamo naraščajoče po indeksu  $i$ . Največjo vsoto dobimo kot  $p^* = \max\{p_n, q_n\}$ .

- (b) Za izračun vrednosti  $p_i$  in  $q_i$  pri vsakem  $i$  potrebujemo konstantno časa. Ker naredimo  $n$  takih korakov, je časovna zahtevnost algoritma  $O(n)$ .

- (c) Izračunajmo vrednosti  $p_i$  in  $q_i$  ( $2 \leq i \leq 9$ ).

$$\begin{aligned} p_2 &= \max\{0, 0, 0 - 6 + 3\} = 0 & q_2 &= \max\{0, 0, 0 + 6 - 3\} = 3 \\ p_3 &= \max\{0, \underline{3}, 0 - 3 + (-4)\} = 3 & q_3 &= \max\{0, 3, 0 + 3 + (-4)\} = 7 \\ p_4 &= \max\{3, \underline{7}, 0 - (-4) + 2\} = 7 & q_4 &= \max\{3, \underline{7}, 3 + (-4) - 2\} = 7 \\ p_5 &= \max\{\underline{7}, 7, 3 - 2 + (-3)\} = 7 & q_5 &= \max\{7, 7, 7 + 2 - (-3)\} = 12 \\ p_6 &= \max\{7, 12, \underline{7 - (-3) + 5}\} = 15 & q_6 &= \max\{7, \underline{12}, 7 + (-3) - 5\} = 12 \\ p_7 &= \max\{\underline{15}, 12, 7 - 5 + 9\} = 15 & q_7 &= \max\{\underline{15}, 12, 7 + 5 - 9\} = 15 \\ p_8 &= \max\{\underline{15}, 15, 15 - 9 + 1\} = 15 & q_8 &= \max\{15, 15, \underline{15 + 9 - 1}\} = 23 \\ p_9 &= \max\{15, \underline{23}, 15 - 1 + 2\} = 23 & q_9 &= \max\{15, \underline{23}, 15 + 1 - 2\} = 23 \end{aligned}$$

Optimalno pokritje ima torej vsoto  $p^* = \max\{p_9, q_9\} = 23$ . Poglejmo, kam moramo postaviti domine.

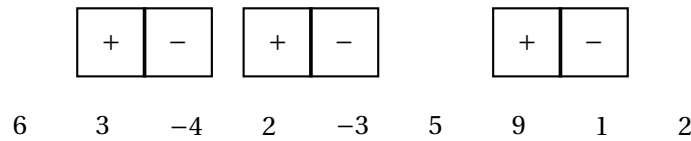
$$\begin{aligned} p^* = p_9 = q_8 = q_6 + a_7 - a_8 & & [+ -] & \text{na } (7, 8) \\ q_6 = q_5 = q_3 + a_4 - a_5 & & [+ -] & \text{na } (4, 5) \\ q_3 = q_1 + a_2 - a_3 & & [+ -] & \text{na } (2, 3) \end{aligned}$$

Postavitev je prikazana na sliki 79.

### Naloga 4.22.

- (a) Zapišimo definicijo funkcije  $q(x)$  za  $0 \leq x \leq 4$ .

$$\begin{aligned} q(x) &= \max(\{0.15x\} \cup \{0.1 \mid x = 1\} \cup \{0.35 \mid x = 2\} \cup \{0.5 \mid x = 3\}) \\ &= \begin{cases} 0.35 & x = 2 \text{ (Diskretna d.d.z.)} \\ 0.5 & x = 3 \text{ (Diskretna d.d.z.)} \\ 0.15x & \text{sicer (Zvezna d.d.z.)} \end{cases} \end{aligned}$$



Slika 79: Optimalno pokritje za nalogo 4.21(c).

- (b) Naj bosta  $v_1(x)$  in  $v_2(x)$  največji pričakovani vrednosti naložbenih strategij, pri katerih imamo na voljo  $x$  milijonov evrov oziroma to količino v celoti vložimo v naložbo in zavarovanje. Zapišimo rekurzivni enačbi.

$$v_2(x) = \max \{ (x - y)(3 + 0.4q(y)) \mid 0 \leq y \leq \min\{4, x\} \}$$

$$v_1(x) = \max \{ x - y + v_2(y) \mid 0 \leq y \leq x \}$$

- (c) Najprej izrazimo  $v_2(x)$  glede na vrednost  $x$  ( $0 \leq x \leq 50$ ).

$$v_2(x) = \max \{ \{3.14(x - 2) \mid x \geq 2\} \cup \{3.2(x - 3) \mid x \geq 3\} \cup \{ (x - y)(3 + 0.06y) \mid 0 \leq y \leq \min\{4, x\}, y \notin \{2, 3\} \} \}$$

Naj bo  $f(y)$  izraz v zadnjem oklepaju. Opazimo, da gre za kvadraten polinom v  $y$  z negativnim vodilnim členom, tako da lahko s pomočjo odvajanja poiščemo njegov maksimum.

$$f(y) = -0.06y^2 + (0.06x - 3)y + 3x$$

$$f'(y) = -0.12y + 0.06x - 3 = 0$$

$$y = x/2 - 25 \leq 0$$

Opazimo, da je maksimum vedno dosežen za vrednost  $y$ , ki ni večja od spodnje meje ustreznega intervala, tako da je znotraj njega maksimum dosežen pri  $y = 0$  in znaša  $f(0) = 3x$  - tj., premije ne plačamo. S primerjavo vseh treh izrazov tako dobimo

$$v_2(x) = \begin{cases} 3x & 0 \leq x \leq 314/7 \text{ (brez zavarovanja)} \\ 3.14(x - 2) & 314/7 < x \leq 50 \text{ (Diskretna d.d.z. s premijo 2 mio evrov)} \end{cases}$$

Optimalno strategijo vlaganja sedaj dobimo tako, da izračunamo  $v_1(50)$ .

$$v_1(50) = \max \{ \{50 + 2y \mid 0 \leq y \leq 314/7\} \cup \{43.72 + 2.14y \mid 314/7 < x \leq 50\} \}$$

Ker oba izraza naraščata z  $y$ , zadostuje preveriti njuni vrednosti pri zgornji meji ustreznega intervala. Tako opazimo, da največjo vrednost dobimo pri  $y = 50$ . Optimalna strategija vlaganja je torej taka, pri kateri vlagatelj 48 milijonov evrov vloži v naložbo, 2 milijona evrov porabi za premijo pri zavarovalnici Diskretna d.d.z., zase pa ne obdrži ničesar. Pričakovana vrednost naložbene strategije je tako  $v^* = v_1(50) = 150.72$  milijonov evrov.

**Naloga 4.23.**

- (a) Naj bo  $v_i$  najnižja cena izgradnje počivališč na odseku od začetka avtoceste do lokacije  $x_i$ , če zadnje počivališče zgradimo na tej lokaciji. Zapišimo začetne pogoje in rekurzivne enačbe.

$$v_0 = x_0 = 0$$

$$v_i = c_i + \min(\{v_j \mid 0 \leq j \leq i-1, x_i - x_j \leq K\} \cup \{\infty\}) \quad (1 \leq i \leq n)$$

Vrednosti  $v_i$  računamo naraščajoče po indeksu  $i$ . Najnižjo ceno izgradnje počivališč na celotnem odseku dobimo kot

$$v^* = \min(\{v_j \mid 0 \leq j \leq n, M - x_j \leq K\} \cup \{\infty\}).$$

Če velja  $v^* = \infty$ , potem izgradnja počivališč pod danimi pogoji ni mogoča.

- (b) V algoritmu izračunamo  $n$  vrednosti  $v_i$ , pri čemer pri vsaki obravnavamo največ  $n$  možnosti; prav tako obravnavamo največ  $n$  možnosti pri računanju  $v^*$ . Časovna zahtevnost algoritma je torej  $O(n^2)$ .
- (c) Izračunajmo vrednosti  $v_i$  ( $1 \leq i \leq 8$ ). Ker se lokacije zaporednih počivališč razlikujejo za manj kot  $K$ , prav tako pa sta prva in zadnja možna lokacija oddaljeni manj kot  $K$  od začetka oziroma konca odseka, bodo vse vrednosti  $v_i$  končne, enako pa velja tudi za  $v^*$ .

$x_1 = 5$	$v_1 = 18 + \min\{0\}$	$= 18$
$x_2 = 12$	$v_2 = 11 + \min\{0, 18\}$	$= 11$
$x_3 = 22$	$v_3 = 21 + \min\{0, 18, 11\}$	$= 21$
$x_4 = 34$	$v_4 = 16 + \min\{18, \underline{11}, 21\}$	$= 27$
$x_5 = 49$	$v_5 = 23 + \min\{\underline{21}, 27\}$	$= 44$
$x_6 = 65$	$v_6 = 15 + \min\{\underline{44}\}$	$= 59$
$x_7 = 83$	$v_7 = 19 + \min\{\underline{59}\}$	$= 78$
$x_8 = 91$	$v_8 = 13 + \min\{\underline{59}, 78\}$	$= 72$

Ker sta samo zadnji dve lokaciji oddaljeni manj kot  $K$  od konca odseka, je najmanjša cena izgradnje počivališč enaka  $v^* = \min\{78, 72\} = 72$ . Rekonstruirajmo še optimalno postavitev.

$v^* = v_8 = c_8 + v_6$	počivališče na $x_8 = 91$
$v_6 = c_6 + v_5$	počivališče na $x_6 = 65$
$v_5 = c_5 + v_3$	počivališče na $x_5 = 49$
$v_3 = c_3 + v_0$	počivališče na $x_3 = 22$

**Naloga 4.24.**

Opazimo, da za izračun vrednosti  $c_{ij}$  potrebujemo samo take vrednosti  $c_{i'j'}$ , za katere velja  $i + j \equiv i' + j' \pmod{2}$ . V primeru  $m = 1$  tako za izračun  $c_{1j}$  potrebujemo le  $c_{0,j-1}, c_{2,j-1}, c_{0,j+1}$ , v primeru  $n = 1$  pa za izračun  $c_{i1}$  potrebujemo le  $c_{i-1,0}, c_{i+1,0}, c_{i-1,2}$ .

V primeru, ko velja  $m, n \geq 2$ , bo potrebno poznati vse vrednosti  $c_{0j}$  ( $0 \leq j \leq n+1, j \equiv m+n \pmod{2}$ ),  $c_{i0}$  ( $0 \leq i \leq m+1, i \equiv m+n \pmod{2}$ ),  $c_{m+1,j}$  ( $1 \leq j \leq n-1, j+1 \equiv n \pmod{2}$ ) in  $c_{n+1,i}$  ( $1 \leq i \leq n-1, i+1 \equiv m \pmod{2}$ ). Ker pa v podani rekurzivni zvezi pride do zanke (npr. za izračun  $c_{ij}$  potrebujemo  $c_{i+1,j-1}$ , za izračun slednje vrednosti pa spet potrebujemo  $c_{ij}$ ), morajo na teh začetnih vrednostih veljati določeni pogoji, da bo rešitev sploh obstajala in da bo enolično določena.

**Naloga 4.25.**

Ker je zaslužek premo sorazmeren času najema, bomo tukaj optimizirali skupen čas najema dvorane. Če želimo iz časa (predpostavimo, da je podan v minutah) izračunati zaslužek, ga lahko pomnožimo s 5€.

- (a) Postopek ni optimalen, saj lahko najdemo protiprimer, kjer opisani postopek ne da optimalne rešitve. Denimo, da imamo ponudbe  $(0, 2), (1, 6), (5, 7)$  – opisani postopek izbere prvi in zadnji interval s skupnim trajanjem 4, medtem ko je optimalna rešitev izbira drugega intervala s trajanjem 5.
- (b) Če je  $I = (z_t, k_t)$  zadnji interval v optimalni izbiri OPT, potem so kandidati za predzadnji interval vsi intervali oblike  $(z_i, k_i)$ , kjer je  $k_i \leq z_t$ . Tako naj bo  $x_i$  največji čas najema dvorane, če je zadnji uporabljeni interval  $(z_i, k_i)$ . Brez škode za splošnost lahko predpostavimo, da za vsak  $i$  ( $1 \leq i \leq n$ ) velja  $0 \leq z_i < k_i$  ter da so intervali  $(z_i, k_i)$  urejeni naraščajoče po  $k_i$ . Zapišimo začetne pogoje in rekurzivne enačbe.

$$x_0 = k_0 = 0$$

$$x_i = k_i - z_i + \max\{x_j \mid 0 \leq j < i, k_j \leq z_i\}$$

Urejenost intervalov nam zagotovi, da velja  $k_j > z_i$  za vse  $j \geq i$ , tako da nam teh primerov ni potrebno obravnavati. Največji skupni čas najema potem dobimo kot  $x^* = \max\{x_i \mid 1 \leq i \leq n\}$ .

- (c) Izračunati je potrebno  $n$  vrednosti  $x_i$ , za izračun  $x_i$  pa je potrebno pregledati največ  $i$  vrednosti  $x_j$ . Skupna časovna zahtevnost je torej  $O(n^2)$ .
- (d) Najprej bomo podatke uredili po časih končanja – imeli bomo torej

$$((z_i, k_i))_{i=1}^8 = ((15, 60), (30, 90), (30, 105), (75, 135), \\ (90, 150), (60, 180), (120, 210), (165, 255)).$$

Izračunajmo sedaj vrednosti  $x_i$  ( $1 \leq i \leq 8$ ).

$$\begin{aligned}
 x_1 &= 45 + 0 & &= 45 \\
 x_2 &= 60 + 0 & &= 60 \\
 x_3 &= 75 + 0 & &= 75 \\
 x_4 &= 60 + \max\{0, \underline{45}\} & &= 105 \\
 x_5 &= 60 + \max\{0, 45, \underline{60}\} & &= 120 \\
 x_6 &= 120 + \max\{0, \underline{45}\} & &= 165 \\
 x_7 &= 90 + \max\{0, 45, 60, \underline{75}\} & &= 165 \\
 x_8 &= 90 + \max\{0, 45, 60, 75, 105, \underline{120}\} & &= 210
 \end{aligned}$$

Največji skupni čas najema je torej  $x^* = 210$ . Poiščimo še intervale, v katerih naj upravljalec odda dvorano.

$$\begin{aligned}
 x^* &= x_8 = k_8 - z_8 + x_5 & \text{interval } (165, 255) \\
 x_5 &= k_5 - z_5 + x_2 & \text{interval } (90, 150) \\
 x_2 &= k_2 - z_2 + x_0 & \text{interval } (30, 90)
 \end{aligned}$$

#### Naloga 4.26.

Ker je predvideno povpraševanje znano vnaprej, bo zadostovalo, da minimiziramo stroške, saj je prodajna cena fiksna.

- (a) Naj bodo  $s_{ij}$  najmanjši skupni stroški, ki jih lahko ima Jatifi za nabavo pomaranč do  $j$ -tega meseca, če zadnjič kupuje v  $i$ -tem mesecu. Poleg tega naj bo  $H_{ij}$  skupna cena skladiščenja škatle, kupljene v mesecu  $i$  in prodane v mesecu  $j$ . Zapišimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned}
 c_0 &= h_0 = \infty \\
 s_{00} &= 0 \\
 H_{jj} &= 0 & (0 \leq j \leq n) \\
 s_{jj} &= K_j + r_j c_j + \min\{s_{i,j-1} \mid 0 \leq i \leq j-1\} & (1 \leq j \leq n) \\
 H_{ij} &= h_{j-1} + H_{i,j-1} & (0 \leq i < j \leq n) \\
 s_{ij} &= s_{i,j-1} + r_j(c_i + H_{ij}) & (0 \leq i < j \leq n)
 \end{aligned}$$

Vrednosti  $H_{ij}$  in  $s_{ij}$  računamo v leksikografskem vrstnem redu glede na indeksa  $(i, j)$ , pri čemer upoštevamo  $0 \cdot \infty = 0$  (na ta način se izognemo plačilu fiksnih stroškov za morebitne začetne mesece brez povpraševanja). Najmanjše skupne stroške za celotno obdobje dobimo kot  $s^* = \min\{s_{in} \mid 0 \leq i \leq n\}$ .

Algoritem izračuna  $O(n^2)$  vrednosti, pri čemer za izračun vrednosti  $s_{jj}$  ( $1 \leq j \leq n$ ) porabi  $O(j)$  korakov, ostale vrednosti pa izračuna v konstantnem času. Časovna zahtevnost algoritma je tako  $O(n^2)$ .

- (b) Ker velja  $r_1 > 0$ , imamo  $H_{0j} = s_{0j} = \infty$  ( $1 \leq j \leq n$ ). Izračunajmo še vrednosti  $H_{ij}$  in  $s_{ij}$  ( $1 \leq i \leq j \leq n$ ).

$$\begin{array}{lll}
H_{11} = 0 & s_{11} = 2 + 20 \cdot 3 + 0 & = 62 \\
H_{12} = 1 & s_{12} = 62 + 40 \cdot (3 + 1) & = 222 \\
H_{13} = 2 & s_{13} = 222 + 15 \cdot (3 + 2) & = 297 \\
H_{14} = 3 & s_{14} = 297 + 10 \cdot (3 + 3) & = 357 \\
H_{22} = 0 & s_{22} = 3 + 40 \cdot 4 + 62 & = 225 \\
H_{23} = 1 & s_{23} = 225 + 15 \cdot (4 + 1) & = 300 \\
H_{24} = 2 & s_{24} = 300 + 10 \cdot (4 + 2) & = 360 \\
H_{33} = 0 & s_{33} = 4 + 15 \cdot 3 + \min\{222, 225\} & = 271 \\
H_{34} = 1 & s_{34} = 271 + 10 \cdot (3 + 1) & = 311 \\
H_{44} = 0 & s_{44} = 2 + 10 \cdot 4 + \min\{297, 300, 271\} & = 313
\end{array}$$

Najmanjši skupni stroški za celotno obdobje so torej  $s^* = 311$ . Poiščimo še optimalno strategijo kupovanja pomaranč.

$$\begin{aligned}
s^* = s_{34} &= K_3 + r_3 c_3 + r_4 (c_3 + H_{34}) + s_{12} && \text{kupi v 3. mesecu za 3. in 4. mesec} \\
s_{12} &= K_1 + r_1 c_1 + r_2 (c_1 + H_{12}) + s_{00} && \text{kupi v 1. mesecu za 1. in 2. mesec}
\end{aligned}$$

#### Naloga 4.27.

Zapišimo začetne pogoje in rekurzivne enačbe za reševanje danega problema.

$$\begin{aligned}
b_{ii} &= a_{ii} && (1 \leq i \leq n) \\
b_{ij} &= a_{ij} + b_{i,j-1} && (1 \leq i < j \leq n) \\
c_{0j} &= 1 && (1 \leq j \leq n) \\
c_{ii} &= c_{i-1,i} + b_{ii} && (1 \leq i \leq n) \\
c_{ij} &= \min\{c_{i,j-1}, c_{i-1,j} + b_{ij}\} && (1 \leq i < j \leq n)
\end{aligned}$$

Pokažimo, da zgornje enačbe ustrezajo podani rekurziji. Najprej dokažimo, da velja

$$b_{ij} = \sum_{u=i}^j a_{iu}.$$

Zgornja enakost očitno velja za  $j = i$ . Denimo, da velja tudi za  $j = j_0 \geq i$ . Potem velja

$$b_{i,j_0+1} = a_{i,j_0+1} + b_{ij_0} = a_{i,j_0+1} + \sum_{u=i}^{j_0} a_{iu} = \sum_{u=i}^{j_0+1} a_{iu},$$

tako da po indukciji zgornja trditev velja za vse  $j \geq i$ . Nadalje dokažimo, da velja

$$c_{ij} = \min\{c_{i-1,k} + b_{ik} \mid i \leq k \leq j\}.$$



Zgornja enakost spet očitno velja za  $j = i$ . Denimo, da velja tudi za  $j = j_0 \geq i$ . Potem velja

$$\begin{aligned} c_{i,j_0+1} &= \min\{c_{ij_0}, c_{i-1,j_0+1} + b_{i,j_0+1}\} \\ &= \min\{\min\{c_{i-1,k} + b_{ik} \mid i \leq k \leq j_0\}, c_{i-1,j_0+1} + b_{i,j_0+1}\} \\ &= \min\{c_{i-1,k} + b_{ik} \mid i \leq k \leq j_0 + 1\}, \end{aligned}$$

tako da po indukciji zgornja trditev velja za vse  $j \geq i$ . Tako velja

$$c_{ij} = \min \left\{ c_{i-1,k} + \sum_{u=i}^k a_{iu} \mid i \leq k \leq j \right\}.$$

Vrednost  $c_{nn}$  je tako mogoče izračunati tako, da izračunamo vrednosti  $b_{ij}$  in  $c_{ij}$  za vsak par  $(i, j)$  z  $1 \leq i \leq j \leq n$  v leksikografskem vrstnem redu. Ker je mogoče vsako vrednost izračunati v konstantnem času, je mogoče celoten izračun opraviti v času  $O(n^2)$ .

#### Naloga 4.28.

- (a) Denimo, da imamo  $n$  kupcev in  $m$  predmetov ( $m \leq n$ ). Naj bo  $v_{ij}$  največja vsota, ki si jo lahko prislužimo s prodajo prvih  $i$  predmetov prvim  $j$  kupcem. Zapišimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned} v_{0j} &= 0 & (1 \leq j \leq n) \\ v_{i,i-1} &= v_{i-1,i-1} & (1 \leq i \leq m) \\ v_{ij} &= \max\{\{v_{i,j-1}\} \cup \{v_{i-1,j-1} + c_j \mid T_j = P_i\}\} & (0 \leq i \leq m, i \leq j \leq n) \end{aligned}$$

Vrednosti  $v_{ij}$  računamo v leksikografskem vrstnem redu glede na indeksa  $(i, j)$ . Največji zaslužek dobimo kot  $v^* = v_{mn}$ . Optimalno rešitev rekonstruiramo tako, da za vsak  $j = n, n-1, \dots, 1$  in trenutno vrednost  $i$  z začetno vrednostjo  $i = m$  preverimo, ali velja  $v_{ij} = v_{i,j-1}$  – če to velja, potem  $i$ -tega predmeta nismo prodali  $i$ -temu kupcu, sicer pa smo ga in vrednost  $i$  zmanjšamo za 1.

- (b) Algoritem izračuna  $\sum_{i=1}^m (n-i+1)$  vrednosti  $v_{ij}$ , vsak izračun pa opravi v konstantnem času. Ker za rekonstrukcijo rešitve potrebuje  $n$  korakov, je skupna časovna zahtevnost  $O(mn)$ .

- (c) Izračunajmo vrednosti  $v_{ij}$  ( $1 \leq i \leq 2, i \leq j \leq 5$ ).

$$\begin{aligned} v_{11} &= \max\{0, \underline{0+2}\} = 2 \\ v_{12} &= 2 \\ v_{13} &= \max\{2, \underline{0+7}\} = 7 \\ v_{14} &= \max\{\underline{7}, 0+1\} = 7 \end{aligned}$$

$$\begin{aligned}
v_{15} &= 7 \\
v_{22} &= \max\{2, \underline{2+2}\} = 4 \\
v_{23} &= 4 \\
v_{24} &= 4 \\
v_{25} &= \max\{4, \underline{7+1}\} = 8
\end{aligned}$$

Igralec lahko torej prisluži največ  $v^* = v_{25} = 8$  enot denarja. Rekonstruirajmo še optimalno rešitev.

$$\begin{aligned}
v_{25} &= v_{14} + c_5 && \text{drugi predmet prodamo petemu kupcu} \\
v_{14} &= v_{13} = v_{12} + c_3 && \text{prvi predmet prodamo tretjemu kupcu}
\end{aligned}$$

**Naloga 4.29.**

- (a) Denimo, da imamo v tabeli nize  $w_h$  ( $1 \leq h \leq m$ ) dolžine  $\ell_h \leq k$ , ter da imamo podan niz  $D = (u_i)_{i=1}^n$ . Naj bo  $d_j$  dolžina najkrajšega kodiranja niza  $D_j = (u_i)_{i=1}^j$  s kodnimi besedami iz tabele. Zapišimo začetni pogoj in rekurzivne enačbe.

$$\begin{aligned}
d_0 &= 0 \\
d_j &= 1 + \min\left(\{d_{j-\ell_h} \mid 1 \leq h \leq m \wedge \ell_h \leq j \wedge D_j = D_{j-\ell_h} \| w_h\} \cup \{\infty\}\right) \\
&\quad (1 \leq j \leq n)
\end{aligned}$$

Vrednosti  $d_j$  računamo v naraščajočem vrstnem redu glede na indeks  $j$ . Če velja  $d_n = \infty$ , potem ustrezno kodiranje ne obstaja, sicer pa dolžino najkrajšega kodiranja dobimo kot  $d^* = d_n$ .

- (b) Izračunajmo vrednosti  $d_j$  ( $1 \leq j \leq n$ ). Ker niz  $D$  sestoji samo iz znakov  $a$  in  $b$ , ki se kot niza pojavita v tabeli, bodo vse vrednosti  $d_j$  končne.

$$\begin{aligned}
u_1 &= b & d_1 &= 1 + \min\{\underline{0_b}\} &= 1 \\
u_2 &= a & d_2 &= 1 + \min\{1_a, \underline{0_{ba}}\} &= 1 \\
u_3 &= b & d_3 &= 1 + \min\{\underline{1_b}\} &= 2 \\
u_4 &= a & d_4 &= 1 + \min\{2_a, \underline{1_{ba}}\} &= 2 \\
u_5 &= b & d_5 &= 1 + \min\{2_b, \underline{1_{abab}}\} &= 2 \\
u_6 &= b & d_6 &= 1 + \min\{\underline{2_b}\} &= 3 \\
u_7 &= a & d_7 &= 1 + \min\{3_a, \underline{2_{ba}}\} &= 3 \\
u_8 &= a & d_8 &= 1 + \min\{\underline{3_a}\} &= 4 \\
u_9 &= b & d_9 &= 1 + \min\{\underline{4_b}\} &= 5 \\
u_{10} &= a & d_{10} &= 1 + \min\{5_a, \underline{4_{ba}}\} &= 5
\end{aligned}$$

$$\begin{aligned} u_{11} &= b & d_{11} &= 1 + \min\{5_b, 3_{abab}\} = 4 \\ u_{12} &= a & d_{12} &= 1 + \min\{4_a, 5_{abab}\} = 5 \end{aligned}$$

Dolžina najkrajšega kodiranja je torej  $d^* = d_{12} = 5$ . Rekonstruirajmo še ustrezno kodiranje.

$$\begin{aligned} d_{12} &= 1 + d_{11} & a \\ d_{11} &= 1 + d_7 & abab \\ d_7 &= 1 + d_5 & ba \\ d_5 &= 1 + d_1 & abab \\ d_1 &= 1 + d_0 & b \end{aligned}$$

Najkrajše kodiranje je torej res  $(b, abab, ba, abab, a)$ .

- (c) V algoritmu izračunamo  $n$  spremenljivk, pri izračunu vsake pa primerjamo  $m$  nizov, pri čemer vsakič primerjamo največ  $k$  znakov. Časovna zahtevnost algoritma je torej  $O(nmk)$ .

#### Naloga 4.30.

Naj bo  $v_i$  višina najvišjega stolpa, v katerem je na vrhu  $i$ -ta kocka. Zapišimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned} a_0 &= b_0 = \infty \\ v_0 &= 0 \\ v_i &= \max\{v_j + c_i \mid 0 \leq j \leq i-1, a_i < a_j, b_i < b_j\} \quad (1 \leq i \leq n) \end{aligned}$$

Vrednosti  $v_i$  računamo naraščajoče po indeksu  $i$  ( $1 \leq i \leq n$ ). Največjo višino dobimo kot  $v^* = \max\{v_i \mid 1 \leq i \leq n\}$ .

#### Naloga 4.31.

- (a) Naj bosta  $(a_i)_{i=1}^m$  in  $(b_j)_{j=1}^n$  podani zaporedji nukleotidov, spremenljivka  $r_{uv}$  ( $0 \leq u \leq m, 0 \leq v \leq n$ ) pa naj predstavlja najmanjše število operacij, s katerim pridemo od zaporedja  $(a_i)_{i=1}^u$  do zaporedja  $(b_j)_{j=1}^v$ . Zapišimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned} r_{0v} &= v & (0 \leq v \leq n) \\ r_{u0} &= u & (0 \leq u \leq m) \\ r_{uv} &= 1 + \min\left(\{r_{u-1,v-1}, r_{u-1,v}, r_{u,v-1}\} \cup \{r_{u-1,v-1} - 1 \mid a_u = b_v\} \right. \\ &\quad \left. \cup \{r_{u-2,v-2} \mid u, v \geq 2 \wedge a_{u-1} = b_v \wedge a_u = b_{v-1}\} \right) & (1 \leq u \leq m, 1 \leq v \leq n) \end{aligned}$$

Vrednosti  $r_{uv}$  računamo v leksikografskem vrstnem redu glede na indeksa  $(u, v)$ , najmanjše potrebno število operacij pa dobimo kot  $r^* = r_{mn}$ . Algoritem, ki sledi iz zgornjih rekurzivnih enačb, teče v času  $O(mn)$ .

(b) Zapišimo matriko vrednosti  $(r_{uv})_{u,v=0}^8$ .

$r_{uv}$	$\epsilon$	A	C	C	A	T	G	T	A
$\epsilon$	0	1	2	3	4	5	6	7	8
A	1	0	1	2	3	4	5	6	7
A	2	1	1	2	2	3	4	5	6
C	3	2	1	1	2	3	4	5	6
A	4	3	2	2	1	2	3	4	5
G	5	4	3	3	2	2	2	3	4
T	6	5	4	4	3	2	2	2	3
T	7	6	5	5	4	3	3	2	3
A	8	7	6	6	5	4	4	3	2

Najmanjše število operacij je tako  $r_{88} = 2$ , dosežemo pa ga tako, da opravimo substitucijo  $A \rightarrow C$  na drugem mestu in transpozicijo  $GT \rightarrow TG$  na šestem in sedmem mestu.

#### Naloga 4.32.

Denimo, da imamo zaporedje simbolov  $(s_h)_{h=0}^n$  in zaporedje veznikov  $(v_h)_{h=1}^n$ , pri čemer se veznik  $v_h$  pojavi med simboloma  $s_{h-1}$  in  $s_h$ . Naj bosta  $T_{ij}$  in  $F_{ij}$  števili takih postavitvev oklepajev v zaporedje simbolov  $(s_h)_{h=i}^j$  z vezniki  $(v_h)_{h=i+1}^j$ , da ima ustrezni izraz vrednost  $\top$  oziroma  $\perp$ . Zapišimo začetne pogoje in rekurzivne enačbe.

$$T_{ii} = \begin{cases} 1 & s_i = \top \\ 0 & s_i = \perp \end{cases} \quad (0 \leq i \leq n)$$

$$F_{ii} = 1 - T_{ii} \quad (0 \leq i \leq n)$$

$$T_{ij} = \sum_{h=i+1}^j \begin{cases} T_{i,h-1} T_{hj} & v_h = \wedge \\ T_{i,h-1} T_{hj} + F_{i,h-1} T_{hj} + T_{i,h-1} F_{hj} & v_h = \vee \\ F_{i,h-1} T_{hj} + T_{i,h-1} F_{hj} & v_h = \oplus \end{cases} \quad (0 \leq i < j \leq n)$$

$$F_{ij} = \sum_{h=i+1}^j \begin{cases} F_{i,h-1} T_{hj} + T_{i,h-1} F_{hj} + F_{i,h-1} F_{hj} & v_h = \wedge \\ F_{i,h-1} F_{hj} & v_h = \vee \\ T_{i,h-1} T_{hj} + F_{i,h-1} F_{hj} & v_h = \oplus \end{cases} \quad (0 \leq i < j \leq n)$$

Vrednosti  $T_{ij}$  in  $F_{ij}$  računamo v leksikografskem vrstnem redu glede na  $(j-i, i)$ , število postavitvev oklepajev, da je vrednost celotnega izraza enaka  $\top$  ali  $\perp$ , pa dobimo kot  $T^* = T_{0n}$  oziroma  $F^* = F_{0n}$ . Algoritem, ki sledi iz zgornjih rekurzivnih enačb, teče v času  $O(n^3)$ .

### Naloga 4.33.

- (a) Naj bodo  $s_{ij}$  pričakovani stroški, ki bodo nastali, če Stekelce pri  $i$ -tem naročilu ( $i = 1, 2$ ) potrebuje še  $j$  kroglic ( $j = 0, 1, 2$ ). Določimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned}s_{22} &= \min(\{k\} \cup \{z_2 + cx + (p + x(1-p))p^{x-1}k \mid x \geq 2, x \in \mathbb{Z}\}) \\s_{21} &= \min(\{k\} \cup \{z_2 + cx + p^x k \mid x \geq 1, x \in \mathbb{Z}\}) \\s_{12} &= \min(\{s_{22}\} \cup \{z_1 + cx + p^x s_{22} + x(1-p)p^{x-1}s_{21} \mid x \geq 1, x \in \mathbb{Z}\})\end{aligned}$$

Tukaj  $c$  predstavlja ceno posamezne kroglice,  $z_i$  ( $i = 1, 2$ ) stroške zagona izdelave pri  $i$ -tem naročilu,  $k$  kazen ob neizstavitvi naročila, in  $p$  verjetnost, da bo posamezna kroglica neuporabna. Pričakovani stroški so  $s^* = s_{12}$ .

Izračunajmo zgornje vrednosti pri  $c = 100\text{€}$ ,  $z_1 = z_2 = 300\text{€}$ ,  $k = 800\text{€}$  in  $p = 0.5$ . Opazimo, da imajo izrazi na desni natanko en lokalni minimum (za  $x \in \mathbb{R}$ ), tako da zadostuje obravnavati le tiste vrednosti  $x$ , kjer vrednost izraza še pada.

$$\begin{aligned}s_{22} &= \min\{800\text{€}, 1\,100\text{€}, 1\,000\text{€}, 950\text{€}, 950\text{€}, \dots\} &&= 800\text{€} \\s_{21} &= \min\{800\text{€}, 800\text{€}, 700\text{€}, 700\text{€}, \dots\} &&= 700\text{€} \\s_{12} &= \min\{800\text{€}, 1\,150\text{€}, 1\,050\text{€}, 962.5\text{€}, 925\text{€}, 934.375\text{€}, \dots\} &&= 800\text{€}\end{aligned}$$

Kroglic se ji torej ne izplača naročati, saj bo do najmanjših pričakovanih stroškov prišlo, če enostavno plača kazen.

- (b) Rekurzivne enačbe iz prejšnje točke rešimo še za primer, ko velja  $z_1 = 150\text{€}$ . Ker so ostali parametri nespremenjeni, zadostuje, da znova izračunamo le vrednost  $s_{12}$ .

$$s_{12} = \min\{800\text{€}, 1\,000\text{€}, 900\text{€}, 812.5\text{€}, 775\text{€}, 784.375\text{€}, \dots\} = 775\text{€}$$

Pri prvem naročilu naj torej naroči 4 kroglice. Če bo le ena uporabna, naj drugič naroči še 2 ali 3 kroglice, če pa nobena ne bo uporabna, naj plača kazen.

### Naloga 4.34.

Naj bo  $p_{ij}$  največja teža spusta v trikotniku z vrhom v  $a_{ij}$ . Določimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned}p_{nj} &= a_{nj} && (1 \leq j \leq n) \\p_{ij} &= a_{ij} + \max\{p_{i+1,j}, p_{i+1,j+1}\} && (1 \leq j \leq i \leq n-1)\end{aligned}$$

Vrednosti  $p_{ij}$  računamo v padajočem leksikografskem vrstnem redu glede na  $(i, j)$ . Težo najtežega spusta dobimo kot  $p^* = p_{11}$ . Sam spust  $(a_{i,j_i})_{i=1}^n$  lahko dobimo tako, da izračunamo

$$j_1 = 1$$

$$j_i = \begin{cases} j_{i-1} & p_{i,j_{i-1}} \geq p_{i,j_{i-1}+1}, \text{ in} \\ j_{i-1} + 1 & \text{sicer} \end{cases} \quad (2 \leq i \leq n)$$

v naraščajočem vrstnem redu glede na indeks  $i$ .

**Naloga 4.35.**

Naj bodo  $s_{ij}$  pričakovani stroški, ki jih bo imel Antonio, če mora naročiti  $i$  prstanov, na voljo pa ima še  $j$  naročil. Zapišimo začetne pogoje in rekurzivne enačbe.

$$s_{ij} = 0\text{€} \quad (i \leq 0, 0 \leq j \leq 2)$$

$$s_{i0} = 1600\text{€} \quad (1 \leq i \leq 2)$$

$$s_{ij} = 200\text{€} + \min \left\{ h \cdot 100\text{€} + \frac{1}{2^h} \sum_{k=0}^h \binom{h}{k} s_{i-h,j-1} \mid h \geq 1 \right\} \quad (1 \leq i, j \leq 2)$$

Izračunajmo najprej vrednosti  $s_{11}$  in  $s_{21}$ , nato pa še skupne pričakovane stroške  $s^* = s_{22}$ . Opazimo, da minimiziramo vsoto linearno naraščajočega in eksponentno padajočega člena, tako da lahko računanje ustavimo, ko začne ta vsota naraščati.

$$s_{11} = 200\text{€} + \min\{900\text{€}, 600\text{€}, \underline{500\text{€}}, \underline{500\text{€}}, 550\text{€}, \dots\} = 700\text{€}$$

$$s_{21} = 200\text{€} + \min\{1700\text{€}, 1400\text{€}, 1100\text{€}, 900\text{€}, 800\text{€}, \underline{775\text{€}}, 800\text{€}, \dots\} = 975\text{€}$$

$$s_{22} \approx 200\text{€} + \min\{937.5\text{€}, 793.75\text{€}, 684.38\text{€}, \underline{635.94\text{€}}, 639.84\text{€}, \dots\} \approx 835.94\text{€}$$

Antonio naj torej naroči 4 prstane. Če uspe graviranje samo enega prstana, naj v drugo naroči še 3 ali 4 prstane, če pa ne uspe graviranje nobenega prstana, naj v drugo naroči še 6 prstanov. Pričakovani stroški bodo tedaj približno 835.94€.

**Naloga 4.36.**

Naj bo  $v_{ij}$  največja vsota števil izmed  $a_1, a_2, \dots, a_i$ , ki ne preseže vrednosti  $j$ . Definirajmo še  $s = \lfloor \frac{1}{2} \sum_{i=1}^n a_i \rfloor$  (tj., največja možna vrednost za  $\min\{\sum S_1, \sum S_2\}$ ). Določimo začetne pogoje in rekurzivne enačbe.

$$v_{00} = v_{i0} = v_{0j} = 0 \quad (1 \leq i \leq n, 1 \leq j \leq s)$$

$$v_{ij} = \max(\{v_{i-1,j}\} \cup \{a_i + v_{i-1,j-a_i} \mid a_i \leq j\}) \quad (1 \leq i \leq n, 1 \leq j \leq s)$$

Vrednosti  $v_{ij}$  računamo v leksikografskem vrstnem redu indeksov (npr. najprej naraščajoče po  $i$ , nato pa naraščajoče po  $j$ ). Optimalna vrednost  $v^* = v_{ns}$  nam pove največjo vrednost za  $\sum S_1$  (brez škode za splošnost lahko predpostavimo, da ta ne preseže  $\sum S_2$ ); razliko dobimo kot  $\sum_{i=1}^n a_i - 2v^*$ .

**Naloga 4.37.**

Definirajmo

$$a'_{ij} = \begin{cases} -\infty & a_{ij} = 4, \text{ in} \\ a_{ij} & \text{sicer.} \end{cases} \quad (1 \leq j \leq i \leq n)$$

- (a) Naj bo  $p_{ij}$  največja teža za Marina veljavne poti od vznožja do  $a_{ij}$ . Določimo začetne pogoje in rekurzivne enačbe.

$$p_{nj} = a'_{nj} \quad (1 \leq j \leq n)$$

$$p_{ij} = a'_{ij} + \max\{p_{i+1,j}, p_{i+1,j+1}\} \quad (1 \leq j \leq i \leq n-1)$$

Vrednosti  $p_{ij}$  računamo v padajočem leksikografskem vrstnem redu glede na  $(i, j)$ . Če Marin s svojim znanjem ne more priti do vrha po veljavni poti, potem velja  $p_{11} = -\infty$ , sicer pa težo najtežje veljavne poti dobimo kot  $p^* = p_{11}$ . Samo pot  $(a_{n-i,j_i})_{i=0}^{n-1}$  lahko dobimo tako, da izračunamo

$$j_{n-1} = 1$$

$$j_i = \begin{cases} j_{i+1} & p_{n-i,j_{i+1}} \geq p_{n-i,j_{i+1}+1}, \text{ in} \\ j_{i+1} + 1 & \text{sicer} \end{cases} \quad (0 \leq i \leq n-2)$$

v padajočem vrstnem redu glede na indeks  $i$ .

- (b) Naj bosta  $q_{ij}$  in  $r_{ij}$  največji teži za Mirka veljavne poti od vznožja do  $a_{ij}$ , če dovolimo  $a_{ij} = 4$  oziroma če tega ne dovolimo. Določimo začetne pogoje in rekurzivne enačbe.

$$q_{nj} = a_{nj} \quad (1 \leq j \leq n)$$

$$r_{nj} = a'_{nj} \quad (1 \leq j \leq n)$$

$$q_{ij} = a_{ij} + \max\{r_{i+1,j}, r_{i+1,j+1}\} \quad (1 \leq j \leq i \leq n-1)$$

$$r_{ij} = a'_{ij} + \max\{q_{i+1,j}, q_{i+1,j+1}, r_{i+1,j}, r_{i+1,j+1}\} \quad (1 \leq j \leq i \leq n-1)$$

Vrednosti  $q_{ij}$  in  $r_{ij}$  računamo v padajočem leksikografskem vrstnem redu glede na  $(i, j)$ . Če Mirko s svojim znanjem ne more priti do vrha po veljavni poti, potem velja  $q_{11} = r_{11} = -\infty$ , sicer pa težo najtežje veljavne poti dobimo kot  $q^* = \max\{q_{11}, r_{11}\}$ . Samo pot  $(a_{n-i,j_i})_{i=0}^{n-1}$  lahko dobimo tako, da izračunamo

$$j_{n-1} = 1$$

$$j_i = \begin{cases} j_{i+1} & a_{n-i-1,j_{i+1}} < 4 \wedge \\ & r_{n-i-1,j_{i+1}} - a_{n-i-1,j_{i+1}} \in \{q_{n-i,j_{i+1}}, r_{n-i,j_{i+1}}\}, \\ j_{i+1} & a_{n-i-1,j_{i+1}} = 4 \wedge r_{n-i,j_{i+1}} \geq r_{n-i,j_{i+1}+1}, \text{ in} \\ j_{i+1} + 1 & \text{sicer} \end{cases} \quad (0 \leq i \leq n-2)$$

v padajočem vrstnem redu glede na indeks  $i$ .

**Naloga 4.38.**

Naj bosta  $p_{ij}$  in  $q_{ij}$  največji oziroma najmanjši znesek, ki ga lahko doseže igralec  $A$  iz zaporedja  $c_i, c_{i+1}, \dots, c_j$ , če je na potezi igralec  $A$  oziroma igralec  $B$ . Določimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned}
 p_{ii} &= c_i & (1 \leq i \leq n) \\
 q_{ii} &= 0 & (1 \leq i \leq n) \\
 q_{i,i+1} &= 0 & (1 \leq i \leq n-1) \\
 p_{ij} &= \max\{c_i + q_{i+1,j}, c_j + q_{i,j-1}\} & (1 \leq i < j \leq n) \\
 q_{ij} &= \min\{p_{i+2,j}, p_{i+1,j-1}, p_{i,j-2}\} & (1 \leq i \leq j-2 \leq n-2)
 \end{aligned}$$

Vrednosti  $p_{ij}$  in  $q_{ij}$  računamo v leksikografskem vrstnem redu glede na  $(j-i, i)$  – pravzaprav zadostuje, da izračunamo vrednosti  $p_{ij}$  z  $j-i \equiv n-1 \pmod{3}$  in vrednosti  $q_{ij}$  z  $j-i \equiv n-2 \pmod{3}$ . Maksimalni znesek igralca  $A$  ob optimalni igri igralca  $B$  dobimo kot  $p^* = p_{1n}$ .

**Naloga 4.39.**

- (a) Naj bo  $v_{ij}$  maksimalni profit, ki ga lahko kmet doseže po  $i$  letih, če mu bo na začetku  $i$ -tega leta (po prodaji) ostalo  $j$  ovac. Zapišimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned}
 v_{0k_0} &= 0 \\
 v_{0j} &= -\infty & (0 \leq j < k_0) \\
 v_{ij} &= \max\left\{v_{i-1,h} + p_i(2h-j) \mid j/2 \leq h \leq 2^{i-1}k_0\right\} & (1 \leq i \leq n, 0 \leq j \leq 2^i k_0)
 \end{aligned}$$

Vrednosti  $v_{ij}$  računamo v leksikografskem vrstnem redu glede na indeksa  $(i, j)$ . Maksimalni profit dobimo kot  $v^* = v_{n0}$ .

- (b) Izračunajmo vrednosti  $v_{ij}$  ( $1 \leq i \leq 2, 0 \leq j \leq 2 \cdot 2^i$ ) in  $v^* = v_{30}$ .

$$\begin{aligned}
 v_{10} &= 0\text{€} + 4 \cdot 100\text{€} & = 400\text{€} \\
 v_{11} &= 0\text{€} + 3 \cdot 100\text{€} & = 300\text{€} \\
 v_{12} &= 0\text{€} + 2 \cdot 100\text{€} & = 200\text{€} \\
 v_{13} &= 0\text{€} + 1 \cdot 100\text{€} & = 100\text{€} \\
 v_{14} &= 0\text{€} + 0 \cdot 100\text{€} & = 0\text{€} \\
 v_{20} &= \max\{400\text{€} + 0 \cdot 130\text{€}, 300\text{€} + 2 \cdot 130\text{€}, 200\text{€} + 4 \cdot 130\text{€}, \\
 &\quad 100\text{€} + 6 \cdot 130\text{€}, \underline{0\text{€} + 8 \cdot 130\text{€}}\} & = 1040\text{€} \\
 v_{21} &= \max\{300\text{€} + 1 \cdot 130\text{€}, 200\text{€} + 3 \cdot 130\text{€}, \\
 &\quad 100\text{€} + 5 \cdot 130\text{€}, \underline{0\text{€} + 7 \cdot 130\text{€}}\} & = 910\text{€} \\
 v_{22} &= \max\{300\text{€} + 0 \cdot 130\text{€}, 200\text{€} + 2 \cdot 130\text{€}, \\
 &\quad 100\text{€} + 4 \cdot 130\text{€}, \underline{0\text{€} + 6 \cdot 130\text{€}}\} & = 780\text{€}
 \end{aligned}$$



$$\begin{aligned}
v_{23} &= \max\{200\text{€} + 1 \cdot 130\text{€}, 100\text{€} + 3 \cdot 130\text{€}, \underline{0\text{€} + 5 \cdot 130\text{€}}\} &= 650\text{€} \\
v_{24} &= \max\{200\text{€} + 0 \cdot 130\text{€}, 100\text{€} + 2 \cdot 130\text{€}, \underline{0\text{€} + 4 \cdot 130\text{€}}\} &= 520\text{€} \\
v_{25} &= \max\{100\text{€} + 1 \cdot 130\text{€}, \underline{0\text{€} + 3 \cdot 130\text{€}}\} &= 390\text{€} \\
v_{26} &= \max\{100\text{€} + 0 \cdot 130\text{€}, \underline{0\text{€} + 2 \cdot 130\text{€}}\} &= 260\text{€} \\
v_{27} &= 0\text{€} + 1 \cdot 130\text{€} &= 130\text{€} \\
v_{28} &= 0\text{€} + 0 \cdot 130\text{€} &= 0\text{€} \\
v_{30} &= \max\{1\,040\text{€} + 0 \cdot 120\text{€}, 910\text{€} + 2 \cdot 120\text{€}, 780\text{€} + 4 \cdot 120\text{€}, \\
&\quad 650\text{€} + 6 \cdot 120\text{€}, 520\text{€} + 8 \cdot 120\text{€}, 390\text{€} + 10 \cdot 120\text{€}, \\
&\quad 260\text{€} + 12 \cdot 120\text{€}, 130\text{€} + 14 \cdot 120\text{€}, \underline{0\text{€} + 16 \cdot 120\text{€}}\} &= 1\,920\text{€}
\end{aligned}$$

Kmet lahko torej doseže profit največ 1 920€. Poiščimo še optimalno strategijo prodaje ovac.

$$\begin{aligned}
v^* &= v_{30} = v_{28} + 16p_3 && \text{ob prehodu v leto 3 proda vseh 16 ovac} \\
v_{28} &= v_{14} + 0p_2 && \text{ob prehodu v leto 2 ne prodaja, ostane mu 8 ovac} \\
v_{14} &= v_{02} + 0p_1 && \text{ob prehodu v leto 1 ne prodaja, ostanejo mu 4 ovce}
\end{aligned}$$

#### Naloga 4.40.

- (a) Naj bo  $q_{kj}$  maksimalna verjetnost, da bo prvih  $k$  opravil uspešno opravljenih, če na njih dela  $j$  delavcev. Zapišimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned}
q_{1j} &= p_1(j) && (0 \leq j \leq w) \\
q_{kj} &= \max\{q_{k-1,i} p_k(j-i) \mid 0 \leq i \leq j\} && (2 \leq k \leq N, 0 \leq j \leq w)
\end{aligned}$$

Vrednosti  $1_{kj}$  računamo v leksikografskem vrstnem redu glede na indeksa  $(k, j)$ . Maksimalno verjetnost dobimo kot  $q^* = q_{Nw}$ .

- (b) Izračunajmo vrednosti  $q_{2j}$  ( $0 \leq j \leq w$ ) in  $q^* = q_{35}$ .

$$\begin{aligned}
q_{20} &= 0 \cdot 0 && = 0 \\
q_{21} &= \max\{0 \cdot 0.4, 0.5 \cdot 0\} && = 0 \\
q_{22} &= \max\{0 \cdot 0.7, \underline{0.5 \cdot 0.4}, 0.6 \cdot 0\} && = 0.2 \\
q_{23} &= \max\{0 \cdot 0.8, \underline{0.5 \cdot 0.7}, 0.6 \cdot 0.4, 0.7 \cdot 0\} && = 0.35 \\
q_{24} &= \max\{0 \cdot 0.9, 0.5 \cdot 0.8, \underline{0.6 \cdot 0.7}, 0.7 \cdot 0.4, 0.8 \cdot 0\} && = 0.42 \\
q_{25} &= \max\{0 \cdot 0.95, 0.5 \cdot 0.9, 0.6 \cdot 0.8, \underline{0.7 \cdot 0.7}, 0.8 \cdot 0.4, 0.85 \cdot 0\} && = 0.49 \\
q_{35} &= \max\{0 \cdot 0.85, 0 \cdot 0.85, 0.2 \cdot 0.8, \underline{0.35 \cdot 0.6}, 0.42 \cdot 0.3, 0.49 \cdot 0\} && = 0.21
\end{aligned}$$

Maksimalna verjetnost uspešno opravljenih opravil je torej  $q^* = q_{35} = 0.21$ . Poiščimo še optimalno razporeditev delavcev.

$q^* = q_{35} = q_{23}p_3(2)$	2 delavca za tretje opravilo
$q_{23} = q_{11}p_2(2)$	2 delavca za drugo opravilo
$q_{11} = p_1(1)$	1 delavec za prvo opravilo

**Naloga 4.41.**

Naj bo  $s = (a_i)_{i=1}^n$  dani niz in  $W$  množica veljavnih turkmenskih besed, spremenljivka  $v_j$  ( $0 \leq j \leq n$ ) pa nam pove, ali je mogoče niz  $s_j = (a_i)_{i=1}^j$  razbiti na zaporedje veljavnih turkmenskih besed. Zapišimo začetni pogoj in rekurzivne enačbe.

$$v_0 = \top$$

$$v_j = \exists w \in W : (|w| \leq j \wedge v_{j-|w|} \wedge s_j = s_{j-|w|} \| w) \quad (1 \leq j \leq n)$$

Vrednosti  $v_j$  računamo v naraščajočem vrstnem redu glede na indeks  $j$ . Informacijo o tem, ali je mogoče niz  $s$  razbiti na zaporedje veljavnih turkmenskih besed, dobimo kot  $d^* = d_n$ .

**Naloga 4.42.**

Naj bo  $p_i$  velikost največje skupine delavcev izmed  $d_1, d_2, \dots, d_i$ , da se nihče v tej skupini ne bo bal drugega v tej skupini. Določimo začetni pogoj in rekurzivne enačbe.

$$p_0 = 0$$

$$p_i = \max\{p_{i-1}, p_{i-k_i-1} + 1\} \quad (1 \leq i \leq n)$$

Vrednosti  $p_i$  računamo naraščajoče glede na indeks  $i$ . Maksimalno velikost skupine dobimo kot  $p^* = p_n$ .

Iskano skupino lahko iz vrednosti  $k_i$  ( $1 \leq i \leq n$ ) in  $p_i$  ( $1 \leq i \leq n$ ) rekonstruiramo s sledečim algoritmom.

```

function REKONSTRUIRAJSKUPINO( $((k_i)_{i=1}^n, (p_i)_{i=1}^n)$ )
   $S \leftarrow$  prazna množica
   $i \leftarrow n$ 
  while  $i > 0$  do
    if  $p_i = p_{i-1}$  then                                upoštevamo  $p_0 = 0$ 
       $i \leftarrow i - 1$ 
    else
       $S.add(i)$ 
       $i \leftarrow i - k_i - 1$ 
    end if
  end while
  return  $S$ 
end function

```

**Naloga 4.43.**

- (a) Naj bo  $V_i$  največja vsota nesosednjih členov zaporedja, ki jo lahko dobimo, če gledamo samo elemente od 1 do  $i$ , in vključuje  $a_i$ .

$$\begin{aligned} V_{-1} &= 0 \\ V_0 &= 0 \\ V_i &= \max_{j < i-1} (V_j) + a_i \quad (1 \leq i \leq n) \end{aligned}$$

Vidimo, da na  $i$ -tem koraku iščemo največjo prejšnjo nesosednjo vsoto  $V_j$ , ki ji bomo priključili element  $a_i$ , torej izračunamo največjo vsoto, ki zadošča pogoju nesosednosti in vključuje člen  $a_i$ .

Za izračun vrednosti  $V_i$  moramo imeti podane že vse prejšnje  $V_j$ , torej vse  $V_j$  z  $j < i$ . To dosežemo tako, da te vrednosti računamo naraščajoče z začetkom v 1 in koncem v  $n$ . Po izračunu vseh vrednosti  $V_i$  bo maksimalna vrednost iskanega problema enaka  $V^* = \max_{1 \leq i \leq n} (V_i)$ .

Podamo lahko še alternativno rekurzivno zvezo za ta problem, in sicer definirajmo  $U_i$  kot največjo vsoto nesosednjih členov zaporedja, ki jo dobimo, če gledamo samo elemente od 1 do  $i$ . Ta definicija se razlikuje od zgornje po tem, da dobljena vsota ne vključuje nujno elementa  $a_i$ .

$$\begin{aligned} U_{-1} &= 0 \\ U_0 &= 0 \\ U_i &= \max(U_{i-2} + a_i, U_{i-1}) \quad (1 \leq i \leq n) \end{aligned}$$

Ideja tega rekurzivnega zapisa je podobna zgornji, le da se namesto iskanja maksimuma odločamo, ali bomo  $i$ -ti člen vključili v vsoto oziroma ga bomo izpustili.

Vrstni red iskanja vrednosti  $U_i$  določimo z naraščajočim zaporedjem indeksov  $i$ , ko  $1 \leq i \leq n$ , saj imamo tako pred reševanjem problema  $U_i$  definirane vse predhodne vrednosti. Ko izračunamo vse vrednosti  $U_i$ , je maksimum vsote zaporedja enak  $U^* = U_n$ , kar sledi neposredno iz definicije vrednosti  $U_n$ .

- (b) Algoritem bo sledil drugi rekurzivni zvezi.

```
function MAXVSOTA( $(a_i)_{i=1}^n$ )
   $U_{-1} \leftarrow 0$ 
   $U_0 \leftarrow 0$ 
  for  $i = 1, 2, \dots, n$  do
     $U_i \leftarrow \max(U_{i-2} + a_i, U_{i-1})$ 
  end for
   $\ell \leftarrow$  prazen seznam
   $i \leftarrow n$ 
  while  $i > 0$  do
    if  $U_i = U_{i-1}$  then
```

$i$	$U_{-1}$	$U_0$	$U_1$	$U_2$	$U_3$	$U_4$	$U_5$	$U_6$	$U_7$	$U_8$	$U_9$
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	3	0	0	0	0	0	0	0
3	0	0	0	3	10	0	0	0	0	0	0
4	0	0	0	3	10	10	0	0	0	0	0
5	0	0	0	3	10	10	17	0	0	0	0
6	0	0	0	3	10	10	17	31	0	0	0
7	0	0	0	3	10	10	17	31	31	0	0
8	0	0	0	3	10	10	17	31	31	31	0
9	0	0	0	3	10	10	17	31	31	31	31

Tabela 16: Potek izvajanja algoritma za nalogo 4.43.

```

         $i \leftarrow i - 1$ 
    else
         $\ell.append(i)$ 
         $i \leftarrow i - 2$ 
    end if
end while
 $\ell.reverse()$ 
return  $(U_n, \ell)$ 
end function

```

Algoritem enostavno upošteva drugo rekurzivno zvezo in iterativno, naraščajoče po  $i$ , računa vrednosti  $U_i$ , dokler ne pride do  $U_n$ , kar vrne kot optimalno vrednost. Časovna zahtevnost je očitno  $O(n)$ , saj zanko izvedemo enkrat, v vsakem obhodu pa porabimo konstantno časa.

Izvajanje algoritma na primeru  $a = [-2, 3, 10, -4, 7, 21, 5, 0, -10]$  si lahko ogledamo v tabeli 16. Iz zadnje vrstice lahko ugotovimo, da je podzaporedje z največjo vsoto  $(a_3, a_6) = (10, 21)$ .

Algoritem vrne tudi indekse elementov podzaporedja, ki maksimizira vsoto. To doseže tako, da primerja sosednje elemente zaporedja  $(U_i)_{i=0}^n$  – če se  $U_{i-1}$  in  $U_i$  razlikujeta, potem mora očitno veljati  $U_i = U_{i-2} + a_i$ , sicer pa element  $a_i$  ni bil uporabljen v vsoti. Po koncu izvajanja zanke tako dobimo seznam indeksov elementov zaporedja, ki smo jih dejansko prišteli v vsoto.

#### Naloga 4.44.

- (a) Definirajmo usmerjen graf  $G_a$ , ki ima za vozlišča števila  $1, 2, \dots, n$  in povezavo  $i \rightarrow j$  natanko tedaj, ko velja  $i < j \leq i + a_i$ . Skratka, vsakemu členu (indeksu)

zaporedja dodelimo vozlišče in ga povežemo s tistimi, ki so iz njega dosegljiva v enem koraku.

Problem smo prevedli na osnovno nalogo dinamičnega programiranja, saj moramo poiskati najkrajšo pot med začetno in končno točko v usmerjenem, acikličnem grafu  $G_a$ .

Naj bo  $d_i$  najkrajša pot od vozlišča  $i$  do vozlišča  $j$  v grafu  $G_a$ . Rekurzivno zvezo potem lahko zapišemo kot

$$\begin{aligned} d_n &= 0 \\ d_i &= 1 + \min \{d_k \mid i + 1 \leq k \leq \min(i + a_i, n)\} \quad (1 \leq i < n) \end{aligned}$$

Da bomo imeli vse prejšnje probleme rešene pred trenutnim, bomo vrednosti  $d_i$  računali padajoče po  $i$  za  $1 \leq i \leq n$ , saj začnemo v zadnjem vozlišču  $n$  in se nato pomikamo nazaj. Po razrešitvi vseh problemov optimum  $d^*$  dobimo kot vrednost  $d_1$ , torej  $f(S) = d^* = d_1$ , kar sledi iz definicije vrednosti  $d_i$ .

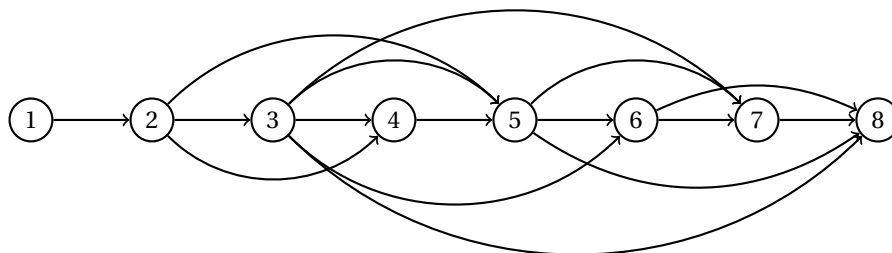
(b) Algoritem bo sledil rekurzivni zvezi.

```
function SKOKI( $((a_i)_{i=1}^n)$ )
   $d_n \leftarrow 0$ 
  for  $i = n - 1, n - 2, \dots, 1$  do
     $d_i, p_i \leftarrow \min \{(1 + d_k, k) \mid i + 1 \leq k \leq \min(i + a_i, n)\}$ 
  end for
   $pot \leftarrow$  prazen seznam
   $i \leftarrow 1$ 
  while  $i < n$  do
     $pot.append(i)$ 
     $i \leftarrow p_i$ 
  end while
   $pot.append(n)$ 
  return ( $d_1, pot$ )
end function
```

Algoritem ima časovno zahtevnost enako  $O(|V| + |E|)$ , kjer sta  $V$  in  $E$  množici vozlišč in povezav v grafu  $G_a$ , saj za vsako vozlišče pregledamo njegove vhodne povezave. Število vozlišč v našem grafu je  $n$ , število povezav pa je odvisno od elementov v vhodnem seznamu  $S$ . V najslabšem primeru lahko za vhod velja  $a_i \geq n - i$  za vsak  $i$ . Časovna zahtevnost našega algoritma bo potem  $O(n^2)$ , saj je število vozlišč enako  $n$ , število povezav pa lahko izračunamo kot

$$(n - 1) + (n - 2) + \dots + 1 + 0 = \frac{n(n - 1)}{2} = \frac{n^2 - n}{2} = O(n^2).$$

Poglejmo si delovanje algoritma na primeru  $S = [1, 3, 5, 1, 3, 10, 6, 13]$ . Ustrezni graf  $G_a$  je prikazan na sliki 80. V tabeli 17 je prikazan potek izvajanja algoritma. Algoritem nam vrne optimalno pot  $[1, 2, 3, 8]$ , torej je  $f(S) = 3$ .



Slika 80: Predstavitev skokov z grafom za nalogo 4.44(b).

$i$	$a_i$	$d_i$	$p_i$
8	13	0	
7	6	1	8
6	10	1	8
5	3	1	8
4	1	2	5
3	5	1	8
2	3	2	3
1	1	3	2

Tabela 17: Potek izvajanja algoritma za nalogo 4.44(b).

#### Naloga 4.45.

- (a) Naj bo  $p_{hik}$  število stolpcev v največji strnjeni podmatriki s samimi enicami z vrsticami od  $h$ -te do  $i$ -te vrstice matrice  $A$ , katere zadnji stolpec je  $k$ -ti stolpec matrice  $A$ . Poleg tega naj bo  $s_{ij}$  vsota prvih  $i$  komponent (tj., število enic) v  $j$ -tem stolpcu matrice  $A$ . Določimo začetne pogoje in rekurzivne enačbe.

$$s_{0j} = 0 \quad (1 \leq j \leq n)$$

$$s_{ij} = A_{ij} + s_{i-1,j} \quad (1 \leq i, j \leq n)$$

$$p_{hi0} = 0 \quad (1 \leq h \leq i \leq n)$$

$$p_{hik} = \begin{cases} p_{h,i,k-1} + 1 & s_{ik} - s_{h-1,k} = i - h + 1, \text{ in} \\ 0 & \text{sicer} \end{cases} \quad \begin{matrix} (1 \leq h \leq i \leq n, \\ 1 \leq k \leq n) \end{matrix}$$

Najprej za vsak  $j$  ( $1 \leq j \leq n$ ) izračunamo vrednosti  $s_{ij}$  naraščajoče po indeksu  $i$ , nato pa za vsaka  $h$  in  $i$  ( $1 \leq h \leq i \leq n$ ) izračunamo vrednosti  $p_{hik}$  naraščajoče po indeksu  $k$ . Največjo velikost strnjene podmatrike samih enic dobimo kot

$$N(A) = \max \{ (i - h + 1) p_{hik} \mid 1 \leq h \leq i \leq m, 0 \leq k \leq n \}.$$

- (b) Zapišimo algoritem, ki bo vrnil velikost največje strnjene podmatrike samih enic skupaj s paroma, ki določata prvo in zadnjo vrstico oziroma stolpec. Za razliko od zgornjih rekurzivnih enačb bo tukaj  $p_{hik}$  označeval velikost matrike in ne samo števila stolpcev.

```

function ENICE( $A \in \{0, 1\}^{n \times n}$ )
  for  $j = 1, \dots, n$  do                                računanje  $s_{ij}$ 
     $s_{0j} \leftarrow 0$                                     začetni pogoj
    for  $i = 1, \dots, n$  do
       $s_{ij} \leftarrow A_{ij} + s_{i-1,j}$                     rekurzivna enačba
    end for
  end for
   $p^*, h^*, i^*, j^*, k^* \leftarrow 0, 0, 0, 0, 0$           ničelna rešitev
  for  $h = 1, \dots, n$  do                                računanje  $p_{hik}$ 
    for  $i = h, \dots, n$  do
       $\ell \leftarrow i - h + 1$                             število vrstic
       $p_{hi0} \leftarrow 0$                                 začetni pogoj
       $j \leftarrow 1$                                     podmatriko začnemo s prvim stolpcem
      for  $k = 1, \dots, n$  do
        if  $s_{ik} - s_{h-1,k} = \ell$  then                    preverimo, ali imamo
           $p_{hik} \leftarrow p_{h,i,k-1} + \ell$                 v trenutnem stolpcu same enice
          if  $p_{hik} > p^*$  then                                preverimo, ali imamo boljše rešitev
             $p^*, h^*, i^*, j^*, k^* \leftarrow p_{hik}, h, i, j, k$ 
          end if
        else                                                če nimamo samih enic, bomo naslednjo
           $j \leftarrow k + 1$                                 podmatriko začeli v naslednjem stolpcu
           $p_{hik} \leftarrow 0$ 
        end if
      end for
    end for
  end for
  return  $p^*, (h^*, i^*), (j^*, k^*)$ 
end function

```

Za računanje  $s_{ij}$  potrebujemo  $O(n^2)$  korakov, za računanje  $p_{hik}$  pa  $O(n^3)$  korakov. Skupna časovna zahtevnost algoritma je torej  $O(n^3)$ .

#### Naloga 4.46.

Problem bomo razdelili na tri dele, pri čemer bomo v  $i$ -tem delu obravnavali situacijo, ko vlagamo v prvih  $i$  investicij. Vrednost  $p_i(x)$  naj torej označuje največji zaslužek, če v prvih  $i$  investicij vložimo znesek  $x$ .

Ker vrednost  $r_1(x)$  narašča z  $x$ , velja  $p_1(x) = r_1(x)$ . Sedaj določimo  $p_2(x)$ .

$$\begin{aligned}
 p_2(x) &= \max \{ p_1(y) + r_2(x-y) \mid 0 \leq y \leq x \} \\
 &= \begin{cases} \max \left( \begin{aligned} &\{3x-3y+7 \mid 0 \leq y < 1, y \leq x-1\} \\ &\cup \{3x+4y+9 \mid 1 \leq y \leq x-1\} \\ &\cup \{0 \mid x-1 < y < 1\} \\ &\cup \{7y+2 \mid x-1 < y \leq x, y \geq 1\} \end{aligned} \right) & \text{če } x \geq 1, \text{ in} \\ 0 & \text{sicer} \end{cases} \\
 &= \begin{cases} 0 & 0 \leq x < 1 \\ 3x+7 & 1 \leq x < \frac{5}{4}, \\ 7x+2 & \frac{5}{4} \leq x < 2, \\ 7x+5 & x \geq 2 \end{cases}
 \end{aligned}$$

Največji zaslužek bomo dobili kot  $p^* = p_3(6)$ .

$$\begin{aligned}
 p_3(6) &= \max \{ p_2(z) + r_3(6-z) \mid 0 \leq z \leq 6 \} \\
 &= \max \left( \begin{aligned} &\{-4z+29 \mid 0 \leq z < 1\} \\ &\cup \left\{ -z+36 \mid 1 \leq z < \frac{5}{4} \right\} \\ &\cup \left\{ 3z+31 \mid \frac{5}{4} \leq z < 2 \right\} \\ &\cup \{3z+34 \mid 2 \leq z \leq 5\} \\ &\cup \{7z+5 \mid 5 < z \leq 6\} \end{aligned} \right) \\
 &= 49
 \end{aligned}$$

Ker velja  $p^* = p_3(6) = p_2(5) + r_3(1) = r_1(4) + r_2(1) + r_3(1)$ , največji zaslužek 49000€ dosežemo, če vložimo 4000€ v prvo investicijo in po 1000€ v drugi dve investiciji.

#### Naloga 4.47.

- (a) Naj bo  $p_i$  najmanjša cena postavitve baznih postaj do  $i$ -te milje tako, da se v celoti pokrije interval  $[0, i]$ . Določimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned}
 p_{-1} &= p_0 = 0 \\
 p_i &= \min \{ a_{i-1} + p_{i-2}, a_i + p_{i-1} \} \quad (1 \leq i \leq n)
 \end{aligned}$$

Vrednosti  $p_i$  računamo naraščajoče po indeksu  $i$  ( $0 \leq i \leq n$ ). Najmanjšo ceno postavitve baznih postaj dobimo s  $p^* = p_n$ .

- (b) Za izračun vrednosti  $p_i$  za posamezen  $i$  potrebujemo konstantno mnogo časa. Ker ta izračun opravimo  $(n-1)$ -krat, je torej časovna zahtevnost ustreznega algoritma  $O(n)$ .



(c) Izračunajmo vrednosti  $p_i$  ( $1 \leq i \leq 10$ ).

$$\begin{aligned}
 p_1 &= \min\{\underline{4} + 0, 6 + 0\} &= 4 \\
 p_2 &= \min\{6 + 0, \underline{1} + 4\} &= 5 \\
 p_3 &= \min\{\underline{1} + 4, 10 + 5\} &= 5 \\
 p_4 &= \min\{\underline{10} + 5, 14 + 5\} &= 15 \\
 p_5 &= \min\{\underline{14} + 5, 21 + 15\} &= 19 \\
 p_6 &= \min\{21 + 15, \underline{15} + 19\} &= 34 \\
 p_7 &= \min\{\underline{15} + 19, 6 + 34\} &= 34 \\
 p_8 &= \min\{\underline{6} + 34, 10 + 34\} &= 40 \\
 p_9 &= \min\{10 + 34, \underline{3} + 40\} &= 43 \\
 p_{10} &= \min\{\underline{3} + 40, 2 + 43\} &= 43
 \end{aligned}$$

Cena postavitve baznih postaj je torej  $p^* = p_{10} = 43$ . Poglejmo, kam moramo postaviti bazne postaje.

$$\begin{array}{ll}
 p_{10} = a_9 + p_8 & \text{postaja na 9} \\
 p_8 = a_7 + p_6 & \text{postaja na 7} \\
 p_6 = a_6 + p_5 & \text{postaja na 6} \\
 p_5 = a_4 + p_3 & \text{postaja na 4} \\
 p_3 = a_2 + p_1 & \text{postaja na 2} \\
 p_1 = a_0 & \text{postaja na 0}
 \end{array}$$

(d) Naj bo  $q_i$  najmanjša cena postavitve baznih postaj (pri čemer dovolimo tudi večje postaje) do  $i$ -te milje tako, da se v celoti pokrije interval  $[0, i]$ . Določimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned}
 b_{-1} &= \infty \\
 q_{-3} = q_{-2} = q_{-1} &= q_0 = 0 \\
 q_i &= \min\{b_{i-2} + \min\{q_{i-4}, q_{i-3}\}, \\
 &\quad b_{i-1} + \min\{q_{i-3}, q_{i-2}\}, \\
 &\quad b_i + \min\{q_{i-2}, q_{i-1}\}, \\
 &\quad a_{i-1} + q_{i-2}, \\
 &\quad a_i + q_{i-1}\} \quad (1 \leq i \leq n)
 \end{aligned}$$

Vrednosti  $q_i$  računamo naraščajoče po indeksu  $i$  ( $0 \leq i \leq n$ ). Najmanjšo ceno postavitve baznih postaj dobimo s  $q^* = q_n$ .

(e) Izračunajmo vrednosti  $q_i$  ( $1 \leq i \leq 10$ ).

$$\begin{aligned}
 q_1 &= \min\{\infty + 0, 10 + 0, 12 + 0, \underline{4 + 0}, 6 + 0\} &= 4 \\
 q_2 &= \min\{10 + 0, 12 + 0, \underline{3 + 0}, 6 + 0, 1 + 4\} &= 3 \\
 q_3 &= \min\{12 + 0, \underline{3 + 0}, 18 + 3, 1 + 4, 10 + 3\} &= 3 \\
 q_4 &= \min\{\underline{3 + 0}, 18 + 3, 24 + 3, 10 + 3, 14 + 3\} &= 3 \\
 q_5 &= \min\{18 + 3, 24 + 3, 25 + 3, \underline{14 + 3}, 21 + 3\} &= 17 \\
 q_6 &= \min\{24 + 3, 25 + 3, \underline{20 + 3}, 21 + 3, 15 + 17\} &= 23 \\
 q_7 &= \min\{25 + 3, \underline{20 + 3}, 11 + 17, 15 + 17, 6 + 23\} &= 23 \\
 q_8 &= \min\{\underline{20 + 3}, 11 + 17, 16 + 23, 6 + 23, 10 + 23\} &= 23 \\
 q_9 &= \min\{11 + 17, 16 + 23, 7 + 23, 10 + 23, \underline{3 + 23}\} &= 26 \\
 q_{10} &= \min\{16 + 23, 7 + 23, 4 + 23, \underline{3 + 23}, 2 + 26\} &= 26
 \end{aligned}$$

Cena postavitve baznih postaj je torej  $q^* = q_{10} = 26$ . Poglejmo, kam moramo postaviti bazne postaje.

$$\begin{aligned}
 q_{10} &= a_9 + q_8 && \text{manjša postaja na 9} \\
 q_8 &= b_6 + q_4 && \text{večja postaja na 6} \\
 q_4 &= b_2 + q_0 && \text{večja postaja na 2}
 \end{aligned}$$

#### Naloga 4.48.

(a) Zapišimo začetni pogoj in rekurzivne enačbe.

$$\begin{aligned}
 c_0 &= 0 \\
 c_i &= c_{i-1} + p_i \quad (1 \leq i \leq n)
 \end{aligned}$$

Vrednosti  $c_i$  ( $0 \leq i \leq n$ ) računamo naraščajoče po indeksu  $i$ , za izračun pa porabimo  $O(n)$  časa.

(b) Naj bo  $x_{ij}$  ( $1 \leq i \leq j \leq n$ ) cena postavitve razdelilnikov, ki razdelijo vodo za delavnice od  $i$  do  $j$ . Zapišimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned}
 x_{ii} &= 0 && (1 \leq i \leq n) \\
 x_{ij} &= c_j - c_{i-1} + \min\{x_{ik} + x_{k+1,j} \mid i \leq k < j\} && (1 \leq i < j \leq n)
 \end{aligned}$$

Vrednosti  $x_{ij}$  ( $1 \leq i \leq j \leq n$ ) računamo naraščajoče po razliki  $j - i$ , nato pa še naraščajoče po indeksu  $i$ . Optimalno rešitev dobimo kot  $x^* = x_{1n}$ . Časovna zahtevnost izračuna je  $O(n^3)$ .

(c) Najprej izračunajmo vrednosti  $c_i$  ( $1 \leq i \leq 6$ ).

$$\begin{aligned}c_1 &= 0 + 4 = 4 \\c_2 &= 4 + 19 = 23 \\c_3 &= 23 + 17 = 40 \\c_4 &= 40 + 7 = 47 \\c_5 &= 47 + 5 = 52 \\c_6 &= 52 + 9 = 61\end{aligned}$$

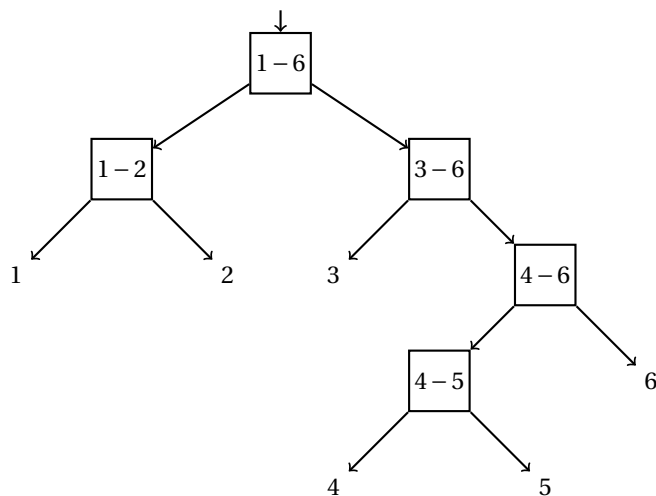
Sedaj izračunajmo še vrednosti  $x_{ij}$  ( $1 \leq i < j \leq 6$ ).

$$\begin{aligned}x_{12} &= 23 - 0 + 0 + 0 &= 23 \\x_{23} &= 40 - 4 + 0 + 0 &= 36 \\x_{34} &= 47 - 23 + 0 + 0 &= 24 \\x_{45} &= 52 - 40 + 0 + 0 &= 12 \\x_{56} &= 61 - 47 + 0 + 0 &= 14 \\x_{13} &= 40 - 0 + \min\{0 + 36, \underline{23} + 0\} &= 63 \\x_{24} &= 47 - 4 + \min\{\underline{0} + 24, 36 + 0\} &= 67 \\x_{35} &= 52 - 23 + \min\{\underline{0} + 12, 24 + 0\} &= 41 \\x_{46} &= 61 - 40 + \min\{0 + 14, \underline{12} + 0\} &= 33 \\x_{14} &= 47 - 0 + \min\{0 + 67, \underline{23} + 24, 63 + 0\} &= 94 \\x_{25} &= 52 - 4 + \min\{\underline{0} + 41, 36 + 12, 67 + 0\} &= 89 \\x_{36} &= 61 - 23 + \min\{\underline{0} + 33, 24 + 14, 41 + 0\} &= 71 \\x_{15} &= 52 - 0 + \min\{0 + 89, \underline{23} + 41, 63 + 12, 94 + 0\} &= 116 \\x_{26} &= 61 - 4 + \min\{0 + 71, \underline{36} + 33, 67 + 14, 89 + 0\} &= 126 \\x_{16} &= 61 - 0 + \min\{0 + 126, \underline{23} + 71, 63 + 33, 94 + 14, 116 + 0\} &= 155\end{aligned}$$

Cena postavitve razdelilnikov je torej  $x^* = x_{16} = 155$ . Poglejmo, kako naj jih postavimo.

$$\begin{array}{ll}x_{16} = c_6 - c_0 + x_{12} + x_{36} & \text{delimo na 1, 2 in 3 do 6} \\x_{12} = c_2 - c_0 + x_{11} + x_{22} & \text{delimo na 1 in 2} \\x_{36} = c_6 - c_2 + x_{33} + x_{46} & \text{delimo na 3 in 4 do 6} \\x_{46} = c_6 - c_3 + x_{45} + x_{66} & \text{delimo na 4, 5 in 6} \\x_{45} = c_5 - c_3 + x_{44} + x_{55} & \text{delimo na 4 in 5}\end{array}$$

Postavitev je prikazana na sliki 81.



Slika 81: Postavitev razdelilnikov za nalogo 4.48(c).

**Naloga 4.49.**

- (a) Naj bo  $p_{ij}$  največja vsota, ki jo lahko dosežemo z zaporedjem skokov od  $A_{11}$  do  $A_{ij}$ . Določimo začetni pogoj in rekurzivne enačbe.

$$\begin{aligned}
 p_{11} &= A_{11} \\
 p_{1j} &= A_{1j} + \max\{p_{1\ell} \mid 1 \leq \ell \leq j-1\} & (2 \leq j \leq n) \\
 p_{i1} &= A_{i1} + \max\{p_{\ell 1} \mid 1 \leq \ell \leq i-1\} & (2 \leq i \leq m) \\
 p_{ij} &= A_{ij} + \max(\{p_{i\ell} \mid 1 \leq \ell \leq j-1\} \cup \{p_{\ell j} \mid 1 \leq \ell \leq i-1\}) & (2 \leq i \leq m, \\
 & & 2 \leq j \leq n)
 \end{aligned}$$

Vrednosti  $p_{ij}$  računamo v leksikografskem vrstnem redu indeksov (npr. najprej naraščajoče po  $i$ , nato pa naraščajoče po  $j$ ). Maksimalno vsoto obiskanih mest dobimo kot  $p^* = p_{mn}$ . Rekurzivne enačbe lahko rešimo v času  $O(mn(m+n))$ .

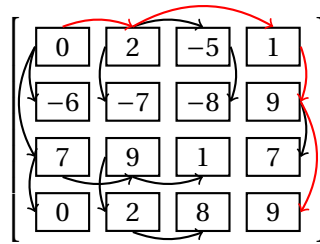
- (b) Matrika vrednosti  $(p_{ij})_{i,j=1}^4$  je prikazana na sliki 82, pri čemer je pri vsaki vrednosti puščica od tiste vrednosti, ki je bila uporabljena pri njenem izračunu. Z rdečo barvo je prikazano zaporedje  $(1, 1), (1, 2), (1, 4), (2, 4), (4, 4)$  z največjo vsoto  $p^* = 9$ .

**Naloga 4.50.**

- (a) Naj bo  $d_i$  ( $i = 1, 2$ ) optimalni dobiček pri vlaganju v  $i$ -to podjetje. Potem velja

$$d_i = \begin{cases} \max\{(n_i + k_i x)(n_3 + k_3(m-x)) \mid a_i \leq x \leq m - a_3\}, & \text{če } a_i \leq m - a_3, \text{ in} \\ 0 & \text{sicer.} \end{cases}$$

Optimalni dobiček potem dobimo kot  $d^* = \max\{d_1, d_2\}$ .



Slika 82: Reševanje in optimalna rešitev za nalogo 4.49(b).

- (b) Naj bo  $q_i(x)$  ( $i = 1, 2$ ) kvadratni polinom, ki nastopa v zgornjem izrazu. Da izračunamo  $d_1$  in  $d_2$ , bomo poiskali maksimum polinomov  $q_1$  in  $q_2$  v ustreznih mejah.

Vrednost  $d_1$  je maksimalna vrednost izraza

$$q_1(x) = (8 + 4x)(4 + 10 - x) = -4x^2 + 48x + 112$$

za  $x \in [4, 7]$ . Ker ima polinom negativen vodilni člen, je njegov globalni maksimum tam, kjer ima njegov odvod ničlo.

$$\begin{aligned} q_1'(x) &= -8x + 48 = 0 \\ x &= 6 \end{aligned}$$

Ker maksimum leži na iskanem intervalu, velja  $d_1 = q_1(6) = 256$ .

Vrednost  $d_2$  je maksimalna vrednost izraza

$$q_2(x) = (12 + 2x)(4 + 10 - x) = -2x^2 + 16x + 168$$

za  $x \in [5, 7]$ . Ker ima polinom negativen vodilni člen, je njegov globalni maksimum tam, kjer ima njegov odvod ničlo.

$$\begin{aligned} q_2'(x) &= -4x + 16 = 0 \\ x &= 4 \end{aligned}$$

Ker maksimum leži levo od iskanega intervala, leži maksimum na njegovem levem robu, torej velja  $d_2 = q_2(5) = 198$ .

Optimalni dobiček je tako  $d^* = \max\{256, 198\} = 256$  in ga dosežemo tako, da vložimo 6 v prvo podjetje in 4 v marketing.

**Naloga 4.51.**

- (a) Naj bo  $p_i$  najmanjša cena postavitve stebrov do  $i$ -tega mesta, če zadnji steber stoji na  $i$ -tem mestu (pri čemer upoštevamo, da moramo “stebra” postaviti tudi na začetno in končno točko). Določimo začetne pogoje in rekurzivne enačbe.

$$p_1 = c_1$$

$$p_i = c_i + \min\{p_j \mid 1 \leq j \leq i-1, x_i - x_j \leq \min\{r_i, r_j\}\} \quad (2 \leq i \leq n)$$

Vrednosti  $p_i$  računamo naraščajoče po indeksu  $i$  ( $1 \leq i \leq n$ ). Najmanjšo ceno postavitve stebrov dobimo s  $p^* = p_n$ .

- (b) Zapišimo psevdokodo algoritma.

```

function STEBRI( $(x_i)_{i=1}^n, (c_i)_{i=1}^n, (r_i)_{i=1}^n$ )
     $p_1 \leftarrow c_1$ 
    for  $i = 2, 3, \dots, n$  do
         $p_i \leftarrow \infty$ 
         $j \leftarrow i - 1$ 
        while  $j > 0 \wedge x_i - x_j \leq r_i$  do
            if  $p_j < p_i \wedge x_i - x_j \leq r_j$  then
                 $p_i \leftarrow p_j$ 
            end if
             $j \leftarrow j - 1$ 
        end while
         $p_i \leftarrow p_i + c_i$ 
    end for
    return  $p_n$ 
end function

```

Za izračun vrednosti  $p_i$  za posamezen  $i$  potrebujemo  $O(i)$  časa. Ker ta izračun opravimo za  $i = 2, 3, \dots, n$ , je torej časovna zahtevnost ustreznega algoritma  $O(n^2)$ .

- (c) Izračunajmo vrednosti  $p_i$  ( $2 \leq i \leq 12$ ).

$$p_2 = 8 + 0 = 8$$

$$p_3 = 9 + \min\{0, 8\} = 9$$

$$p_4 = 32 + \min\{0, 8, 9\} = 32$$

$$p_5 = 14 + 32 = 46$$

$$p_6 = 18 + \min\{32, 46\} = 50$$

$$p_7 = 9 + \min\{46, 50\} = 55$$

$$p_8 = 27 + 55 = 82$$

$$p_9 = 47 + 82 = 129$$

$$p_{10} = 16 + \min\{82, 129\} = 98$$

$$p_{11} = 22 + \min\{129, 98\} = 120$$

$$p_{12} = 0 + \min\{98, 120\} = 98$$

Najmanjša cena postavitve stebrov je torej  $p^* = p_{12} = 98$ . Poglejmo, kam moramo postaviti stebre.

$p_{12} = c_{12} + p_{10}$	steber na $x_{10} = 857$
$p_{10} = c_{10} + p_8$	steber na $x_8 = 688$
$p_8 = c_8 + p_7$	steber na $x_7 = 509$
$p_7 = c_7 + p_5$	steber na $x_5 = 258$
$p_5 = c_5 + p_4$	steber na $x_4 = 119$
$p_4 = c_4 + p_1$	

#### Naloga 4.52.

- (a) Naj bodo  $x_i, y_i$  in  $z_i$  ( $1 \leq i \leq n$ ) največje vsote, ki jih dosežemo, če so najbolj desni elementi v  $S$  v prvi, drugi oziroma obeh vrsticah  $i$ -tega stolpca. Določimo začetne pogoje in rekurzivne enačbe.

$$\begin{aligned} x_0 &= y_0 = z_0 = 0 \\ x_i &= a_i + \max\{0, x_{i-1}, z_{i-1}\} & (1 \leq i \leq n) \\ y_i &= b_i + \max\{0, y_{i-1}, z_{i-1}\} & (1 \leq i \leq n) \\ z_i &= a_i + b_i + \max\{0, x_{i-1}, y_{i-1}, z_{i-1}\} & (1 \leq i \leq n) \end{aligned}$$

Vrednosti  $x_i, y_i, z_i$  računamo naraščajoče po indeksu  $i$  ( $1 \leq i \leq n$ ). Največjo vsoto množice  $S$  dobimo kot  $z^* = \max\{x_i, y_i, z_i \mid (0 \leq i \leq n)\}$ .

- (b) Algoritem izračuna  $O(n)$  spremenljivk, za vsako pa porabi konstantno časa. Časovna zahtevnost algoritma je torej  $O(n)$ .
- (c) Izračunajmo vrednosti  $x_i, y_i$  in  $z_i$  ( $1 \leq i \leq 5$ ).

$x_1$		$= 4$
$y_1$		$= -1$
$z_1 = 4 + (-1)$		$= 3$
$x_2 =$	$4 + \max\{0, \underline{4}, 3\}$	$= 8$
$y_2 =$	$-2 + \max\{0, -1, \underline{3}\}$	$= 1$
$z_2 = 4 + (-2) + \max\{0, \underline{4}, -1, 3\}$		$= 6$
$x_3 =$	$-3 + \max\{0, \underline{8}, 6\}$	$= 5$
$y_3 =$	$4 + \max\{0, 1, \underline{6}\}$	$= 10$
$z_3 =$	$-3 + 4 + \max\{0, \underline{8}, 1, 6\}$	$= 9$
$x_4 =$	$4 + \max\{0, 5, \underline{9}\}$	$= 13$

$$\begin{aligned}
y_4 &= -4 + \max\{0, \underline{10}, 9\} = 6 \\
z_4 &= 4 + (-4) + \max\{0, 5, \underline{10}, 9\} = 10 \\
x_5 &= -3 + \max\{0, \underline{13}, 10\} = 10 \\
y_5 &= 2 + \max\{0, 6, \underline{10}\} = 12 \\
z_5 &= -3 + 2 + \max\{0, \underline{13}, 6, 10\} = 12
\end{aligned}$$

Največja vsota množice  $S$  je torej  $z^* = \max\{x_i, y_i, z_i \mid (0 \leq i \leq 5)\} = 13$ . Poglejmo, kateri elementi so v množici  $S$ .

$$\begin{aligned}
z^* &= x_4 = a_4 + z_3 & a_4 &= 4 \\
z_3 &= a_3 + b_3 + x_2 & a_3 &= -3, b_3 = 4 \\
x_2 &= a_2 + x_1 & a_2 &= 4 \\
x_1 &= a_1 & a_1 &= 4
\end{aligned}$$

Imamo torej sledečo optimalno rešitev:

$$\begin{bmatrix} \boxed{4} & \boxed{4} & \boxed{-3} & \boxed{4} & -3 \\ -1 & -2 & \boxed{4} & -4 & 2 \end{bmatrix}$$

#### Naloga 4.53.

- (a) Naj bodo  $x_{ij}$ ,  $y_{ij}$  in  $z_{ij}$  ( $1 \leq i \leq j \leq n$ ) najmanjša začetna sredstva, potrebna za zasedbo mest od  $i$ -tega do  $j$ -tega, sredstva, ki jih imamo po tem na voljo, če začnemo z  $x_{ij}$  sredstvi, ter indeks mesta, ki je bilo pri tem nazadnje zasedeno. Zapišimo psevdokodo algoritma, iz katerega bodo razvidni tudi začetni pogoji in rekurzivne enačbe.

```

function ZASEDBA( $((s_i)_{i=1}^n, (t_i)_{i=1}^n)$ )
  for  $i = 1, \dots, n$  do                                     začetni pogoji
     $x_{ii} \leftarrow s_i$ 
     $y_{ii} \leftarrow t_i$ 
  end for
  for  $h = 1, \dots, n-1$  do
    for  $i = 1, \dots, n-h$  do
       $j \leftarrow i+h$                                        zasedamo mesta od  $i$  do  $j$ 
       $x_{ij}, y_{ij}, z_{ij} \leftarrow \min\{(x_{i+1,j} + \max\{0, s_i - y_{i+1,j}\}, t_i + \max\{0, y_{i+1,j} - s_i\}, i),$ 
                                                                 $(x_{i,j-1} + \max\{0, s_j - y_{i,j-1}\}, t_j + \max\{0, y_{i,j-1} - s_j\}, j)\}$ 
    end for
  end for
   $\ell \leftarrow$  prazen seznam
   $i, j \leftarrow 1, n$ 
  while  $i < j$  do                                       rekonstruiramo zaporedje zasedanja mest
     $h \leftarrow z_{ij}$ 
     $\ell.append(h)$ 
    if  $h = i$  then
       $i \leftarrow i+1$ 

```



```

else
     $j \leftarrow j - 1$ 
end if
end while
 $\ell.append(i)$ 
 $\ell.reverse()$ 
return  $(x_{1n}, \ell)$ 
end function

```

Da algoritem deluje pravilno, bo za vse  $i, j$  ( $1 \leq i \leq j \leq n$ ) moralo veljati

$$y_{ij} - x_{ij} = \sum_{h=i}^j (t_h - s_h).$$

Opazimo, da rekurzivne enačbe v zgornjem algoritmu zadoščajo temu pogoju.

- (b) Algoritem izračuna  $O(n^2)$  spremenljivk, za vsako pa porabi konstantno časa. Pri rekonstrukciji rešitve algoritem naredi še  $O(n)$  korakov, pri čemer za vsakega spet porabi konstantno časa. Skupna časovna zahtevnost je torej  $O(n^2)$ .
- (c) Izračunajmo vrednosti  $x_{ij}$  in  $y_{ij}$  ( $1 \leq i < j \leq 5$ ).

$x_{12} = \min\{10 + 8 - 4, \underline{8 + 10 - 6}\}$	$= 12$	$y_{12}$	$= 4$
$x_{23} = \min\{\underline{9 + 10 - 7}, 10 + 9 - 4\}$	$= 12$	$y_{23}$	$= 4$
$x_{34} = \min\{\underline{11}, 9 + 11 - 7\}$	$= 11$	$y_{34} = 7 + 12 - 9$	$= 10$
$x_{45} = \min\{9 + 11 - 3, \underline{11}\}$	$= 11$	$y_{34} = 3 + 12 - 9$	$= 6$
$x_{13} = \min\{\underline{12 + 8 - 4}, 12 + 9 - 4\}$	$= 16$	$y_{13}$	$= 6$
$x_{24} = \min\{\underline{11}, 12 + 11 - 4\}$	$= 11$	$y_{24}$	$= 4$
$x_{35} = \min\{11 + 9 - 6, \underline{11}\}$	$= 11$	$y_{35} = 3 + 10 - 9$	$= 4$
$x_{14} = \min\{\underline{11 + 8 - 4}, 16 + 11 - 6\}$	$= 15$	$y_{14}$	$= 6$
$x_{25} = \min\{11 + 10 - 4, \underline{11 + 9 - 4}\}$	$= 16$	$y_{25}$	$= 3$
$x_{15} = \min\{16 + 8 - 3, \underline{15 + 9 - 6}\}$	$= 18$	$y_{15}$	$= 3$

Za zasedbo mest torej potrebujemo začetna sredstva vsaj  $x_{15} = 18$ . Rekonstruirajmo vrstni red zasedbe mest – dobili ga bomo v obratnem vrstnem redu.

$x_{15} = x_{14} + s_5 - y_{14}$	nazadnje zasedemo mesto 5
$x_{14} = x_{24} + s_1 - y_{24}$	pred tem zasedemo mesto 1
$x_{24} = x_{34}$	pred tem zasedemo mesto 2
$x_{34} = x_{44}$	pred tem zasedemo mesto 3
	začnemo z zasedbo mesta 4



## 2.5 Algoritmi na grafih

### Naloga 5.1.

Graf  $G$  bomo predstavili kot matriko  $G.A$ , indeksirano z vozlišči, kar bomo implementirali s slovarjem slovarjev. Napišimo ustrezne metode.

```
function INIT( $G$ )                                     časovna zahtevnost:  $O(1)$ 
     $G.A \leftarrow$  prazen slovar
end function

function ADDVERTEX( $G, u$ )                             časovna zahtevnost:  $O(n)$ 
     $G.A[u] \leftarrow$  slovar z vrednostjo 0 za vsak ključ slovarja  $G.A$ 
    for  $v \in G.A$  do
         $G.A[v][u] \leftarrow 0$ 
    end for
end function

function ADDEDGE( $G, u, v$ )                             časovna zahtevnost:  $O(1)$ 
     $G.A[u][v] \leftarrow 1$ 
     $G.A[v][u] \leftarrow 1$ 
end function

function DELEDGE( $G, u, v$ )                             časovna zahtevnost:  $O(1)$ 
     $G.A[u][v] \leftarrow 0$ 
     $G.A[v][u] \leftarrow 0$ 
end function

function DELVERTEX( $G, u$ )                             časovna zahtevnost:  $O(n)$ 
    izbriši  $G.A[u]$ 
    for  $v \in G.A$  do
        izbriši  $G.A[v][u]$ 
    end for
end function

function ADJ( $G, u$ )                                    časovna zahtevnost:  $O(n)$ 
    return  $\{v \in G.A[u] \mid G.A[u][v] = 1\}$ 
end function
```

V podanih časovnih zahtevnostih  $n$  predstavlja število vozlišč v grafu. Prostorska zahtevnost podatkovne strukture je  $O(n^2)$ .

Opomnimo, da lahko preverjanje, ali sta dve vozlišči sosedni (torej če velja  $v \in \text{ADJ}(G, u)$  za dani vozlišči  $u, v$ ), opravimo v času  $O(1)$ .

### Naloga 5.2.

Graf  $G$  bomo predstavili kot slovar  $G.E$ , ki vsakemu vozlišču priredi množico njegovih sosedov. Napišimo ustrezne metode.

```
function INIT( $G$ )                                     časovna zahtevnost:  $O(1)$ 
     $G.E \leftarrow$  prazen slovar
end function

function ADDVERTEX( $G, u$ )                             časovna zahtevnost:  $O(1)$ 
     $G.E[u] \leftarrow$  prazna množica
end function

function ADDEDGE( $G, u, v$ )                             časovna zahtevnost:  $O(1)$ 
     $G.E[u].\text{add}(v)$ 
```

```

    G.E[v].add(u)
end function
function DELEDGE(G, u, v)                                časovna zahtevnost:  $O(1)$ 
    G.E[u].remove(v)
    G.E[v].remove(u)
end function
function DELVERTEX(G, u)                                  časovna zahtevnost:  $O(n)$ 
    izbriši G.E[u]
    for  $v \in G.E$  do
        G.E[v].remove(u)
    end for
end function
function ADJ(G, u)                                         časovna zahtevnost:  $O(d_G(u))$ 
    return G.E[u]
end function

```

V podanih časovnih zahtevnostih  $n$  predstavlja število vozlišč v grafu,  $d_G(u)$  pa stopnjo vozlišča  $u$  v grafu  $G$ . Prostorska zahtevnost podatkovne struktura je  $O(m + n)$ , kjer  $m$  predstavlja število povezav v grafu.

Opomnimo, da lahko preverjanje, ali sta dve vozlišči sosedni (torej če velja  $v \in \text{ADJ}(G, u)$  za dani vozlišči  $u, v$ ), opravimo v času  $O(1)$ .

### Naloga 5.3.

Strukturo iz naloge 5.1 spremenimo tako, da lahko hrani poljubno dvojiško matriko (torej ne nujno simetrične). To naredimo tako, da popravimo metodi ADDEDGE in DELEDGE, in sicer tako, da spreminjamo samo vrednost  $G.A[u][v]$ . Časovne zahtevnosti metod in prostorska zahtevnost strukture se pri tem ne spremenijo.

Strukturo iz naloge 5.1 spremenimo tako, da relacija, določena s slovarjem  $G.E$ , ni nujno simetrična. To naredimo tako, da popravimo metodi ADDEDGE in DELEDGE, in sicer tako, da spreminjamo samo množico  $G.E[u]$ . Tudi tukaj se časovne in prostorske zahtevnosti ne spremenijo.

V digrafih lahko ločimo sosednost po vhodnih in izhodnih povezavah. V ta namen bi lahko za vsako imeli svojo metodo. Pri matrični predstavitvi lahko samo zamenjamo indeksa v metodi ADJ, če pa želimo ohraniti časovno zahtevnost pri predstavitvi s sezname sosedov, pa bi morali poleg slovarja za izhodne povezave hraniti še slovar za vhodne povezave.

### Naloga 5.4.

Uporabili bomo predstavitev grafa s seznamom sosedov (glej nalogo 5.2).

```

function TRIKOTNIK(G)
    for  $u \in G.E$  do
        for  $v \in \text{ADJ}(G, u)$  do
            for  $w \in \text{ADJ}(G, v)$  do
                if  $u \in \text{ADJ}(G, w)$  then
                    return TRUE
                end if
            end for
        end for
    end for
end function

```

```

        end if
    end for
end for
end for
return FALSE
end function

```

Da pregledamo vse pare  $(u, v)$ , porabimo  $O(n)$  korakov, da obiščemo vsa vozlišča, in še  $O(m)$  korakov, da obiščemo vse povezave. Tukaj je  $n$  število vozlišč in  $m$  število povezav grafa. Za vsak tak par pregledamo še sosedu  $w$  vozlišča  $v$  – teh je vsakič največ  $\Delta$  (največja stopnja grafa  $G$ ). Nazadnje še v času  $O(1)$  preverimo, ali sta vozlišči  $u$  in  $w$  sosedni. Skupna časovna zahtevnost algoritma je torej  $O(n + m\Delta)$ .

### Naloga 5.5.

Zapišimo algoritem, ki bo vrnil zvezdo digrafa  $D$ , če ta obstaja, in NULL sicer.

```

function ZVEZDA( $D = (V, E)$ )
     $z \leftarrow \text{NULL}$                                 začnemo brez kandidata
    for  $u \in V$  do
         $S \leftarrow \text{ADJ}(D, u)$                     sosedu vozlišča  $u$ 
        if  $|S| == |V| - 1 \wedge u \notin S$  then          ali je  $u$  kandidat?
            if  $z \neq \text{NULL}$  then
                return NULL                        če smo že imeli kandidata, nimamo zvezde
            end if
             $z \leftarrow u$                             nastavimo kandidata
        else if  $|S| \neq 0$  then                        nekandidati ne smejo imeti povezav
            return NULL
        end if
    end for
    return  $z$ 
end function

```

Če je digraf  $D$  predstavljen kot seznam sosedov (glej nalogi 5.2 in 5.3), pri čemer množice sosedov hranijo svojo velikost, bo algoritem tekel v času  $O(n)$ , kjer je  $n$  število vozlišč.

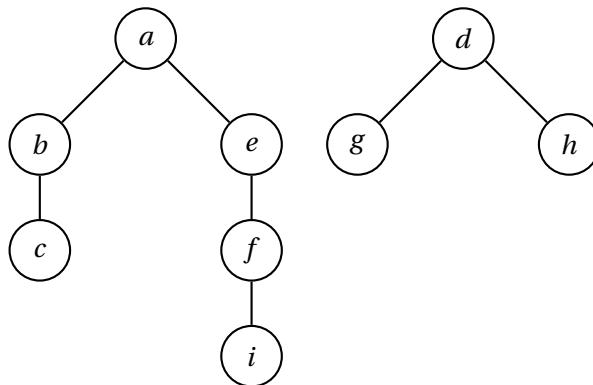
### Naloga 5.6.

Sledili bomo naslednjemu algoritmu.

```

function BFS( $G = (V, E), S, \text{VISIT}$ )
     $\text{visited} \leftarrow$  slovar z vrednostjo FALSE za vsak  $v \in V$ 
     $\text{pred} \leftarrow$  slovar z vrednostjo NULL za vsak  $v \in V$ 
    for  $s \in S$  do
        if  $\neg \text{visited}[s]$  then
             $\text{visited}[s] \leftarrow \text{TRUE}$ 
             $Q \leftarrow [s]$ 
            while  $\neg Q.\text{isEmpty}()$  do
                 $u \leftarrow Q.\text{pop}()$ 
                 $\text{VISIT}(u, \text{pred}[u])$ 
                for  $v \in \text{ADJ}(G, u)$  do

```



Slika 83: Gozd iskanja v širino za nalogo 5.6.

```

if  $\neg \text{visited}[v]$  then
     $\text{visited}[v] \leftarrow \text{TRUE}$ 
     $\text{pred}[v] \leftarrow u$ 
     $Q.\text{append}(v)$ 
end if
end for
end while
end if
end for
return  $\text{pred}$ 
end function
  
```

Časovna zahtevnost preiskovanja je  $O(m + n)$  (kjer je  $n$  število vozlišč in  $m$  število povezav grafa), poleg tega pa algoritem opravi še  $O(n)$  klicev funkcije VISIT.

Potek zgornjega algoritma na grafu s slike 13 (pri čemer za množico začetnih vozlišč  $S$  vzamemo množico vseh vozlišč  $V$ , za VISIT pa vzamemo NOOP, torej ne naredimo ničesar), pri čemer sledimo abecednemu vrstnemu redu vozlišč, je prikazan v tabeli 18. Gozd iskanja v širino je prikazan na sliki 83, iz katere je razvidno, da so drevesne povezave

$(a, b), (b, c), (a, e), (e, f), (f, i), (d, g), (d, h)$ .

### Naloga 5.7.

Premier grafa je največja razdalja med dvema njegovima vozliščema. Poiskali ga bomo tako, da bomo opravili iskanje v širino iz vsakega vozlišča grafa (pomagali si bomo torej z algoritmom BFS iz naloge 5.6) ter vrnili največjo globino dobljenega drevesa.

```

function PREMIER( $G = (V, E)$ )
     $D \leftarrow 0$ 
     $d \leftarrow$  prazen slovar
    function VISIT( $u, v$ )
        if  $v = \text{NULL}$  then
             $d[u] \leftarrow 0$ 
  
```

$s$	$u$	$v$	$Q$	množica označenih vozlišč
$a$			$[a]$	$\{a\}$
$a$	$a$	$b$	$[b]$	$\{a, b\}$
$a$	$a$	$e$	$[b, e]$	$\{a, b, e\}$
$a$	$b$	$c$	$[e, c]$	$\{a, b, c, e\}$
$a$	$b$	$e$		$\{a, b, c, e\}$
$a$	$e$	$a$	$[c]$	$\{a, b, c, e\}$
$a$	$e$	$b$		$\{a, b, c, e\}$
$a$	$e$	$f$	$[c, f]$	$\{a, b, c, e, f\}$
$a$	$c$	$b$	$[f]$	$\{a, b, c, e, f\}$
$a$	$c$	$f$		$\{a, b, c, e, f\}$
$a$	$f$	$c$		$\{a, b, c, e, f\}$
$a$	$f$	$e$		$\{a, b, c, e, f\}$
$a$	$f$	$i$	$[i]$	$\{a, b, c, e, f, i\}$
$a$	$i$	$f$	$[\ ]$	$\{a, b, c, e, f, i\}$
$b$				$\{a, b, c, e, f, i\}$
$c$				$\{a, b, c, e, f, i\}$
$d$			$[d]$	$\{a, b, c, d, e, f, i\}$
$d$	$d$	$g$	$[g]$	$\{a, b, c, d, e, f, g, i\}$
$d$	$d$	$h$	$[g, h]$	$\{a, b, c, d, e, f, g, h, i\}$
$d$	$g$	$d$	$[h]$	$\{a, b, c, d, e, f, g, h, i\}$
$d$	$g$	$h$		$\{a, b, c, d, e, f, g, h, i\}$
$d$	$h$	$d$		$\{a, b, c, d, e, f, g, h, i\}$
$d$	$h$	$g$	$[\ ]$	$\{a, b, c, d, e, f, g, h, i\}$
$e$				$\{a, b, c, d, e, f, g, h, i\}$
$f$				$\{a, b, c, d, e, f, g, h, i\}$
$g$				$\{a, b, c, d, e, f, g, h, i\}$
$h$				$\{a, b, c, d, e, f, g, h, i\}$
$i$				$\{a, b, c, d, e, f, g, h, i\}$

Tabela 18: Potek izvajanja algoritma za nalogo 5.6.

```

    else
         $d[u] \leftarrow d[v] + 1$ 
    end if
end function
for  $v \in V$  do
    for  $u \in V$  do
         $d[u] \leftarrow \infty$ 
    end for
    BFS( $G, \{v\}, \text{VISIT}$ )
     $D \leftarrow \max\{D, \max\{d[u] \mid u \in V\}\}$ 
end for
return  $D$ 
end function

```

Časovna zahtevnost algoritma je  $O(n(m + n))$ , kjer je  $n$  število vozlišč in  $m$  število povezav podanega grafa.

### Naloga 5.8.

Sledili bomo naslednjemu algoritmu. Predpostavili bomo, da imamo na voljo podatkovno strukturo za prednostno vrsto, ki za vsak element hrani njegovo prioriteto (to lahko tudi spreminjamo). Podatkovna struktura ima metodo `pop()`, ki vrne in odstrani element z najmanjšo prioriteto skupaj z njegovo prioriteto. Tukaj bodo elementi vozlišča grafa, prioritete pa dolžine najkrajših najdenih poti od začetnega vozlišča.

```

function DIJKSTRA( $G = (V, E), s \in V, L: E \rightarrow \mathbb{R}_+$ )
     $d \leftarrow$  slovar z vrednostjo  $\infty$  za vsako vozlišče  $v \in V$ 
     $pred \leftarrow$  slovar z vrednostjo NULL za vsako vozlišče  $v \in V$ 
     $Q \leftarrow$  prednostna vrsta s prioriteto  $\infty$  za vsako vozlišče  $v \in V$  (na vrhu je vozlišče z najmanjšo prioriteto)
     $Q[s] \leftarrow 0$ 
    while  $\neg Q.\text{isEmpty}()$  do
         $u, d[u] \leftarrow Q.\text{pop}()$ 
        for  $v \in \text{ADJ}(G, u)$  do
            if  $v \in Q \wedge Q[v] > d[u] + L_{uv}$  then
                 $Q[v] \leftarrow d[u] + L_{uv}$ 
                 $pred[v] \leftarrow u$ 
            end if
        end for
    end while
    return ( $d, pred$ )
end function

```

Če za prednostno vrsto uporabimo običajen slovar, je časovna zahtevnost metode `pop()` linearna v številu elementov slovarja, spreminjanje vrednosti v slovarje pa se opravi v konstantnem času. Časovna zahtevnost zgornjega algoritma je tako  $O(n^2)$ , kjer je  $n$  število vozlišč v grafu.

Če pa za prednostno vrsto vzamemo kopico, sta časovni zahtevnosti metode `pop()` in spreminjanja vrednosti v kopici logaritemski v velikosti kopice. Časovna zahtevnost zgornjega algoritma je v tem primeru  $O((m + n) \log n)$ , kjer je  $m$  število



povezav v grafu.

Potek zgornjega algoritma na grafu s slike 14, pri čemer sledimo abecednemu vrstnemu redu vozlišč, je prikazan v tabeli 19. Drevo najkrajših poti je prikazano na sliki 84.

### Naloga 5.9.

Trditev je napačna, saj z odštevanem konstantnega števila vsaki povezavi na grafu nekatere poti skrajšamo za več kot druge. Razlog za to je, da bo konstanta odšteta na vsaki povezavi, torej bodo poti, ki vsebujejo več povezav, prejele večjo olajšavo. Da bo to jasno, si pogledjmo primer, za katerega bomo našli dve različni drevesi najkrajših poti pri utežeh  $\ell$  in  $\ell'$ . Za primer vzemimo graf s slike 85. Vidimo, da je na začetnem grafu optimalna pot  $a \rightarrow b \rightarrow c$ , a ker vsebuje dve povezavi, se po prištevanju konstante 3 celotna pot podaljša za 6, pot  $a \rightarrow c$  pa se podaljša le za 3 in tako postane optimalna.

### Naloga 5.10.

- (a) Algoritem BFS iz naloge 5.6 bomo uporabili na pomožnem grafu s povezavami, dolgimi največ  $L$ . Preiskovanje bomo začeli v vozlišču  $t$ , tako da bomo s sledenjem predhodnikom iz vozlišča  $s$  rekonstruirali iskano pot.

**function** OBSTAJAPOT( $G = (V, E), s, t, \ell : E \rightarrow \mathbb{R}_+, L$ )

$E' \leftarrow \{e \in E \mid \ell_e \leq L\}$

$G' \leftarrow (V, E')$

$pred \leftarrow \text{BFS}(G', \{t\}, \text{NOOP})$

**if**  $s \neq t \wedge pred[s] = \text{NULL}$  **then**

**return** NULL

**end if**

$pot \leftarrow [s]$

$u \leftarrow s$

**while**  $pred[u] \neq \text{NULL}$  **do**

$u \leftarrow pred[u]$

$pot.append(u)$

**end while**

**return**  $pot$

**end function**

Časovna zahtevnost algoritma je  $O(m + n)$ , kjer je  $n$  število vozlišč in  $m$  število povezav podanega grafa.

- (b) Uporabili bomo algoritem MINNAJDALJŠAPOVEZAVA. Gre za varianto algoritma DIJKSTRA iz naloge 5.8, ki namesto dolžine poti za vsako vozlišče hrani dolžino najdaljše povezave v poti do njega.

**function** MINNAJDALJŠAPOVEZAVA( $G = (V, E), s, t, \ell : E \rightarrow \mathbb{R}_+$ )

$Q \leftarrow$  prednostna vrsta s prioriteto  $\infty$  za vsako vozlišče  $v \in V$

$Q[s] \leftarrow 0$

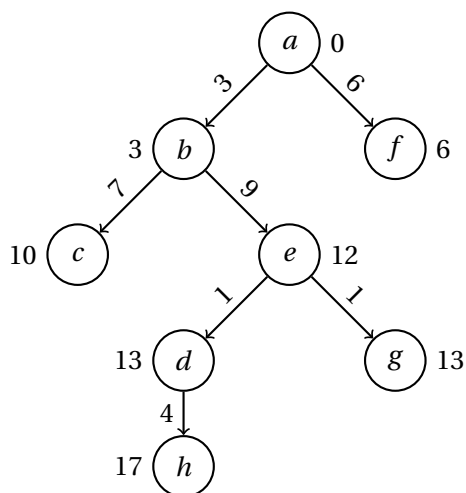
**while**  $\neg Q.isEmpty()$  **do**

$u, d \leftarrow Q.pop()$

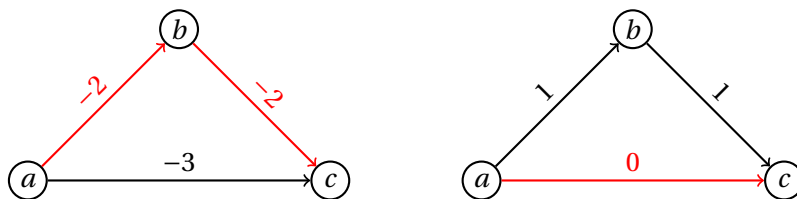
**if**  $u = t$  **then**

$u$	$v$	$Q$	razdalje in predhodniki
		$[(a, 0)]$	$[0, \infty, \infty, \infty, \infty, \infty, \infty, \infty]$
$a$	$b$	$[(b, 3)]$	$[0, 3_a, \infty, \infty, \infty, \infty, \infty, \infty]$
$a$	$f$	$[(b, 3), (f, 6)]$	$[0, 3_a, \infty, \infty, \infty, 6_a, \infty, \infty]$
$b$	$c$	$[(f, 6), (c, 10)]$	$[0, 3_a, 10_b, \infty, \infty, 6_a, \infty, \infty]$
$b$	$e$	$[(f, 6), (c, 10), (e, 12)]$	$[0, 3_a, 10_b, \infty, 12_b, 6_a, \infty, \infty]$
$f$	$g$	$[(c, 10), (e, 12), (g, 14)]$	$[0, 3_a, 10_b, \infty, 12_b, 6_a, 14_g, \infty]$
$c$	$d$	$[(e, 12), (g, 14), (d, 17)]$	$[0, 3_a, 10_b, 17_c, 12_b, 6_a, 14_g, \infty]$
$e$	$d$	$[(d, 13), (g, 14), (d, 17)]$	$[0, 3_a, 10_b, 13_e, 12_b, 6_a, 14_g, \infty]$
$e$	$g$	$[(g, 13), (d, 13), (g, 14), (d, 17)]$	$[0, 3_a, 10_b, 13_e, 12_b, 6_a, 13_e, \infty]$
$g$	$f$	$[(d, 13), (g, 14), (d, 17)]$	$[0, 3_a, 10_b, 13_e, 12_b, 6_a, 13_e, \infty]$
$d$	$b$	$[(g, 14), (d, 17)]$	$[0, 3_a, 10_b, 13_e, 12_b, 6_a, 13_e, \infty]$
$d$	$h$	$[(g, 14), (d, 17), (h, 17)]$	$[0, 3_a, 10_b, 13_e, 12_b, 6_a, 13_e, 17_d]$
$g$	$f$	$[(d, 17), (h, 17)]$	$[0, 3_a, 10_b, 13_e, 12_b, 6_a, 13_e, 17_d]$
$d$	$b$	$[(h, 17)]$	$[0, 3_a, 10_b, 13_e, 12_b, 6_a, 13_e, 17_d]$
$d$	$h$	$[(h, 17)]$	$[0, 3_a, 10_b, 13_e, 12_b, 6_a, 13_e, 17_d]$
$h$	$g$	$[\ ]$	$[0, 3_a, 10_b, 13_e, 12_b, 6_a, 13_e, 17_d]$

Tabela 19: Potek izvajanja algoritma za nalogo 5.8.



Slika 84: Drevo najkrajših poti za nalogo 5.8.



Slika 85: Protiprimer trditve za nalogo 5.9.

```

    return d
  end if
  for  $v \in \text{ADJ}(G, u)$  do
    if  $v \in Q \wedge Q[v] > \max\{d, \ell_{uv}\}$  then
       $Q[v] \leftarrow \max\{d, \ell_{uv}\}$ 
    end if
  end for
end while
return  $\infty$ 
end function

```

Tako kot Dijkstrov algoritem tudi zgornji algoritem teče v času  $O(m \log n)$  (kjer je  $n$  število vozlišč in  $m$  število povezav podanega grafa – ob predpostavki, da je graf povezan), če za prednostno vrsto uporabimo kopico.

### Naloga 5.11.

Zapišimo algoritem, ki poišče najkrajšo pot od  $s$  do  $t$ , pri čemer se preiskovanje začne v obeh koncih poti hkrati. Tako bomo v prednostno vrsto vsako vozlišče postavili dvakrat, preiskovanje pa bomo končali, ko bomo neko vozlišče drugič odstranili iz vrste. Iskana pot bo šla bodisi skozi to vozlišče, bodisi po povezavi med vozliščema, ki smo ju odstranili iz vrste po tem, ko sta bili doseženi iz različnih koncev iskane poti.

**function** DIJKSTRA2( $G = (V, E), s, t, L : E \rightarrow \mathbb{R}_+$ )

$Q \leftarrow$  prednostna vrsta s prioriteto  $\infty$  za  $(v, 0)$  in  $(v, 1)$  za vsako vozlišče  $v \in V$

$d_0 \leftarrow$  prazen slovar

$d_1 \leftarrow$  prazen slovar

$pred_0 \leftarrow$  slovar z vrednostjo NULL za vsak  $v \in V$

$pred_1 \leftarrow$  slovar z vrednostjo NULL za vsak  $v \in V$

$Q[s, 0] \leftarrow 0$

$Q[t, 1] \leftarrow 0$

$(u, i), r \leftarrow Q.\text{pop}()$

**while**  $(u, 1 - i) \in Q$  **do**

$d_i[u] \leftarrow r$

**for**  $v \in \text{ADJ}(G, u)$  **do**

**if**  $(v, i) \in Q \wedge Q[v, i] > d_i[u] + L_{uv}$  **then**

$Q[v, i] \leftarrow d_i[u] + L_{uv}$

$pred_i[v] \leftarrow u$

**end if**

**end for**

vzamemo prvo vozlišče iz vrste

nadaljujemo, če je vozlišče še v vrsti

zapomnimo si razdaljo

posodobimo razdalje sosedom

```

    if Q.isEmpty() then
        return ( $\infty$ , NULL)
    end if
    ( $u, i$ ),  $r \leftarrow Q.pop()$ 
end while
 $r, u \leftarrow \min(\{(r + d_{1-i}[u], u)\} \cup \{(d_i[v] + Q[v, 1 - i], v) \mid v \in d_i\})$ 
 $v \leftarrow u$ 
 $pot_0 \leftarrow [u]$ 
while  $pred_0[u] \neq \text{NULL}$  do
     $u \leftarrow pred_0[u]$ 
     $pot_0.append(u)$ 
end while
 $pot_0.reverse()$ 
 $pot_1 \leftarrow []$ 
while  $pred_1[v] \neq \text{NULL}$  do
     $v \leftarrow pred_1[v]$ 
     $pot_1.append(v)$ 
end while
return ( $r, pot_0 + pot_1$ )
end function

```

če je vrsta prazna, ni poti od  $s$  do  $t$

vzamemo naslednje vozlišče

poiščemo vozlišče, preko katerega poteka najkrajša pot

rekonstruiramo del poti od  $s$

rekonstruiramo del poti do  $t$

Pokažimo, da algoritem deluje pravilno. Naj bosta  $S$  in  $T$  množici vozlišč, ki jih algoritem ob izteku prve zanke **while** doseže iz  $s$  oziroma  $t$  (tj., odstrani ustrezni vnos v  $Q$ ) – do teh vozlišč torej poznamo dolžino najkrajše poti od  $s$  oziroma do  $t$ . V preseku množic  $S$  in  $T$  je natanko eno vozlišče, in sicer  $u$ . Za vsako vozlišče  $v \notin S \cup T$  velja  $d_G(s, v) \geq d_G(s, u)$  in  $d_G(v, t) \geq d_G(u, t)$ , tako da če najkrajša pot od  $s$  do  $t$  vsebuje vozlišče  $v$ , obstaja tudi najkrajša pot, ki vsebuje vozlišče  $u$ . Tako vidimo, da najkrajša pot med  $s$  in  $t$  bodisi vsebuje vozlišče  $u$ , bodisi vsebuje povezavo s krajiščema v  $S$  in  $T$ .

Če najkrajša pot med  $s$  in  $t$  bodisi vsebuje vozlišče  $u$ , potem algoritem očitno pravilno najde najkrajšo pot. Denimo sedaj, da najkrajša pot med  $s$  in  $t$  vsebuje povezavo  $vw$ , kjer je  $v \in S$  in  $w \in T$ . Očitno tudi najkrajši poti od  $s$  do  $w$  in od  $v$  do  $t$  vsebujeta povezavo  $vw$  (sicer bi obstajala krajša pot od  $s$  do  $t$  preko  $v$  ali  $w$ ), zato ob izteku prve zanke **while** velja  $d_G(s, w) = Q[w, 0]$  in  $d_G(v, t) = Q[v, 1]$ . Tako algoritem obravnava najkrajšo pot od  $s$  do  $t$  preko vozlišča  $v$  ali  $w$  in zato pravilno najde najkrajšo pot.

Tako kot Dijkstrov algoritem tudi zgornji algoritem teče v času  $O(m \log n)$  (kjer je  $n$  število vozlišč in  $m$  število povezav podanega grafa – ob predpostavki, da je graf povezan), če za prednostno vrsto uporabimo kopico.

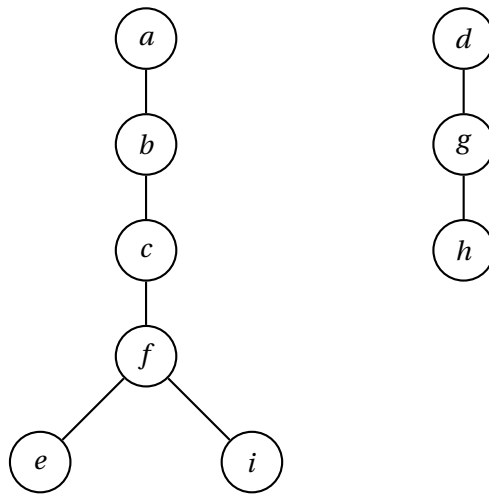
### Naloga 5.12.

Sledili bomo naslednjemu algoritmu.

```

function DFS( $G = (V, E)$ ,  $S$ , PREVISIT, POSTVISIT)
    for  $v \in V$  do
        visited[ $v$ ]  $\leftarrow$  FALSE
    end for

```



Slika 86: Gozd iskanja v globino za nalogo 5.12.

```

function RAZIŠČI( $v, w$ )
  if visited[ $v$ ] then
    return
  end if
  visited[ $v$ ]  $\leftarrow$  TRUE
  PREVISIT( $v, w$ )
  for  $u \in \text{ADJ}(G, v)$  do:
    RAZIŠČI( $u, v$ )
  end for
  POSTVISIT( $v, w$ )
end function
for  $v \in S$  do
  RAZIŠČI( $v, \text{NULL}$ )
end for
end function

```

Časovna zahtevnost preiskovanja je  $O(m + n)$  (kjer je  $n$  število vozlišč in  $m$  število povezav grafa), poleg tega pa algoritem opravi še  $O(n)$  klicev funkcij PREVISIT in POSTVISIT.

Potek zgornjega algoritma na grafu s slike 13 (pri čemer za množico začetnih vozlišč  $S$  vzamemo množico vseh vozlišč  $V$ , za PREVISIT in POSTVISIT pa vzamemo NOOP, torej ne naredimo ničesar), pri čemer sledimo abecednemu vrstnemu redu vozlišč, je prikazan v tabeli 20. Gozd iskanja v globino je prikazan na sliki 86, iz katere je razvidno, da so drevesne povezave

$(a, b), (b, c), (c, f), (f, e), (f, i), (d, g), (g, h)$ .

$v$	$u$	množica označenih vozlišč
$a$		$\{a\}$
$a$	$b$	$\{a, b\}$
$b$	$a$	$\{a, b\}$
$b$	$c$	$\{a, b, c\}$
$c$	$b$	$\{a, b, c\}$
$c$	$f$	$\{a, b, c, f\}$
$f$	$c$	$\{a, b, c, f\}$
$f$	$e$	$\{a, b, c, e, f\}$
$e$	$a$	$\{a, b, c, e, f\}$
$e$	$b$	$\{a, b, c, e, f\}$
$e$	$f$	$\{a, b, c, e, f\}$
$f$	$i$	$\{a, b, c, e, f, i\}$
$i$	$f$	$\{a, b, c, e, f, i\}$
$b$	$e$	$\{a, b, c, e, f, i\}$
$a$	$e$	$\{a, b, c, e, f, i\}$
$b$		$\{a, b, c, e, f, i\}$
$c$		$\{a, b, c, e, f, i\}$
$d$		$\{a, b, c, d, e, f, i\}$
$d$	$g$	$\{a, b, c, d, e, f, g, i\}$
$g$	$d$	$\{a, b, c, d, e, f, g, i\}$
$g$	$h$	$\{a, b, c, d, e, f, g, h, i\}$
$h$	$d$	$\{a, b, c, d, e, f, g, h, i\}$
$h$	$g$	$\{a, b, c, d, e, f, g, h, i\}$
$d$	$h$	$\{a, b, c, d, e, f, g, h, i\}$

Tabela 20: Potek izvajanja algoritma za nalogo 5.12.

**Naloga 5.13.**

Sledili bomo naslednjemu algoritmu.

```

function BELLMANFORD( $G = (V, E), s \in V, L : E \rightarrow \mathbb{R}$ )
   $d \leftarrow$  slovar z vrednostjo  $\infty$  za vsako vozlišče  $v \in V$ 
   $pred \leftarrow$  slovar z vrednostjo NULL za vsako vozlišče  $v \in V$ 
   $d[s] \leftarrow 0$ 
   $i \leftarrow 0$ 
   $trenutna \leftarrow \{s\}$ 
  while  $\neg trenutna.isEmpty()$  do
     $i \leftarrow i + 1$ 
    if  $i > |V|$  then
      return graf ima negativen cikel
    end if
     $naslednja \leftarrow$  prazna množica
    for  $u \in trenutna$  do
      for  $v \in \text{ADJ}(G, u)$  do
        if  $d[v] > d[u] + L_{uv}$  then
           $d[v] \leftarrow d[u] + L_{uv}$ 
           $pred[v] \leftarrow u$ 
           $naslednja.add(v)$ 
        end if
      end for
    end for
     $trenutna \leftarrow naslednja$ 
  end while
  return  $(d, pred)$ 
end function

```

V  $i$ -tem obhodu zanke **while** slovar  $d$  vsebuje dolžine tistih najkrajših poti od  $s$  do vsakega drugega vozlišča, ki vsebujejo največ  $i$  povezav. Če graf nima negativnih ciklov, je v vsaki najkrajši poti največ  $n - 1$  povezav (kjer je  $n$  število vozlišč) in zato algoritem konča najkasneje po  $n$ -tem obhodu zanke **while**. V nasprotnem primeru obstaja najkrajša pot z neskončno povezavami, kar algoritem zazna ob vstopu v  $(n + 1)$ -vi obhod zanke. Časovna zahtevnost algoritma je tako  $O(mn)$ , kjer je  $m$  število povezav grafa.

Potek zgornjega algoritma na grafu s slike 15 je prikazan v tabeli 21. Drevo najkrajših poti je prikazano na sliki 87.

**Naloga 5.14.**

- (a) Upoštevamo, da je graf  $G = (V, E)$  acikličen, torej za vsaki vozlišči  $u, v \in V$  velja

$$u \rightarrow v \implies \text{postlabel}(u) > \text{postlabel}(v),$$

kjer je  $\text{postlabel}$  števec, s katerim označimo vozlišče, ko ga v algoritmu DFS docela preiščemo.

Trditev lahko dokažemo z obravnavanjem dveh primerov, in sicer, ko z DFS pridemo v vozlišče  $u$  pred  $v$ , in ko pridemo v vozlišče  $v$  pred  $u$ . V prvem

$i$	$u$	$v$	$L_{uv}$	razdalje in predhodniki
				$[0, \infty, \infty, \infty, \infty, \infty, \infty, \infty]$
1	$a$	$b$	3	$[0, 3_a, \infty, \infty, \infty, \infty, \infty, \infty]$
1	$a$	$f$	6	$[0, 3_a, \infty, \infty, \infty, \infty, 6_a, \infty]$
2	$b$	$c$	7	$[0, 3_a, 10_b, \infty, \infty, \infty, 6_a, \infty]$
2	$b$	$e$	9	$[0, 3_a, 10_b, \infty, 12_b, 6_a, \infty, \infty]$
2	$f$	$g$	8	$[0, 3_a, 10_b, \infty, 12_b, 6_a, 14_f, \infty]$
3	$c$	$d$	-7	$[0, 3_a, 10_b, 3_c, 12_b, 6_a, 14_f, \infty]$
3	$e$	$d$	1	$[0, 3_a, 10_b, 3_c, 12_b, 6_a, 14_f, \infty]$
3	$e$	$g$	1	$[0, 3_a, 10_b, 3_c, 12_b, 6_a, 13_e, \infty]$
3	$g$	$f$	-1	$[0, 3_a, 10_b, 3_c, 12_b, 6_a, 13_e, \infty]$
4	$d$	$b$	9	$[0, 3_a, 10_b, 3_c, 12_b, 6_a, 13_e, \infty]$
4	$d$	$h$	4	$[0, 3_a, 10_b, 3_c, 12_b, 6_a, 13_e, 7_d]$
4	$g$	$f$	-1	$[0, 3_a, 10_b, 3_c, 12_b, 6_a, 13_e, 7_d]$
5	$h$	$g$	-5	$[0, 3_a, 10_b, 3_c, 12_b, 6_a, 2_h, 7_d]$
6	$g$	$f$	-1	$[0, 3_a, 10_b, 3_c, 12_b, 1_g, 2_h, 7_d]$
7	$f$	$g$	8	$[0, 3_a, 10_b, 3_c, 12_b, 1_g, 2_h, 7_d]$

Tabela 21: Potek izvajanja algoritma za nalogo 5.13.

primeru bomo pred določitvijo pooznake vozlišču  $u$  morali obiskati vse njegove sosedbe, torej tudi  $v$ , ki mu bomo določili pooznako pred  $u$ . V drugem primeru pridemo prej do  $v$ , a ker je graf acikliččen, od  $v$  ne bomo prišli do  $u$ , torej bomo spet  $v$  obdelali prej.

Trditev nam omogoča zapis krajšega algoritma, ki bo uporabil algoritem DFS iz naloge 5.12. Njegova ideja je, da padajoče uredimo vozlišča po njihovih poznakah in tako dobimo inverzno topološko ureditev. To dosežemo tako, da po popolni obdelavi vozlišča tega postavimo na sklad.

```

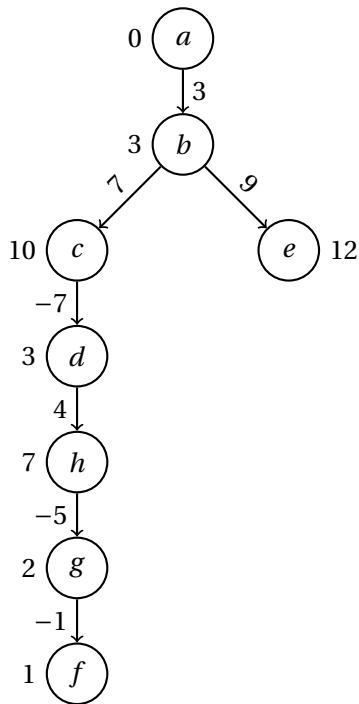
function TOPO( $G = (V, E)$ )
   $toposklad \leftarrow []$ 
  function POSTVISIT( $u, v$ )
     $toposklad.append(u)$ 
  end function
  DFS( $G, V, \text{NOP}, \text{POSTVISIT}$ )
   $toposklad.reverse()$ 
  return  $toposklad$ 
end function

```

Časovna zahtevnost algoritma je  $O(m + n)$ , kjer je  $n$  število vozlišč in  $m$  število povezav grafa.

Klic  $\text{TOPO}(G)$  nam vrne ureditev vozlišč  $[g, a, h, b, c, f, d, e]$ .





Slika 87: Drevo najkrajših poti za nalogo 5.13.

- (b) Tukaj bomo uporabili rezultat iz prejšnje točke in osnoven koncept dinamičnega programiranja.

Označimo z  $L_{uv}$  utež povezave  $u \rightarrow v$ . Naj  $d_G(u)$  predstavlja najkrajšo razdaljo od izbranega vozlišča  $s$  do vozlišča  $u$ .

$$d_G(s) = 0$$

$$d_G(u) = \min_{v \rightarrow u} (d_G(v) + L_{vu}) \quad (u \in V \setminus \{s\})$$

Da bomo imeli vse potrebne  $d_G(v)$  definirane pred izračunom  $d_G(u)$ , poskrbimo z rezultatom iz prejšnje točke, saj pri topološki ureditvi vodijo vse povezave le naprej. Po premiku na naslednje vozlišče tako ne potrebujemo kasnejših vozlišč, pač pa le prejšnja, za katera vrednost že poznamo.

Imamo torej vse, kar potrebujemo za izračun najkrajše poti. Za beleženje vozlišč, skozi katera vodi ta pot, bomo uporabili seznam predhodnikov, ki bo nakazoval, katero vozlišče smo izbrali za predhodnika.

$$pred(s) = \text{NULL}$$

$$pred(u) = \operatorname{argmin}_{v \rightarrow u} (d_G(v) + L_{vu}) \quad (u \in V \setminus \{s\})$$

Zapišimo algoritem, ki poišče razdalje od  $s$  do ostalih vozlišč. Poleg razdalj bo za vsako vozlišče povedal še, iz katerega vozlišča smo prišli do njega.

```
function NAJKRAJŠAPOT( $G = (V, E), s, L: E \rightarrow \mathbb{R}$ )
     $d_G \leftarrow$  slovar z vrednostjo  $\infty$  za vsako vozlišče  $v \in V$ 
     $d_G[s] \leftarrow 0$ 
     $pred \leftarrow$  slovar z vrednostjo NULL za vsako vozlišče  $v \in V$ 
    for  $u \in \text{TOPO}(G)$  do
        for  $v \in \text{ADJ}(G, u)$  do
            if  $d_G[v] > d_G[u] + L_{uv}$  then
                 $d_G[v] \leftarrow d_G[u] + L_{uv}$ 
                 $pred[v] \leftarrow u$ 
            end if
        end for
    end for
    return ( $d_G, pred$ )
end function
```

Časovna zahtevnost algoritma je  $O(m + n)$ , kjer je  $n$  število vozlišč in  $m$  število povezav grafa.

Če želimo rekonstruirati pot, sledimo predhodnikom podanega končnega vozlišča  $t$ , dokler ne pridemo do  $s$ .

```
function REKONSTRUIRAJPOT( $pred, t$ )
     $u \leftarrow t$ 
     $pot \leftarrow [t]$ 
    while  $pred[u] \neq \text{NULL}$  do
         $u \leftarrow pred[u]$ 
         $pot.append(u)$ 
    end while
     $pot.reverse()$ 
    return  $pot$ 
end function
```

Klic  $\text{NAJKRAJŠAPOT}(G, g, L)$  vrne seznam  $d_G$  z razdaljami od vozlišča  $g$  do ostalih vozlišč grafa in seznam  $pred$  s prednikom vsakega vozlišča v najkrajši poti od  $g$ . Potek izvajanja algoritma je prikazan v tabeli 22. Seznam vozlišč, skozi katera je algoritem potoval, da je opravil pot od  $g$  do  $e$ , nato dobimo s klicem  $\text{REKONSTRUIRAJPOT}(pred, e)$  – to so  $[g, a, b, e]$ . Dolžino te poti lahko preberemo kot  $d_G[e] = -1$ .

Pripomnimo, da algoritem deluje pravilno ne glede na položaj začetnega vozlišča  $s$  v topološki ureditvi. Za vsako vozlišče  $u$ , ki v topološki ureditvi nastopa pred  $s$ , namreč velja  $d_G(u) = \infty$ , tako da se ne nastavi kot predhodnik nobenemu vozlišču. Tako se vozlišču  $s$  ne bo nastavil predhodnik. Algoritem lahko nekoliko pohitimo tako, da gledamo vozlišča v topološki ureditvi le od  $s$  naprej, prejšnja pa ignoriramo, saj iz  $s$  ne moremo priti do njih. Prav tako bi se lahko ustavili, ko dosežemo vozlišče  $t$ .

- (c) Uporabimo algoritem  $\text{NAJDALJŠAPOT}$ , ki deluje na enak način kot  $\text{NAJKRAJŠAPOT}$ , le da namesto  $\infty$  vzame v  $d_G$  za začetne vrednosti  $-\infty$ , v zanki pa na

vsakem koraku preverja pogoj  $d_G[v] < d_G[u] + L_{uv}$ . Časovna zahtevnost algoritma tako ostane enaka. Klic REKONSTRUIRAJPOT( $pred, e$ ), kjer je  $d_G, pred$  izhod klica NAJDALJŠAPOT( $G, g, L$ ), vrne pot  $[g, h, b, c, f, e]$  dolžine 24. Potek izvajanja algoritma je prikazan v tabeli 23.

### Naloga 5.15.

- (a) Algoritem deluje podobno kot algoritem NAJKRAJŠAPOT iz naloge 5.14(b), le da vsakič prišteje število možnih poti iz trenutnega vozlišča. Pri tem sledi topološki ureditvi, ki jo izračuna z algoritmom TOPO iz naloge 5.14(a).

```

function ŠTEVILOPOTI( $G = (V, E), s, t, L : E \rightarrow \mathbb{R}$ )
   $C \leftarrow$  slovar z vrednostjo 0 za vsako vozlišče  $v \in V$ 
   $C[s] \leftarrow 1$ 
  for  $u \in \text{TOPO}(G)$  do
    for  $v \in \text{ADJ}(G, u)$  do
       $C[v] \leftarrow C[v] + C[u] \cdot L_{uv}$ 
    end for
  end for
  return  $C[t]$ 
end function

```

Časovna zahtevnost algoritma je  $O(m + n)$ , kjer je  $n$  število vozlišč in  $m$  število povezav grafa.

- (b) Potek izvajanja algoritma na grafu s slike 17 je prikazan v tabeli 24. Pri tem je uporabljena topološka ureditev vozlišč  $[s, a, b, d, c, e, f, g, t]$ . Od vozlišča  $s$  do vozlišča  $t$  je mogoče priti na 58 000 načinov.

### Naloga 5.16.

Naj bo  $G = (V, E)$  usmerjen graf z  $n = |V|$  vozlišči. Za vsako vozlišče  $u \in V$  in vsak  $i$  ( $1 \leq i \leq n$ ) uvedimo spremenljivko  $x_{ui}$ , katere vrednost interpretiramo kot

$$x_{ui} = \begin{cases} 1, & \text{vozlišče } u \text{ je } i\text{-to v topološki ureditvi, in} \\ 0 & \text{sicer.} \end{cases}$$

Zapišimo linearne omejitve.

$$\forall u \in V \forall i \in \{1, \dots, n\} : \quad 0 \leq x_{ui} \leq 1, \quad x_{ui} \in \mathbb{Z}$$

Vsako vozlišče je na natanko enem mestu:

$$\forall u \in V : \quad \sum_{i=1}^n x_{ui} = 1$$

Na vsakem mestu je natanko eno vozlišče:

$$\forall i \in \{1, \dots, n\} : \quad \sum_{u \in V} x_{ui} = 1$$

$u$	$v$	$d_G$													$\text{pred}[v]$
		$d_G(v)$	$\overset{?}{>}$	$d_G(u)$	$+$	$L_{uv}$	$a$	$b$	$c$	$d$	$e$	$f$	$g$	$h$	
$g$	$a$	$\infty$	$>$	$0$	$+$	$(-1)$	$-1$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$0$	$\infty$	$g$
$g$	$d$	$\infty$	$>$	$0$	$+$	$6$	$-1$	$\infty$	$\infty$	$6$	$\infty$	$\infty$	$0$	$\infty$	$g$
$g$	$f$	$\infty$	$>$	$0$	$+$	$6$	$-1$	$\infty$	$\infty$	$6$	$\infty$	$6$	$0$	$\infty$	$g$
$g$	$h$	$\infty$	$>$	$0$	$+$	$6$	$-1$	$\infty$	$\infty$	$6$	$\infty$	$6$	$0$	$6$	$g$
$a$	$b$	$\infty$	$>$	$-1$	$+$	$2$	$-1$	$1$	$\infty$	$6$	$\infty$	$6$	$0$	$6$	$a$
$a$	$c$	$\infty$	$>$	$-1$	$+$	$(-2)$	$-1$	$1$	$-3$	$6$	$\infty$	$6$	$0$	$6$	$a$
$a$	$h$	$6$	$>$	$-1$	$+$	$0$	$-1$	$1$	$-3$	$6$	$\infty$	$6$	$0$	$-1$	$a$
$h$	$b$	$1$	$\nless$	$-1$	$+$	$3$	$-1$	$1$	$-3$	$6$	$\infty$	$6$	$0$	$-1$	$a$
$h$	$f$	$6$	$>$	$-1$	$+$	$6$	$-1$	$1$	$-3$	$6$	$\infty$	$5$	$0$	$-1$	$h$
$b$	$c$	$-3$	$\nless$	$1$	$+$	$6$	$-1$	$1$	$-3$	$6$	$\infty$	$5$	$0$	$-1$	$a$
$b$	$e$	$\infty$	$>$	$1$	$+$	$(-2)$	$-1$	$1$	$-3$	$6$	$-1$	$5$	$0$	$-1$	$b$
$c$	$d$	$6$	$>$	$-3$	$+$	$2$	$-1$	$1$	$-3$	$-1$	$-1$	$5$	$0$	$-1$	$c$
$c$	$f$	$5$	$>$	$-3$	$+$	$6$	$-1$	$1$	$-3$	$-1$	$-1$	$3$	$0$	$-1$	$c$
$f$	$e$	$-1$	$\nless$	$3$	$+$	$6$	$-1$	$1$	$-3$	$-1$	$-1$	$3$	$0$	$-1$	$b$
$d$	$e$	$-1$	$\nless$	$-1$	$+$	$7$	$-1$	$1$	$-3$	$-1$	$-1$	$3$	$0$	$-1$	$b$

Tabela 22: Potek izvajanja algoritma NAJKRAJŠAPOT za nalogo 5.14(b).

$u$	$v$	$d_G$													$\text{pred}[v]$
		$d_G(v)$	$\overset{?}{<}$	$d_G(u)$	$+$	$L_{uv}$	$a$	$b$	$c$	$d$	$e$	$f$	$g$	$h$	
$g$	$a$	$-\infty$	$<$	$0$	$+$	$(-1)$	$-1$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$0$	$-\infty$	$g$
$g$	$d$	$-\infty$	$<$	$0$	$+$	$6$	$-1$	$-\infty$	$-\infty$	$6$	$-\infty$	$-\infty$	$0$	$-\infty$	$g$
$g$	$f$	$-\infty$	$<$	$0$	$+$	$6$	$-1$	$-\infty$	$-\infty$	$6$	$-\infty$	$6$	$0$	$-\infty$	$g$
$g$	$h$	$-\infty$	$<$	$0$	$+$	$6$	$-1$	$-\infty$	$-\infty$	$6$	$-\infty$	$6$	$0$	$6$	$g$
$a$	$b$	$-\infty$	$<$	$-1$	$+$	$2$	$-1$	$1$	$-\infty$	$6$	$-\infty$	$6$	$0$	$6$	$a$
$a$	$c$	$-\infty$	$<$	$-1$	$+$	$(-2)$	$-1$	$1$	$-3$	$6$	$-\infty$	$6$	$0$	$6$	$a$
$a$	$h$	$6$	$\nless$	$-1$	$+$	$0$	$-1$	$1$	$-3$	$6$	$-\infty$	$6$	$0$	$6$	$g$
$h$	$b$	$1$	$<$	$6$	$+$	$3$	$-1$	$9$	$-3$	$6$	$-\infty$	$6$	$0$	$6$	$h$
$h$	$f$	$6$	$<$	$6$	$+$	$6$	$-1$	$9$	$-3$	$6$	$-\infty$	$12$	$0$	$6$	$h$
$b$	$c$	$-3$	$<$	$9$	$+$	$6$	$-1$	$9$	$15$	$6$	$-\infty$	$12$	$0$	$6$	$b$
$b$	$e$	$-\infty$	$<$	$9$	$+$	$(-2)$	$-1$	$9$	$15$	$6$	$7$	$12$	$0$	$6$	$b$
$c$	$d$	$6$	$<$	$15$	$+$	$2$	$-1$	$9$	$15$	$17$	$7$	$12$	$0$	$6$	$c$
$c$	$f$	$12$	$<$	$15$	$+$	$6$	$-1$	$9$	$15$	$17$	$7$	$21$	$0$	$6$	$c$
$f$	$e$	$7$	$<$	$21$	$+$	$6$	$-1$	$9$	$15$	$17$	$27$	$21$	$0$	$6$	$f$
$d$	$e$	$27$	$\nless$	$17$	$+$	$7$	$-1$	$9$	$15$	$17$	$27$	$21$	$0$	$6$	$f$

Tabela 23: Potek izvajanja algoritma NAJDALJŠAPOT za nalogo 5.14(c).

$u$	$v$	$L_{uv}$	$C[s, a, b, c, d, e, f, g, t]$
			[1, 0, 0, 0, 0, 0, 0, 0, 0]
$s$	$a$	10	[1, 10, 0, 0, 0, 0, 0, 0, 0]
$s$	$b$	10	[1, 10, 10, 0, 0, 0, 0, 0, 0]
$s$	$c$	10	[1, 10, 10, 10, 0, 0, 0, 0, 0]
$s$	$d$	10	[1, 10, 10, 10, 10, 0, 0, 0, 0]
$a$	$b$	5	[1, 10, 60, 10, 10, 0, 0, 0, 0]
$a$	$e$	5	[1, 10, 60, 10, 10, 50, 0, 0, 0]
$b$	$e$	2	[1, 10, 60, 10, 10, 170, 0, 0, 0]
$d$	$c$	5	[1, 10, 60, 60, 10, 170, 0, 0, 0]
$c$	$f$	2	[1, 10, 60, 60, 10, 170, 120, 0, 0]
$e$	$g$	10	[1, 10, 60, 60, 10, 170, 120, 1700, 0]
$f$	$g$	10	[1, 10, 60, 60, 10, 170, 120, 2900, 0]
$g$	$t$	10	[1, 10, 60, 60, 10, 170, 120, 2900, 58000]

Tabela 24: Potek izvajanja algoritma za nalogo 5.15.

Vsaka povezava gre v smeri povečanja mesta:

$$\forall uv \in E : \sum_{i=1}^n i(x_{vi} - x_{ui}) \geq 1$$

Ciljna funkcija tukaj ni pomembna – vsaka dopustna rešitev nam da topološko urejanje.

#### Naloga 5.17.

Algoritem deluje tako, da v vrsto najprej doda vozlišča z vhodno stopnjo 0, nato pa pobira vozlišča iz vrste in jih odstranjuje iz grafa, pri čemer se v vrsto dodajajo vozlišča, ki jim pri tem izhodna stopnja pade na 0. Če se zgodi, da je v nekem trenutku v vrsti več kot eno vozlišče, si algoritem to zapomni – če algoritem tako obišče vsa vozlišča (tj., obstaja topološka ureditev), potem je topološka ureditev več kot ena in algoritem vrne TRUE. Če pa je topološka ureditev ena sama ali pa te ni, algoritem vrne FALSE.

**function** VEČTOPO( $G = (V, E)$ )

$stopnja \leftarrow$  slovar vhodnih stopenj vozlišč grafa  $G$

$Q \leftarrow$  prazna vrsta

$več \leftarrow$  FALSE

$prvo \leftarrow$  FALSE

$i \leftarrow 0$

števec obiskanih vozlišč

**for**  $v \in V$  **do**

začetno iskanje izvorov

**if**  $stopnja[v] = 0$  **then**

$Q.push(v)$

**if**  $prvo$  **then**

če je bil predhodno najden izvor,

$več \leftarrow$  TRUE

rešitev ni enolična

**else**

```

        prvo ← TRUE                                prvi najden izvor
    end if
end if
end for
while ¬Q.isEmpty() do
    u ← Q.pop()
    i ← i + 1                                       povečamo števec
    prvo ← FALSE
    for v ∈ ADJ(G, u) do                           izhodni sosedi vozlišča u
        stopnja[v] ← stopnja[v] - 1
        if stopnja[v] = 0 then                       nov izvor po odstranjevanju u
            Q.push(v)
            if prvo then
                več ← TRUE
            else
                prvo ← TRUE
            end if
        end if
    end for
end while
return več ∧ (i = |V|)    vrni TRUE, če obstaja topološka ureditev in ta ni enolična
end function

```

### Naloga 5.18.

- (a) Iz danega neusmerjenega grafa  $G = (V, E)$  in funkcije uteži vozlišč  $c : V \rightarrow [0, 1]$  bomo konstruirali usmerjen graf  $G' = (V, E')$ , kjer je  $E' = \{uv, vu \mid uv \in E\}$  (tj., vsako neusmerjeno povezavo iz  $G$  nadomestimo z nasprotno usmerjenima povezavama med krajiščema), in funkcijo uteži povezav  $L : E' \rightarrow [0, 1]$  s predpisom  $L(uv) = c(u)$  (tj., cena povezave je enaka ceni začetnega vozlišča v  $G$ ). Na grafu  $G'$  s funkcijo uteži povezav  $L$  nato poženemo varianto algoritma DIJKSTRA iz naloge 5.8.

```

function DIJKSTRAPROB( $G = (V, E), s \in V, L : E \rightarrow [0, 1]$ )
     $d \leftarrow$  slovar z vrednostjo 0 za vsako vozlišče  $v \in V$ 
     $pred \leftarrow$  slovar z vrednostjo NULL za vsako vozlišče  $v \in V$ 
     $Q \leftarrow$  prednostna vrsta s prioriteto 0 za vsako vozlišče  $v \in V$  (na vrhu je vozlišče
    z največjo prioriteto)
     $Q[s] \leftarrow 1$ 
    while ¬ $Q.isEmpty()$  do
         $u, d[u] \leftarrow Q.pop()$ 
        for  $v \in ADJ(G, u)$  do
            if  $v \in Q \wedge Q[v] < d[u] \cdot L_{uv}$  then
                 $Q[v] \leftarrow d[u] \cdot L_{uv}$ 
                 $pred[v] \leftarrow u$ 
            end if
        end for
    end while
    return ( $d, pred$ )
end function

```

LA	SF	PH	LV	RE	EP	AQ	DE	SLC	DA	OC	KC	OM	ME	SL	CH
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
*	1 <sub>LA</sub>	1 <sub>LA</sub>	1 <sub>LA</sub>												
	*			1 <sub>SF</sub>											
		*			0.8 <sub>PH</sub>	0.8 <sub>PH</sub>									
			*				0.5 <sub>LV</sub>	0.5 <sub>LV</sub>							
				*				0.7 <sub>RE</sub>							
					*				0.48 <sub>EP</sub>						
						*	0.56 <sub>AQ</sub>			0.56 <sub>AQ</sub>					
							0.63 <sub>SLC</sub>	*							
							*				0.504 <sub>DE</sub>	0.504 <sub>DE</sub>			
										*					
											*				
													0.336 <sub>OC</sub>		
														0.3528 <sub>KC</sub>	
									*						0.2016 <sub>OM</sub>
													0.384 <sub>DA</sub>		
													*		
														*	0.31752 <sub>SL</sub>
															*

Tabela 25: Potek izvajanja algoritma za nalogo 5.18(b).

V primerjavi z algoritmom DIJKSTRA smo torej zamenjali seštevanje z množenjem in obrnili primerjave, poleg tega pa smo še ustrezno spremenili začetne vrednosti. Preslikava  $x \mapsto -\log x$  pošlje uteži na interval  $[0, \infty]$ , pri čemer se urejenost obrne, množenje se pa nadomesti s seštevanjem – zgornji algoritem je torej ekvivalenten algoritmu DIJKSTRA s tako preslikanimi utežmi, in ima tako isto časovno zahtevnost (tj.,  $O(n^2)$ ), če za prednostno vrsto uporabimo običajen slovar, in  $O(m \log n)$ , če za prednostno vrsto vzamemo kopico, kjer je  $n$  število vozlišč in  $m$  število povezav v grafu).

- (b) Potek izvajanja zgornjega algoritma je prikazan v tabeli 25, iz katere razberemo, da je najvarnejša pot LA – SF – RE – SLC – DE – KC – SL – CH, z verjetnostjo, da nas ne oropajo, enako 0.31752.

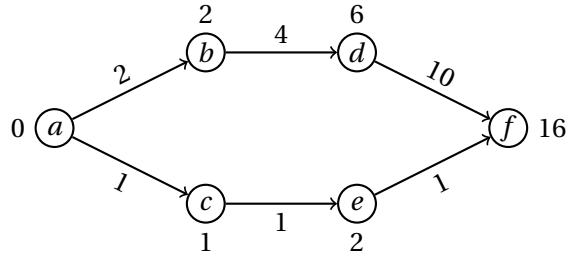
### Naloga 5.19.

Ideja algoritma je tesno povezana z algoritmom DIJKSTRA iz naloge 5.8. Če so vse vhodne razdalje pravilne, lahko predpostavimo, da so bile poiskane z Dijkstrovim algoritmom. Če torej sledimo poti, ki jo ta opravi, bomo lahko preverili prav vse nastavljene razdalje  $\delta_v$ .

```

function RAZDALJE( $G = (V, E), s \in V, L: E \rightarrow \mathbb{R}_+, (\delta_v)_{v \in V}$ )
  if  $\delta_s \neq 0$  then
    return FALSE
  end if
   $doseženi \leftarrow$  slovar z vrednostjo FALSE za vsak  $v \in V$ 
   $\ell \leftarrow [s]$ 
  while  $\neg \ell.isEmpty()$  do
     $u \leftarrow \ell.pop()$ 

```



Slika 88: Primer grafa za nalogo 5.19.

```

doseženi[u] ← TRUE
for v ∈ ADJ(G, u) do
    if δu + Luv = δv then
        if ¬doseženi[v] then
            ℓ.append(v)
        end if
    else if δu + Luv < δv then
        return FALSE
    end if
end for
end while
return ∀v ∈ V : doseženi[v]
end function

```

V primeru, ko vrednosti  $\delta_v$  ustrezajo razdaljam od  $s$ , bo algoritem deloval pravilno, kar sledi iz pravilnosti Dijkstrovega algoritma in predpostavke, da so vsa vozlišča dosegljiva iz  $s$ .

Če bo za neko vozlišče  $v$  vrednost  $\delta_v$  manjša od razdalje  $d_G(s, v)$ , potem algoritem vozlišče  $v$  ne bo označil. Če pa je vrednost  $\delta_v$  večja od razdalje  $d_G(s, v)$ , algoritem morda lahko označi vozlišče  $v$ , a se to lahko zgodi le, če pridemo do njega z manj kot optimalno potjo – to vozlišče je torej dosegljivo še iz vsaj ene druge poti. Predpostavimo, da je  $t$  najvišje vozlišče v drevesu najkrajših poti, ki ga algoritem označi, čeprav velja  $\delta_t > d_G(s, t)$ , in naj bo  $r$  njegov predhodnik v drevesu. Ker velja  $\delta_r = d_G(s, r)$ , bo algoritem pri pregledovanju sosedov vozlišča  $r$  ugotovil, da velja  $\delta_r + L_{rt} < \delta_t$ , in torej vrnil FALSE.

Algoritem pregleda vsako vozlišče največ enkrat in pri vsakem njegovem sosedu porabi konstantno časa. Časovno zahtevnost lahko torej izračunamo kot

$$\sum_{v \in V} (O(1) + \deg_G(u) \cdot O(1)) = O(|V|) + O(|E|) = O(|V| + |E|).$$

Poglejmo si, kako algoritem deluje na grafu  $G = (V, E)$  s slike 88, pri čemer vzamemo  $\delta_v = d_G(a, v)$  za vsak  $v \in V \setminus \{f\}$  in  $\delta_f = 16$ . Izvajanje algoritma z začetnim vozliščem  $a$  je prikazano v tabeli 26. Na zadnjem koraku so vsa vozlišča označena, a velja  $d_u + L_{uv} < \delta_v$ , tako da algoritem vrne FALSE.



$u$	$v$	$\delta_u$	$\delta_v$	$L_{uv}$	$dosežen[v]$
$a$	$b$	0	2	2	FALSE
$a$	$c$	0	1	1	FALSE
$b$	$d$	2	6	4	FALSE
$c$	$e$	1	2	1	FALSE
$d$	$f$	6	16	10	FALSE
$e$	$f$	2	16	1	TRUE

Tabela 26: Potek izvajanja algoritma na primeru s slike 88 za nalogo 5.19.

### Naloga 5.20.

- Konstruiramo graf  $G$ , katerega vozlišča so kraji z našega seznama, med vozliščema pa je povezava, če lahko cesto med ustreznima krajema prevozimo v enem dnevu. Če vozlišči  $s$  in  $t$  predstavljata začetno točko in destinacijo, potem lahko na grafu  $G$  izvedemo iskanje v širino z začetkom v vozlišču  $s$  ter v dobljenem drevesu poiščemo pot do vozlišča  $t$ .
- Iz grafa  $G$  izpeljemo graf  $G'$ , ki ga dobimo tako, da odstranimo notranja vozlišča poti iz prejšnje točke (tj.,  $s$  in  $t$  pustimo v  $G'$ ). Potem lahko na grafu  $G'$  izvedemo iskanje v širino z začetkom v vozlišču  $t$  ter v dobljenem drevesu poiščemo pot do vozlišča  $s$ .
- Iskanje v širino na grafu  $G$  s slike 19 nam da drevo s slike 89, s katere je razvidna pot LJ – WI – BN – LU – AM. Graf  $G'$  torej dobimo tako, da iz  $G$  odstranimo vozlišča WI, BN in LU. Iskanje v širino nam da drevo s slike 90, s katere je razvidna povratna pot AM – BL – PR – BS – BU – LJ.

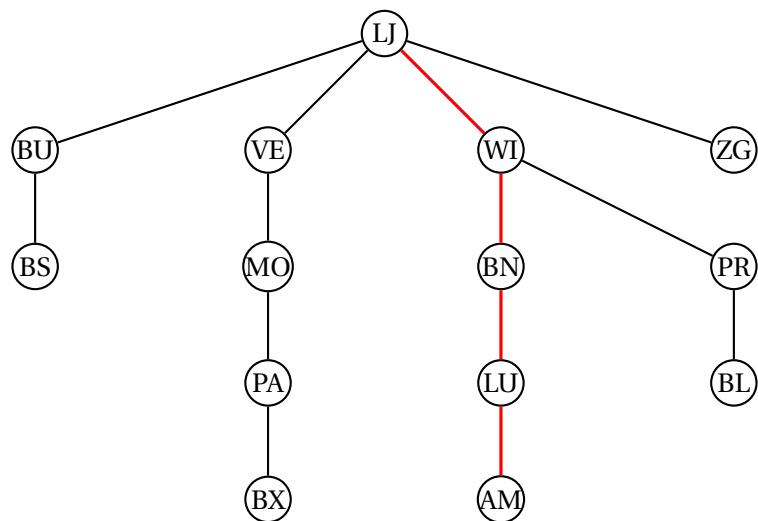
### Naloga 5.21.

- Drevo iskanja v globino je prikazano na sliki 91, pri čemer sta pri vsakem vozlišču  $u$  prikazani še vrednosti  $(\ell_u, p_u)$ .

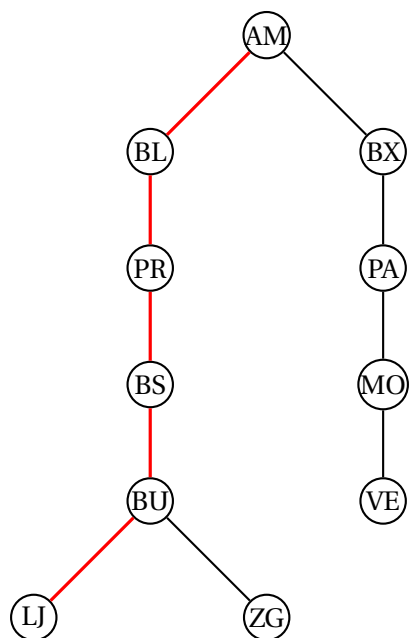
Prerezna vozlišča grafa s slike 20 so  $a$ ,  $b$  in  $c$ . Koren drevesa iskanja v globino  $a$  je prerezna vozlišče, ker ima več kot enega naslednika. Ostala prerezna vozlišča prepoznamo po tem, da imajo takega naslednika v drevesu iskanja v globino, da v njegovem poddrevesu ni vozlišč s prečnimi povezavami do predhodnikov obravnavanega vozlišča – drugače povedano, notranje vozlišče  $u$  v drevesu iskanja v globino je prerezna vozlišče natanko tedaj, ko obstaja njegov naslednik  $w$ , za katerega velja  $p_w \geq \ell_u$ .

- Zapišimo rekurzivno formulo.

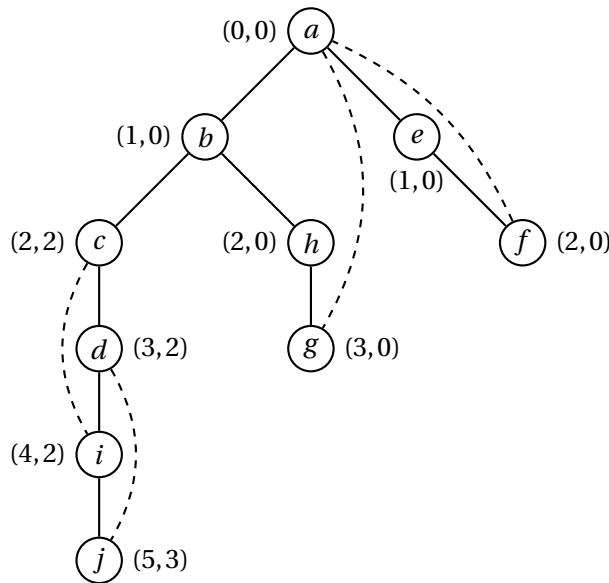
$$p_u = \min(\{p_w \mid w \in V, u \rightarrow w\} \cup \{\ell_w \mid w \in V, u \sim w \vee u = w\})$$



Slika 89: Drevo iskanja v širino na grafu  $G$  za nalogo 5.20.



Slika 90: Drevo iskanja v širino na grafu  $G'$  za nalogo 5.20.



Slika 91: Drevo iskanja v globino za nalogo 5.21.

(c) Zapišimo psevdokodo za funkcijo PREVISIT.

```

function PREVISIT( $u, v$ )
  if  $v = \text{NULL}$  then
     $\ell_u \leftarrow 0$ 
  else
     $\ell_u \leftarrow \ell_v + 1$ 
  end if
end function

```

(d) Zapišimo psevdokodo za funkcijo POSTVISIT.

```

function POSTVISIT( $u, v$ )
  if  $(v = \text{NULL} \wedge |\text{ADJ}(G, u)| > 1) \vee (v \neq \text{NULL} \wedge \exists w \in \text{ADJ}(G, u) : p_w \leq \ell_u)$  then
    izhod.append( $u$ )
  end if
   $p_u \leftarrow \min(\{p_w \mid w \in \text{ADJ}(G, u) \wedge \ell_w > \ell_u\} \cup \{\ell_w \mid w \in \text{ADJ}(G, u) \vee w = u\})$ 
end function

```

(e) Iskanje v globino teče v času  $O(m + T)$ , pri čemer je  $m$  število povezav v grafu,  $T$  pa je čas, porabljen za klic funkcij PREVISIT in POSTVISIT za vsako vozlišče grafa. Funkcija PREVISIT teče v konstantnem času, klic funkcije POSTVISIT( $u, v$ ) pa teče v času  $O(d_G(u))$ , kjer je  $d_G(u)$  stopnja vozlišča  $u$  v grafu  $G$ . Ker je vsota stopenj vozlišč grafa enaka dvakratniku števila povezav, lahko sklenemo, da celoten algoritem teče v času  $O(m)$ .

**Naloga 5.22.**

- (a) Topološko ureditev dobimo z uporabo algoritma TOPO iz naloge 5.14(a). Dobimo  $[s, b, a, d, h, e, c, f, i, g, t]$ . Graf v tej ureditvi si lahko ogledamo na sliki 92.
- (b) Najkrajšo pot od vozlišča  $s$  do vozlišča  $t$  lahko izračunamo z algoritmom NAJKRAJŠAPOT iz naloge 5.14(b). Ta nam vrne pot  $s - a - d - h - i - t$  dolžine 11. Delovanje algoritma si lahko ogledamo v tabeli 27.
- (c) Ker se algoritem odloča neodvisno od svojih prejšnjih odločitev, so vsi dogodki neodvisni. Verjetnost, da pridemo do vozlišča  $v$ , bo vsota verjetnosti vseh dogodkov potovanja po (različnih) poteh, ki se končajo v  $v$ . Verjetnost potovanja po določeni poti pa bo produkt verjetnosti potovanja po vsaki vsebujoči povezavi. Za verjetnosti  $q_v$  v usmerjenem acikličnem grafu torej velja zveza

$$q_v = \sum_{u \rightarrow v} p_{uv} q_u$$

$$q_s = 1.$$

Vrednosti  $q_v$  bomo iskali po topološkem vrstnem redu.

```

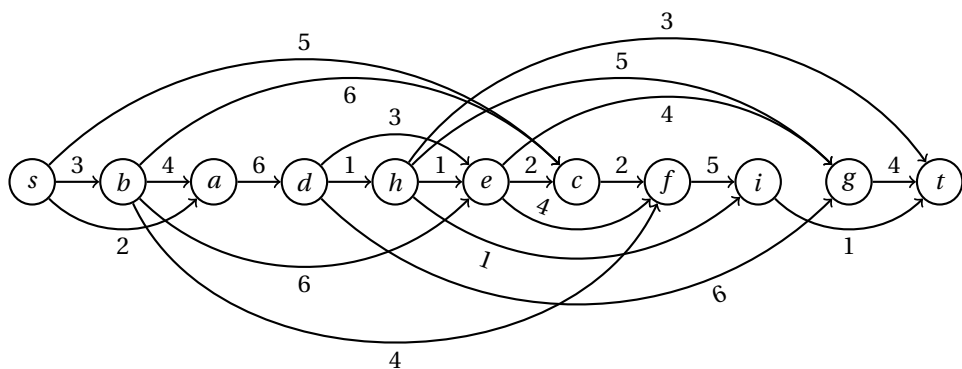
function DAGVERJETNOSTI( $G = (V, E), s, \ell : E \rightarrow \mathbb{R}_+$ )
   $q \leftarrow$  slovar z vrednostjo 0 za vsako vozlišče  $v \in V$ 
   $q[s] \leftarrow 1$ 
  for  $u \in \text{TOPO}(G)$  do
     $p \leftarrow \sum_{w \in \text{ADJ}_G(u)} \ell_{uw}$ 
    for  $v \in \text{ADJ}_G(u)$  do
       $q[v] \leftarrow q[v] + q[u] \ell_{uv} / p$ 
    end for
  end for
  return  $q$ 
end function

```

Časovna zahtevnost algoritma je  $O(m)$ , saj gremo po vsaki povezavi preko vsakega vozlišča enkrat, pri čemer pa moramo seveda poskrbeti, da vsote v števcu definicije vrednosti  $p_{uv}$  ne računamo vsakič sproti, pač pa to storimo pred posodabljanjem verjetnosti sosedom vozlišča  $u$ .

**Naloga 5.23.**

- (a) Imamo neusmerjen graf  $G = (V, E)$  z utežmi na vozliščih  $c_u$  ( $u \in V$ ) in povezavah  $\ell_{uv}$  ( $uv \in E$ ). Problem bi bilo najlažje reševati z Dijkstrovim algoritmom na usmerjenem grafu  $D = (V, A)$ , kjer je  $A = \{uv, vu \mid uv \in E\}$  množica usmerjenih povezav, z utežmi  $L_{uv} = \ell_{uv} + c_v$  ( $uv \in A$ ).
- (b) Potek reševanja problema je prikazan v tabeli 28. Najcenejša pot od LJ do BX je LJ – WI – BN – LU – BX s ceno 150.



Slika 92: Predstavitev topološko urejenega grafa za nalogo 5.22(a).

$u$	$v$	$pred[v]$	$d_G[v]$
$s$	$a$	$s$	2
$s$	$b$	$s$	3
$s$	$c$	$s$	5
$b$	$a$	$s$	2
$b$	$c$	$s$	5
$b$	$e$	$b$	9
$b$	$f$	$b$	7
$a$	$d$	$a$	8
$d$	$e$	$b$	9
$d$	$g$	$d$	14
$d$	$h$	$d$	9
$h$	$e$	$b$	9
$h$	$g$	$d$	14
$h$	$i$	$h$	10
$h$	$t$	$h$	12
$e$	$c$	$s$	5
$e$	$f$	$b$	7
$e$	$g$	$e$	13
$g$	$t$	$h$	12
$c$	$f$	$b$	7
$f$	$i$	$h$	10
$i$	$t$	$i$	11

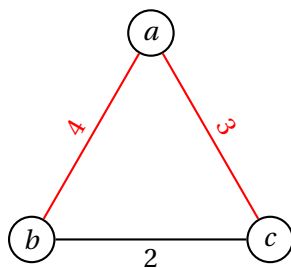
Tabela 27: Potek izvajanja algoritma NAJKRAJŠAPOT za nalogo 5.22(b).

$u$	$v$	$Q$	BX	AM	BL	PR	BS	PA	LU	BN	WI	BU	MO	VE	LJ	ZG
		[(LJ, 0)]	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	0	$\infty$
LJ	VE	[(VE, 35)]												35 <sub>LJ</sub>		
LJ	WI	[(WI, 33), (VE, 35)]									33 <sub>LJ</sub>					
LJ	ZG	[(ZG, 20), (WI, 33), (VE, 35)]														20 <sub>LJ</sub>
LJ	BU	[(ZG, 20), (BU, 25), (WI, 33), (VE, 35)]										25 <sub>LJ</sub>				
ZG	LJ	[(BU, 25), (WI, 33), (VE, 35)]														
ZG	BU	[(BU, 25), (WI, 33), (VE, 35)]														
BU	LJ	[(WI, 33), (VE, 35)]														
BU	ZG	[(WI, 33), (VE, 35)]														
BU	WI	[(WI, 33), (VE, 35)]														
BU	BS	[(WI, 33), (VE, 35), (BS, 40)]					40 <sub>BU</sub>									
WI	LJ	[(VE, 35), (BS, 40)]														
WI	BU	[(VE, 35), (BS, 40)]														
WI	BS	[(VE, 35), (BS, 40)]														
WI	PR	[(VE, 35), (BS, 40), (PR, 53)]				53 <sub>WI</sub>										
WI	BN	[(VE, 35), (BS, 40), (PR, 53), (BN, 68)]								68 <sub>WI</sub>						
VE	LJ	[(BS, 40), (PR, 53), (BN, 68), (MO, 79)]														
VE	MO	[(BS, 40), (PR, 53), (BN, 68), (MO, 79)]											79 <sub>VE</sub>			
BS	BU	[(PR, 53), (BN, 68), (MO, 79)]														
BS	WI	[(PR, 53), (BN, 68), (MO, 79)]														
BS	PR	[(PR, 53), (BN, 68), (MO, 79)]														
PR	BS	[(BN, 68), (MO, 79)]														
PR	WI	[(BN, 68), (MO, 79)]														
PR	BL	[(BN, 68), (MO, 79), (BL, 83)]			83 <sub>PR</sub>											
BN	WI	[(MO, 79), (BL, 83)]														
BN	MO	[(MO, 79), (BL, 83)]														
BN	LU	[(MO, 79), (BL, 83), (LU, 101)]							101 <sub>BN</sub>							
MO	VE	[(BL, 83), (LU, 101)]														
MO	BN	[(BL, 83), (LU, 101)]														
MO	PA	[(BL, 83), (LU, 101), (PA, 133)]						133 <sub>MO</sub>								
BL	PR	[(LU, 101), (PA, 133)]														
BL	AM	[(LU, 101), (AM, 127), (PA, 133)]		127 <sub>BL</sub>												
LU	BN	[(AM, 127), (PA, 133)]														
LU	AM	[(AM, 127), (PA, 133)]														
LU	BX	[(AM, 127), (PA, 133), (BX, 150)]	150 <sub>LU</sub>													
LU	PA	[(AM, 127), (PA, 133), (BX, 150)]														
AM	BL	[(PA, 133), (BX, 150)]														
AM	LU	[(PA, 133), (BX, 150)]														
AM	BX	[(PA, 133), (BX, 150)]														
PA	MO	[(BX, 150)]														
PA	LU	[(BX, 150)]														
PA	BX	[(BX, 150)]														
BX	AM	[]														
BX	LU	[]														
BX	PA	[]														

Tabela 28: Potek izvajanja algoritma za nalogo 5.23.

### Naloga 5.24.

- (a) Trditev ni resnična. Protiprimer je podan na sliki 93, ki z rdečo barvo prikazuje drevo najkrajših poti iz vozlišča  $a$  – to ne vsebuje najkrajše povezave  $bc$ .
- (b) Trditev je resnična. Da jo dokažemo, predpostavimo, da vpeto drevo  $T$  ne vsebuje najkrajše povezave  $e$ . Potem graf  $T + e$  vsebuje cikel, ki vsebuje povezavo  $e' \neq e$ . Po predpostavki je povezava  $e'$  daljša od povezave  $e$ , tako da je graf  $T + e - e'$  vpeto drevo v grafu  $G$  z manjšo dolžino kot  $T$ . Drevo  $T$  tako ne more biti minimalno vpeto drevo.
- (c) Očitno je, da je vsako vpeto drevo v  $G'$  tudi vpeto drevo v  $G$ . Naj bo  $T$  drevo, ki je vpeto v grafu  $G$ , ne pa tudi v grafu  $G'$ . Drevo  $T$  potem vsebuje povezavo  $e$ . Naj bo  $e'$  povezava iz  $C - e$  – po predpostavki je  $T' = T - e + e'$  vpeto drevo v  $G$ , katerega dolžina ni večja kot dolžina drevesa  $T$ . Ker pa  $T'$  ne vsebuje povezave  $e$ , je tudi vpeto drevo v grafu  $G'$ . Dolžina minimalnega vpetega



Slika 93: Protiprimer za nalogi 5.24(a) in 5.28(a, c).

drevesa  $T_{\min}$  v  $G'$  tako ne presega dolžine nobenega vpetega drevesa v  $G$ , zaradi česar lahko sklenemo, da je  $T_{\min}$  tudi minimalno vpeto drevo v  $G$ .

### Naloga 5.25.

Sledili bomo naslednjemu algoritmu.

```

function FLOYDWARSHALL( $G = (V, E), L : E \rightarrow \mathbb{R}$ )
   $d \leftarrow$  slovar z vrednostjo  $\infty$  za vsak par vozlišč  $u, v \in V$ 
   $pred \leftarrow$  slovar z vrednostjo NULL za vsak par vozlišč  $u, v \in V$ 
  for  $u \in V$  do
     $d[u, u] \leftarrow 0$ 
  end for
  for  $uv \in E$  do
     $d[u, v] \leftarrow L_{uv}$ 
     $pred[u, v] \leftarrow u$ 
  end for
  for  $w \in V$  do
    for  $u \in V$  do
      if  $d[u, w] + d[w, u] < 0$  then
        return graf ima negativen cikel
      end if
      for  $v \in V$  do
         $\ell \leftarrow d[u, w] + d[w, v]$ 
        if  $\ell < d[u, v]$  then
           $d[u, v] \leftarrow \ell$ 
           $pred[u, v] \leftarrow pred[w, v]$ 
        end if
      end for
    end for
  end for
  return ( $d, pred$ )
end function

```

Po vsakem obhodu zunanje zanke **for** slovar  $d$  vsebuje dolžine tistih najkrajših poti, ki v svoji notranjosti (tj., brez začetnega in končnega vozlišča) obiščejo le tista vozlišča, ki jih je ta zanka že obravnavala. Algoritem pri tem preverja, ali je našel zanko negativne dolžine – tedaj sklene, da v grafu obstaja negativen

$w$	$u$	$v$	$d[uv]$	$pred[u, v]$	$w$	$u$	$v$	$d[uv]$	$pred[u, v]$
	$a$	$b$	3	$a$	$e$	$a$	$g$	13	$e$
	$a$	$f$	6	$a$	$e$	$b$	$g$	10	$e$
	$b$	$c$	7	$b$	$e$	$c$	$g$	12	$e$
	$b$	$e$	9	$b$	$e$	$d$	$g$	19	$e$
	$c$	$d$	-7	$c$	$g$	$b$	$f$	9	$g$
	$d$	$b$	9	$d$	$g$	$c$	$f$	11	$g$
	$d$	$h$	4	$d$	$g$	$d$	$f$	18	$g$
	$e$	$d$	1	$e$	$g$	$e$	$f$	0	$g$
	$e$	$g$	1	$e$	$g$	$h$	$f$	-6	$g$
	$f$	$g$	8	$f$	$h$	$a$	$f$	1	$g$
	$g$	$f$	-1	$g$	$h$	$a$	$g$	2	$h$
	$h$	$g$	-5	$h$	$h$	$b$	$f$	-2	$g$
$b$	$a$	$c$	10	$b$	$h$	$b$	$g$	-1	$h$
$b$	$a$	$e$	12	$b$	$h$	$c$	$f$	-9	$g$
$b$	$d$	$c$	16	$b$	$h$	$c$	$g$	-8	$h$
$b$	$d$	$e$	18	$b$	$h$	$d$	$f$	-2	$g$
$c$	$a$	$d$	3	$c$	$h$	$d$	$g$	-1	$h$
$c$	$b$	$d$	0	$c$	$h$	$e$	$f$	-1	$g$
$d$	$a$	$h$	7	$d$	$h$	$e$	$g$	0	$h$
$d$	$b$	$h$	4	$d$					
$d$	$c$	$b$	2	$d$					
$d$	$c$	$e$	11	$b$					
$d$	$c$	$h$	-3	$d$					
$d$	$e$	$b$	10	$d$					
$d$	$e$	$c$	17	$b$					
$d$	$e$	$h$	5	$d$					

Tabela 29: Potek izvajanja algoritma za nalogo 5.25.

cikel. Če graf nima negativnih ciklov, po izhodu iz zunanje zanke slovar  $d$  vsebuje dolžine najkrajših poti med vsemi pari vozlišč. Časovna zahtevnost algoritma je  $O(n^3)$ , kjer je  $n$  število vozlišč grafa.

Potek zgornjega algoritma na grafu s slike 15 je prikazan v tabeli 29 (prikazani so samo koraki, ko se katera od vrednosti v slovarju  $d$  spremeni). Izračunane razdalje so povzete v tabeli 30, v kateri je prikazano tudi, preko katerega vozlišča gre pot, preden obiše ciljno vozlišče (prazni vnosi pomenijo, da ustrezná pot ne obstaja).

#### Naloga 5.26.

Dano cestno omrežje lahko predstavimo kot usmerjen graf  $G$ , kjer uteži predstavljajo čas potovanja po cestni povezavi. Na grafu  $G$  lahko uporabimo Floyd-



	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>
<i>a</i>	0	$3_a$	$10_b$	$3_c$	$12_b$	$1_g$	$2_h$	$7_d$
<i>b</i>		0	$7_b$	$0_c$	$9_b$	$-2_g$	$-1_h$	$4_d$
<i>c</i>		$2_d$	0	$-7_c$	$11_b$	$-9_g$	$-8_h$	$-3_d$
<i>d</i>		$9_d$	$16_b$	0	$18_b$	$-2_g$	$-1_h$	$4_d$
<i>e</i>		$10_d$	$17_b$	$1_e$	0	$-1_g$	$0_h$	$5_d$
<i>f</i>						0	$8_f$	
<i>g</i>						$-1_g$	0	
<i>h</i>						$-6_g$	$-5_h$	0

Tabela 30: Najkrajše razdalje za nalogo 5.25.

Warshallov algoritem, opisan v rešitvi naloge 5.25, za izračun razdalje med vsemi pari vozlišč. Nato zgradimo dvodelen graf  $H$ , v katerem imamo po eno vozlišče za vsakega ponesrečenca in po  $\lceil \frac{n}{k} \rceil$  vozlišč za vsako bolnišnico, vozlišči pa sta povezani, če je razdalja od ponesrečenca do bolnišnice v grafu  $G$  največ pol ure. Na grafu  $H$  nato z madžarsko metodo poiščemo največje prirejanje  $M$ . Če ima najdeno prirejanje  $n$  povezav, potem smo našli ustrezno dodelitev ponesrečencev bolnišnicam, sicer pa ta ne obstaja.

#### Naloga 5.27.

- (a) Naj bo  $G = (V, E)$  graf, v katerem iščemo iskano rešitev. Za iskanje najdražje poti v grafu bomo uporabili predelano različico algoritma BELLMANFORD iz rešitve naloge 5.13. Za razliko od algoritma BELLMANFORD naš algoritem za namen problema shranjuje predhodnike za vsako najkrajšo pot, najdeno v posameznem obhodu glavne zanke, te pa uporabi pri iskanju poti, ki jo na koncu tudi sestavi s pomočjo pomožne funkcije REKONSTRUIRAJPOT. V algoritmu že pred izvajanjem zank nastavimo začetne vrednosti spremenljivk *konec* in *globina*, ki jih med samim algoritmom posodabljam v primeru najdenega večjega zaslužka.

```

function MAKSIMALNIDOBICEK( $G = (V, E), s \in V, L : E \rightarrow \mathbb{R}$ )
   $d[0, \dots, |V|] \leftarrow$  seznam slovarjev z vrednostjo  $-\infty$  za vsako vozlišče  $v \in V$ 
   $pred[1, \dots, |V|] \leftarrow$  seznam slovarjev z vrednostjo NULL za vsako vozlišče  $v \in V$ 
   $d[0][s] \leftarrow 0$ 
   $i \leftarrow 0$ 
   $trenutna \leftarrow \{s\}$ 
   $konec, globina \leftarrow s, 0$                                 nastavimo začetne vrednosti
  while  $\neg trenutna.isEmpty()$  do
     $i \leftarrow i + 1$ 
    if  $i > |V|$  then                                       obstaja pozitiven cikel
       $w \leftarrow trenutna.pop()$                             z  $w$  označimo konec poti, ki vsebuje cikel
       $pot \leftarrow \text{REKONSTRUIRAJPOT}(pred, w, |V|)$         naredimo pot do  $w$ 
       $konec \leftarrow \text{NULL}$ 
       $pregledani \leftarrow$  prazen slovar

```

```

for  $j = 0, 1, \dots, |V|$  do           v poti poiščemo vozlišče, ki se pojavi dvakrat
     $u \leftarrow \text{pot}[j]$ 
    if  $u \in \text{pregledani}$  then           najdena ponovitev
         $h \leftarrow \text{pregledani}[u]$        ločimo pot na pot do cikla in cikel
        return  $(\infty, \text{pot}[0, 1, \dots, h], \text{pot}[h + 1, h + 2, \dots, j])$ 
    end if
     $\text{pregledani}[u] \leftarrow i$ 
end for
end if
 $\text{naslednja} \leftarrow$  prazna množica
for  $u \in V$  do
     $d[i][u] \leftarrow d[i - 1][u]$ 
     $\text{pred}[i][u] \leftarrow u$ 
end for
for  $u \in \text{trenutna}$  do
    for  $v \in \text{ADJ}(G, u)$  do
        if  $d[i][v] < d[i - 1][u] + L_{uv}$  then           najdemo večji zaslužek do  $v$ 
             $d[i][v] \leftarrow d[i - 1][u] + L_{uv}$ 
             $\text{pred}[i][v] \leftarrow u$ 
             $\text{naslednja.add}(v)$ 
            if  $d[i][v] > d[\text{globina}][\text{konec}]$  then
                 $\text{konec}, \text{globina} \leftarrow v, i$            najdemo nov največji zaslužek
            end if
        end if
    end for
end for
 $\text{trenutna} \leftarrow \text{naslednja}$ 
end while           v grafu ni pozitivnega cikla
return  $(d[\text{globina}][\text{konec}], \text{REKONSTRUIRAJPOT}(\text{pred}, \text{konec}, \text{globina}), \text{NULL})$ 
end function

```

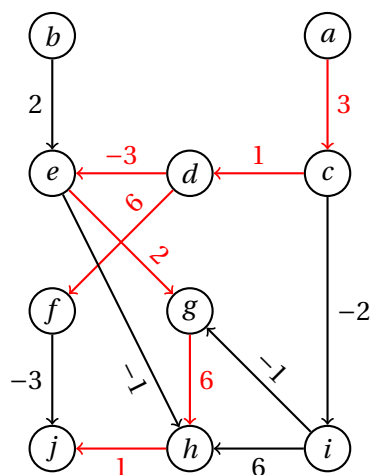
Zapišimo še nekoliko prirejeno funkcijo REKONSTRUIRAJPOT, ki deluje podobno kot istoimenska funkcija v rešitvi naloge 5.14.

```

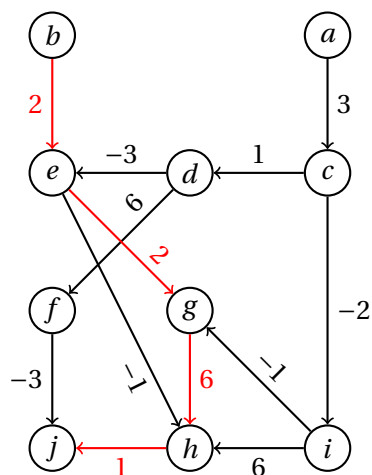
function REKONSTRUIRAJPOT( $\text{pred}, t, i$ )
     $u \leftarrow t$ 
     $\text{pot} \leftarrow [t]$ 
    while  $i \geq 0$  do
        if  $u \neq \text{pred}[i][u]$  then
             $u \leftarrow \text{pred}[i][u]$ 
             $\text{pot.append}(u)$ 
        end if
         $i \leftarrow i - 1$ 
    end while
     $\text{pot.reverse}()$ 
    return  $\text{pot}$ 
end function

```

Časovna zahtevnost algoritma MAKSIMALNIDOBICEK je kot pri navadnem Bellman-Fordovem algoritmu enaka  $O(mn)$ , kjer je  $n$  število vozlišč in  $m$  število povezav grafa.



Slika 94: Rešitvi pri začetnem stanju  $a$  za nalogo 5.27.



Slika 95: Rešitev pri začetnem stanju  $b$  za nalogo 5.27.

- (b) Lahko se prepričamo, da je graf s slike 22 acikličen. Če torej na njem izvedemo algoritem iz točke (a), bo ta vrnil končno pot, ki torej ne predstavlja trajnega dobičkonosnega poslovanja.

V primeru, da začnemo v stanju  $a$ , nam algoritem vrne pot  $a - c - d - f$  in dobiček 10, enak dobiček pa dosežemo tudi s potjo  $a - c - d - e - g - h - j$ . V primeru, da začnemo v stanju  $b$ , pa dobimo pot  $b - e - g - h - j$  in dobiček 11.

**Naloga 5.28.**

Trditvi (a) in (c) sta napačni. Protiprimer dobimo, če vzamemo graf s slike 93 ter množici  $A = \{a, c\}$  in  $B = \{b\}$ . Potem je  $e = bc$ , toda nobena najkrajša pot med vozliščema  $a$  in  $b$  ne vsebuje povezave  $e$ .

Trditev (b) je pravilna, saj lahko vzamemo najkrajšo pot med krajiščema povezave  $e$ . Vsaka pot med tema vozliščema, ki ne vsebuje povezave  $e$ , namreč vsebuje vsaj eno povezavo, ki je vsaj tako dolga kot  $e$ , zaradi pozitivnosti dolžin povezav pa je taka pot vsaj tako dolga kot pot, ki sestoji samo iz povezave  $e$ .

**Naloga 5.29.**

Naj bodo  $(z_i, k_i)$  ( $i \leq n$ ) pari z začetnimi in končnimi urami izpitov. Zapišimo algoritem, ki reši dani problem – ta bo klical algoritma NAJDALJŠAPOT in REKONSTRUIRAJPOT iz rešitve naloge 5.14.

```

function NAJDALJŠESTRNJENAZAPOREDJE( $\{(z_i, k_i)\}_{i=1}^n$ )
     $V \leftarrow \{s, t, z_i, k_i \mid 1 \leq i \leq n\}$ 
     $E \leftarrow \{sz_i, z_ik_i, k_it \mid i \leq n\}$ 
    for  $i \in \{1, 2, \dots, n\}$  do
         $L_{sz_i} \leftarrow 0$ 
         $L_{z_ik_i} \leftarrow k_i - z_i$ 
         $L_{k_it} \leftarrow 0$ 
    end for
     $d_G, pred \leftarrow \text{NAJDALJŠAPOT}(G = (V, E), s, L)$ 
    return REKONSTRUIRAJPOT( $pred, t$ )
end function

```

Pri podanih podatkih algoritem najprej konstruira graf s slike 96, iz katerega je razvidno, da je ena od najdaljših poti od  $s$  do  $t$  dosežena s povezavami, ki ustrezajo intervalom (7, 10), (10, 13) in (13, 15).

**Naloga 5.30.**

Na grafu lahko ponovimo iskanje v širino ali iskanje v globino z začetkom v vsakem vozlišču, ter preverimo, ali smo vsakič dosegli vsa vozlišča. V obeh primerih je časovna zahtevnost postopka  $O(mn)$ , kjer je  $m$  število povezav in  $n$  število vozlišč v grafu.

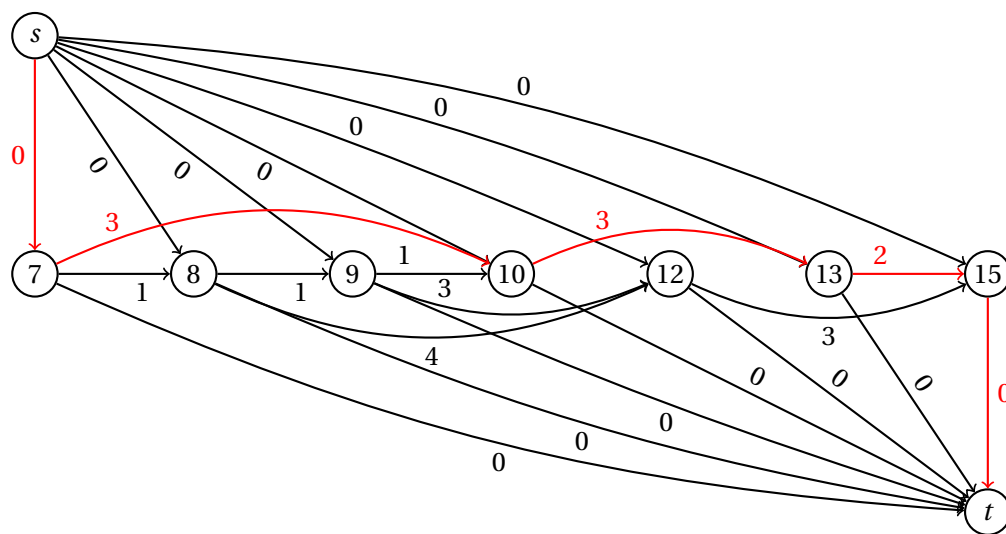
**Naloga 5.31.**

Zapišimo psevdokodo algoritma, ki kliče funkcijo TOPO iz naloge 5.14(a).

```

function ŠTEVILOVSEHPOTI( $G = (V, E), s$ )
     $C \leftarrow$  slovar z vrednostjo 0 za vsako vozlišče  $v \in V$ 
     $C[s] \leftarrow 1$ 
    for  $u \in \text{TOPO}(G)$  do
        for  $v \in \text{ADJ}(G, u)$  do
             $C[v] \leftarrow C[v] + C[u]$ 
        end for
    end for
    return  $C$ 
end function

```



Slika 96: Graf za nalogo 5.29.

$u$	$s$	$a$	$b$	$c$	$d$	$t$
	1	0	0	0	0	0
$s$	1	1	0	1	0	0
$a$	1	1	1	2	0	0
$b$	1	1	1	3	1	0
$c$	1	1	1	3	4	3
$d$	1	1	1	3	4	7

Tabela 31: Potek izvajanja algoritma za nalogo 5.31.

Potek izvajanja algoritma na grafu s slike 23 z začetnim vozliščem  $s$ , pri čemer je uporabljena topološka ureditev  $s, a, b, c, d, t$ , je prikazan v tabeli 31.

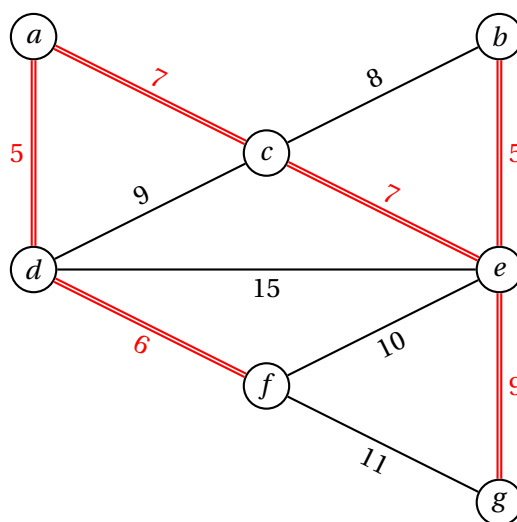
### Naloga 5.32.

S Primovim algoritmom bomo poiskali najlažje vpeto drevo v grafu s slike 24. Algoritem deluje tako, da začne z drevesom z enim samim (poljubno izbranim) vozliščem, nato pa v vsakem koraku v drevo doda najcenejšo izmed povezav, ki imajo natanko eno krajišče v drevesu, dokler v drevesu niso vsa vozlišča. Potek izvajanja algoritma je prikazan v tabeli 32, najcenejše vpeto drevo s ceno 39 pa je prikazano na sliki 97. Skupna cena gradnje je torej 39 milijonov evrov.

Če mora vsak most pridobiti še potrdilo inšpektorjev, se rešitev ne spremeni, saj ima vsako vpeto drevo enako število povezav. Cena gradnje se v tem primeru poveča na 45 milijonov evrov.

dodano vozlišče	teža drevesa	vrsta povezav
<i>a</i>	0	<i>ad</i> : 5, <i>ac</i> : 7
<i>d</i>	5	<i>df</i> : 6, <i>ac</i> : 7, <i>dc</i> : 9, <i>de</i> : 15
<i>f</i>	11	<i>ac</i> : 7, <i>dc</i> : 9, <i>fe</i> : 10, <i>fg</i> : 11, <i>de</i> : 15
<i>c</i>	18	<i>ce</i> : 7, <i>cb</i> : 8, <i>fe</i> : 10, <i>fg</i> : 11, <i>de</i> : 15
<i>e</i>	25	<i>eb</i> : 5, <i>cb</i> : 8, <i>eg</i> : 9, <i>fg</i> : 11
<i>b</i>	30	<i>eg</i> : 9, <i>fg</i> : 11
<i>g</i>	39	

Tabela 32: Potek izvajanja algoritma za nalogo 5.32.



Slika 97: Najcenejše vpeto drevo za nalogo 5.32.

### Naloga 5.33.

Ker v grafu s slike 25 ni negativnih uteži, bomo najcenejšo pot poiskali z Dijkstrovim algoritmom. Potek algoritma je prikazan v tabeli 33, pri čemer so uporabljene oznake iz algoritma DIJKSTRA iz naloge 5.8. Iz končnega stanja lahko razberemo, da je najcenejša pot  $s - a - c - d - t$  s ceno 13.

### Naloga 5.34.

- (a) Skonstruiramo graf, v katerem vsakemu delegatu priredimo vozlišče; če delegata  $X$  in  $Y$  govorita skupni jezik pri stopnjah obvladovanja  $s_X$  in  $s_Y$ , med njima postavimo povezavo z utežjo  $\max\{s_X, s_Y\}$  (če govorita več skupnih jezikov, potem zadostuje ena sama povezava med  $X$  in  $Y$ , in sicer tista z najmanjšo utežjo). Na tem grafu nato poiščemo najcenejšo pot od  $A$  do  $B$ , kar lahko storimo z Dijkstrovim algoritmom, saj so vse uteži nenegativne.

$u$	$v$	$Q$	$s$	$a$	$b$	$c$	$d$	$e$	$f$	$t$
		$s:0$	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
$s$	$a$	$a:2$	0	$2_s$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
$s$	$c$	$a:2, c:5$	0	$2_s$	$\infty$	$5_s$	$\infty$	$\infty$	$\infty$	$\infty$
$s$	$e$	$a:2, e:4, c:5$	0	$2_s$	$\infty$	$5_s$	$\infty$	$4_s$	$\infty$	$\infty$
$a$	$b$	$e:4, c:5, b:14$	0	$2_s$	$14_a$	$5_s$	$\infty$	$4_s$	$\infty$	$\infty$
$a$	$c$	$c:4, e:4, b:14$	0	$2_s$	$14_a$	$4_a$	$\infty$	$4_s$	$\infty$	$\infty$
$a$	$d$	$c:4, e:4, d:9, b:14$	0	$2_s$	$14_a$	$4_a$	$9_a$	$4_s$	$\infty$	$\infty$
$c$	$d$	$e:4, d:8, b:14$	0	$2_s$	$14_a$	$4_a$	$8_c$	$4_s$	$\infty$	$\infty$
$c$	$e$	$e:4, d:8, b:14$	0	$2_s$	$14_a$	$4_a$	$8_c$	$4_s$	$\infty$	$\infty$
$c$	$f$	$e:4, f:7, d:8, b:14$	0	$2_s$	$14_a$	$4_a$	$8_c$	$4_s$	$7_c$	$\infty$
$e$	$f$	$f:7, d:8, b:14$	0	$2_s$	$14_a$	$4_a$	$8_c$	$4_s$	$7_c$	$\infty$
$f$	$d$	$d:8, b:14$	0	$2_s$	$14_a$	$4_a$	$8_c$	$4_s$	$7_c$	$\infty$
$f$	$t$	$d:8, b:14, t:14$	0	$2_s$	$14_a$	$4_a$	$8_c$	$4_s$	$7_c$	$14_f$
$d$	$t$	$t:13, b:14$	0	$2_s$	$14_a$	$4_a$	$8_c$	$4_s$	$7_c$	$13_d$
$t$	$b$	$b:14$	0	$2_s$	$14_a$	$4_a$	$8_c$	$4_s$	$7_c$	$13_d$

Tabela 33: Potek izvajanja algoritma za nalogo 5.33.

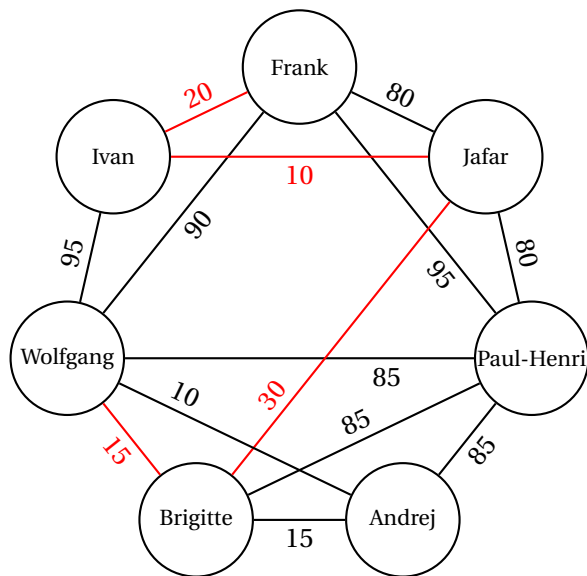
Frank	Ivan	Paul-Henri	Brigitte	Andrej	Wolfgang	Jafar
*	$20_{\text{Frank}}$	$95_{\text{Frank}}$			$90_{\text{Frank}}$	$80_{\text{Frank}}$
	*					$30_{\text{Ivan}}$
			$60_{\text{Jafar}}$			*
			*	$75_{\text{Brigitte}}$	$75_{\text{Brigitte}}$	
				*		
					*	
		*				

Tabela 34: Potek izvajanja algoritma za nalogo 5.34(b).

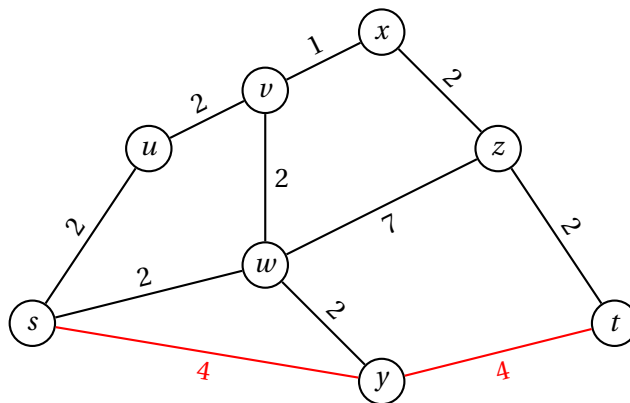
- (b) Po zgornjem postopku zgrajeni graf je prikazan na sliki 98. Postopek izvajanja Dijkstrovega algoritma z začetkom pri Franku je prikazan v tabeli 34, iz katere je razvidno, da bo Frankovo sporočilo najhitreje (v 75 časovnih enotah) doseglo Wolfganga, če ga v španščini preda Ivanu, ta ga v ruščini preda Jafarju, ta ga v nizozemščini preda Brigitte, ki ga nazadnje v nemščini preda Wolfgangu.

### Naloga 5.35.

- (a) Problem lahko prevedemo na problem iskanja najcenejše poti od  $s$  do  $t$  v grafu, kjer je vsaka povezava utežena s številom, ki je za 1 večje od števila



Slika 98: Graf za nalogo 5.34.



Slika 99: Graf in najkrajša pot za nalogo 5.35.

rožic na tej povezavi. Ker so uteži nenegativne, lahko za reševanje uporabimo Dijkstrov algoritem.

- (b) Postopek izvajanja Dijkstrovega algoritma na grafu s slike 99 z začetkom pri  $s$  je prikazan v tabeli 35, iz katere je razvidno, da je najkrajša pot  $s - y - t$ , zanjo pa Gaber potrebuje 8 časovnih enot.

### Naloga 5.36.

- (a) Definirajmo graf  $G = (V, E)$ , katerega množica vozlišč  $V \subset \mathbb{R}^2$  je enaka vhodnemu seznamu koordinat, za par različnih vozlišč  $v_i, v_j \in V$  pa velja  $(v_i, v_j) \in$



$s$	$u$	$v$	$w$	$x$	$y$	$z$	$t$
*	$2_s$		$2_s$		$4_s$		
	*	$4_u$					
		*				$9_w$	
		*		$5_v$			
				*			$8_y$
				*		$7_x$	
						*	
							*

Tabela 35: Potek izvajanja algoritma za nalogo 5.35(b).

$E$  natanko tedaj, ko  $\|v_i - v_j\| \leq d$ . Vozlišči sta torej povezani, če je razdalja med njima manjša od  $d$ , kar pomeni, da Rok lahko skoči iz ene na drugo. Sedaj moramo ugotoviti, če je točka, na kateri počiva Neli, dosegljiva Roku, za kar pa imamo na voljo algoritma DFS iz naloge 5.12 in BFS iz naloge 5.6.

Zapišimo kratek algoritem, pri čemer upoštevamo, da je  $s$  list, na katerem sedi Rok, in  $t$  list, na katerem sedi Neli.

```

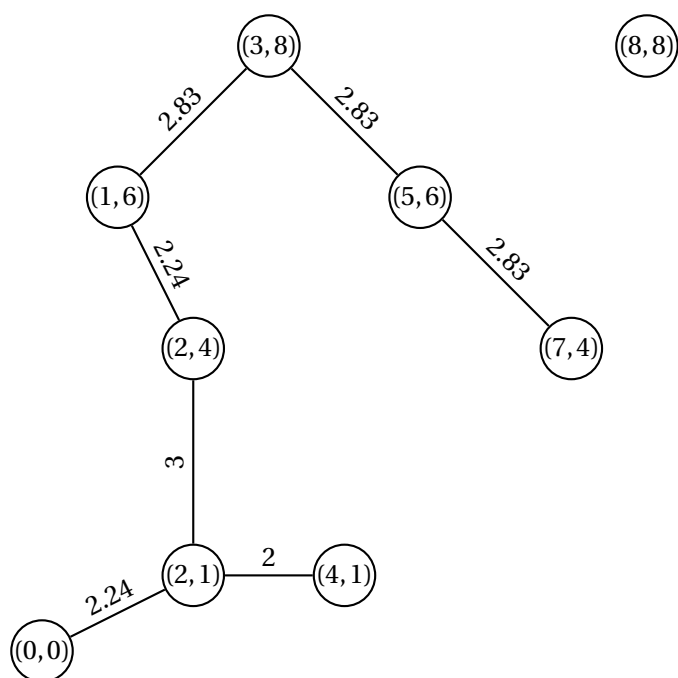
function ŽABE( $G = (V, E)$ ,  $s$ ,  $t$ )
  označen  $\leftarrow$  slovar s ključi  $v \in V$  in vrednostmi FALSE
  function VISIT( $u$ ,  $v$ )
    označen[ $u$ ]  $\leftarrow$  TRUE
  end function
  BFS( $G$ , { $s$ }, VISIT)
  return označen[ $t$ ]
end function

```

- (b) Po zgornjem opisu lahko torej povežemo dane točke v ravnini in dobimo graf  $G$ , prikazan na sliki 100. Graf je seveda neusmerjen, saj smo povezave definirali s simetrično relacijo. Za uteži smo vzeli razdalje med točkami, a jih za rešitev naloge ne bomo potrebovali. Sedaj s klicem  $\text{Ž}_{\text{ABE}}(G, (0, 0), (8, 8))$  ugotovimo, da Nelino vozlišče Roku ni dosegljivo.

### Naloga 5.37.

- (a) Na grafu  $G$  bomo s Primovim algoritmom poiskali najlažje vpeto drevo. Za priklop na zunanje omrežje bomo izbrali sobano, za katero je priklop najcenejši.
- (b) Potek izvajanja algoritma je prikazan v tabeli 36, najcenejše vpeto drevo s ceno 17 guldo pa je prikazano na sliki 101. Za priklop na zunanje omrežje bomo uporabili sobano  $g$ , kjer priklop stane 3 gulde. Skupna cena je torej 20 guldo.



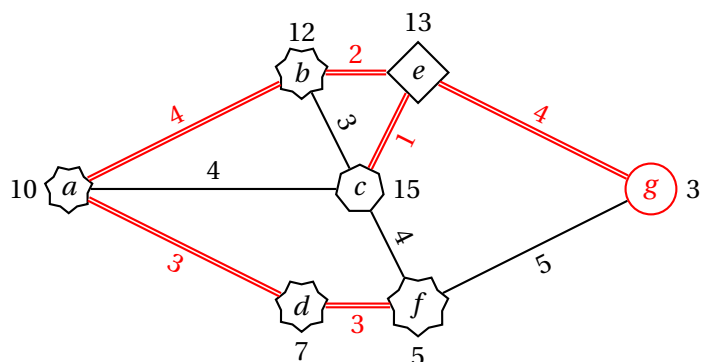
Slika 100: Graf dopustnih skokov za nalogo 5.36.

dodano vozlišče	teža drevesa	vrsta povezav
<i>a</i>	0	$ad : 3, ab : 4, ac : 4$
<i>d</i>	3	$df : 3, ab : 4, ac : 4$
<i>f</i>	6	$ab : 4, ac : 4, fc : 4, fg : 5$
<i>b</i>	10	$be : 2, bc : 3, ac : 4, fc : 4, fg : 5$
<i>e</i>	12	$ec : 1, bc : 3, ac : 4, eg : 4, fc : 4, fg : 5$
<i>c</i>	13	$eg : 4, fg : 5$
<i>g</i>	17	

Tabela 36: Potek izvajanja algoritma za nalogo 5.37.

### Naloga 5.38.

- V tabeli 37 je prikazan vrstni red barvanja vozlišč v rumeno barvo, pri čemer se obarva vozlišče  $v$  ali  $u$ . Zabeleženo je tudi, kdaj se spremeni vrednost števca  $s$ .
- Algoritem vrača število povezanih komponent grafa. Izvaja namreč pregled v širino, pri čemer se števec poveča vsakič, ko naleti na vozlišče, ki ga ni dosegel po predhodnih povečevanjih števca.



Slika 101: Najcenejše vpeto drevo za nalogo 5.37(b).

$s$	$v$	$w$	$u$	$Q$
1	$a$			$[a]$
		$a$	$e$	$[e]$
2	$b$			$[b]$
		$b$	$c$	$[c]$
			$d$	$[c, d]$

Tabela 37: Potek izvajanja algoritma za nalogo 5.38.

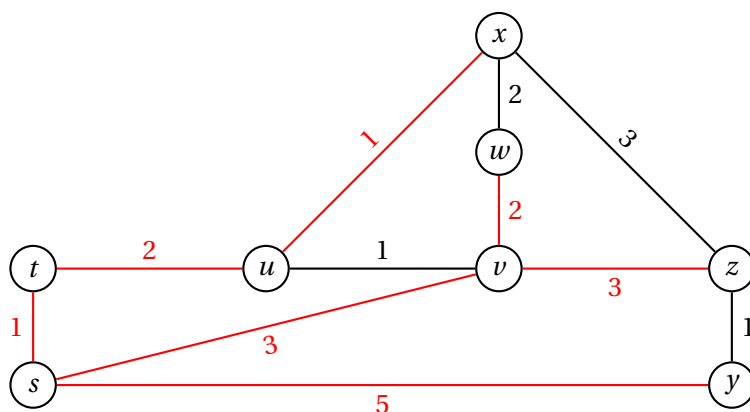
- (c) Časovna zahtevnost algoritma je  $O(n + m)$ , kjer je  $n$  število vozlišč,  $m$  pa število povezav podanega grafa. Algoritem namreč z zunanjo zanko obišče vsako vozlišče enkrat, z notranjo zanko pa vsako povezavo dvakrat.

### Naloga 5.39.

- (a) Problem lahko prevedemo na problem iskanja najcenejših poti od vozlišča  $s$  do ostalih vozlišč v grafu, kjer je vsaka povezava utežena s številom, ki je za 1 večje od uteži na ustrezni povezavi v grafu  $G$ . Vrnemo vozlišče, do katerega je najdena pot najdražja.

Ker so uteži nenegativne, lahko za reševanje uporabimo Dijkstrov algoritem. Če za prednostno vrsto uporabimo seznam, potem ta teče v času  $O(n^2)$  (v primeru, ko je število povezav  $m = O\left(\frac{n^2}{\log n}\right)$ , potem bo hitrejša različica algoritma, pri kateri za prednostno vrsto uporabimo kopico, saj ta teče v času  $O(m \log n)$ ).

- (b) Postopek izvajanja Dijkstrovega algoritma na grafu s slike 102 z začetkom pri  $s$  je prikazan v tabeli 38, iz katere je razvidno, da bo informacijo nakasneje dobil računalnik  $z$ , in sicer po 6 ms.



Slika 102: Graf in drevo najkrajših poti za nalogo 5.39.

$s$	$t$	$u$	$v$	$w$	$x$	$y$	$z$
*	$1_s$		$3_s$			$5_s$	
	*	$3_t$					
		*			$4_u$		
			*	$5_v$			$6_v$
				*			
				*			
					*		
						*	

Tabela 38: Potek izvajanja algoritma za nalogo 5.39(b).

#### Naloga 5.40.

- (a) Denimo, da za vsako povezavo  $e \in E$  poznamo njeno utež  $w_e$ . Naj bo  $x_u$  ( $u \in V$ ) največji dobiček, ki ga lahko Marco doseže, če svojo pot konča v vozlišču  $u$ . Zapišimo rekurzivne enačbe.

$$x_u = \max(\{0\} \cup \{x_v + w_{vu} \mid vu \in E\}) \quad (u \in V)$$

Vrednosti  $x_u$  računamo glede na topološko ureditev grafa. Optimalni dobiček dobimo kot  $x^* = \max\{x_u \mid u \in V\}$ , optimalno pot pa lahko rekonstruiramo glede na vrednosti, ki smo jih uporabili pri računanju  $x^*$ . Časovna zahtevnost algoritma je  $O(n + m)$ , kjer je  $n$  število vozlišč,  $m$  pa število povezav podanega grafa.

- (b) Izračunajmo vrednosti  $x_u$ . Uporabili bomo abecedni vrstni red vozlišč, za katerega opazimo, da ustreza topološki ureditvi.

$$\begin{aligned}
 x_a &= 0 \\
 x_b &= \max\{0, 0 + (-1)\} = 0 \\
 x_c &= \max\{0, 0 + 1\} = 1 \\
 x_d &= \max\{0, 0 + 6\} = 6 \\
 x_e &= \max\{0, 0 + 10, 6 + 5\} = 11 \\
 x_f &= \max\{0, 0 + 6, 1 + 5\} = 6 \\
 x_g &= \max\{0, 1 + 10, 6 + 5\} = 11 \\
 x_h &= \max\{0, 6 + 5, 11 + 2\} = 13 \\
 x_i &= \max\{0, 11 + (-2), 6 + 5\} = 11 \\
 x_j &= \max\{0, 13 + (-5), 11 + (-2)\} = 9 \\
 x_k &= \max\{0, 9 + 8\} = 17 \\
 x_\ell &= \max\{0, 11 + 3, 11 + 1\} = 14
 \end{aligned}$$

Največji dobiček, ki si ga lahko obeta Marco, je torej  $x^* = 17$ . Rekonstruirajmo pot, ki nam da tak dobiček.

$$\begin{array}{ll}
 x^* = x_k = x_j + w_{jk} & \text{gre po povezavi } jk \\
 x_j = x_i + w_{ij} & \text{gre po povezavi } ij \\
 x_i = x_f + w_{fi} & \text{gre po povezavi } fi \\
 x_f = x_c + w_{cf} & \text{gre po povezavi } cf \\
 x_c = x_a + w_{ac} & \text{gre po povezavi } ac \\
 x_a = 0 & \text{začne v vozlišču } a
 \end{array}$$

Optimalna pot je torej  $a \rightarrow c \rightarrow f \rightarrow i \rightarrow j \rightarrow k$ . Ker velja tudi  $x_f = x_b + w_{bf}$  in  $x_b = 0$ , je druga optimalna pot  $b \rightarrow f \rightarrow i \rightarrow j \rightarrow k$ .

#### Naloga 5.41.

- (a) Na grafu s slike 31 s Primovim algoritmom poiščemo najcenejše vpeto drevo. Potek izvajanja algoritma je prikazan v tabeli 39, najcenejše vpeto drevo s ceno 87 pa je prikazano na sliki 103.
- (b) Poiskati želimo najcenejši gozd  $F$  na povezavah danega grafa, tako da v grafu obstaja povezava med vsakim parom povezanih komponent gozda  $F$ . Če v danem grafu skrbimo vse povezave iz  $F$ , bomo torej dobili poln graf s tolikšnim številom vozlišč, kolikor ima  $F$  povezanih komponent – tak poln graf je torej minor v danem grafu.

dodano vozlišče	teža drevesa	vrsta povezav
<i>a</i>	0	<i>ac</i> : 2, <i>ab</i> : 7, <i>ad</i> : 10
<i>c</i>	2	<i>ab</i> : 7, <i>cd</i> : 8, <i>ad</i> : 10, <i>cg</i> : 14
<i>b</i>	9	<i>bf</i> : 6, <i>cd</i> : 8, <i>ad</i> : 10, <i>be</i> : 10, <i>cg</i> : 14
<i>f</i>	15	<i>cd</i> : 8, <i>fe</i> : 8, <i>ad</i> : 10, <i>be</i> : 10, <i>cg</i> : 14, <i>fj</i> : 16
<i>d</i>	23	<i>de</i> : 5, <i>fe</i> : 8, <i>be</i> : 10, <i>cg</i> : 14, <i>fj</i> : 16
<i>e</i>	28	<i>eh</i> : 10, <i>cg</i> : 14, <i>fj</i> : 16, <i>ej</i> : 18
<i>h</i>	38	<i>hi</i> : 6, <i>hg</i> : 12, <i>hk</i> : 12, <i>cg</i> : 14, <i>fj</i> : 16, <i>ej</i> : 18
<i>i</i>	44	<i>ik</i> : 8, <i>hg</i> : 12, <i>hk</i> : 12, <i>cg</i> : 14, <i>ij</i> : 15, <i>fj</i> : 16, <i>ej</i> : 18
<i>k</i>	52	<i>kg</i> : 10, <i>hg</i> : 12, <i>cg</i> : 14, <i>ij</i> : 15, <i>kl</i> : 15, <i>fj</i> : 16, <i>ej</i> : 18
<i>g</i>	62	<i>ij</i> : 15, <i>kl</i> : 15, <i>fj</i> : 16, <i>ej</i> : 18
<i>j</i>	77	<i>jℓ</i> : 10, <i>kℓ</i> : 15
<i>ℓ</i>	87	

Tabela 39: Potek izvajanja algoritma za nalogo 5.41(a).

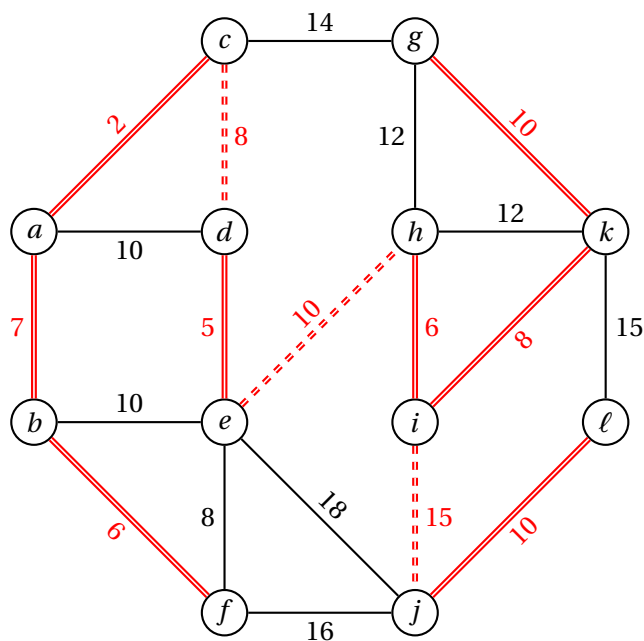
Ker je graf s slike 31 ravninski in torej ne vsebuje minorja  $K_5$ , bo iskani gozd imel največ 4 povezane komponente. Iskanja optimalne rešitve se torej lahko lotimo tako, da največ trem povezavam spremenimo utež na 0 ter s Primovim algoritmom poiščemo najcenejše vpeto drevo v takih grafih (če se v njem pojavi povezava, ki smo ji utež nastavili na 0, smatramo, da te ceste ne bomo prenovili). Izkaže se, da je optimalna rešitev taka kot na sliki 103, pri čemer izpustimo povezave *cd*, *eh* in *ij*. Skupna cena obnove je v tem primeru 54.

#### Naloga 5.42.

Naj bo  $e = (s, t)$ . Če preverimo obstoj poti od  $s$  do  $t$ , ki ne vključuje povezave  $e$ , bomo s tem rešili nalogo, saj takšna pot obstaja natanko tedaj, ko je v grafu cikel, ki vsebuje povezavo  $e$ . Ideja bo torej uporabiti DFS algoritem iz naloge 5.12 na grafu  $G \setminus \{e\}$  z začetkom v  $s$  in tako preveriti, če lahko pridemo do  $t$ .

```

function CIKEL( $G = (V, E)$ ,  $e = (s, t)$ )
     $G' \leftarrow G \setminus \{e\}$ 
     $označeni \leftarrow$  slovar s ključi  $v \in V$  in vrednostmi FALSE
    function VISIT( $u, v$ )
         $označeni[u] \leftarrow$  TRUE
    end function
    DFS( $G', \{s\}, VISIT, NOOP$ )
    return  $označeni[t]$ 
end function
```



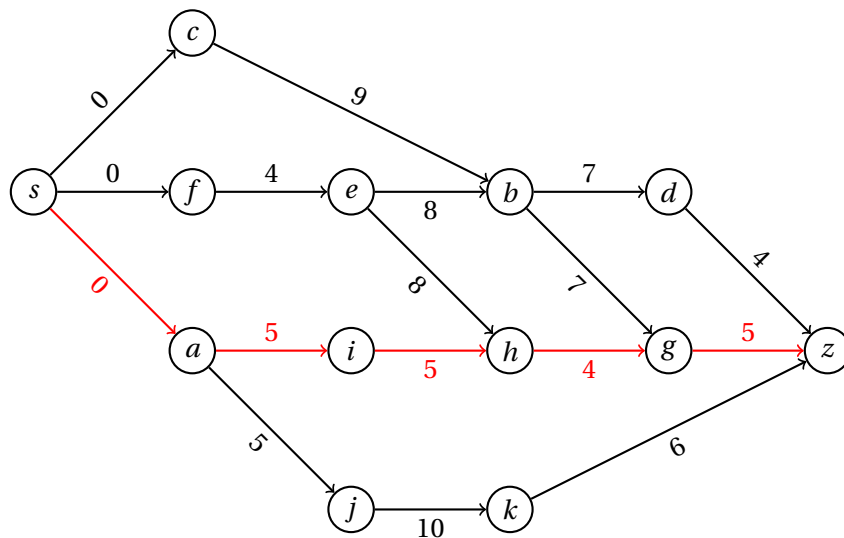
Slika 103: Najcenejše vpeto drevo za nalogo 5.41.

$s$	$c$	$f$	$a$	$e$	$i$	$j$	$b$	$h$	$k$	$d$	$g$	$z$
0	$0_s$	$0_s$	$0_s$	$4_f$	$5_a$	$5_a$	$9_c$	$10_i$	$15_j$	$16_b$	$14_h$	$19_g$

Tabela 40: Izračunane vrednosti v algoritmu za nalogo 5.43(b).

### Naloga 5.43.

- Ustrezni graf je prikazan na sliki 104, iz katere je razvidna topološka ureditev  $s, c, f, a, e, i, j, b, h, k, d, g, z$ . Vozlišče  $s$  je bilo dodano kot izhodiščna točka s povezavami dolžine 0 do predmetov, h katerim lahko Peter pristopi takoj.
- S pomočjo algoritma NAJKRAJŠAPOT iz naloge 5.14(b) poiščemo razdalje od vozlišča  $s$  do ostalih vozlišč grafa – te so zbrane v tabeli 40. Iz izhoda lahko rekonstruiramo najkrajšo pot  $s - a - i - h - g - z$  dolžine 19, ki je prikazana tudi na sliki 104. Peter naj torej zaporedoma opravi predmete  $a, i, h$  in  $g$ , kar mu bo omogočilo, da po 19 tednih pristopi k zaključnemu izpitu.



Slika 104: Graf in najkrajša pot za nalogo 5.43.

**Naloga 5.44.**

(a) Definirali bomo usmerjen graf  $G' = (V', E')$ , kjer je

$$\begin{aligned}
 V' &= \{s, t\} \cup \{v_A, v_B \mid v \in V \setminus \{s, t\}\} \quad \text{in} \\
 E' &= \{u_X v_Y \mid uv \in X \setminus Y, X, Y \in \{A, B\}, u, v \notin \{s, t\}\} \\
 &\quad \cup \{sv_Y \mid sv \in E \setminus Y, Y \in \{A, B\}, v \notin \{s, t\}\} \\
 &\quad \cup \{u_X t \mid ut \in X, X \in \{A, B\}, u \notin \{s, t\}\} \\
 &\quad \cup \{st \mid st \in E\}.
 \end{aligned}$$

Cene povezav ohranimo – velja torej

$$\begin{aligned}
 L'_{u_X v_Y} &= L_{uv} & (u_X v_Y \in E'), \\
 L'_{sv_Y} &= L_{sv} & (sv_Y \in E'), \\
 L'_{u_X t} &= L_{ut} & (u_X t \in E'), \\
 L'_{st} &= L_{st} & (st \in E').
 \end{aligned}$$

Z Dijkstrovim algoritmom lahko poiščemo najkrajšo pot od  $s$  do  $t$  v grafu  $G'$ , ki bo oblike  $s - w_{X_1}^{(1)} - w_{X_2}^{(2)} - \dots - w_{X_k}^{(k)} - t$ , kjer velja  $X_i \in \{A, B\}$  ( $1 \leq i \leq k$ ) in  $X_{i+1} \neq X_i = X_{i+2}$  ( $1 \leq i \leq k-2$ ). Iz te poti lahko izluščimo najkrajšo alternirajočo pot  $s - w^{(1)} - w^{(2)} - \dots - w^{(k)} - t$  v grafu  $G$ .

Graf  $G'$  ima  $2n-2$  vozlišč in  $m$  povezav, kjer je  $n$  število vozlišč in  $m$  število povezav v grafu  $G$ . Če v Dijkstrovem algoritmu za prednostno vrsto uporabimo kopico, je časovna zahtevnost algoritma tako  $O(m \log n)$ .



Trenutno vozlišče	$s$	$r_A$	$r_B$	$u_A$	$u_B$	$v_A$	$v_B$	$w_A$	$w_B$	$t$
	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
$s$	0	$\infty$	$4_s$	$10_s$	$\infty$	$\infty$	$\infty$	$1_s$	$\infty$	$12_s$
$w_A$	0	$\infty$	$3_{w_A}$	$10_s$	$\infty$	$\infty$	$2_{w_A}$	$1_s$	$\infty$	$12_s$
$v_B$	0	$4_{v_B}$	$3_{w_A}$	$10_s$	$\infty$	$\infty$	$2_{w_A}$	$1_s$	$\infty$	$12_s$
$r_B$	0	$4_{v_B}$	$3_{w_A}$	$10_s$	$\infty$	$5_{r_B}$	$2_{w_A}$	$1_s$	$\infty$	$10_{r_B}$
$r_A$	0	$4_{v_B}$	$3_{w_A}$	$10_s$	$\infty$	$5_{r_B}$	$2_{w_A}$	$1_s$	$6_{r_A}$	$10_{r_B}$
$v_A$	0	$4_{v_B}$	$3_{w_A}$	$10_s$	$7_{v_A}$	$5_{r_B}$	$2_{w_A}$	$1_s$	$6_{r_A}$	$10_{r_B}$
$w_B$	0	$4_{v_B}$	$3_{w_A}$	$9_{w_B}$	$7_{v_A}$	$5_{r_B}$	$2_{w_A}$	$1_s$	$6_{r_A}$	$10_{r_B}$
$u_B$	0	$4_{v_B}$	$3_{w_A}$	$9_{w_B}$	$7_{v_A}$	$5_{r_B}$	$2_{w_A}$	$1_s$	$6_{r_A}$	$10_{r_B}$
$u_A$	0	$4_{v_B}$	$3_{w_A}$	$9_{w_B}$	$7_{v_A}$	$5_{r_B}$	$2_{w_A}$	$1_s$	$6_{r_A}$	$10_{r_B}$
$t$	0	$4_{v_B}$	$3_{w_A}$	$9_{w_B}$	$7_{v_A}$	$5_{r_B}$	$2_{w_A}$	$1_s$	$6_{r_A}$	$10_{r_B}$

Tabela 41: Potek reševanja za nalogo 5.44(b).

- (b) Potek izvajanja Dijkstrovega algoritma na pomožnem grafu  $G'$  je prikazan v tabeli 41. Najkrajša pot v grafu  $G'$  je  $s - w_A - r_B - t$  dolžine 10, kar ustreza najkrajši alternirajoči poti  $s - w - r - t$  v grafu  $G$ .

#### Naloga 5.45.

- (a) Algoritem BFS iz naloge 5.6 vrne zeleni slovar. Ker lahko koren poljubno izberemo, lahko slovar  $pred$  dobimo s klicem  $BFS(T, V, \text{NOP})$ . Neposredni nasledniki vozlišča  $u$  v drevesu  $T$  so tisti njegovi sosedi  $v \in \text{ADJ}(T, u)$ , za katere velja  $v \neq pred[u]$ .
- (b) Naj bosta  $x_u$  in  $y_u$  teži najtežje neodvisne množice  $S$  v poddrevesu s korenem  $u$  (glede na slovar  $pred$ ), za katero velja  $u \in S$  oziroma  $u \notin S$ . Potem velja

$$x_u = c_u + \sum_{\substack{v \sim u \\ v \neq pred[u]}} y_v \quad \text{in} \\ y_u = \sum_{\substack{v \sim u \\ v \neq pred[u]}} \max\{x_v, y_v\}.$$

Vrednosti  $x_u$  in  $y_u$  računamo od listov v smeri proti korenu  $r$  (tj., v topološki ureditvi glede na  $pred$ ). Težo najtežje neodvisne množice dobimo kot  $c^* = \max\{x_r, y_r\}$ .

- (c) Da iz vrednosti  $x_u$  in  $y_u$  ( $u \in V$ ) izluščimo najtežjo neodvisno množico  $S$  v drevesu  $T$ , se sprehodimo od korena proti listom (npr. z iskanjem v širino) in posamezno vozlišče  $u$  dodamo v množico  $S$ , če njegov prednik ni v  $S$  in velja  $x_u \geq y_u$ .

```

function NAJTEŽJANEODVISNAMNOŽICA( $T = (V, E), r, (x_u)_{u \in V}, (y_u)_{u \in V}$ )
   $S \leftarrow$  prazna množica
  function VISIT( $u, v$ )
    if  $v \notin S \wedge x_u \geq y_u$  then
       $S.add(u)$ 
    end if
  end function
  BFS( $T, \{r\}, VISIT$ )
  return  $S$ 
end function

```

(d) Naj bo  $n$  število vozlišč drevesa  $T$ . Potem ima  $T$  natanko  $n - 1$  povezav. Za izračun najtežje neodvisne množice naredimo tri preglede v širino, pri čemer v vsakem vozlišču porabimo konstantno mnogo časa. Časovna zahtevnost celotnega algoritma je torej  $O(n)$ .

(e) Izračunajmo vrednosti  $x_u$  in  $y_u$  ( $u \in V$ ), če za koren vzamemo vozlišče  $a$ .

$x_i$	$= 9$	$y_i$	$= 0$
$x_j$	$= 4$	$y_j$	$= 0$
$x_k$	$= 2$	$y_k$	$= 0$
$x_d$	$= 1$	$y_d$	$= 0$
$x_e = 1 + 0 + 0$	$= 1$	$y_e = \max\{9, 0\} + \max\{4, 0\}$	$= 13$
$x_f$	$= 4$	$y_f$	$= 0$
$x_g = 8 + 0$	$= 8$	$y_g = \max\{2, 0\}$	$= 2$
$x_h$	$= 4$	$y_h$	$= 0$
$x_b = 1 + 0 + 13 + 0 = 14$		$y_b = \max\{1, 0\} + \max\{1, 13\} + \max\{4, 0\} = 18$	
$x_c = 9 + 2 + 0 = 11$		$y_c = \max\{8, 0\} + \max\{4, 0\} = 12$	
$x_a = 8 + 18 + 12 = 38$		$y_a = \max\{14, 18\} + \max\{11, 12\} = 30$	

Teža najtežje neodvisne množice  $S$  je torej  $c^* = \max\{x_a, y_a\} = 38$ . Poglejmo, katera vozlišča so v množici  $S$ .

$c^* = x_a$	
$x_a = c_a + y_b + y_c$	$a \in S$
$y_b = x_d + y_e + x_f$	$b \notin S$
$y_c = x_g + x_h$	$c \notin S$
$x_d = c_d$	$d \in S$
$y_e = x_i + x_j$	$e \notin S$
$x_f = c_f$	$f \in S$
$x_g = c_g + y_k$	$g \in S$
$x_h = c_h$	$h \in S$

$$\begin{array}{ll}
x_i = c_i & i \in S \\
x_j = c_j & j \in S \\
y_k = 0 & k \notin S
\end{array}$$

Najtežja neodvisna množica je torej  $S = \{a, d, f, g, h, i, j\}$ .

**Naloga 5.46.**

- (a) Za dani graf  $G = (V, E)$ , vozlišči  $s, t \in V$ , uteži povezav  $L_{uv}$  ( $uv \in E$ ) in uteži vozlišč  $c_v$  ( $v \in V$ ) bomo izračunali nove uteži povezav  $\ell_{uv} = L_{uv} + c_v$  ( $uv \in E$ ), ki predstavljajo skupen strošek prehoda od operaterja  $u$  k operaterju  $v$ . Ker so te uteži še vedno lahko tudi negativne, bomo v grafu  $G$  poiskali najkrajšo pot od  $s$  do  $t$  s pomočjo Bellman-Fordovega algoritma, ki teče v času  $O(mn)$ , kjer je  $n$  število vozlišč in  $m$  število povezav grafa  $G$ .
- (b) Najprej izračunajmo nove uteži povezav.

$$\begin{aligned}
\ell_{\text{brez}, \text{Bobitel}} &= 65 + 15 = 80 \\
\ell_{\text{brez}, \text{Sodafon}} &= 85 + 10 = 95 \\
\ell_{\text{brez}, \text{Rega}} &= 95 + 0 = 95 \\
\ell_{\text{brez}, \text{Fenmobil}} &= 50 + 30 = 80 \\
\ell_{\text{brez}, Z^0} &= 60 + 5 = 65 \\
\ell_{\text{brez}, \text{TeleJanez}} &= 45 + 20 = 65 \\
\ell_{\text{Bobitel}, \text{Sodafon}} &= 10 + 10 = 20 \\
\ell_{\text{Sodafon}, \text{Rega}} &= -5 + 0 = -5 \\
\ell_{\text{Rega}, \text{Fenmobil}} &= -15 + 30 = 15 \\
\ell_{\text{Fenmobil}, Z^0} &= -10 + 5 = -5 \\
\ell_{Z^0, \text{TeleJanez}} &= 5 + 20 = 25 \\
\ell_{\text{TeleJanez}, \text{Bobitel}} &= 0 + 15 = 15 \\
\ell_{\text{TeleJanez}, \text{Fenmobil}} &= -20 + 30 = 10 \\
\ell_{\text{Fenmobil}, \text{Sodafon}} &= 5 + 10 = 15 \\
\ell_{\text{Sodafon}, \text{TeleJanez}} &= 20 + 20 = 40
\end{aligned}$$

Za izračun najcenejše poti od vozlišča *brez* do vozlišča *Rega* bomo sledili algoritmu BELLMANFORD iz naloge 5.13, pri čemer bomo upoštevali vrstni red povezav iz zgornjega seznama. Strnjen potek algoritma (samo koraki, kjer se katera od razdalj spremeni) je prikazan v tabeli 42. Za najcenejše zaporedje prehodov med operaterji tako dobimo *TeleJanez*, *Fenmobil*, *Sodafon*, *Rega*, skupni strošek pa je 75.

$i$	$u$	$v$	$\ell_{uv}$	<i>brez</i>	<i>Bobitel</i>	<i>Sodafon</i>	<i>Rega</i>	<i>Fenmobil</i>	$Z^0$	<i>TeleJanez</i>
1	<i>brez</i>	<i>ostali</i>		0	80 <sub>brez</sub>	95 <sub>brez</sub>	95 <sub>brez</sub>	80 <sub>brez</sub>	65 <sub>brez</sub>	65 <sub>brez</sub>
1	<i>Sodafon</i>	<i>Rega</i>	-5	0	80 <sub>brez</sub>	95 <sub>brez</sub>	90 <sub>Sodafon</sub>	80 <sub>brez</sub>	65 <sub>brez</sub>	65 <sub>brez</sub>
1	<i>TeleJanez</i>	<i>Fenmobil</i>	10	0	80 <sub>brez</sub>	95 <sub>brez</sub>	90 <sub>Sodafon</sub>	75 <sub>TeleJanez</sub>	65 <sub>brez</sub>	65 <sub>brez</sub>
1	<i>Fenmobil</i>	<i>Sodafon</i>	10	0	80 <sub>brez</sub>	90 <sub>Fenmobil</sub>	90 <sub>Sodafon</sub>	75 <sub>TeleJanez</sub>	65 <sub>brez</sub>	65 <sub>brez</sub>
2	<i>Sodafon</i>	<i>Rega</i>	-5	0	80 <sub>brez</sub>	90 <sub>Fenmobil</sub>	85 <sub>Sodafon</sub>	75 <sub>TeleJanez</sub>	65 <sub>brez</sub>	65 <sub>brez</sub>

Tabela 42: Potek izvajanja algoritma za nalogo 5.46(b).

#### Naloga 5.47.

- (a) Pomagali si bomo z iskanjem v širino. Zapišimo algoritem, ki kliče funkcijo BFS iz naloge 5.6.

```

function ŠTEVILONAJKRAJŠIH( $G = (V, E), s$ )
    števil  $\leftarrow$  slovar s ključi  $v \in V$  in vrednostmi 0
    globina  $\leftarrow$  slovar s ključi  $v \in V$  in vrednostmi  $\infty$ 
    števil[ $s$ ]  $\leftarrow$  1
    globina[ $s$ ]  $\leftarrow$  0
    function VISIT( $u, v$ )
        for  $w \in \text{ADJ}(G, u)$  do
            if globina[ $w$ ] =  $\infty$  then
                globina[ $w$ ]  $\leftarrow$  globina[ $u$ ] + 1
            end if
            if globina[ $w$ ] = globina[ $u$ ] + 1 then
                števil[ $w$ ]  $\leftarrow$  števil[ $w$ ] + števil[ $u$ ]
            end if
        end for
    end function
    BFS( $G, \{s\}, \text{VISIT}$ )
    return globina
end function

```

Vidimo, da s klicanjem funkcije VISIT na vsakem vozlišču vsako povezavo obravnavamo največ dvakrat, tako da je časovna zahtevnost algoritma  $O(m)$ .

- (b) Potek izvajanja algoritma je prikazan v tabeli 43, pri čemer je bil upoštevan abecedni vrstni red obiskovanja sosedov posameznega vozlišča. Končni rezultat lahko preberemo iz zadnje vrstice tabele.

#### Naloga 5.48.

- (a) Zgledovali se bomo po algoritmu NAJKRAJŠAPOT iz naloge 5.14. Zapišimo algoritem, ki kliče funkciji TOPO in REKONSTRUIRAJPOT iz iste naloge, poleg tega pa kliče tudi funkcijo ADJIN, ki vrne množico vozlišč v podanem usmerjenem grafu s povezavo do podanega vozlišča.

$u$	globina									število								
	$a$	$b$	$c$	$d$	$e$	$f$	$g$	$h$	$i$	$a$	$b$	$c$	$d$	$e$	$f$	$g$	$h$	$i$
	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	0	0	0	0	0	0	0	0	0	1
$i$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	1	1	1	0	0	0	0	0	0	1	1	1	1
$f$	$\infty$	$\infty$	2	$\infty$	2	1	1	1	0	0	0	1	0	1	1	1	1	1
$g$	$\infty$	$\infty$	2	2	2	1	1	1	0	0	0	2	1	1	1	1	1	1
$h$	$\infty$	$\infty$	2	2	2	1	1	1	0	0	0	2	1	2	1	1	1	1
$c$	3	3	2	2	2	1	1	1	0	2	2	2	1	2	1	1	1	1
$e$	3	3	2	2	2	1	1	1	0	2	4	2	1	2	1	1	1	1
$d$	3	3	2	2	2	1	1	1	0	3	4	2	1	2	1	1	1	1
$a$	3	3	2	2	2	1	1	1	0	3	4	2	1	2	1	1	1	1
$b$	3	3	2	2	2	1	1	1	0	3	4	2	1	2	1	1	1	1

Tabela 43: Potek izvajanja algoritma za nalogo 5.47(b).

```

function JADRNIC(A( $G = (V, E), s, t, (k_u)_{u \in V}$ )
   $c \leftarrow$  slovar z vrednostjo  $-\infty$  za vsako vozlišče  $v \in V$ 
   $pred \leftarrow$  slovar z vrednostjo NULL za vsako vozlišče  $v \in V$ 
  for  $u \in \text{TOPO}(G)$  do
    for  $v \in \text{ADJIN}(G, u)$  do
      if  $c[v] > c[u]$  then
         $c[v] \leftarrow c[u]$ 
         $pred[v] \leftarrow u$ 
      end if
    end for
    if  $c[u] > k_u$  then
       $c[u] \leftarrow k_u$ 
    end if
  end for
  return REKONSTRUIRAJPOT( $pred, t$ )
end function

```

Algoritem opravi topološko urejanje grafa, nato vsako povezavo pregleda enkrat, nazadnje pa opravi še obratni prehod po najdeni poti, tako da je njegova časovna zahtevnost  $O(m)$ .

- (b) Graf s slike 36 ima topološko ureditev KP, PU, RI, ML, ZD, DO, ŠI, ST, HV, DU. Izračunane vrednosti v tabelah  $c$  in  $pred$  v algoritmu iz točke (a) so prikazane v tabeli 44, iz katere je razvidno, da gre lahko na jadrnico največ 15 prijateljev, kar je mogoče doseči z izbiro poti KP – PU – ZD – DO – HV – DU.

KP	PU	RI	ML	ZD	DO	ŠI	ST	HV	DU
20	18 <sub>KP</sub>	14 <sub>PU</sub>	12 <sub>PU</sub>	16 <sub>PU</sub>	15 <sub>ZD</sub>	13 <sub>ZD</sub>	13 <sub>ŠI</sub>	15 <sub>DO</sub>	15 <sub>HV</sub>

Tabela 44: Izračunane vrednosti v algoritmu za nalogo 5.48(b).

**Naloga 5.49.**

- (a) Uporabili bomo prirejeno različico Bellman-Fordovega algoritma, pri katerem bomo naredili le  $t$  obhodov glavne zanke.

```

function OMEJENIBELLMANFORD( $G = (V, E), s \in V, c : E \rightarrow \mathbb{R}, t \in \{0, 1, \dots, |V| - 1\}$ )
   $d[0, \dots, t] \leftarrow$  seznam slovarjev z vrednostjo  $\infty$  za vsako vozlišče  $v \in V$ 
   $pred[1, \dots, t] \leftarrow$  seznam slovarjev z vrednostjo NULL za vsako vozlišče  $v \in V$ 
   $d[0][s] \leftarrow 0$ 
  for  $i = 1, 2, \dots, t$  do
    for  $u \in V$  do
       $d[i][u] \leftarrow d[i-1][u]$ 
       $pred[i][u] \leftarrow u$ 
    end for
    for  $uv \in E$  do
      if  $d[i][v] > d[i-1][u] + c_{uv}$  then
         $d[i][v] \leftarrow d[i-1][u] + c_{uv}$ 
         $pred[i][v] \leftarrow u$ 
      end if
    end for
  end for
  return  $(d, pred)$ 
end function

```

Časovna zahtevnost algoritma je  $O(tm)$ , kjer je  $m$  število povezav grafa. Najkrajšo pot od  $s$  do vozlišča  $u \in V$  z največ  $t$  povezavami lahko rekonstruiramo s klicem funkcije  $\text{REKONSTRUIRAJPOT}(pred, u, t)$  iz rešitve naloge 5.27.

- (b) Potek izvajanja zgornjega algoritma je prikazan v tabeli 45, kjer so prikazani le koraki, ko algoritem najde krajšo pot do vozlišča v primerjavi s prejšnjim obhodom glavne zanke. Iz tabele lahko rekonstruiramo sledeče najkrajše poti z največ 4 povezavami:

- $a$ , dolžina 0
- $ab$ , dolžina 2
- $abc$ , dolžina 3
- $abcd$ , dolžina 4
- $abcde$ , dolžina 5
- $aef$ , dolžina 8

$i$	$a$	$b$	$c$	$d$	$e$	$f$
0	0					
1		$2_a$			$7_a$	
2			$3_b$			$8_e$
3				$4_c$		
4					$6_d$	

Tabela 45: Potek reševanja za nalogo 5.49(b).

### Naloga 5.50.

- (a) Naj bosta  $x_u$  in  $y_u$  teži najtežjega prirejanja  $M$  v poddrevesu s korenem  $u$  (glede na slovar  $pred$ ), za katerega velja, da ne vsebuje nobene povezave s krajiščem  $u$ , oziroma da tako povezavo lahko vsebuje. Potem velja

$$x_u = \sum_{\substack{v \sim u \\ v \neq pred[u]}} y_v \quad \text{in} \\ y_u = x_u + \max(\{0\} \cup \{c_{uv} + x_v - y_v \mid v \sim u, v \neq pred[u]\}).$$

Vrednosti  $x_u$  in  $y_u$  računamo od listov v smeri proti korenu  $r$  (tj., v topološki ureditvi glede na  $pred$ ). Težo najtežje neodvisne množice dobimo kot  $c^* = y_r$ .

- (b) Da iz vrednosti  $x_u$  in  $y_u$  ( $u \in V$ ) izluščimo najtežje prirejanje  $M$  v drevesu  $T$ , se sprehodimo od korena proti listom (npr. z iskanjem v širino) in za posamezno vozlišče  $u$ , za katerega velja  $x_u \neq y_u$  in v  $M$  nimamo povezave do njegovega predhodnika, v množico  $M$  dodamo povezavo do enega od naslednikov  $v$ , za katerega velja  $y_u + y_v - x_u - x_v = c_{uv}$ .

Zapišimo algoritem, ki kliče funkcijo BFS iz naloge 5.6.

```

function NAJTEŽJEPRIREJANJE( $T = (V, E), r, c : E \rightarrow \mathbb{R}, (x_u)_{u \in V}, (y_u)_{u \in V}$ )
   $M \leftarrow$  prazna množica
  function VISIT( $u, w$ )
    if  $x_u \neq y_u \wedge (w = \text{NULL} \vee uw \notin M)$  then
       $v \leftarrow$  eno izmed vozlišč iz  $\text{ADJ}_G(u) \setminus \{w\}$  z  $y_u + y_v - x_u - x_v = c_{uv}$ 
       $M.\text{add}(uv)$ 
    end if
  end function
  BFS( $T, \{r\}, \text{VISIT}$ )
  return  $M$ 
end function

```

- (c) Naj bo  $n$  število vozlišč drevesa  $T$ . Potem ima  $T$  natanko  $n - 1$  povezav. Za izračun najtežjega prirejanja naredimo tri preglede v širino, pri čemer v vsakem vozlišču porabimo konstantno mnogo časa. Časovna zahtevnost celotnega algoritma je torej  $O(n)$ .

(d) Izračunajmo vrednosti  $x_u$  in  $y_u$  ( $u \in V$ ), če za koren vzamemo vozlišče  $a$ .

$$\begin{array}{llll}
 x_i & = 0 & y_i & = 0 \\
 x_j & = 0 & y_j & = 0 \\
 x_k & = 0 & y_k & = 0 \\
 x_d & = 0 & y_d & = 0 \\
 x_e = 0 + 0 & = 0 & y_e = 0 + \max\{0, 4, \underline{9}\} & = 9 \\
 x_f & = 0 & y_f & = 0 \\
 x_g & = 0 & y_g = 0 + \max\{0, \underline{1}\} & = 1 \\
 x_h & = 0 & y_h & = 0 \\
 x_b = 0 + 9 + 0 & = 9 & y_b = 9 + \max\{0, \underline{8}, -1, 4\} & = 17 \\
 x_c = 1 + 0 & = 1 & y_c = 1 + \max\{0, 1, \underline{4}\} & = 5 \\
 x_a = 17 + 5 & = 22 & y_a = 22 + \max\{0, -7, \underline{5}\} & = 27
 \end{array}$$

Teža najtežjega prirejanja  $M$  je torej  $c^* = y_a = 27$ . Poglejmo, katere povezave so v množici  $M$ .

$$\begin{array}{ll}
 c^* = y_a = c_{ac} + x_a + x_c - y_c & ac \in M \\
 y_b = c_{bd} + x_b + x_d - y_d & bd \in M \\
 x_c = y_g + y_h & \\
 y_e = c_{ej} + x_e + x_j - y_j & ej \in M \\
 x_g = c_{gk} + x_g + x_k - y_k & gk \in M
 \end{array}$$

Najtežje prirejanje je torej  $M = \{ac, bd, ej, gk\}$ .



## 2.6 CPM/PERT

### Naloga 6.1.

- (a) Projekt predstavimo z usmerjenim acikličnim grafom  $G = (V, E)$ , kjer vozlišča predstavljajo opravila, posebej pa dodamo še začetno vozlišče  $s$  in končno vozlišče  $t$ . Opravilo  $u$  povežemo z opravilom  $v$ , če je  $u$  pogoj za začetek opravila  $v$ . Poleg tega vozlišče  $s$  povežemo z opravili, ki nimajo predpogojev; opravila, ki niso predpogoj za katero drugo opravilo, pa povežemo z vozliščem  $t$ . Povezave iz opravila  $u$  utežimo s trajanjem opravila  $u$ , na povezave iz  $s$  pa postavimo utež 0.

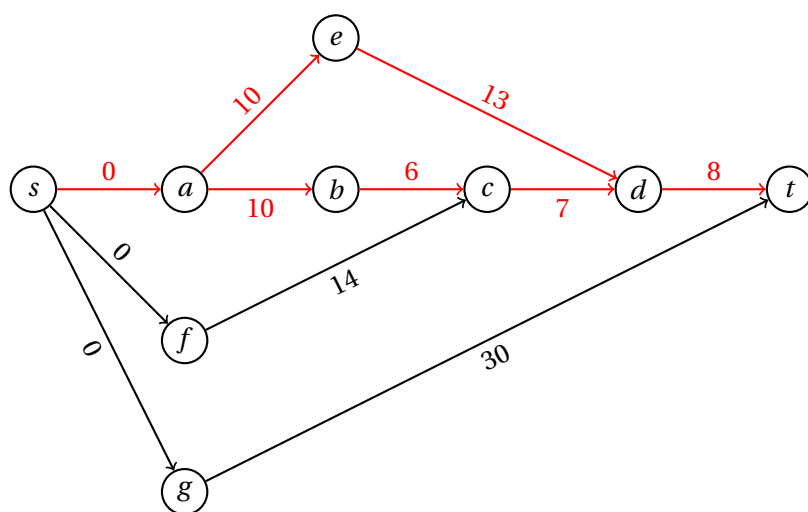
Na grafu  $G$  uporabimo algoritem NAJDALJŠAPOT iz naloge 5.14(c) z začetkom v  $s$  in tako za vsako opravilo dobimo najzgodnejši čas, ko ga lahko začnemo. Dolžina najdaljše poti do vozlišča  $t$  predstavlja najkrajše možno trajanje celotnega projekta. Nato na obratnem grafu  $G' = (V, E')$ , kjer je  $E' = \{vu \mid uv \in E\}$  množica obratnih povezav z enakimi utežmi, še enkrat uporabimo algoritem NAJDALJŠAPOT, tokrat z začetkom v  $t$ , in dobljene razdalje odštejemo od najkrajšega možnega trajanja projekta. Tako za vsako opravilo dobimo še najpoznejši možen čas začetka, da se celotno trajanje projekta ne poveča. Pri kritičnih opravilih sta oba časa enaka.

Projekt lahko predstavimo z uteženim grafom s slike 105, iz katerega je razvidna topološka ureditev  $s, a, f, g, b, e, c, d, t$ . V tabeli 46 so podani rezultati, dobljeni z zgornjim postopkom. Vidimo, da je najkrajše trajanje projekta 31 dni, kritična opravila pa so  $a, b, c, d, e$ .

- (b) Opazimo, da vsa kritična opravila ležijo na dveh poteh od  $s$  do  $t$ . Kritični poti sta torej  $s - a - e - d - t$  in  $s - a - b - c - d - t$ .
- (c) Iz tabele 46 je razvidno, da je najmanj kritično opravilo  $f$ , saj ga lahko brez podaljševanja celotnega trajanja podaljšamo za 2 dni, kar je več kot katerokoli drugo opravilo.
- (d) Skrajšati želimo katero od kritičnih opravil, ki leži na obeh kritičnih poteh (taki sta  $a$  in  $d$ ), saj sicer ne bomo skrajšali celotnega trajanja projekta. Ker ima pot  $s - g - t$  dolžino 30 dni, bomo lahko celotno trajanje skrajšali za največ 1 dan. To lahko dosežemo, če skrajšamo opravilo  $a$ , ki bo tako namesto 10 trajalo 9 dni, skupaj pa bo projekt trajal 30 dni.

### Naloga 6.2.

Zapisali bomo linearni program, ki bo za vsako opravilo imel spremenljivki  $x_u$  in  $y_u$ , ki določata število dni, za katerega skrajšamo trajanje opravila  $u$ , oziroma čas začetka izvajanja opravila  $u$ .



Slika 105: Graf odvisnosti med opravili in kritična pot za nalogo 6.1.

	<i>s</i>	<i>a</i>	<i>f</i>	<i>g</i>	<i>b</i>	<i>e</i>	<i>c</i>	<i>d</i>	<i>t</i>
najzgodnejši začetek	0	0 <sub>s</sub>	0 <sub>s</sub>	0 <sub>s</sub>	10 <sub>a</sub>	10 <sub>a</sub>	16 <sub>b</sub>	23 <sub>c,e</sub>	31 <sub>d</sub>
najpoznejši začetek	0 <sub>a</sub>	0 <sub>e</sub>	2 <sub>c</sub>	1 <sub>t</sub>	10 <sub>c</sub>	10 <sub>d</sub>	16 <sub>d</sub>	23 <sub>t</sub>	31
razlika	0	0*	2	1	0*	0*	0*	0*	0

Tabela 46: Razporejanje opravil za nalogo 6.1.

$$\min \quad 200x_a + 100x_b + 150x_c + 160x_d + 250x_e + 240x_f + 300x_g$$

Opravila brez pogojev:

$$y_a, \quad y_d, \quad y_e \geq 0$$

Odvisnosti med opravili:

$$y_b - y_a + x_a \geq 10$$

$$y_e - y_a + x_a \geq 10$$

$$y_c - y_b + x_b \geq 6$$

$$y_d - y_c + x_c \geq 7$$

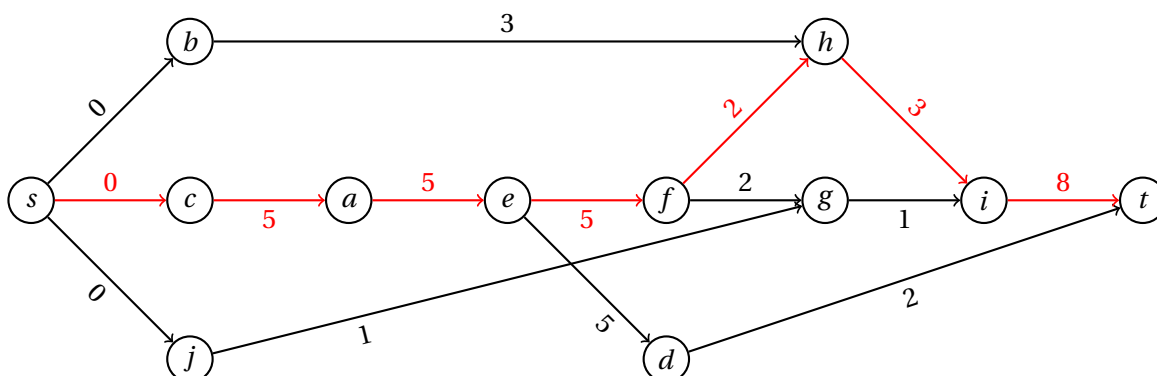
$$y_d - y_e + x_e \geq 13$$

$$y_c - y_f + x_f \geq 14$$

Želeni čas trajanja:

$$y_d - x_d \leq 19$$

$$y_g - x_g \leq -3$$



Slika 106: Graf odvisnosti med opravili in kritična pot za nalogo 6.3.

Minimalno trajanje opravil:

$$0 \leq x_a \leq 3$$

$$0 \leq x_b \leq 1$$

$$0 \leq x_c \leq 2$$

$$0 \leq x_d \leq 2$$

$$0 \leq x_e \leq 4$$

$$0 \leq x_f \leq 3$$

$$0 \leq x_g \leq 5$$

### Naloga 6.3.

- Projekt lahko predstavimo z uteženim grafom s slike 106, iz katerega je razvidna topološka ureditev  $s, b, c, j, a, e, d, f, g, h, i, t$ .
- V tabeli 47 so podani najzgodnejši začetki opravil in najpoznejši začetki, da se celotno trajanje priprave ne podaljša. Priprava bo torej trajala najmanj 28 minut, edina kritična pot pa je  $s - c - a - e - f - h - i - t$ , ki tako vsebuje tudi vsa kritična opravila.
- Iz tabele 47 je razvidno, da je najmanj kritično opravilo  $j$ , saj lahko njegovo trajanje podaljšamo za 18 minut, ne da bi vplivali na trajanje celotnega projekta.

	<i>s</i>	<i>b</i>	<i>c</i>	<i>j</i>	<i>a</i>	<i>e</i>	<i>d</i>	<i>f</i>	<i>g</i>	<i>h</i>	<i>i</i>	<i>t</i>
najprej	0	0 <sub>s</sub>	0 <sub>s</sub>	0 <sub>s</sub>	5 <sub>c</sub>	10 <sub>a</sub>	15 <sub>e</sub>	15 <sub>e</sub>	17 <sub>f</sub>	17 <sub>f</sub>	20 <sub>h</sub>	28 <sub>i</sub>
najkasneje	0 <sub>c</sub>	14 <sub>h</sub>	0 <sub>a</sub>	18 <sub>g</sub>	5 <sub>e</sub>	10 <sub>f</sub>	26 <sub>t</sub>	15 <sub>h</sub>	19 <sub>i</sub>	17 <sub>i</sub>	20 <sub>t</sub>	28
razlika	0	14	0*	18	0*	0*	11	0*	2	0*	0*	0
proste rezerve	0	14	0	16	0	0	11	0	2	0	0	0
varnostne rezerve	0	14	0	18	0	0	11	0	0	0	0	0
neodvisne rezerve	0	14	0	16	0	0	11	0	0	0	0	0

Tabela 47: Razporejanje opravil za nalogo 6.3 in rezerve za nalogo 6.4.

#### Naloga 6.4.

Skupna rezerva opravila je največji čas, za katerega lahko zamaknemo končanje opravila, ne da bi s tem vplivali na čas končanja projekta. Izračunamo jo kot minimalno razliko najkasnejšega začetka naslednika in najzgodnejšega konca predhodnika, od katere odštejemo trajanje opravila. Skupne rezerve računamo pri določitvi kritičnih in najmanj kritičnih opravil ter so za nalogo 6.3 prikazane v vrstici "razlika" tabele 47.

Prosta rezerva opravila je največji čas, za katerega lahko zamaknemo končanje opravila, če z nasledniki začnemo ob najzgodnejšem možnem času. Izračunamo jo kot minimalno razliko najzgodnejšega začetka naslednika in najzgodnejšega konca predhodnika, od katere odštejemo trajanje opravila.

Varnostna rezerva opravila je največji čas, za katerega lahko zamaknemo končanje opravila, ne da bi s tem vplivali na čas končanja projekta, če s predhodniki končamo ob najkasnejšem možnem času. Izračunamo jo kot minimalno razliko najkasnejšega začetka naslednika in najkasnejšega konca predhodnika, od katere odštejemo trajanje opravila.

Neodvisna rezerva opravila je največji čas, za katerega lahko zamaknemo končanje opravila, če z nasledniki začnemo ob najzgodnejšem možnem času in s predhodniki končamo ob najkasnejšem možnem času. Izračunamo jo kot minimalno razliko najzgodnejšega začetka naslednika in najkasnejšega konca predhodnika, od katere odštejemo trajanje opravila.

Tudi proste, varnostne in neodvisne rezerve opravil iz naloge 6.3 so prikazane v tabeli 47.

#### Naloga 6.5.

Prvi kuhar bo prevzel opravila *c*, *a*, *e*, *f*, *h*, *i*, drugi pa opravila *b*, *j*, *d*, *g*. Iz tabele 47 je razvidno, da ga najbolj omejuje razporeditev opravilo *g* – najhitreje ga lahko začne 17 minut po začetku priprave, in tedaj ga konča 18 minut po začetku. Taka razporeditev mu dovoljuje, da z opravilom *d* začne 15 minut po začetku priprave in ga konča do začetka opravila *g*. Opravili *b* in *j* lahko tako opravi pred tem – začne torej 11 minut po začetku priprave in obe zaključi (vrstni red ni pomemben), preden se loti opravila *d*. Drugi kuhar lahko tako zaporedoma izvede opravila *b*, *j*, *d*, *g*, z začetkom 11 minut po začetku priprave in s koncem 7

	$s$	$a$	$b$	$d$	$c$	$e$	$t$
pričakovano trajanje		4	2	3	$\frac{29}{6}$	$\frac{19}{6}$	
varianca		$\frac{1}{9}$	0	$\frac{4}{9}$	$\frac{1}{4}$	$\frac{1}{4}$	
najzgodnejši začetek	0	$0_s$	$4_a$	$4_a$	$6_b$	$7_d$	$10.83_c$
najpoznejši začetek	$0_a$	$0_b$	$4_c$	$4.67_e$	$6_t$	$7.67_t$	10.83
razlika	0	$0^*$	$0^*$	0.67	$0^*$	0.67	0

Tabela 48: Razporejanje opravil za nalogo 6.6.

minut kasneje (torej brez prestanka).

### Naloga 6.6.

- Pričakovana trajanja in variance so prikazane v tabeli 48.
- Projekt lahko predstavimo z uteženim grafom s slike 107, iz katerega je razvidna topološka ureditev  $s, b, d, c, e, t$ .
- V tabeli 48 so podani pričakovani najzgodnejši začetki opravil in najpoznejši začetki, da se celotno trajanje priprave ne podaljša. Pričakovano trajanje projekta je torej  $\mu = 10.83$  tednov, kritična pot pa je  $s - a - b - c - t$ .
- Izračunajmo standardni odklon trajanja pričakovane kritične poti.

$$\sigma = \sqrt{\frac{1}{9} + 0 + \frac{1}{4}} = 0.601$$

Naj bo  $X$  slučajna spremenljivka, ki meri čas izvajanja pričakovane kritične poti. Izračunajmo verjetnost končanja v roku  $T = 11$  tednov.

$$P(X \leq 11) = \Phi\left(\frac{T - \mu}{\sigma}\right) = \Phi(0.277) = 0.609$$

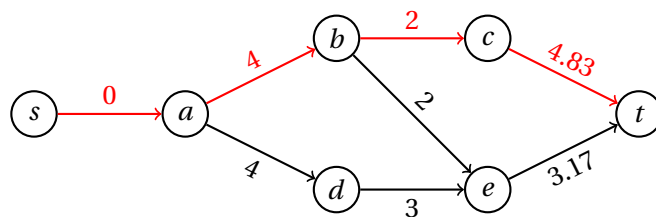
Ob predpostavki, da bodo dela zaključena in bodo tako prejeli izplačilo 250 000€, je pričakovani dobiček enak

$$250\,000\text{€} - (1 - P(X \leq 11)) \cdot 500\,000\text{€} = 54\,622.18\text{€}.$$

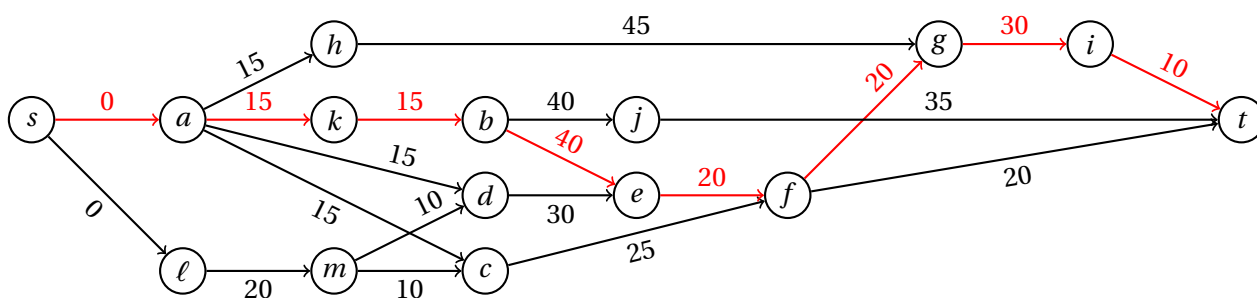
Ker je pričakovani dobiček pozitiven, se podjetju izplača prijaviti na razpis.

### Naloga 6.7.

- Projekt lahko predstavimo z uteženim grafom s slike 108, iz katerega je razvidna topološka ureditev  $s, a, \ell, h, k, m, b, d, c, j, e, f, g, i, t$ .



Slika 107: Graf odvisnosti med opravili in kritična pot za nalogo 6.6.



Slika 108: Graf odvisnosti med opravili in kritična pot za nalogo 6.7.

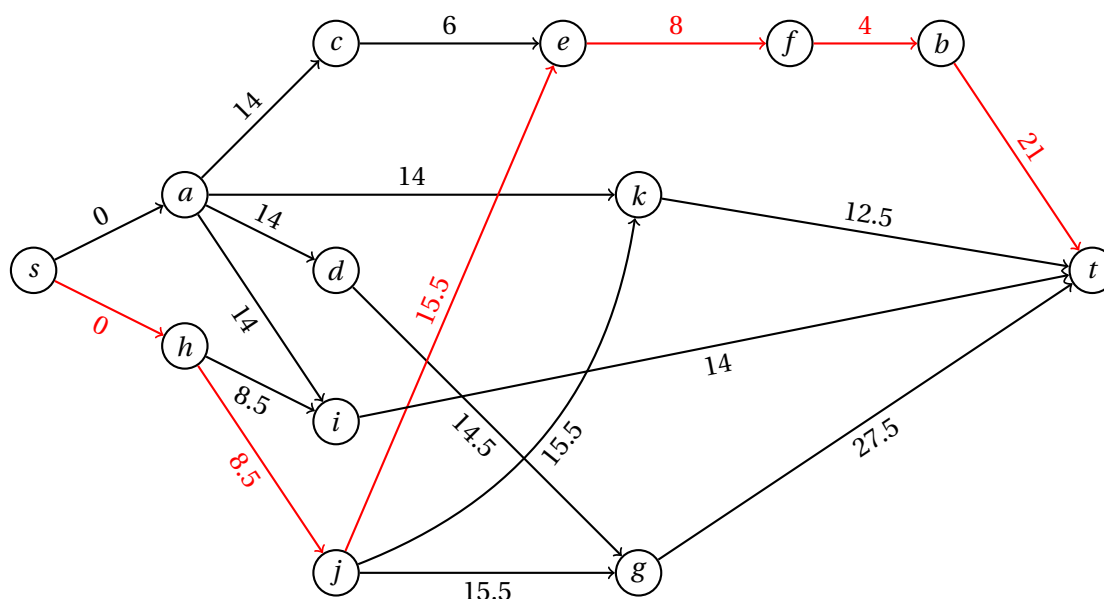
	s	a	ℓ	h	k	m	b	d	c	j	e	f	g	i	t
najprej	0	0 <sub>s</sub>	0 <sub>s</sub>	15 <sub>a</sub>	15 <sub>a</sub>	20 <sub>ℓ</sub>	30 <sub>k</sub>	30 <sub>m</sub>	30 <sub>m</sub>	70 <sub>b</sub>	70 <sub>b</sub>	90 <sub>e</sub>	110 <sub>f</sub>	140 <sub>g</sub>	150 <sub>i</sub>
najkasneje	0 <sub>a</sub>	0 <sub>k</sub>	10 <sub>m</sub>	65 <sub>g</sub>	15 <sub>b</sub>	30 <sub>d</sub>	30 <sub>e</sub>	40 <sub>e</sub>	65 <sub>f</sub>	115 <sub>t</sub>	70 <sub>f</sub>	90 <sub>g</sub>	110 <sub>i</sub>	140 <sub>t</sub>	150
razlika	0	0*	10	50	0*	10	0*	10	35	45	0*	0*	0*	0*	0

Tabela 49: Razporejanje opravil za nalogo 6.7.

- (b) V tabeli 49 so podani najzgodnejši začetki opravil in najpoznejši začetki, da se celotno trajanje projekta ne podaljša. Izdelava bo torej trajala 150 dni, edina kritična pot pa je  $s - a - k - b - e - f - g - i - t$ , ki tako vsebuje tudi vsa kritična opravila.
- (c) Iz tabele 49 je razvidno, da je najmanj kritično opravilo  $h$ , saj lahko njegovo trajanje podaljšamo za 50 dni, ne da bi vplivali na trajanje celotnega projekta.

### Naloga 6.8.

- (a) Projekt lahko predstavimo z uteženim grafom s slike 109, iz katerega je razvidna topološka ureditev  $s, a, h, c, d, j, i, k, e, g, f, b, t$ . Uteži povezav ustrezajo pričakovanim trajanjem opravil, ki so prikazana v tabeli 50.
- (b) V tabeli 50 so podani najzgodnejši začetki opravil in najpoznejši začetki, da se celotno trajanje gradnje ne podaljša. Pričakovano trajanje gradnje je torej  $\mu = 57$  dni, edina kritična pot pa je  $s - h - j - e - f - b - t$ , ki tako vsebuje tudi vsa kritična opravila.



Slika 109: Graf odvisnosti med opravili in kritična pot za nalogo 6.8.

	<i>s</i>	<i>a</i>	<i>h</i>	<i>c</i>	<i>d</i>	<i>j</i>	<i>i</i>	<i>k</i>	<i>e</i>	<i>g</i>	<i>f</i>	<i>b</i>	<i>t</i>
pričakovano		14	8.5	6	14.5	15.5	14	12.5	8	27.5	4	21	
varianca		4	2.25	1	6.25	2.25	1	2.25	1	2.25	0	4	
najprej	0	0 <sub>s</sub>	0 <sub>s</sub>	14 <sub>a</sub>	14 <sub>a</sub>	8.5 <sub>h</sub>	14 <sub>a</sub>	24 <sub>j</sub>	24 <sub>j</sub>	28.5 <sub>d</sub>	32 <sub>e</sub>	36 <sub>f</sub>	57 <sub>b</sub>
najkasneje	0 <sub>h</sub>	1 <sub>d</sub>	0 <sub>j</sub>	18 <sub>e</sub>	15 <sub>g</sub>	8.5 <sub>e</sub>	43 <sub>t</sub>	44.5 <sub>t</sub>	24 <sub>f</sub>	29.5 <sub>t</sub>	32 <sub>b</sub>	36 <sub>t</sub>	57
razlika	0	1	0*	4	1	0*	29	20.5	0*	1	0*	0*	0

Tabela 50: Razporejanje opravil za nalogo 6.8.

- (c) Iz tabele 50 je razvidno, da je najmanj kritično opravilo *i*, saj lahko njegovo trajanje podaljšamo za 29 dni v primerjavi s pričakovanim, ne da bi vplivali na trajanje celotnega projekta.
- (d) Variance trajanj opravil so prikazane v tabeli 50. Izračunajmo standardni odklon trajanja pričakovane kritične poti.

$$\sigma = \sqrt{4 + 1 + 0 + 2.25 + 2.25} = 3.082$$

Naj bo  $X$  slučajna spremenljivka, ki meri čas izvajanja pričakovane kritične poti. Izračunajmo verjetnost končanja v roku  $T = 55$  dni.

$$P(X \leq 55) = \Phi\left(\frac{T - \mu}{\sigma}\right) = \Phi(-0.6489) = 0.2578$$

**Naloga 6.9.**

- (a) Projekt lahko predstavimo z uteženim grafom s slike 110, iz katerega je razvidna topološka ureditev  $s, a, b, c, d, f, g, h, e, t$ . V tabeli 51 so podani najzgodnejši začetki opravil in najpoznejši začetki, da se celotno trajanje izdelave ne podaljša. Izdelava bo trajala najmanj 88 dni, edina kritična pot pa je  $s - b - h - e - t$ , ki tako vsebuje vsa kritična opravila.
- (b) Iz tabele 51 je razvidno, da je najmanj kritično opravilo  $c$ , saj lahko njegovo trajanje podaljšamo za 41 dni, ne da bi vplivali na trajanje celotnega projekta.
- (c) Zapišimo linearni program, ki bo za vsako opravilo imel spremenljivki  $x_u$  in  $y_u$ , ki določata število dni, za katerega skrajšamo trajanje opravila  $u$ , oziroma čas začetka izvajanja opravila  $u$ .

$$\begin{aligned} \min \quad & 1\,000x_a + 1\,500x_b + 800x_c + 1\,200x_d \\ & + 500x_e + 1\,100x_f + 1\,300x_g + 1\,400x_h \end{aligned}$$

Opravila brez pogojev:

$$y_a, y_b, y_c \geq 0$$

Odvisnosti med opravili:

$$\begin{aligned} y_d - y_a + x_a &\geq 40 \\ y_e - y_f + x_f &\geq 10 \\ y_e - y_g + x_g &\geq 12 \\ y_e - y_c + x_c &\geq 90 \\ y_f - y_a + x_a &\geq 40 \\ y_f - y_b + x_b &\geq 50 \\ y_g - y_b + x_b &\geq 50 \\ y_g - y_c + x_c &\geq 35 \\ y_h - y_b + x_b &\geq 50 \end{aligned}$$

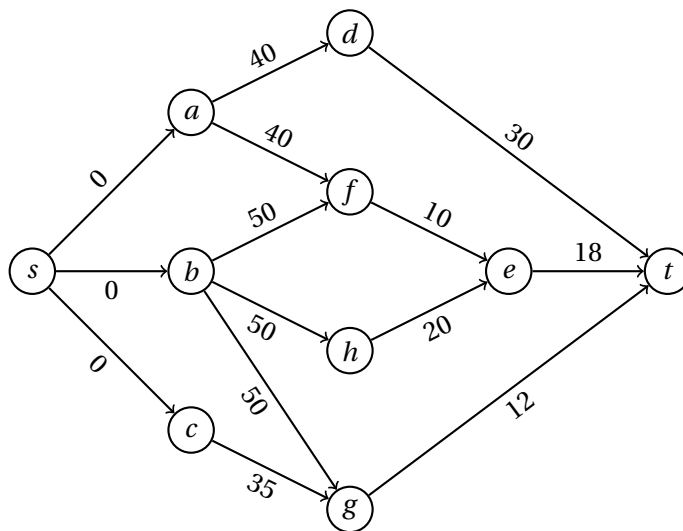
Želeni čas trajanja:

$$\begin{aligned} y_d - x_d &\leq 45 \\ y_e - x_e &\leq 57 \\ y_g - x_g &\leq 63 \end{aligned}$$

Minimalno trajanje opravil:

$$\begin{aligned} 0 &\leq x_a \leq 4 \\ 0 &\leq x_b \leq 2 \\ 0 &\leq x_c \leq 4 \\ 0 &\leq x_d \leq 2 \\ 0 &\leq x_e \leq 2 \end{aligned}$$





Slika 110: Graf odvisnosti med opravili in kritična pot za nalogo 6.9.

	s	a	b	c	d	f	g	h	e	t
najzgodnejši začetek	0	0 <sub>s</sub>	0 <sub>s</sub>	0 <sub>s</sub>	40 <sub>a</sub>	50 <sub>b</sub>	50 <sub>b</sub>	50 <sub>b</sub>	70 <sub>h</sub>	88 <sub>e</sub>
najpoznejši začetek	0	18 <sub>d</sub>	0 <sub>h</sub>	41 <sub>g</sub>	58 <sub>t</sub>	60 <sub>e</sub>	76 <sub>t</sub>	50 <sub>e</sub>	70 <sub>t</sub>	88
razlika	0	18	0*	41	18	10	26	0*	0*	0

Tabela 51: Razporejanje opravil za nalogo 6.9.

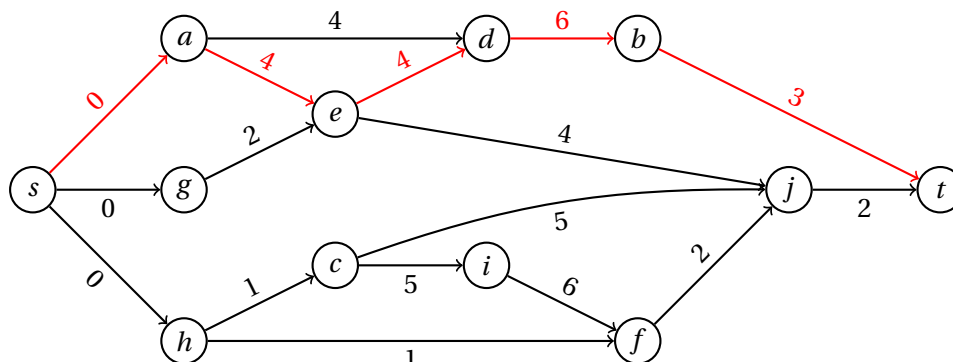
$$0 \leq x_f \leq 1$$

$$0 \leq x_g \leq 2$$

$$0 \leq x_h \leq 2$$

### Naloga 6.10.

- (a) Projekt lahko predstavimo z uteženim grafom s slike 111, iz katerega je razvidna topološka ureditev  $s, a, g, h, c, e, i, f, j, d, b, t$ .
- (b) V tabeli 52 so podani najzgodnejši in najpoznejši začetki opravil, da se celotno trajanje projekta ne podaljša. Izdelava objekta bo torej trajala 17 dni, edina kritična pot pa je  $s - a - e - d - b - t$ , ki tako vsebuje tudi vsa kritična opravila.
- (c) Iz tabele 52 je razvidno, da je najmanj kritično opravilo  $g$ .



Slika 111: Graf odvisnosti med opravili in kritična pot za nalogo 6.10.

	<i>s</i>	<i>a</i>	<i>g</i>	<i>h</i>	<i>c</i>	<i>e</i>	<i>i</i>	<i>f</i>	<i>j</i>	<i>d</i>	<i>b</i>	<i>t</i>
najprej	0	0 <sub>s</sub>	0 <sub>s</sub>	0 <sub>s</sub>	1 <sub>h</sub>	4 <sub>a</sub>	6 <sub>c</sub>	12 <sub>i</sub>	14 <sub>f</sub>	8 <sub>e</sub>	14 <sub>d</sub>	17 <sub>b</sub>
najkasneje	0 <sub>a</sub>	0 <sub>e</sub>	2 <sub>e</sub>	1 <sub>c</sub>	2 <sub>i</sub>	4 <sub>d</sub>	7 <sub>f</sub>	13 <sub>j</sub>	15 <sub>t</sub>	8 <sub>b</sub>	14 <sub>t</sub>	17
razlika	0	0	2	1	1	0	1	1	1	0	0	0

Tabela 52: Razporejanje opravil za nalogo 6.10.

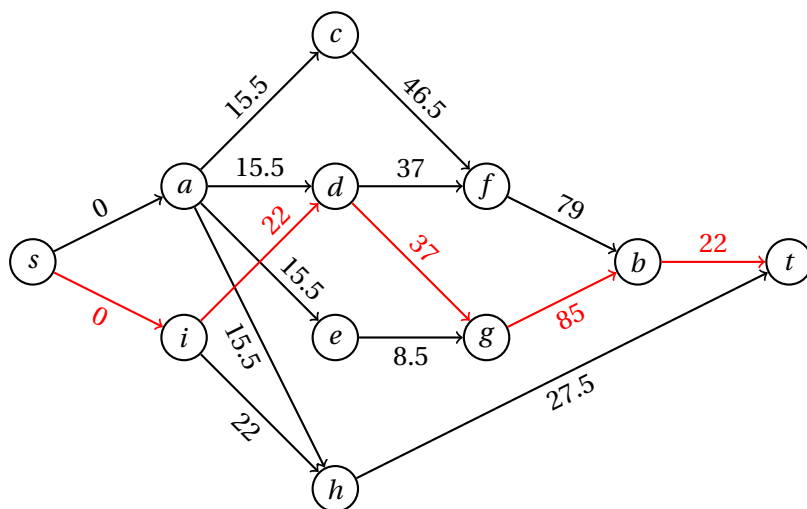
### Naloga 6.11.

- Projekt lahko predstavimo z uteženim grafom s slike 112, iz katerega je razvidna topološka ureditev  $s, a, i, c, d, e, h, f, g, b, t$ . Uteži povezav ustrezajo pričakovanim trajanjem opravil, ki so prikazana v tabeli 53.
- V tabeli 53 so podani najzgodnejši začetki opravil in najpoznejši začetki, da se celotno trajanje gradnje ne podaljša. Pričakovano trajanje gradnje je torej  $\mu = 166$  dni, edina kritična pot pa je  $s - i - d - g - b - t$ , ki tako vsebuje tudi vsa kritična opravila.
- Iz tabele 53 je razvidno, da je najmanj kritično opravilo  $h$ , saj lahko njegovo trajanje podaljšamo za 116.5 dni v primerjavi s pričakovanim, ne da bi vplivali na trajanje celotnega projekta.
- Izračunajmo standardni odklon trajanja pričakovane kritične poti.

$$\sigma = \sqrt{1 + 9 + 16 + 4} = 5.477$$

Naj bo  $X$  slučajna spremenljivka, ki meri čas izvajanja pričakovane kritične poti. Izračunajmo verjetnost končanja v roku  $T = 180$  dni.

$$P(X \leq 180) = \Phi\left(\frac{T - \mu}{\sigma}\right) = \Phi(2.556) = 0.9947$$



Slika 112: Graf odvisnosti med opravili in kritična pot za nalogo 6.11.

	<i>s</i>	<i>a</i>	<i>i</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>h</i>	<i>f</i>	<i>g</i>	<i>b</i>	<i>t</i>
pričakovano		15.5	22	46.5	37	8.5	27.5	79	85	22	
varianca		2.25	1	10.03	9	0.69	2.25	36	16	4	
najprej	0	0 <sub>s</sub>	0 <sub>s</sub>	15.5 <sub>a</sub>	22 <sub>i</sub>	15.5 <sub>a</sub>	22 <sub>i</sub>	62 <sub>c</sub>	59 <sub>d</sub>	144 <sub>g</sub>	166 <sub>b</sub>
najkasneje	0 <sub>i</sub>	3 <sub>c</sub>	0 <sub>d</sub>	18.5 <sub>f</sub>	22 <sub>g</sub>	50.5 <sub>g</sub>	138.5 <sub>t</sub>	65 <sub>b</sub>	59 <sub>b</sub>	144 <sub>t</sub>	166
razlika	0	3	0*	3	0*	35	116.5	3	0*	0*	0

Tabela 53: Razporejanje opravil za nalogo 6.11.

### Naloga 6.12.

- Projekt lahko predstavimo z uteženim grafom s slike 113, iz katerega je razvidna topološka ureditev  $s, a, b, d, c, e, g, f, h, t$ . V tabeli 54 so podani najzgodnejši začetki opravil in najpoznejši začetki, da se celotno trajanje gradnje ne podaljša. Gradnja bo trajala najmanj 215 dni, edina kritična pot pa je  $s - a - c - g - t$ , ki tako vsebuje vsa kritična opravila.
- Iz tabele 54 je razvidno, da je najmanj kritično opravilo  $d$ , saj lahko njegovo trajanje podaljšamo za 100 dni, ne da bi vplivali na trajanje celotnega projekta.
- Zapišimo linearni program, ki bo za vsako opravilo imel spremenljivki  $x_u$  in  $y_u$ , ki določata število dni, za katerega skrajšamo trajanje opravila  $u$ , oziroma čas začetka izvajanja opravila  $u$ .

	<i>s</i>	<i>a</i>	<i>b</i>	<i>d</i>	<i>c</i>	<i>e</i>	<i>g</i>	<i>f</i>	<i>h</i>	<i>t</i>
najzgodnejši začetek	0	0 <sub>s</sub>	0 <sub>s</sub>	45 <sub>a</sub>	45 <sub>a</sub>	135 <sub>c</sub>	135 <sub>c</sub>	165 <sub>e</sub>	200 <sub>f</sub>	215 <sub>g</sub>
najpoznejši začetek	0	0 <sub>c</sub>	25 <sub>c</sub>	145 <sub>f</sub>	45 <sub>g</sub>	140 <sub>f</sub>	135 <sub>t</sub>	170 <sub>h</sub>	205 <sub>t</sub>	215
razlika	0	0*	25	100	0*	5	0*	5	5	0

Tabela 54: Razporejanje opravil za nalogo 6.12.

$$\begin{aligned} \min \quad & 300x_a + 200x_b + 700x_c + 1100x_d \\ & + 1500x_e + 900x_f + 400x_g + 150x_h \end{aligned}$$

Opravila brez pogojev:

$$y_a, \quad y_b \geq 0$$

Odvisnosti med opravili:

$$y_c - y_a + x_a \geq 45$$

$$y_c - y_b + x_b \geq 20$$

$$y_d - y_a + x_a \geq 45$$

$$y_e - y_c + x_c \geq 90$$

$$y_f - y_d + x_d \geq 25$$

$$y_f - y_e + x_e \geq 30$$

$$y_g - y_c + x_c \geq 90$$

$$y_h - y_f + x_f \geq 35$$

Želeni čas trajanja:

$$y_g - x_g \leq 120$$

$$y_h - x_h \leq 190$$

Minimalno trajanje opravil:

$$0 \leq x_a \leq 5$$

$$0 \leq x_b \leq 5$$

$$0 \leq x_c \leq 20$$

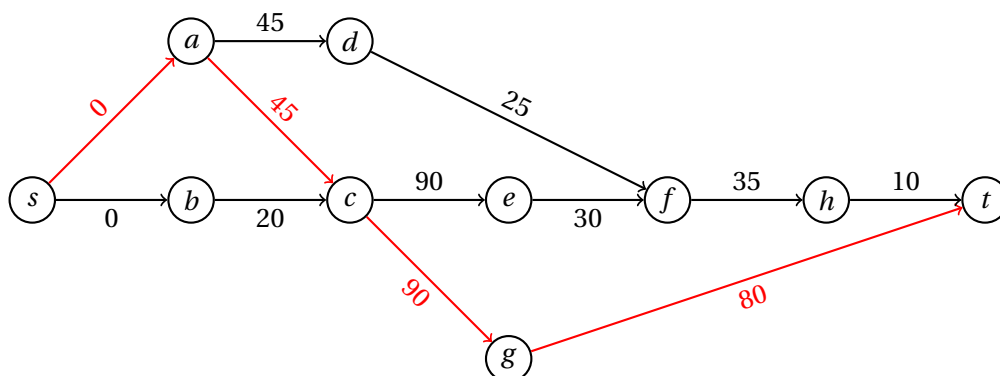
$$0 \leq x_d \leq 3$$

$$0 \leq x_e \leq 4$$

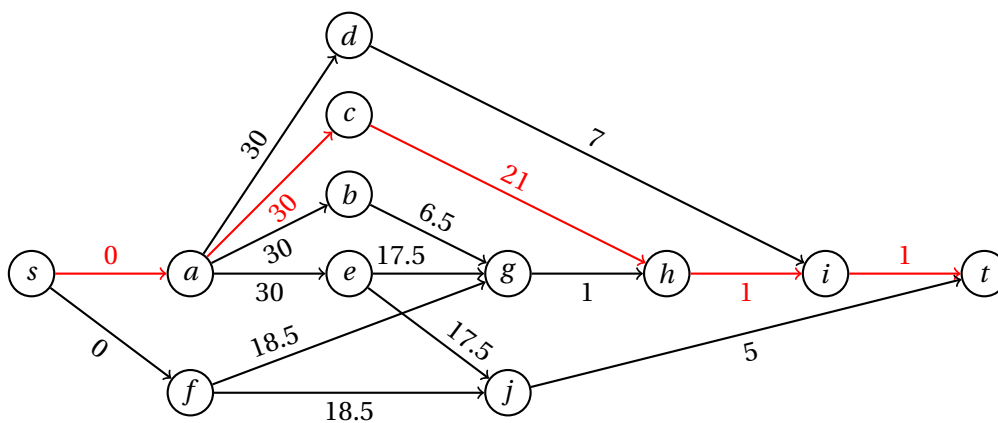
$$0 \leq x_f \leq 7$$

$$0 \leq x_g \leq 15$$

$$0 \leq x_h \leq 2$$



Slika 113: Graf odvisnosti med opravili in kritična pot za nalogo 6.12.



Slika 114: Graf odvisnosti med opravili in kritična pot za nalogo 6.13.

### Naloga 6.13.

- Projekt lahko predstavimo z uteženim grafom s slike 114, iz katerega je razvidna topološka ureditev  $s, a, f, b, c, d, e, g, j, h, i, t$ . Uteži povezav ustrezajo pričakovanim trajanjem opravil, ki so prikazana v tabeli 55.
- V tabeli 55 so podani najzgodnejši začetki opravil in najpoznejši začetki, da se celotno trajanje projekta ne podaljša. Pričakovano trajanje projekta je torej  $\mu = 53$  dni, edina kritična pot pa je  $s - a - c - h - i - t$ , ki tako vsebuje tudi vsa kritična opravila.
- Iz tabele 55 je razvidno, da je najmanj kritično opravilo  $f$ , saj lahko njegovo trajanje podaljšamo za 31.5 dni v primerjavi s pričakovanim, ne da bi vplivali na trajanje celotnega projekta.

	<i>s</i>	<i>a</i>	<i>f</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>g</i>	<i>j</i>	<i>h</i>	<i>i</i>	<i>t</i>
pričakovano		30	18.5	6.5	21	7	17.5	1	5	1	1	
varianca		4	2.25	0.69	1	0.11	0.69	0	0.11	0	0	
najprej	0	0 <sub>s</sub>	0 <sub>s</sub>	30 <sub>a</sub>	30 <sub>a</sub>	30 <sub>a</sub>	30 <sub>a</sub>	47.5 <sub>e</sub>	47.5 <sub>e</sub>	51 <sub>c</sub>	52 <sub>h</sub>	53 <sub>i</sub>
najkasneje	0 <sub>a</sub>	0 <sub>c</sub>	29.5 <sub>j</sub>	43.5 <sub>g</sub>	30 <sub>h</sub>	45 <sub>i</sub>	30.5 <sub>j</sub>	50 <sub>h</sub>	48 <sub>t</sub>	51 <sub>i</sub>	52 <sub>t</sub>	53
razlika	0	0*	29.5	13.5	0*	15	0.5	2.5	0.5	0*	0*	0

Tabela 55: Razporejanje opravil za nalogi 6.13 in 6.14.

**Naloga 6.14.**

Izračunajmo standardni odklon trajanja pričakovane kritične poti, pri čemer uporabimo variance iz tabele 55.

$$\sigma = \sqrt{4 + 1 + 0 + 0} = 2.236$$

Naj bo  $X$  slučajna spremenljivka, ki meri čas izvajanja pričakovane kritične poti. Izračunajmo verjetnost končanja v roku  $T = 55$  dni.

$$P(X \leq 55) = \Phi\left(\frac{T - \mu}{\sigma}\right) = \Phi(0.894) = 0.8144$$

## 2.7 Upravljanje zalog

### Naloga 7.1.

Imamo sledeče podatke (pri časovni enoti 1 teden).

$$v = 600$$

$$K = 25\text{€}$$

$$c = 3\text{€}$$

$$s = 0.05\text{€}$$

Ker artiklov ne proizvajamo (tj., ob dostavi imamo na voljo celotno količino naročila), vzamemo  $\lambda = \infty$  in torej  $\alpha = v$ .

- (a) Ker primanjkljaj ni dovoljen, vzamemo  $p = \infty$  in torej  $\beta = 1$ . S temi podatki poračunamo optimalni interval naročanja  $\tau^*$  in optimalno velikost naročila  $M^*$ .

$$\tau^* = \sqrt{\frac{2K\beta}{s\alpha}} = \sqrt{\frac{50}{30}} \approx 1.291$$
$$M^* = \sqrt{\frac{2K\alpha}{s\beta}} = \sqrt{\frac{30000}{0.05}} \approx 774.597$$

Optimalni interval naročanja je torej 1.291 tedna oziroma 9.037 dni, vsakič naj pa trgovina naroči okoli 775 škatel toaletnega papirja.

- (b) Vzamemo  $p = 2\text{€}$  in torej  $\beta = 1 + s/p = 1.025$ . Izračunajmo optimalni interval naročanja in optimalno vrednost naročila še v tem primeru.

$$\tau^* = \sqrt{\frac{2K\beta}{s\alpha}} = \sqrt{\frac{51.25}{30}} \approx 1.307$$
$$M^* = \sqrt{\frac{2K\alpha}{s\beta}} = \sqrt{\frac{30000}{0.05125}} \approx 765.092$$

Optimalni interval naročanja je v tem primeru torej 1.307 tedna oziroma 9.149 dni, vsakič naj pa trgovina naroči okoli 765 škatel toaletnega papirja.

### Naloga 7.2.

V tovarni bodo naročili  $v = 20000$  zvočnikov, pri čemer jih posamezno naročilo stane  $K = 500\text{€}$ . Pri naročilu velikosti vsaj  $n_i$  je cena posameznega zvočnika enaka  $c_i$ , cena skladiščenja pa je  $s_i = 0.2 \cdot c_i$  ( $1 \leq i \leq 5$ ), pri čemer imamo

$$(n_i)_{i=1}^5 = (1, 2000, 5000, 8000, 20000) \quad \text{in}$$
$$(c_i)_{i=1}^5 = (150\text{€}, 135\text{€}, 125\text{€}, 120\text{€}, 115\text{€}) .$$

Ker zvočnikov ne proizvajajo sami in primanjkljaj ni dovoljen, velja  $\lambda = p = \infty$  in torej  $\alpha = v$ ,  $\beta = 1$ .

Naj bo  $M_i$  optimalna velikost naročila, če lahko zvočnike kupujemo po ceni  $c_i$  ( $1 \leq i \leq 5$ ).

$$M_i = \sqrt{\frac{2K\alpha}{s_i\beta}}$$

$$(M_i)_{i=1}^5 = (816.497, 860.663, 894.427, 912.871, 932.505)$$

Opazimo, da  $M_i \geq n_i$  velja samo pri  $i = 1$ . Za velikost optimalnega naročila  $M_i^*$  po ceni  $c_i$  torej velja  $M_1^* = M_1$  in  $M_i^* = n_i$  ( $2 \leq i \leq 5$ ). Za naročila velikosti  $M_i^*$  ( $1 \leq i \leq 5$ ) izračunajmo skupne stroške naročanja in skladiščenja  $S_i$ .

$$S_i = \frac{M_i^* s_i}{2} + K \frac{v}{M_i^*} + v c_i$$

$$(S_i)_{i=1}^5 = (3024495, 2732000, 2564000, 2496250, 2530500)$$

Opazimo, da najmanjše stroške dosežemo, če naročamo po  $M^* = M_4^* = 8000$  zvočnikov po ceni  $c^* = c_4 = 120\text{€}$ . Izračunajmo še interval naročanja.

$$\tau^* = \frac{M^*}{v} = \frac{8000}{20000} = 0.4$$

Naročamo torej na 0.4 leta oziroma na 146 dni.

### Naloga 7.3.

Imamo sledeče podatke (pri časovni enoti 1 leto).

$$v = 10000 \quad \lambda = 25000$$

$$K = 200\text{€} \quad s = 0.25\text{€}$$

Primanjkljaja ne dovolimo, zato velja  $p = \infty$  in torej  $\beta = 1$ .

Izračunajmo optimalno dolžino cikla  $\tau^*$  in največjo zalogo  $M^*$ .

$$\alpha = v \left(1 - \frac{v}{\lambda}\right) = 10000 \cdot (1 - 0.4) = 6000$$

$$\tau^* = \sqrt{\frac{2K\beta}{s\alpha}} = \sqrt{\frac{400}{1500}} \approx 0.516$$

$$M^* = \sqrt{\frac{2K\alpha}{s\beta}} = \sqrt{\frac{2400000}{0.25}} \approx 3098.387$$

Optimalna dolžina cikla je torej približno 0.516 leta oziroma 188.485 dni, pri tem pa potrebujejo skladišče za 3099 akumulatorjev.

Izračunajmo še trajanje proizvodnje  $t^*$  in število izdelanih akumulatorjev  $q^*$  v posameznem ciklu.

$$t^* = \frac{M^*}{\lambda - v} \approx \frac{3098.387}{15000} \approx 0.207$$

$$q^* = \tau^* v = t^* \lambda \approx 0.516 \cdot 10000 \approx 5163.978$$

Proizvodnja torej traja približno 0.207 leta oziroma 75.394 dni, skupno pa izdelajo okoli 5164 akumulatorjev.



**Naloga 7.4.**

Imamo sledeče podatke (pri časovni enoti 1 teden).

$$v = 10$$

$$s = 0.2\text{€}$$

$$K = 150\text{€}$$

$$\lambda = 12.5$$

$$p = 0.8\text{€}$$

Izračunajmo optimalno dolžino cikla  $\tau^*$  in največjo zalogo  $M^*$ .

$$\alpha = v \left(1 - \frac{v}{\lambda}\right) = 10 \cdot (1 - 0.8) = 2$$

$$\beta = 1 + \frac{s}{p} = 1 + \frac{0.2}{0.8} = 1.25$$

$$\tau^* = \sqrt{\frac{2K\beta}{s\alpha}} = \sqrt{\frac{375}{0.4}} \approx 30.619$$

$$M^* = \sqrt{\frac{2K\alpha}{s\beta}} = \sqrt{\frac{600}{0.25}} \approx 48.990$$

Optimalna dolžina cikla je torej približno 30.619 tednov, pri tem pa Marta potrebuje skladišče za 49 kosov nakita.

Izračunajmo še število izdelanih kosov nakita  $q^*$ , trajanje proizvodnje  $t^*$  in največji primanjkljaj  $m^*$  v posameznem ciklu.

$$q^* = \tau^* v \approx 30.619 \cdot 10 \approx 306.186$$

$$t^* = \frac{q^*}{\lambda} \approx \frac{306.186}{12.5} \approx 24.495$$

$$m^* = \tau^* \alpha - M^* \approx 30.619 \cdot 2 - 48.990 \approx 12.247$$

V vsakem intervalu torej Marta izdelava okoli 306 kosov nakita, pri čemer izdelovanje traja približno 24.495 tednov, primanjkljaj pa ne preseže 13 kosov nakita.

**Naloga 7.5.**

Imamo sledeče podatke (pri časovni enoti 1 teden).

$$v = 60$$

$$\lambda = 90$$

$$K = 180\text{€}$$

$$s = 0.5\text{€}$$

Pri teh podatkih izračunajmo še

$$\alpha = v \left(1 - \frac{v}{\lambda}\right) = 60 \cdot \left(1 - \frac{2}{3}\right) = 20.$$

- (a) Če primanjkljaja ne dovolimo, imamo  $p = \infty$  in torej  $\beta = 1$ . Izračunajmo optimalno dolžino cikla  $\tau^*$ , največjo zalogo  $M^*$  in enotske stroške  $S^*$ .

$$\begin{aligned}\tau^* &= \sqrt{\frac{2K\beta}{s\alpha}} = \sqrt{\frac{360}{10}} = 6 \\ M^* &= \sqrt{\frac{2K\alpha}{s\beta}} = \sqrt{\frac{7200}{0.5}} = 120 \\ S^* &= \sqrt{\frac{2Ks\alpha}{\beta}} = \sqrt{3600} \text{€} = 60 \text{€}\end{aligned}$$

Optimalna dolžina cikla je torej 6 tednov, pri tem pa mora vulkanizer imeti skladišče za 120 pnevmatik. Tedenski stroški proizvodnje in skladiščenja so 60€.

- (b) Izračunajmo še trajanje proizvodnje  $t^*$  in število izdelanih pnevmatik  $q^*$  v posameznem ciklu.

$$\begin{aligned}t^* &= \frac{M^*}{\lambda - \nu} = \frac{120}{30} = 4 \\ q^* &= \tau^* \nu = t^* \lambda = 6 \cdot 60 = 360\end{aligned}$$

Proizvodnja torej traja 4 tedne, skupno pa v tem času izdela 360 pnevmatik.

- (c) Vzamemo  $p = 2\text{€}$  in torej  $\beta = 1 + s/p = 1.25$ . Izračunajmo optimalno dolžino cikla, potrebno velikost skladišča in enotske stroške še v tem primeru.

$$\begin{aligned}\tau^* &= \sqrt{\frac{2K\beta}{s\alpha}} = \sqrt{\frac{450}{10}} \approx 6.708 \\ M^* &= \sqrt{\frac{2K\alpha}{s\beta}} = \sqrt{\frac{7200}{0.625}} \approx 107.331 \\ S^* &= \sqrt{\frac{2Ks\alpha}{\beta}} = \sqrt{\frac{3600}{1.25}} \text{€} \approx 53.666 \text{€}\end{aligned}$$

Optimalni interval naročanja je v tem primeru torej 6.708 tednov, pri tem pa mora vulkanizer imeti skladišče za 108 pnevmatik. Tedenski stroški proizvodnje in skladiščenja so 53.666€.

Izračunajmo še največji primanjkljaj  $m^*$  v posameznem ciklu.

$$m^* = \tau^* \alpha - M^* \approx 6.708 \cdot 20 - 107.331 \approx 26.833$$

Največji primanjkljaj torej ne preseže 27 pnevmatik.

**Naloga 7.6.**

Imamo sledeče podatke (pri časovni enoti 1 mesec).

$$\begin{array}{ll} v = 20 & K = 750\text{€} \\ s = 10\text{€} & p = 50\text{€} \end{array}$$

Pri teh podatkih izračunajmo še

$$\beta = 1 + \frac{s}{p} = 1 + \frac{10}{50} = 1.2.$$

Ker artiklov ne proizvajamo (tj., ob dostavi imamo na voljo celotno količino naročila), vzamemo  $\lambda = \infty$  in torej  $\alpha = v$ .

- (a) Izračunajmo optimalno dolžino cikla  $\tau^*$ , največjo zalogo  $M^*$  in enotske stroške  $S^*$ .

$$\begin{aligned} \tau^* &= \sqrt{\frac{2K\beta}{s\alpha}} = \sqrt{\frac{1800}{200}} = 3 \\ M^* &= \sqrt{\frac{2K\alpha}{s\beta}} = \sqrt{\frac{30000}{12}} = 50 \\ S^* &= \sqrt{\frac{2Ks\alpha}{\beta}} = \sqrt{\frac{300000}{1.2}} \text{€} = 500\text{€} \end{aligned}$$

Optimalna dolžina cikla je torej 3 mesece, pri tem pa mora trgovina imeti skladišče za 50 sedežnih garnitur. Mesečni stroški nabave in skladiščenja so 600€.

- (b) Izračunajmo še največji primanjkljaj  $m^*$  in velikost naročila  $q^*$ .

$$\begin{aligned} m^* &= \tau^* \alpha - M^* = 3 \cdot 20 - 50 = 10 \\ q^* &= \tau^* v = 3 \cdot 20 = 60 \end{aligned}$$

- (c) Drugo skladišče lahko hrani  $n' = 40$  sedežnih garnitur, mesečni strošek hrambe enega kosa pa je  $s' = 6\text{€}$ . Najprej preverimo, kakšna bi bila optimalna velikost skladišča pri takih stroških.

$$\begin{aligned} \beta' &= 1 + \frac{s'}{p} = 1 + \frac{6}{50} = 1.12 \\ M' &= \sqrt{\frac{2K\alpha}{s'\beta'}} = \sqrt{\frac{30000}{6.72}} \approx 66.815 \end{aligned}$$

Ker je  $M' > n'$ , zapišimo formulo za enotske stroške za primer, ko imamo v skladišču največ  $n'$  kosov.

$$S' = \frac{2K\alpha + (s' + p)n'^2}{2\alpha\tau'} + p \left( \frac{\alpha\tau'}{2} - n' \right) = 500\text{€} \cdot \tau' - 2000\text{€} + \frac{2990\text{€}}{\tau'}$$

Ker iščemo dolžino intervala, kjer dosežemo minimalne stroške, poiščimo, kje ima odvod zgornjega izraza po  $\tau'$  ničlo za  $\tau' > 0$ .

$$0\text{€} = 500\text{€} - \frac{2990\text{€}}{\tau'^2}$$

$$\tau' = \sqrt{\frac{2990}{500}} \approx 2.445$$

Optimalna dolžina cikla ob uporabi drugega skladišča je torej 2.445 meseca. Vstavimo v zgornjo formulo, da dobimo mesečne stroške.

$$S' = 500\text{€} \cdot 2.445 - 2000\text{€} + \frac{2990\text{€}}{2.445} \approx 445.404\text{€}$$

Ker so stroški nižji kot pri prvem skladišču, se trgovini izplača sprejeti ponudbo.

### Naloga 7.7.

Imamo sledeče podatke (pri časovni enoti 1 dan).

$$\begin{array}{ll} v = 300 & K = 240\text{€} \\ \lambda = 400 & s = 0.1\text{€} \end{array}$$

Pri teh podatkih izračunajmo še

$$\alpha = v \left( 1 - \frac{v}{\lambda} \right) = 300 \cdot \frac{1}{4} = 75.$$

- (a) Ker primanjkljaj ni dovoljen, vzamemo  $p = \infty$  in torej  $\beta = 1$ . Izračunajmo optimalno dolžino cikla  $\tau^*$ , največjo zalogo  $M^*$ , trajanje proizvodnje  $t^*$  ter število izdelanih mask  $q^*$  v posameznem ciklu.

$$\begin{aligned} \tau^* &= \sqrt{\frac{2K\beta}{s\alpha}} = \sqrt{\frac{480}{7.5}} = 8 \\ M^* &= \sqrt{\frac{2K\alpha}{s\beta}} = \sqrt{\frac{36000}{0.1}} = 600 \\ t^* &= \frac{M^*}{\lambda - v} = \frac{600}{100} = 6 \\ q^* &= \tau^* v = t^* \lambda = 8 \cdot 300 = 2400 \end{aligned}$$

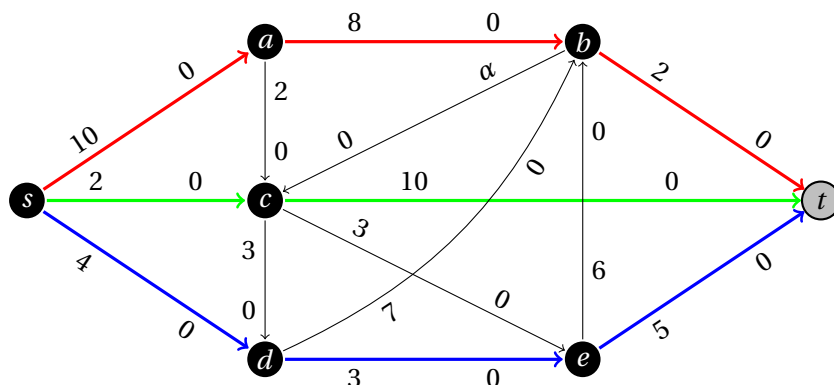
Optimalna dolžina cikla je torej 8 dni, od tega proizvodnja teče 6 dni. V tem času proizvedejo 2400 mask, v skladišču pa mora biti prostora za 600 mask.

- (b) Sedaj imamo  $p = 0.4\text{€}$  in torej  $\beta = 1 + s/p = 1.25$ . Izračunajmo optimalno dolžino cikla  $\tau^*$ , največjo zalogo  $M^*$ , največji primanjkljaj  $m^*$ , trajanje proizvodnje  $t^*$  ter število izdelanih mask  $q^*$  v posameznem ciklu.

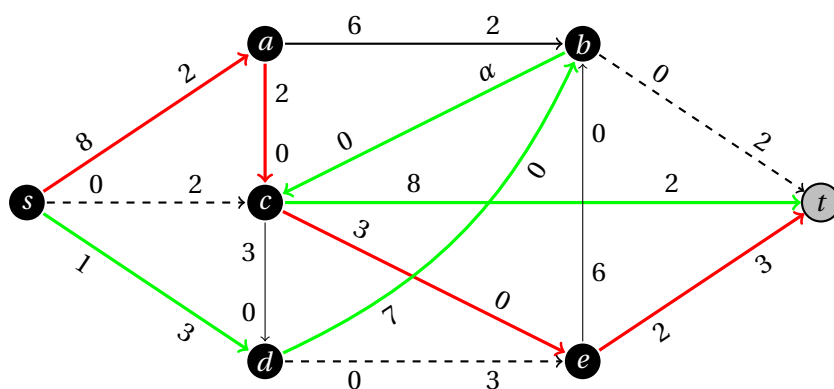
$$\begin{aligned}\tau^* &= \sqrt{\frac{2K\beta}{s\alpha}} = \sqrt{\frac{600}{7.5}} && \approx 8.944 \\ M^* &= \sqrt{\frac{2K\alpha}{s\beta}} = \sqrt{\frac{36000}{0.125}} && \approx 536.656 \\ m^* &= \tau^* \alpha - M^* \approx 670.820 - 536.656 \approx 134.164 \\ t^* &= \frac{M^* + m^*}{\lambda - \nu} \approx \frac{670.820}{100} && = 6.708 \\ q^* &= \tau^* \nu \approx 8.944 \cdot 300 && = 2683.282\end{aligned}$$

V tem primeru je torej optimalna dolžina cikla 8.944 dni, od tega proizvodnja teče 6.708 dni. V tem času proizvedejo približno 2683 mask, v skladišču pa mora biti prostora za 537 mask. Največji primanjkljaj ne preseže 135 mask.





Slika 115: Prvi korak za nalogo 8.1.

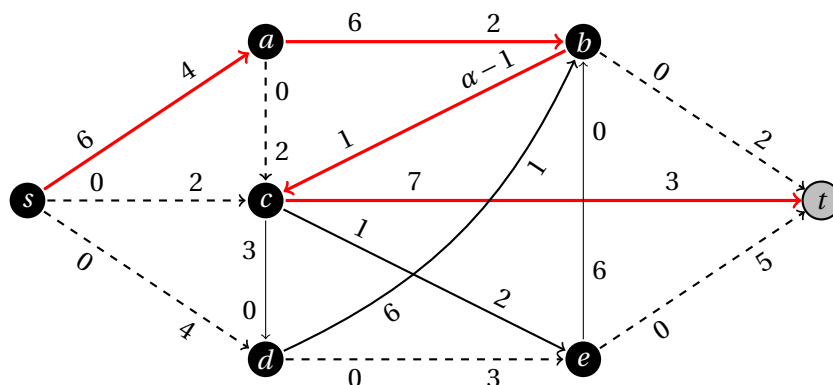


Slika 116: Drugi korak za nalogo 8.1.

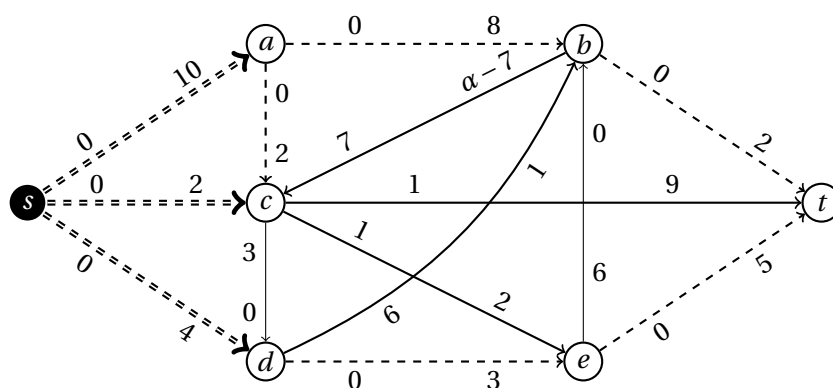
## 2.8 Pretoki in prerezi

### Naloga 8.1.

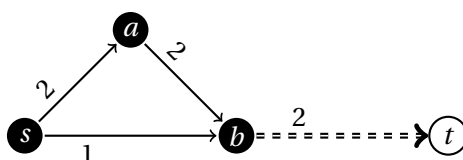
Predpostavimo, da je vrednost parametra  $\alpha$  dovolj velika, da ne vpliva na pretok skozi omrežje. Ob tej predpostavki poiščemo maksimalni pretok – postopek reševanja je prikazan na slikah 115, 116, 117 in 118. Dobimo maksimalni pretok  $10 + 2 + 4 = 2 + 9 + 5 = 16$ , najmanjša vrednost  $\alpha$ , pri kateri ga lahko dosežemo, pa bo enaka najmanjšemu možnemu pretoku po povezavi  $b \rightarrow c$ . Ta pretok bi lahko zmanjšali, če bi našli cikel iz premih povezav z neničelnimi pretoki in nezasičenih nasprotnih povezav, ki vsebuje povezavo  $b \rightarrow c$ . Toda iz slike 118 vidimo, da lahko iz vozlišča  $c$  nadaljujemo le kot  $c \rightarrow t \rightarrow c$  ali  $c \rightarrow e \rightarrow t \rightarrow c$  in tako ne moremo doseči vozlišča  $b$ . Pretoka po povezavi  $b \rightarrow c$  torej ni mogoče zmanjšati, tako da je iskana vrednost enaka  $\alpha = 7$ .



Slika 117: Tretji korak za nalogo 8.1.



Slika 118: Maksimalni pretok in minimalni prerez za nalogo 8.1.



Slika 119: Primer omrežja za nalogo 8.2.

### Naloga 8.2.

Ni nujno, da je povezava z najmanjšo kapaciteto vsebovana v minimalnem prerezu. Primer je podan na sliki 119 – povezava z najmanjšo kapaciteto je  $s \rightarrow b$ , medtem ko edini minimalni prerez sestoji iz povezave  $b \rightarrow t$ .

### Naloga 8.3.

Predpostavimo, da so intervali podani kot pari celih števil  $(a_i, b_i)$  z  $a_i < b_i$  ( $1 \leq i \leq n$ ).



- (a) Naj bo  $G = (V, E)$  usmerjen graf z množico vozlišč  $V = \{1, 2, \dots, n\}$  in povezavo  $i \rightarrow j$  ( $i, j \in V$ ) natanko tedaj, ko velja  $b_i = a_j$ . Ker za vsak  $i \in V$  velja  $a_i < b_i$ , je graf  $G$  acikličen. Naša optimizacijska naloga je poiskati čim manjše število usmerjenih poti, ki pokrijejo vsako vozlišče grafa  $G$  natanko enkrat. Pri tem dovoljujemo tudi poti dolžine 0 – ta pokrijejo natanko eno vozlišče. Vsaka taka pot torej ustreza zaporedju terminov, ki jih bo pazil posamezen asistent.
- (b) Definirajmo omrežje  $G' = (V', E')$ , kjer je

$$V' = \{s, t\} \cup \{u_i, v_i \mid i \in V\},$$

$$E' = \{s \rightarrow u_i, v_i \rightarrow t \mid i \in V\} \cup \{u_i \rightarrow v_j \mid i \rightarrow j \in E\},$$

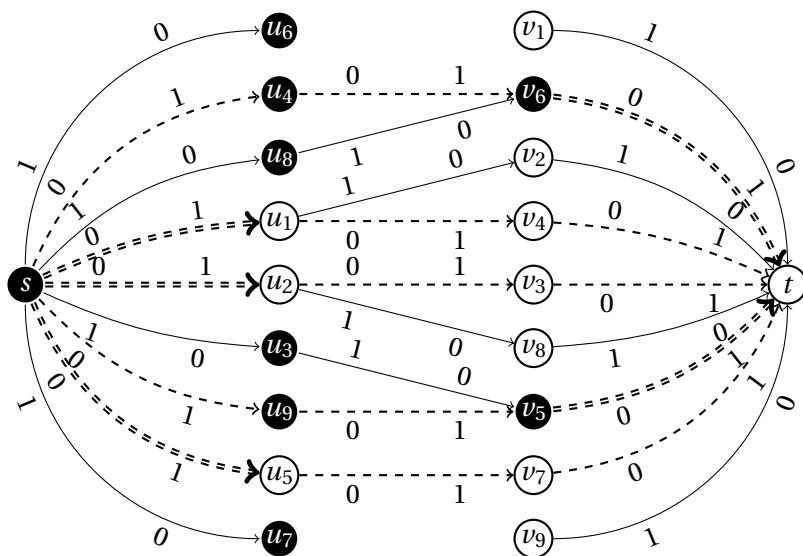
vse povezave pa imajo kapaciteto 1.

Denimo, da imamo celoštevilsko rešitev problema maksimalnega pretoka od vozlišča  $s$  do vozlišča  $t$  v omrežju  $G'$  – ker so kapacitete cela števila, nam Ford-Fulkersonov algoritem zagotavlja njen obstoj. Ker gre vsaka enota pretoka čez natanko eno povezavo  $u_i \rightarrow v_j$ , ki ustreza povezavi  $i \rightarrow j$  v grafu  $G$ , skupni pretok  $f$  ustreza dopustni rešitvi problema iz prejšnje točke z vsoto dolžin poti enako  $f$  – število poti je torej  $n - f$ . Tako maksimalni pretok ustreza optimalni rešitvi.

- (c) Na sliki 120 je prikazan maksimalen pretok za omrežje iz prejšnje točke pri danih podatkih, pri čemer so podani intervali oštevilčeni od 1 do 9. Najdeni maksimalni pretok  $f^* = 5$  torej ustreza dodelitvi terminov štirim asistentom: prvi pazi v terminih (7,8), (8,12) in (12,15), drugi v terminih (8,9) in (9,10), tretji v terminih (7,10), (10,13) in (13,15), četrti pa v terminu (9,12).

#### Naloga 8.4.

- (a) V poti  $s \rightarrow a \rightarrow c \rightarrow d \rightarrow t$  nas najbolj omejujeta povezavi  $s \rightarrow a$  in  $d \rightarrow t$  s kapaciteto 2, v poti  $s \rightarrow c \rightarrow d \rightarrow b \rightarrow t$  pa povezava  $d \rightarrow b$  s kapaciteto 1. Ker ima edina skupna povezava  $c \rightarrow d$  kapaciteto 5, bo skupen pretok po povečanju po teh dveh poteh enak 3. Pridruženo omrežje je prikazano na sliki 121.
- (b) Na omrežju s slike 121 je  $s \rightarrow c \leftarrow a \rightarrow b \rightarrow t$  edina možna povečujoča pot. Pri  $0 \leq x < 2$  nas s količino  $x$  najbolj omejuje povezava  $a \rightarrow b$  – tedaj dobimo maksimalni pretok  $3 \leq 3 + x < 5$  in minimalni prerez  $\{a \rightarrow b, d \rightarrow b, d \rightarrow t\}$ , glej sliko 122.
- Pri  $x \geq 2$  nas s količino 2 najbolj omejuje povezava  $c \leftarrow a$  – tedaj dobimo maksimalni pretok 5 in minimalni prerez  $\{s \rightarrow a, d \rightarrow b, d \rightarrow t\}$ , glej sliko 123.
- (c) Iz slik 122 in 123 je razvidno, da lahko po poti  $s \rightarrow c \rightarrow d$  povečamo pretok za največ 2, tako da lahko s povečanjem kapacitete povezave  $d \rightarrow t$  za 1 za toliko



Slika 120: Maksimalni pretok in minimalni prerez za nalogo 8.3(c).

povečamo maksimalni pretok omrežja. Ker lahko tudi po povezavi  $b \rightarrow t$  povečamo pretok za vsaj 2, enako dosežemo tudi s povečanjem kapacitete povezave  $d \rightarrow b$  za 1.

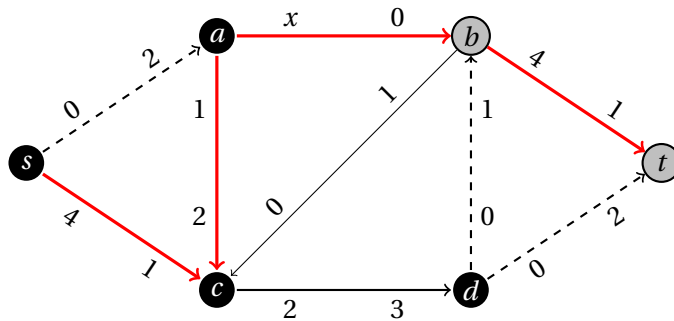
Ker lahko po povezavah  $s \rightarrow a$  in  $c \leftarrow a$  povečamo pretok v vozlišče  $a$  za največ  $2 - x$ , sledi, da lahko s povečanjem kapacitete povezave  $a \rightarrow b$  za 1 za toliko povečamo maksimalni pretok omrežja le v primeru, kadar velja  $0 \leq x \leq 1$ .

#### Naloga 8.5.

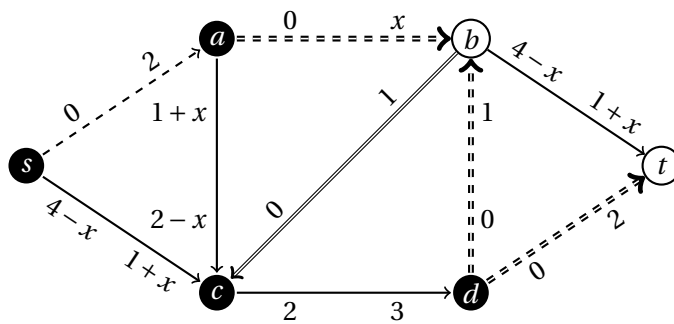
- S Ford-Fulkersonovim algoritmom poiščemo maksimalni  $(f, b)$ -tok na omrežju s slike 41. Postopek reševanja je prikazan na slikah 124, 125 in 126. Dobimo maksimalni pretok  $8 + 3 = 5 + 3 + 3 = 11$ .
- Zadostuje, da poiščemo tak par terminalov, ki maksimizira vsoto kapacitet izstopnih povezav (brez morebitnih povezav med izbranimi terminaloma). Če je eden od izbranih vstopnih terminalov  $f$ , potem največjo skupno kapaciteto 20 dosežemo, če za drugi vstopni terminal izberemo  $c$  ali  $e$ .

#### Naloga 8.6.

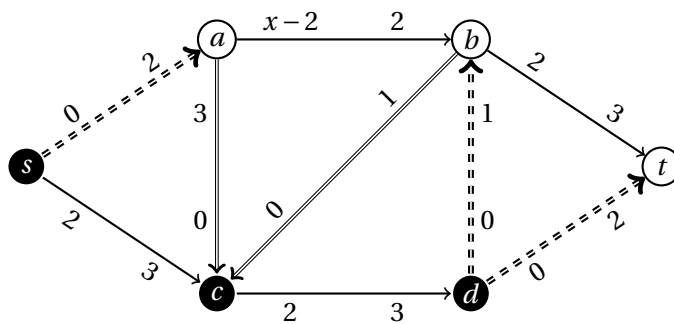
- Pravilno rešitev dobimo, če algoritem zaporedoma izbere povečujoče poti  $s \rightarrow u \rightarrow y \rightarrow t$  (pretok 3, odstraniti se povezavi  $u \rightarrow y$  in  $y \rightarrow t$ ),  $s \rightarrow u \rightarrow x \rightarrow t$  (pretok 3, odstrani se povezava  $u \rightarrow x$  in  $s \rightarrow v \rightarrow z \rightarrow t$  (pretok 2, odstrani se povezava  $s \rightarrow v$ ). Dobimo maksimalni pretok  $3 + 3 + 2 = 8$ , glej sliko 127.



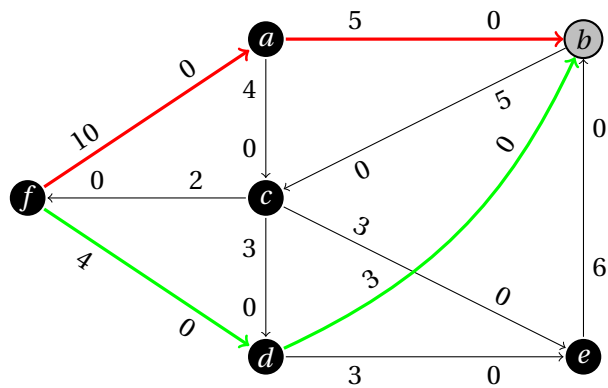
Slika 121: Pridruženo omrežje za nalogo 8.4(a) in povečujoča pot pri  $x > 0$ .



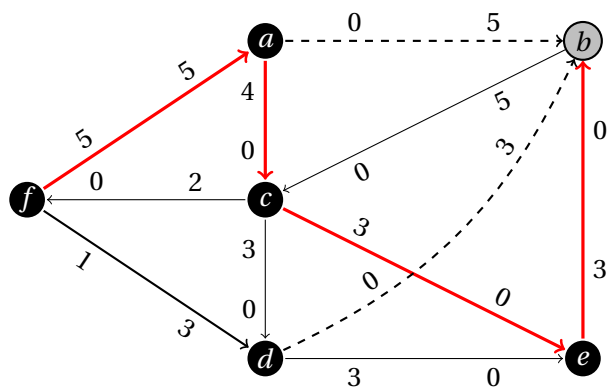
Slika 122: Maksimalni pretok pri  $0 \leq x < 2$  za nalogo 8.4(b).



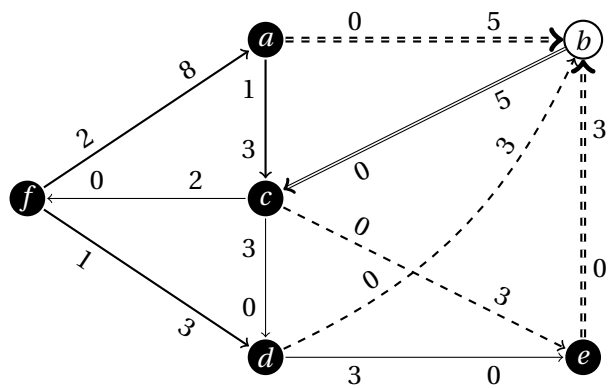
Slika 123: Maksimalni pretok pri  $x \geq 2$  za nalogo 8.4(b).



Slika 124: Prvi korak za nalogo 8.5(a).



Slika 125: Drugi korak za nalogo 8.5(a).



Slika 126: Maksimalni pretok in minimalni prerez za nalogo 8.5(a).

(b) Na sliki 128 je prikazano omrežje z maksimalnim pretokom  $1 + 1 = 2$ . Če algoritem GREEDYFLOW na tem omrežju v prvem koraku izbere pot  $s \rightarrow a \rightarrow b \rightarrow t$  z minimalno kapaciteto 1, potem vse povezave iz te poti odstrani in tako ne najde druge povečujoče poti.

(c) Za vsako omrežje obstaja izbira poti, da bo algoritem GREEDYFLOW dal maksimalni pretok. Zapišimo postopek, ki za dani graf  $G = (V, E)$ , slovar kapacitet povezav  $C$  in vozlišči  $s, t \in V$  vrne ustrezno zaporedje poti skupaj s povečanjem pretoka, ki ga prinese posamezna pot. V postopku bomo uporabili dvostrano vrsto (*double-ended queue*), na kateri bomo uporabili metode  $\text{append}(x)$  (dodajanje  $x$  na konec),  $\text{push}(x)$  (dodajanje  $x$  na začetek) in  $\text{pop}()$  (odstranjevanje z začetka).

```

function ZAPOREDJEPOTI( $G = (V, E), C, s, t$ )
     $Q \leftarrow$  prazna dvostrana vrsta
    if  $s = t$  then                                     če je izvor enak ponoru,
         $Q.\text{append}((s, \infty))$                              lahko prepeljemo poljubno količino
        return  $Q$ 
    end if
     $M \leftarrow$  minimalni  $(s, t)$ -prerez omrežja  $G$  s kapacitetami povezav  $C$ 
     $S, T \leftarrow$  množici z  $V = S \cup T, S \cap T = \emptyset, s \in S$  in  $t \in T$ , določeni s prerezom  $M$ 
     $G_s, G_t \leftarrow$  podgrafa grafa  $G$ , inducirana z množico vozlišč  $S$  oziroma  $T$ 
    for  $e = u \rightarrow v \in M$  do
         $Q_s \leftarrow \text{ZAPOREDJEPOTI}(G_s, C, s, u)$            rekurzivna klica dobita kopijo  $C$ ,
         $Q_t \leftarrow \text{ZAPOREDJEPOTI}(G_t, C, v, t)$          tako da ostane slovar nespremenjen
        while  $C[e] > 0$  do
             $P_s, \gamma_s \leftarrow Q_s.\text{pop}()$ 
             $P_t, \gamma_t \leftarrow Q_t.\text{pop}()$ 
             $\gamma \leftarrow \min\{\gamma_s, c[e], \gamma_t\}$ 
            for  $e_s \in P_s$  do                               popravimo povezave v  $G_s$ 
                if  $C[e_s] = \gamma$  then
                    odstrani  $e_s$  iz  $G_s$ 
                else
                     $C[e_s] \leftarrow C[e_s] - \gamma$ 
                end if
            end for
             $C[e] \leftarrow C[e] - \gamma$ 
            for  $e_t \in P_t$  do                               popravimo povezave v  $G_t$ 
                if  $C[e_t] = \gamma$  then
                    odstrani  $e_t$  iz  $G_t$ 
                else
                     $C[e_t] \leftarrow C[e_t] - \gamma$ 
                end if
            end for
             $Q.\text{append}((P_s \| e \| P_t, \gamma))$                 dodamo novo pot v  $G$ 
            if  $\gamma < \gamma_s$  then                             če nobene povezave v  $P_s$  ne zasitimo,
                 $Q_s.\text{push}((P_s, \gamma_s - \gamma))$              jo vrnemo na začetek vrste
            end if                                           z zmanjšano kapaciteto
            if  $\gamma < \gamma_t$  then                             podobno še za  $P_t$ 

```

```

         $Q_t.\text{push}((P_t, \gamma_t - \gamma))$ 
    end if
end while
end for
return  $Q$ 
end function

```

vrnemo vrsto poti v  $G$

Algoritem deluje rekurzivno, pri čemer si pomaga z izračunom minimalnega prereza omrežja, ki tako graf razdeli na dva dela. Za vsako povezavo  $e = u \rightarrow v$  iz prereza se tako izračunata ustrezni zaporedji poti (kot vrsti) za pretok od  $s$  do  $u$  oziroma od  $v$  do  $t$  v ustreznem podomrežju. Zaporedje poti za omrežje  $G$  se potem dopolnjuje tako, da se preko povezave  $e$  združita poti z vrha obeh vrst, pri čemer se lahko pot vrne na vrh vrste, če se nobena povezava ne zasiti; ko se pa katera od povezav zasiti, se odstrani iz ustreznega grafa. Ker tak postopek posnema obnašanje algoritma GREEDYFLOW, se nikoli ne zgodi, da bi sestavili pot, ki vsebuje katero že odstranjeno povezavo. Obravnava povezave  $e$  se zaključi, ko se ta zasiti – tedaj vrsti poti v ustreznih podomrežjih nista nujno prazni, se pa ob predpostavki, da rekurzivni klici vračajo ustrezna zaporedja, tudi predhodno ne izpraznijo. Postopek se nato nadaljuje z naslednjo povezavo iz  $M$ . Ker se zaporedji poti za podomrežji ponovno izračunata, najdene poti ne bodo vsebovale predhodno odstranjenih povezav. Do robnega primera pride, ko je izvor enak ponoru – to se zgodi v rekurzivnih klicih, ki obravnavajo povezavo iz minimalnega prereza s krajiščem  $s$  oziroma  $t$ . Ker je v tem primeru zaporedje poti ustrezno, lahko po indukciji sklepamo, da algoritem deluje pravilno.

(d) Naj bo  $G_n = (V_n, E_n)$  ( $n \geq 2$ ) omrežje, kjer je

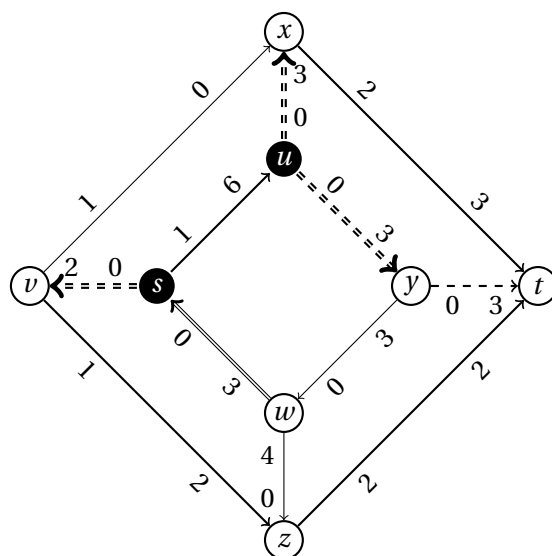
$$V_n = \{s, t\} \cup \{a_i, b_i \mid 1 \leq i \leq n\} \quad \text{in}$$

$$E_n = \{s \rightarrow a_i, a_i \rightarrow b_i, b_i \rightarrow t \mid 1 \leq i \leq n\} \cup \{b_i \rightarrow a_{i+1} \mid 1 \leq i \leq n-1\},$$

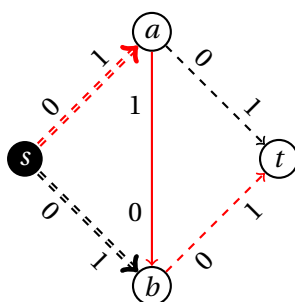
pri čemer imajo vse povezave kapaciteto 1. Maksimalen pretok  $n$  dobimo tako, da peljemo količino 1 po poteh  $s \rightarrow a_i \rightarrow b_i \rightarrow t$  ( $1 \leq i \leq n$ ) – minimalni prerez tedaj sestoji iz povezav  $s \rightarrow a_i$  ( $1 \leq i \leq n$ ). Če pa v prvem koraku algoritma GREEDYFLOW izberemo pot

$$s \rightarrow a_1 \rightarrow b_1 \rightarrow a_2 \rightarrow b_2 \rightarrow \dots \rightarrow a_n \rightarrow b_n \rightarrow t,$$

postanejo vse povezave v tej poti zasičene in se zato odstranijo iz grafa, zaradi česar algoritem druge povečujoče poti ne najde in tako konča s pretokom 1. Tako dobimo razliko  $n - 1$  in razmerje  $n$  za poljubno velik  $n$ .



Slika 127: Maksimalni pretok in minimalni prerez za nalogo 8.6(a).



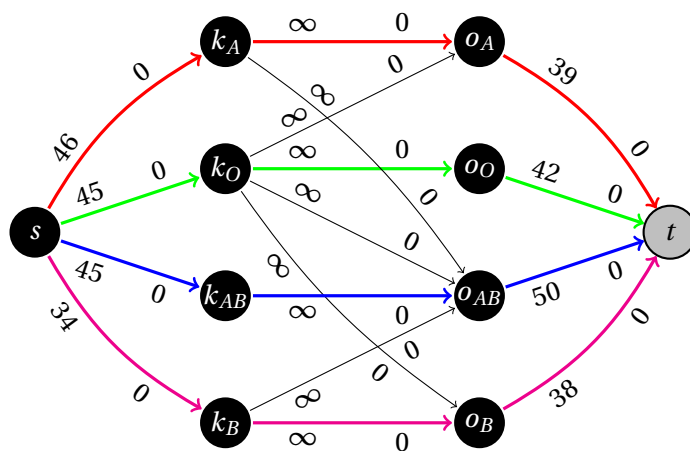
Slika 128: Primer omrežja in povečujoče poti za nalogo 8.6(b).

### Naloga 8.7.

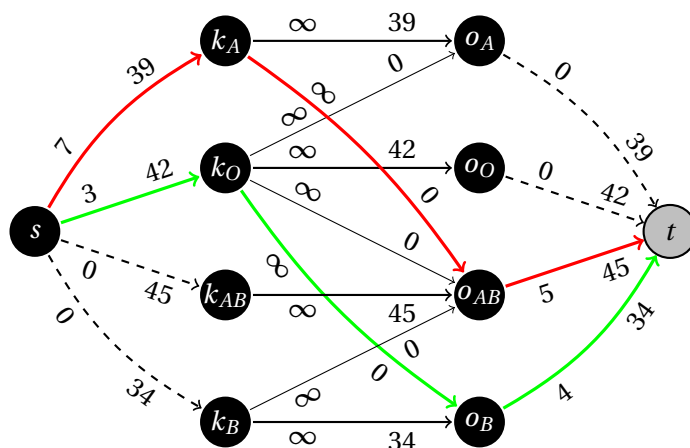
Za naravno število  $n$  naj bo  $G_n = (V_n, E_n)$  usmerjena pot od  $s$  do  $t$  z  $n$  povezavami, tj.,

$$V_n = \{s = v_0, v_1, \dots, v_n = t\} \quad \text{in} \\ E_n = \{v_{i-1} \rightarrow v_i \mid 1 \leq i \leq n\},$$

kjer imajo vse povezave kapaciteto 1. Maksimalni pretok v takem omrežju je tako 1, vsaka povezava pa sama sestavlja minimalen prerez. Tako graf  $G_n$  ustreza zahtevi iz točke (c). Za točko (a) lahko vzamemo  $G = G_1$ , za točko (b) pa  $G' = G_n$  za katerikoli  $n \geq 2$ .



Slika 129: Omrežje in prvi korak za nalogo 8.8.



Slika 130: Drugi korak za nalogo 8.8.

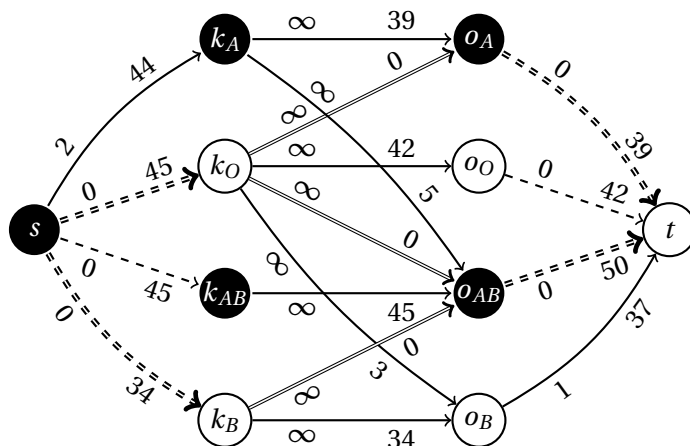
### Naloga 8.8.

Ustrezno omrežje je prikazano na sliki 129, kjer je prikazana tudi začetna izbira povečujočih poti za Ford-Fulkersonov algoritem. Nadaljevanje postopka je prikazano na sliki 130, maksimalni pretok pa je prikazan na sliki 131, iz katere razberemo, da lahko oskrbimo največ  $44 + 45 + 45 + 34 = 39 + 42 + 50 + 37 = 168$  pacientov.

### Naloga 8.9.

- (a) Iz slike 131 vidimo, da ostaneta neporabljeni dve enoti krvi. Ker sta povezuje  $s \rightarrow k_O$  in  $s \rightarrow k_{AB}$  v minimalnem prerezu ter obstaja cikel  $s \rightarrow k_A \rightarrow o_{AB} \leftarrow k_{AB} \leftarrow s$ , po katerem lahko pretok povečamo za največ 2, sledi, da sta neporabljeni enoti krvi iz krvnih skupin A ali AB.





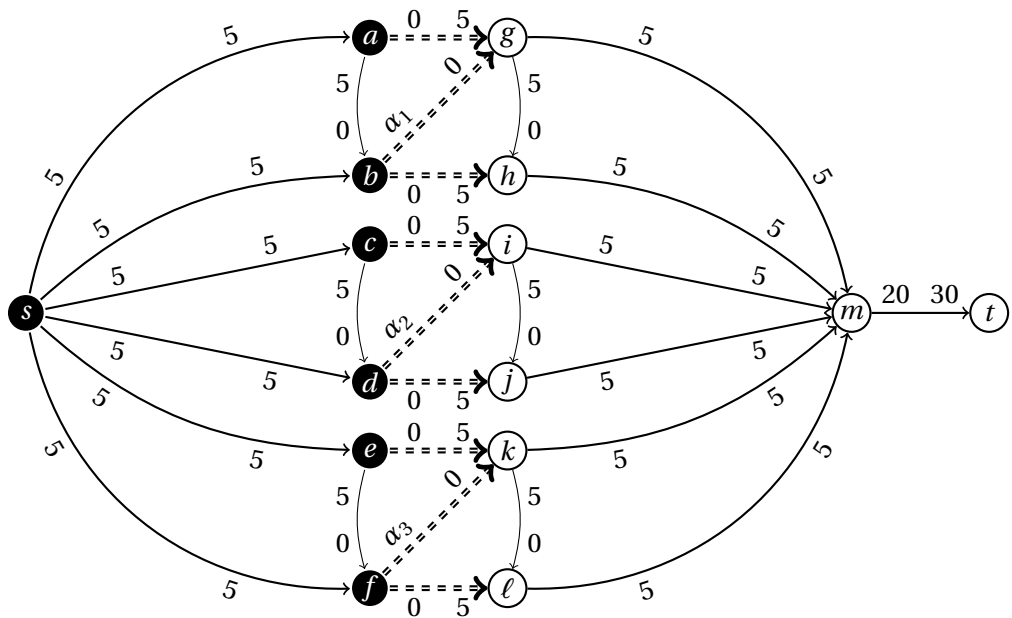
Slika 131: Maksimalni pretok in minimalni prerez za nalogo 8.8.

- (b) Iz slike 131 vidimo, da na kliniki potrebujejo kri še za eno osebo. Ker obstaja cikel  $t \leftarrow o_O \leftarrow k_O \rightarrow o_B \rightarrow t$ , po katerem lahko pretok povečamo za največ 1, ter sta povezavi  $s \rightarrow k_O$  in  $s \rightarrow k_{AB}$  v minimalnem prerezu in povezava  $k_B \rightarrow o_B$  ne predstavlja omejitve pri povečevanju pretoka, sledi, da morajo na kliniki naročiti enoto krvi iz krvne skupine O ali B.

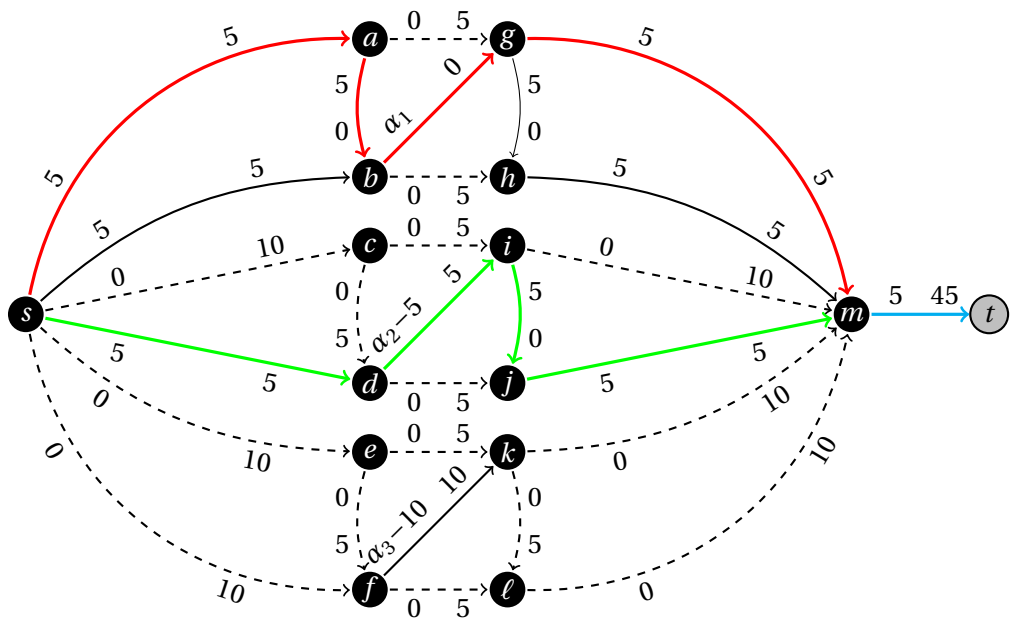
#### Naloga 8.10.

- (a) Po vsaki od poti  $s \rightarrow a \rightarrow g \rightarrow m \rightarrow t$ ,  $s \rightarrow b \rightarrow h \rightarrow m \rightarrow t$ ,  $s \rightarrow c \rightarrow i \rightarrow m \rightarrow t$ ,  $s \rightarrow d \rightarrow j \rightarrow m \rightarrow t$ ,  $s \rightarrow e \rightarrow k \rightarrow m \rightarrow t$  in  $s \rightarrow f \rightarrow \ell \rightarrow m \rightarrow t$  lahko povečamo pretok za 5. Skupni pretok je torej 30, glej sliko 132.
- (b) Na sliki 133 vidimo, kako lahko povečujemo pretok skozi čistilne naprave ob predpostavki, da so vrednosti  $\alpha_i$  ( $i \in \{1, 2, 3\}$ ) dovolj velike. Prva povečujoča pot je speljana skozi postajo  $\{a, b, g, h\}$  – po njej lahko pretok povečamo za največ 5. Druga povečujoča pot je speljana skozi postajo  $\{c, d, i, j\}$  – po njej lahko pretok prav tako povečamo za največ 5. Maksimalen pretok skozi čistilno postajo je prikazan za  $\{e, f, k, \ell\}$ . Tako lahko skozi posamezno postajo pretok povečamo za največ 10.

Iz slike 132 je razvidno, da lahko pretok čez povezavo  $m \rightarrow t$  povečamo za največ 20. Tako bomo postavili cevi skupne kapacitete 20, kar nas bo stalo  $20 \cdot 100\text{€} = 2000\text{€}$ . To lahko dosežemo z inštalacijo dveh cevi kapacitete 10, kar nas dodatno stane  $2 \cdot 10000\text{€} = 20000\text{€}$ . Skupna cena projekta je torej  $22000\text{€}$ .



Slika 132: Maksimalni pretok in minimalni prerez za nalogo 8.10(a).



Slika 133: Povečevanje pretokov za nalogo 8.10(b).

### Naloga 8.11.

- (a) Sestavimo omrežje  $G = (V, E)$ , kjer je množica vozlišč

$$V = \{s, t\} \cup \{w_i \mid 1 \leq i \leq p\} \cup \{x_j \mid 1 \leq j \leq q\} \cup \{y_k, z_k \mid 1 \leq k \leq n\},$$

v množici povezav  $E$  pa imamo povezave  $s \rightarrow w_i$  s kapaciteto  $u_i$  ( $1 \leq i \leq p$ ),  $s \rightarrow x_j$  s kapaciteto  $v_j$  ( $1 \leq j \leq q$ ),  $w_i \rightarrow y_k$  s kapaciteto 1, če je osebi  $k$  všeč sendvič  $i$ ,  $x_j \rightarrow z_k$  s kapaciteto 1, če je osebi  $k$  všeč sok  $j$ , ter  $y_k \rightarrow t$  in  $z_k \rightarrow t$  s kapaciteto 1 ( $1 \leq k \leq n$ ). Želena razdelitev hrane in pijače obstaja natanko tedaj, ko so v dobljenem maksimalnem pretoku vse povezave  $y_k \rightarrow t$  in  $z_k \rightarrow t$  ( $1 \leq k \leq n$ ) zasičene.

- (b) Ustrezno omrežje je prikazano na sliki 134, kjer je prikazana tudi začetna izbira povečujočih poti za Ford-Fulkersonov algoritem. Oznake v indeksih vozlišč ustrezajo prvim črkam v imenih sendvičev, sokov in oseb (brez strešic na šumnikih). Nadaljevanje postopka je prikazano na sliki 135, maksimalni pretok pa je prikazan na sliki 136. Ker v dobljenem maksimalnem pretoku povezava  $y_M \rightarrow t$  ni zasičena, sledi, da ne obstaja zelena razdelitev sendvičev. Ker so pa vse povezave  $x_j \rightarrow z_k$  zasičene ( $j \in \{A, B, C, J, R\}$ ,  $k \in \{B, D, J, M, P, S\}$ ), sledi, da zelena razdelitev sokov obstaja.

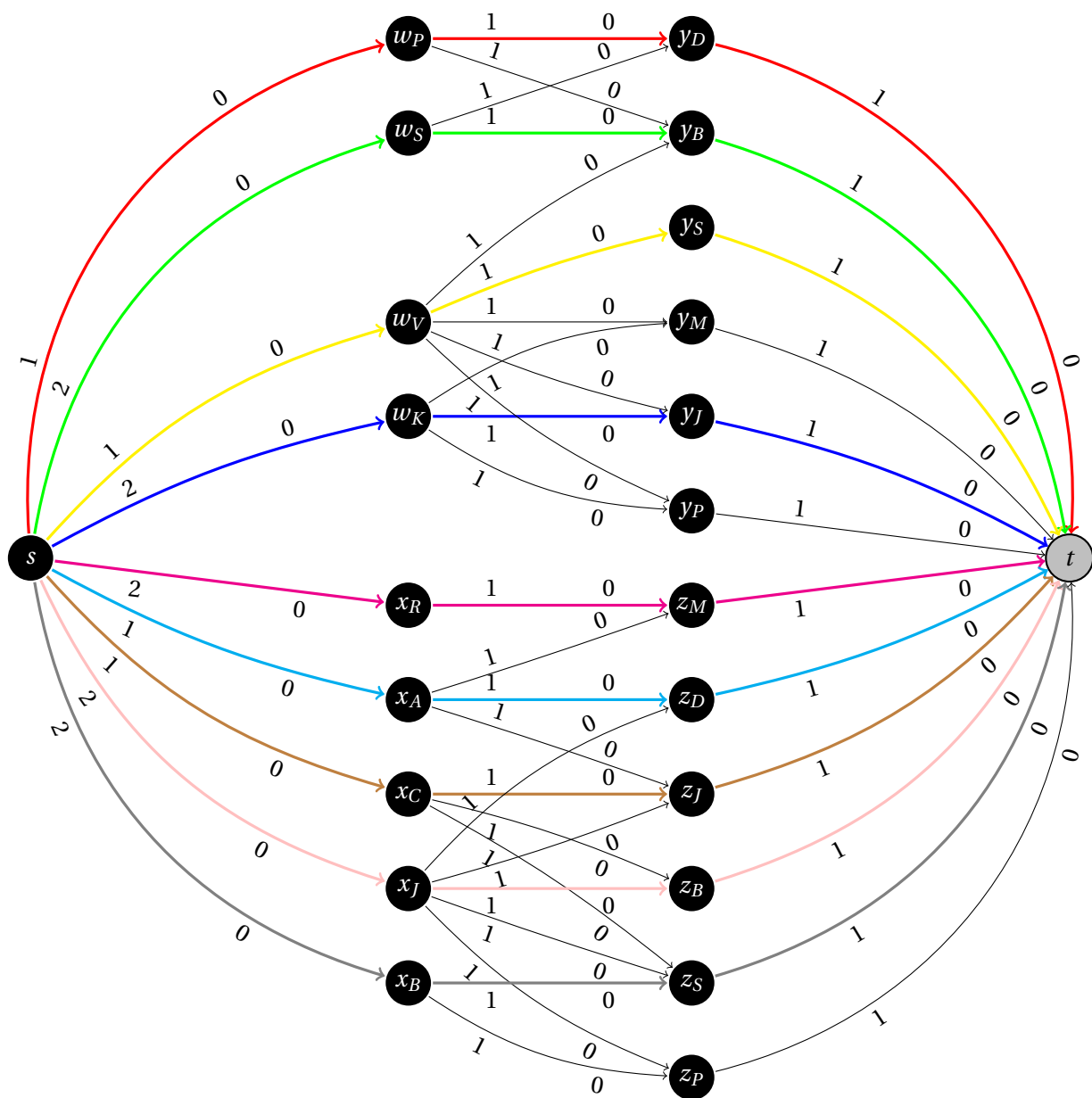
### Naloga 8.12.

Zapišimo želeni algoritem.

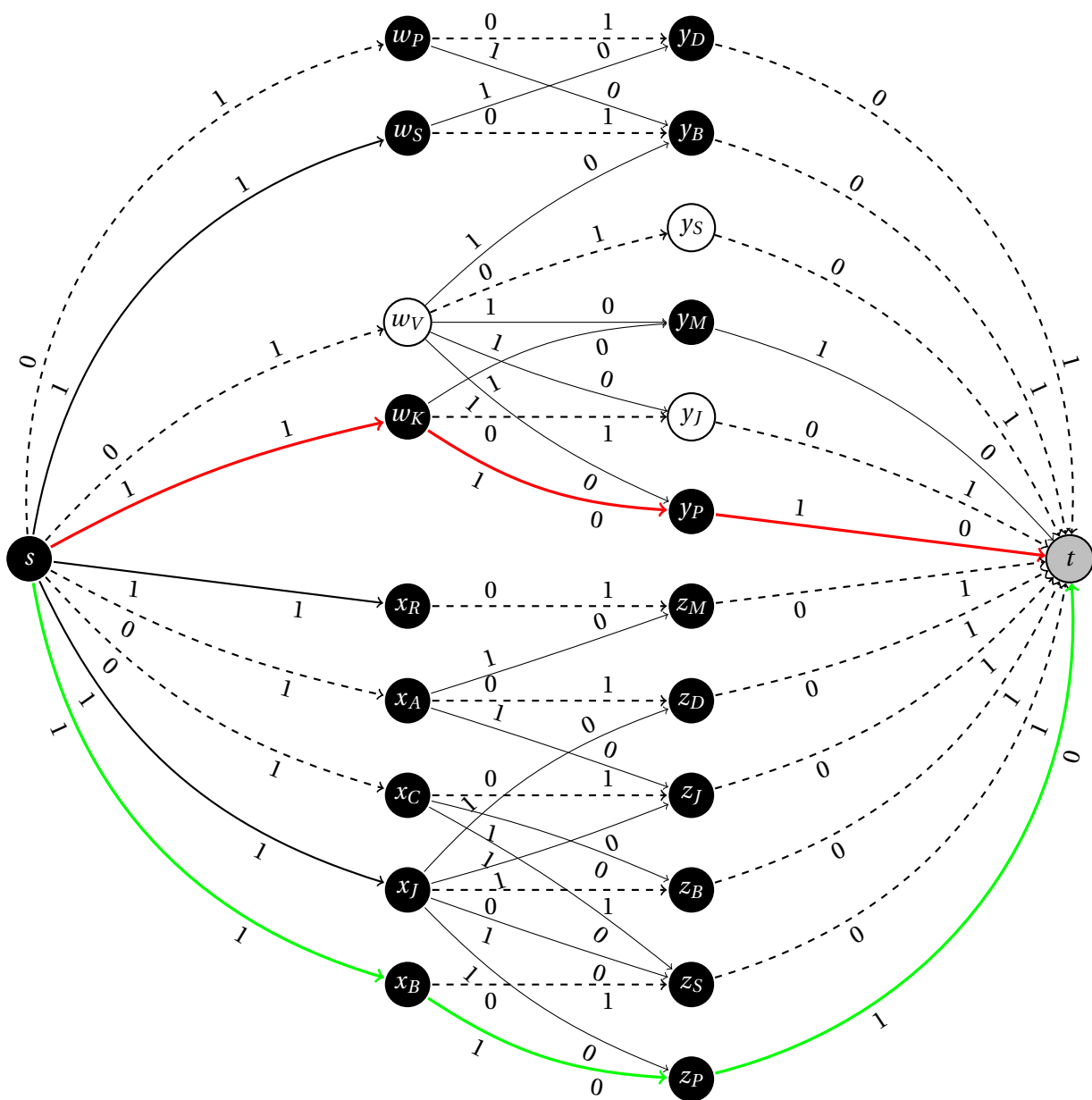
```

function NAČRTRPOV( $n, m, L$ )
  if kateri od vnosov v  $L$  se ponovi then
    return Izvedba ni možna – lokacije bank niso vse različne
  end if
   $V \leftarrow \{s, t\} \cup \{u_{ij}, v_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq m\}$ 
   $E \leftarrow \{s \rightarrow u_{ij} \mid (i, j) \in L\}$ 
     $\cup \{u_{ij} \rightarrow v_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq m\}$ 
     $\cup \{v_{ij} \rightarrow u_{i-1,j}, v_{i-1,j} \rightarrow u_{ij} \mid 2 \leq i \leq n, 1 \leq j \leq m\}$ 
     $\cup \{v_{ij} \rightarrow u_{i,j-1}, v_{i,j-1} \rightarrow u_{ij} \mid 1 \leq i \leq n, 2 \leq j \leq m\}$ 
     $\cup \{v_{i1} \rightarrow t, v_{im} \rightarrow t \mid 2 \leq i \leq n-1\}$ 
     $\cup \{v_{1j} \rightarrow t, v_{nj} \rightarrow t \mid 2 \leq j \leq m-1\}$ 
     $\cup \{v_{ij} \rightarrow t \mid i \in \{1, n\}, j \in \{1, m\}\}$ 
   $C \leftarrow$  slovar kapacitet z vrednostjo 1 za vsako povezavo iz  $E$ 
   $F \leftarrow$  FORDFULKERSON( $G = (E, V), C, s, t$ )      vrne slovar pretokov za vsak  $e \in E$ 
  if  $\exists (i, j) \in L : F[s \rightarrow u_{ij}] < 1$  then
    return Izvedba ni možna – ni mogoče zagotoviti disjunktnih izhodnih poti
  end if
   $S \leftarrow$  prazen seznam
  for  $(i, j) \in L$  do                                sestavimo poti
     $P \leftarrow$  prazen seznam
     $x \leftarrow u_{ij}$ 

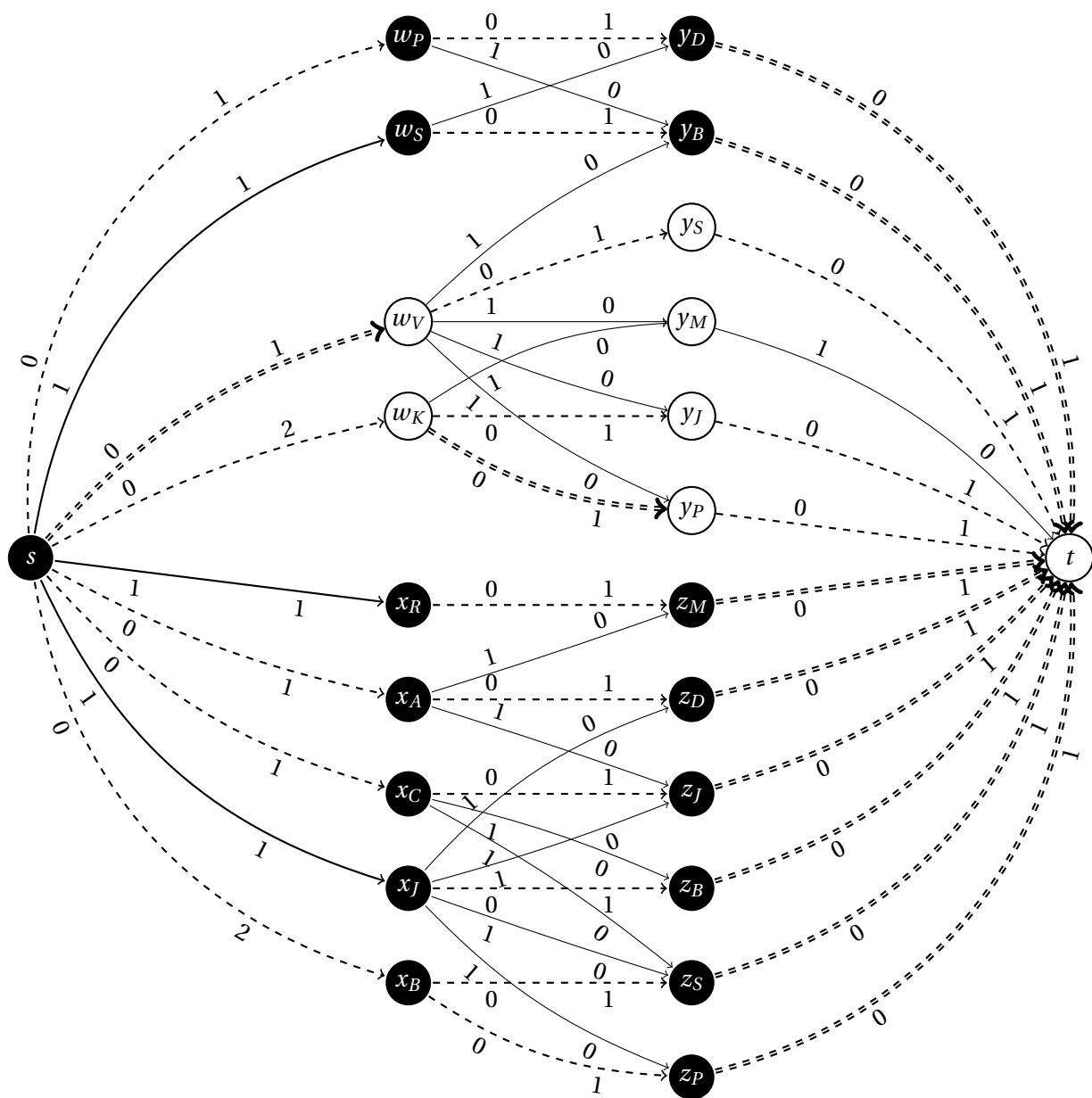
```



Slika 134: Omrežje in prvi korak za nalogo 8.11(b).



Slika 135: Drugi korak za nalogo 8.11(b).



Slika 136: Maksimalni prerez in minimalni pretok za nalogo 8.11(b).

```

while  $x \neq t$  do
     $P.append((i, j))$ 
     $i', j' \leftarrow$  indeksa, da je  $x = u_{i'j'}$ 
     $x' \leftarrow$  vozlišče, da je  $F[v_{i'j'} \rightarrow x'] = 1$ 
     $x, i, j \leftarrow x', i', j'$ 
end while
 $S.append(P)$ 
end for
return  $S$ 
end function

```

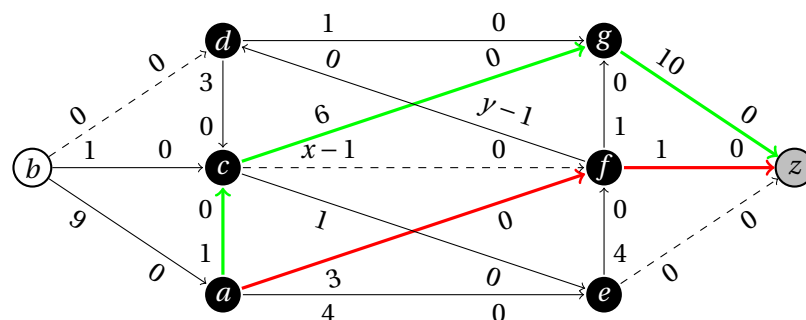
Algoritem sestavi usmerjen graf, v katerem je križišče na lokaciji  $(i, j)$  predstavljeno s povezavo  $u_{ij} \rightarrow v_{ij}$ , med sosednjima križiščema  $(i, j)$  in  $(i', j')$  pa sta povezavi  $v_{ij} \rightarrow u_{i'j'}$  in  $v_{i'j'} \rightarrow u_{ij}$ . Poleg tega so v grafu še povezave od izvora  $s$  do vozlišča  $u_{ij}$  za vsako banko na križišču  $(i, j)$ , ki jo želijo oropati, in povezave  $v_{ij}$  do ponora  $t$  za vsako križišče  $(i, j)$  na robu centra. Ker imajo vse povezave kapaciteto 1, bo mogoče skozi vsako križišče potovati največ enkrat – tako bo mogoče tudi vsako ulico uporabiti največ enkrat. Maksimalen pretok bo  $|L|$  (ali ekvivalentno, pretok po povezavi  $s \rightarrow u_{ij}$  bo 1 za vsak  $(i, j) \in L$ ) natanko tedaj, ko je mogoče za vsako banko najti pot iz centra, ne da bi se te poti srečale v kakšnem križišču.

### Naloga 8.13.

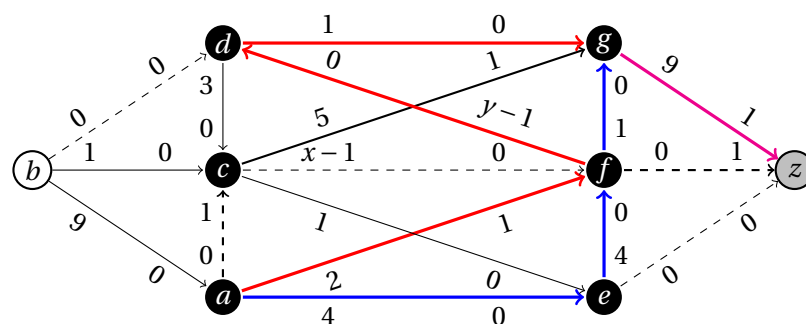
Delali bomo z omrežjem, v katerem je kapaciteta povezav za 1 manjša od nosilnosti ustreznega rova.

- (a) Začetno omrežje je prikazano na sliki 137, kjer sta prikazani tudi povečujoči poti  $a \rightarrow f \rightarrow z$  in  $a \rightarrow c \rightarrow g \rightarrow z$  s kapaciteto 1. Naslednji dve poti,  $a \rightarrow e \rightarrow f \rightarrow g \rightarrow z$  in  $a \rightarrow f \rightarrow d \rightarrow g \rightarrow z$ , prav tako s kapaciteto 1, sta prikazani na sliki 138. Tako dobimo omrežje s slike 139, iz katerega je razvidno, da je maksimalni pretok 4. Organizatorji lahko torej po trasi spustijo 4 škrate, npr. po vsaki od prej omenjenih poti po enega.
- (b) Iz slike 139 je razvidno, da gre povezava  $c \rightarrow f$  v nasprotni smeri minimalnega prereza, tako da s povečevanjem njene kapacitete ne bomo povečali maksimalnega pretoka, medtem ko je povezava  $f \rightarrow d$  v minimalnem prerezu. Za primer  $y > 2$  je prikazana je povečujoča pot  $a \rightarrow e \rightarrow f \rightarrow d \rightarrow c \rightarrow g \rightarrow z$ , po kateri lahko pretok povečamo za največ 3.

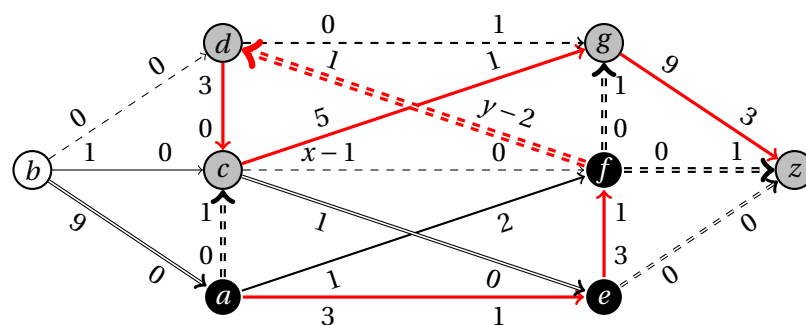
Dobljeno omrežje za primer  $y \geq 5$  je prikazano na sliki 140. Opazimo, da lahko v primeru  $y = 5$  v prikazanem minimalnem prerezu povezavi  $d \rightarrow c$  in  $d \rightarrow g$  nadomestimo s povezavo  $f \rightarrow d$ . Tako vidimo, da se nam vrednost  $y$  izplača povečati na največ 5 – ker je bila predhodna vrednost enaka 2, to ustreza 30 dnevom dela. Tedaj bodo organizatorji lahko spustili še 3 škrate po prej omenjeni poti, s čimer bodo lahko število udeležencev povečali na 7.



Slika 137: Omrežje in prvi korak za nalogo 8.13(a).



Slika 138: Drugi korak za nalogo 8.13(a).



Slika 139: Maksimalni pretok in minimalni prerez za nalogo 8.13(a) ter povečujoča pot pri  $y > 2$ .

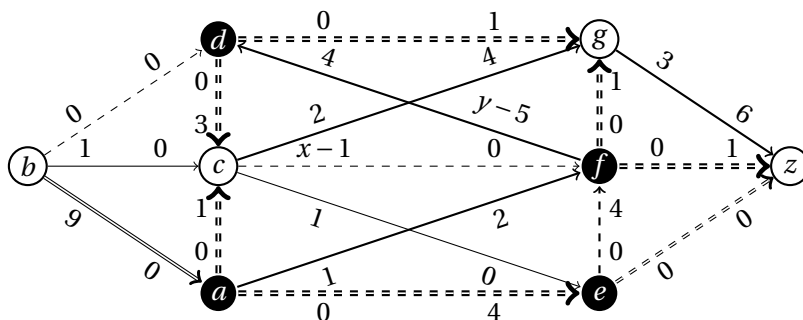
#### Naloga 8.14.

Naj bo  $G = (V, E)$  graf, ki predstavlja naše omrežje, pri čemer velja  $s, t \in V$ . Definirajmo omrežje  $G' = (V', E')$ , kjer je

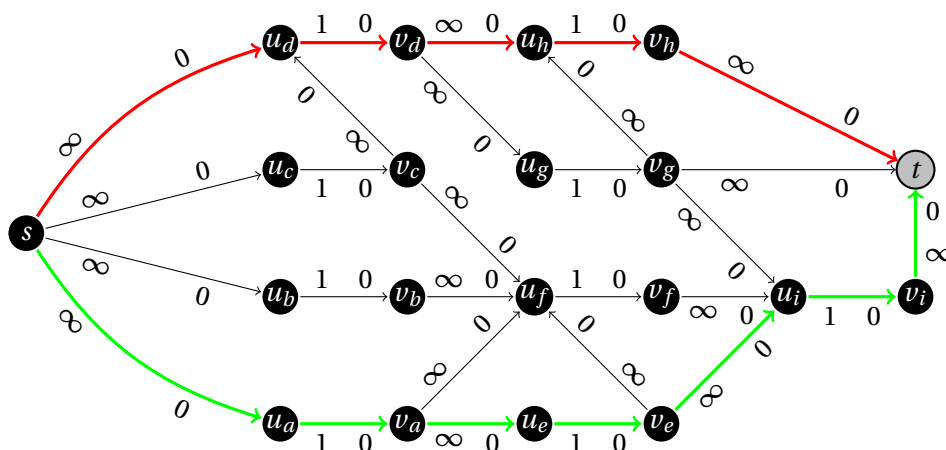
$$V' = \{s = v_s, t = u_t\} \cup \{u_x, v_x \mid x \in V \setminus \{s, t\}\} \quad \text{in} \\ E' = \{v_x \rightarrow u_y \mid x \rightarrow y \in E\} \cup \{u_x \rightarrow v_x \mid x \in V \setminus \{s, t\}\},$$

povezave  $v_x \rightarrow u_y$  ( $x \rightarrow y \in E$ ) imajo kapaciteto  $\infty$ , povezave  $u_x \rightarrow v_x$  ( $x \in V \setminus \{s, t\}$ )





Slika 140: Maksimalni pretok in minimalni prerez za nalogo 8.13(b).



Slika 141: Omrežje in povečujoče poti za nalogo 8.14.

pa imajo kapaciteto 1. Maksimalni pretok v omrežju ustreza številu strežnikov, ki jih moramo onemogočiti, da prekinemo komunikacijo od  $s$  do  $t$ , minimalni prerez pa vsebuje povezave  $u_x \rightarrow v_x$  za tiste strežnike  $x$ , ki jih moramo onemogočiti.

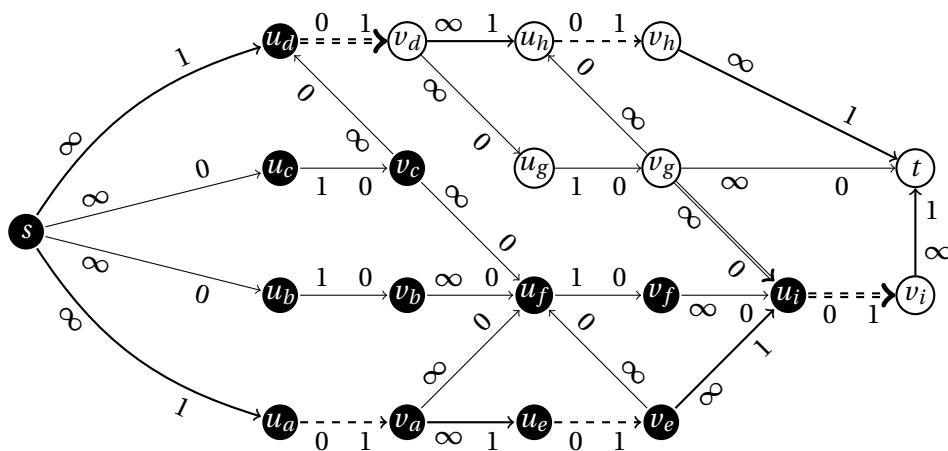
Omrežje za graf s slike 46 je prikazano na sliki 141, kjer sta prikazani tudi povečujoči poti, s katerima pridemo do maksimalnega pretoka 2 in minimalnega prereza, prikazanega na sliki 142. Iz slike je razvidno, da moramo za prekinitev komunikacije od  $s$  do  $t$  onemogočiti strežnika  $d$  in  $i$ .

#### Naloga 8.15.

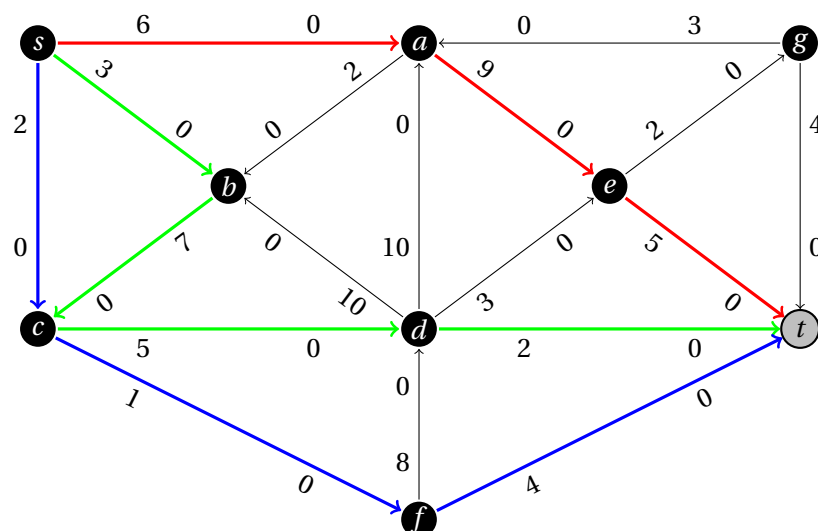
Poiščemo disjunktne povečujoče poti. Postopek reševanja je prikazan na slikah 143, 144, 145 in 146. Dobimo maksimalni pretok  $6 + 2 + 2 = 2 + 5 + 1 + 2 = 10$ .

#### Naloga 8.16.

Poiščemo disjunktne povečujoče poti. Postopek reševanja je prikazan na slikah 147, 148 in 149. Dobimo maksimalni pretok  $2 + 3 + 2 + 3 = 1 + 4 + 4 + 1 = 10$ .



Slika 142: Maksimalni pretok in minimalni prerez za nalogo 8.14.



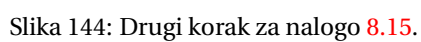
Slika 143: Prvi korak za nalogo 8.15.

### Naloga 8.17.

Najprej poiščemo maksimalni pretok za  $\alpha = 0$  (tj., brez povezave s parametrom), glej sliko 150. Dobimo maksimalni pretok  $0 + 3 = 3 + 0 + 0 = 3$ , glej sliko 151.

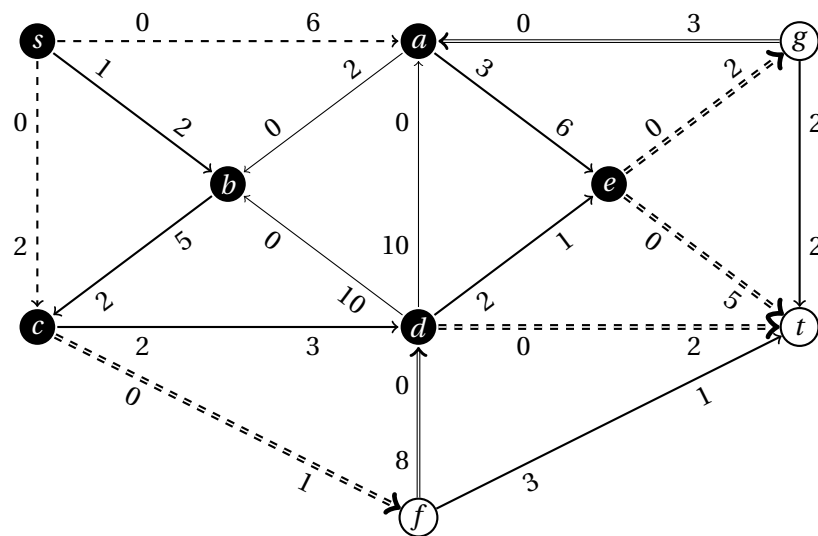
Za  $\alpha > 0$  poiščemo povečujočo pot. Če je  $0 < \alpha \leq 3$ , povečamo pretok po povečujoči poti za  $\alpha$  in tako dobimo maksimalen pretok  $p^* = \alpha + 3$  ( $3 < p^* \leq 6$ ), saj sta obe povezavi iz izvora zasičeni, glej sliko 152. V nasprotnem primeru ( $\alpha > 3$ ) pretok povečamo za 3 in poiščemo novo povečujočo pot.

Če je  $3 < \alpha \leq 5$ , povečamo pretok po povečujoči poti za  $\alpha - 3$  in tako dobimo maksimalen pretok  $p^* = \alpha + 3$  ( $6 < p^* \leq 8$ ), saj sta obe povezavi iz izvora zasičeni,

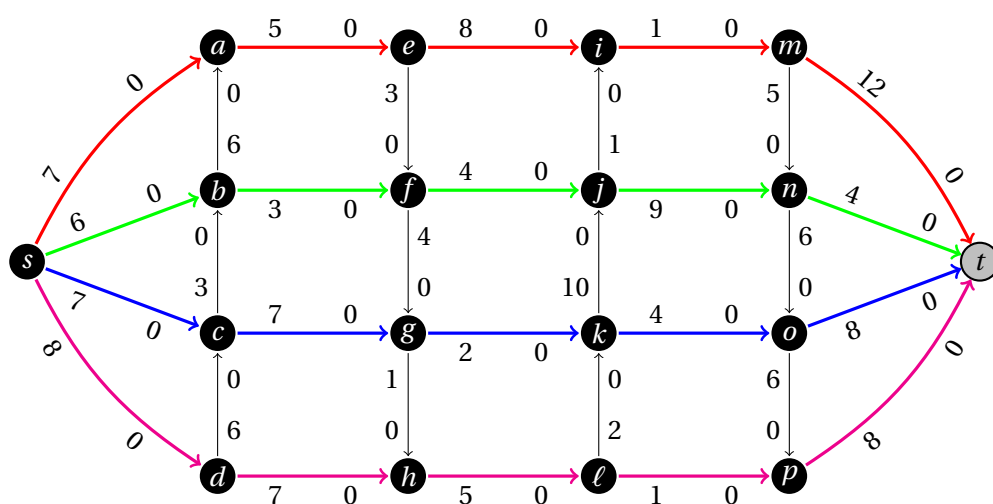


Za  $\alpha > 5$  dobimo maksimalni pretok  $5 + 3 = 3 + 2 + 3 = 8$ , glej sliko 154.

Poiščemo disjunktne povečujoče poti. Postopek reševanja je prikazan na slikah 155, 156, 157 in 158. Dobimo maksimalni pretok  $5 + 10 + 8 = 8 + 4 + 3 + 8 = 23$ . Če želimo povečati pretok, lahko povečamo kapaciteto take povezave v mini-

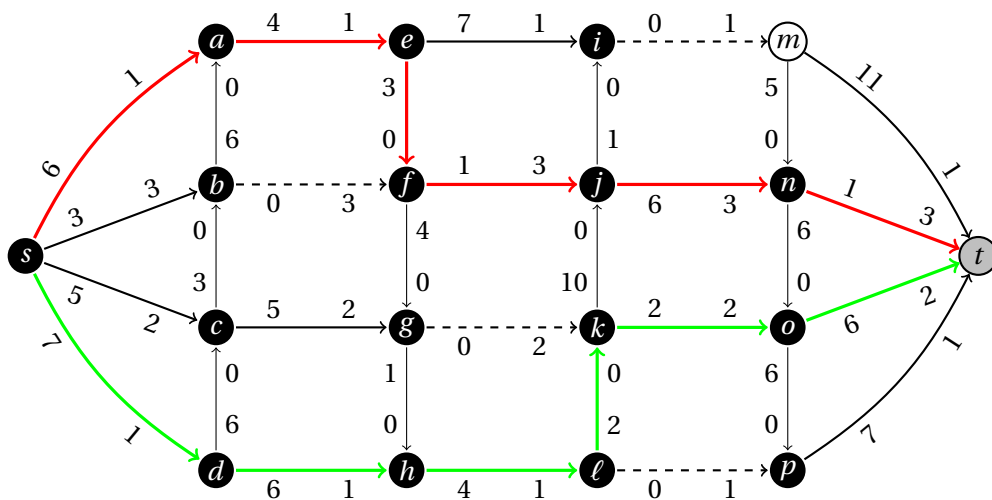


Slika 146: Maksimalni pretok in minimalni prerez za nalogo 8.15.

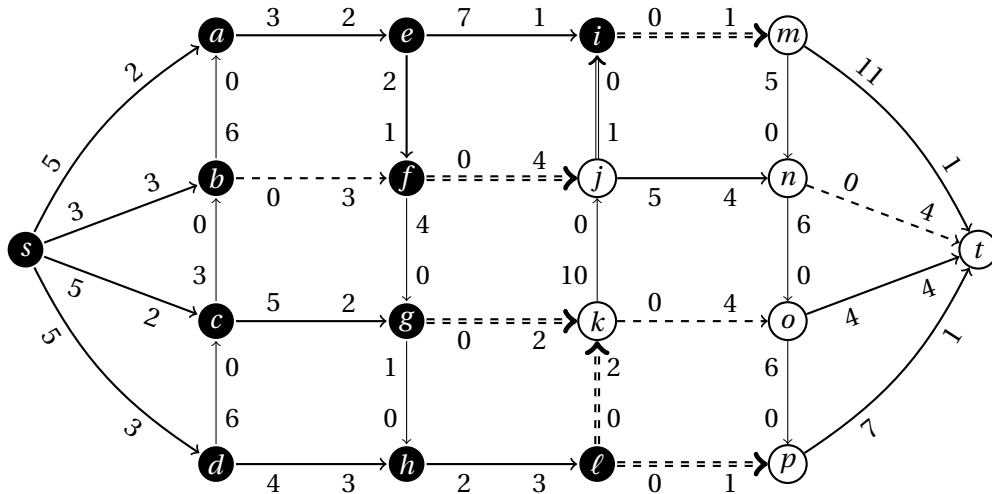


Slika 147: Prvi korak za nalogo 8.16.

malnem prerezu, da je mogoče od njenega končnega vozlišča priti do vozlišča  $T$  po nezasičenih (ali obratnih nepraznih) povezavah. Na grafu s slike 158 so taka vozlišča označena s sivo barvo, povezave, katerih kapaciteto se nam izplača povečati, pa z vijolično barvo.



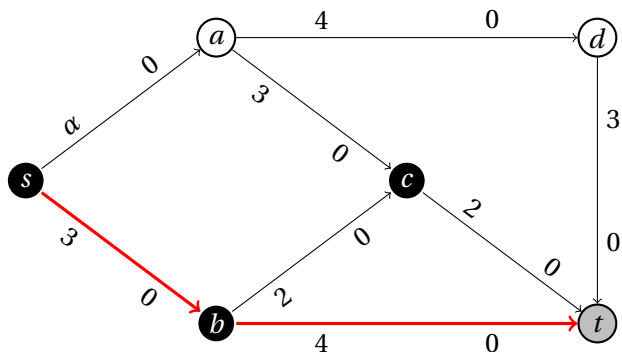
Slika 148: Drugi korak za nalogo 8.16.



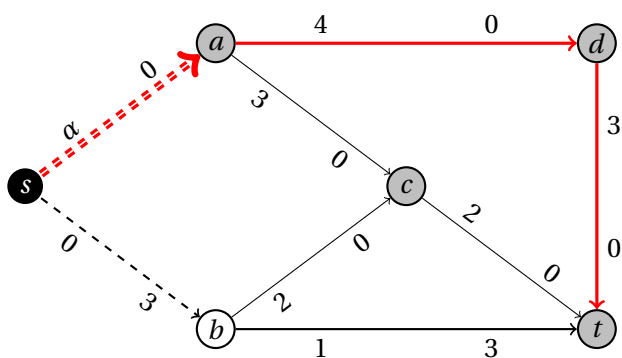
Slika 149: Maksimalni pretok in minimalni prerez za nalogo 8.16.

### Naloga 8.19.

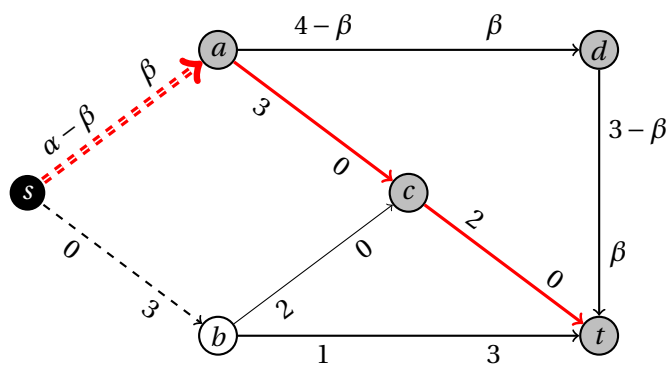
Če najdemo povečujočo pot z najmanjšo kapaciteto  $\infty$ , je problem neomejen. Postopek reševanja je prikazan na slikah 159, 160 in 161. Povečujoče poti s kapaciteto  $\infty$  ne najdemo, dobimo pa maksimalen pretok  $3 + 44 + 13 = 44 + 16 = 60$ .



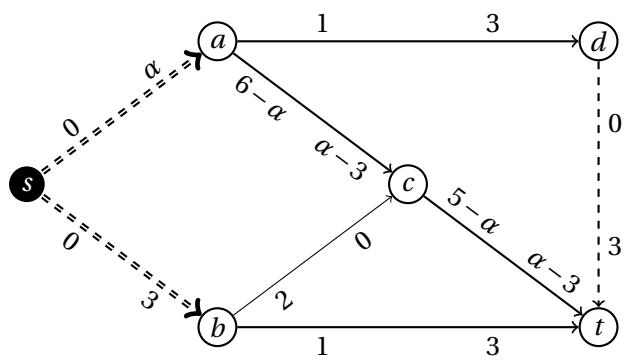
Slika 150: Prvi korak za nalogo 8.17.



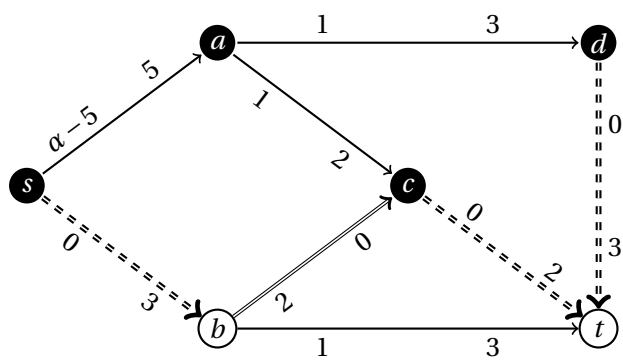
Slika 151: Maksimalni pretok pri  $\alpha = 0$  in drugi korak pri  $\alpha > 0$  za nalogo 8.17.



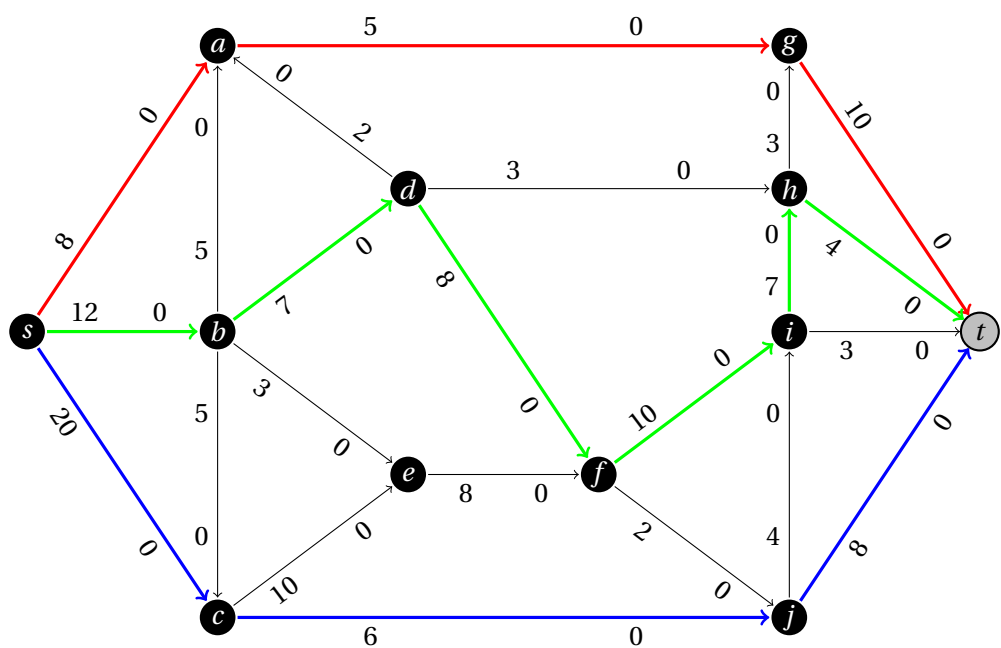
Slika 152: Maksimalni pretok pri  $0 < \alpha = \beta \leq 3$  in tretji korak pri  $\alpha > \beta = 3$  za nalogo 8.17.



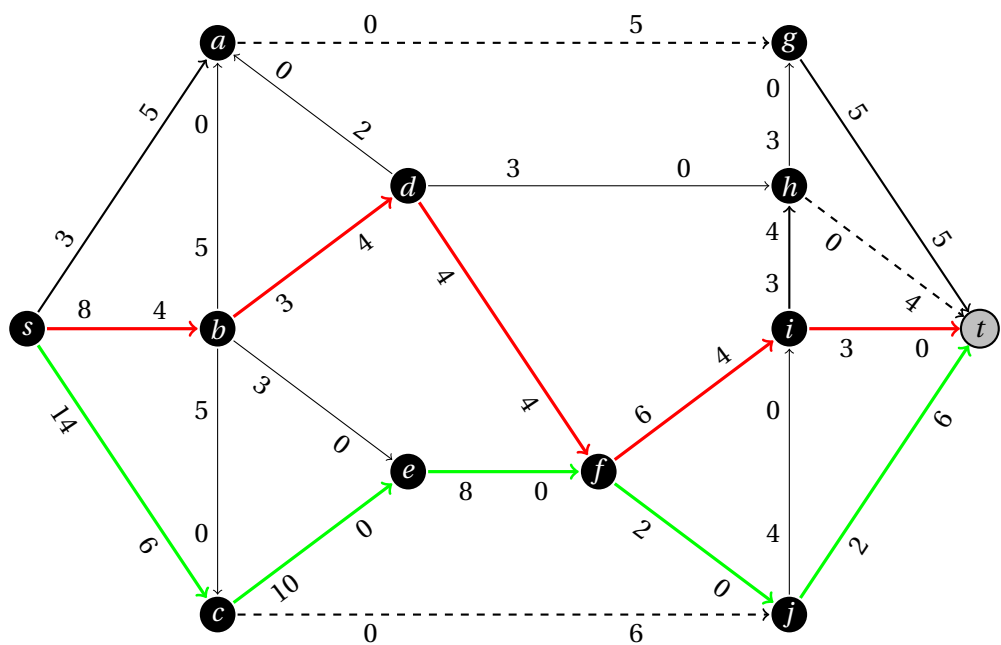
Slika 153: Maksimalni pretok pri  $3 < \alpha \leq 5$  za nalogo 8.17.



Slika 154: Maksimalni pretok pri  $\alpha > 5$  za nalogo 8.17.

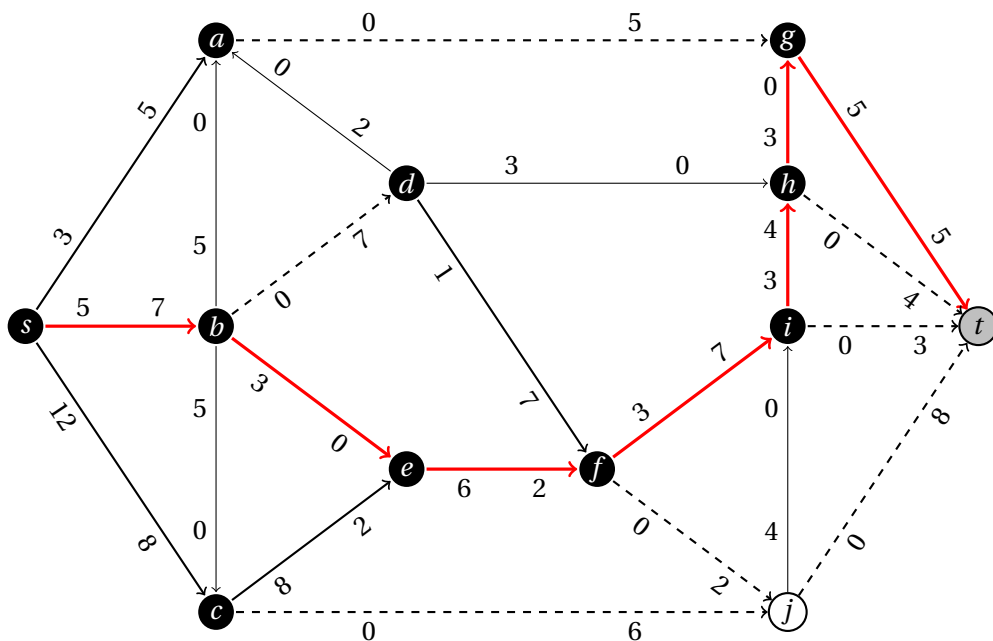


Slika 155: Prvi korak za nalogo 8.18.

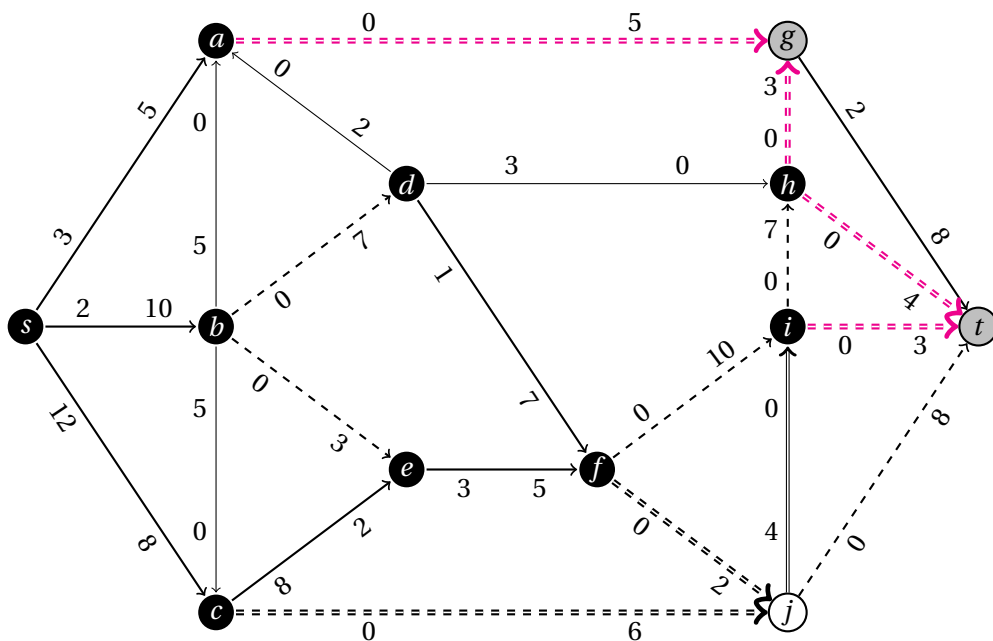


Slika 156: Drugi korak za nalogo 8.18.

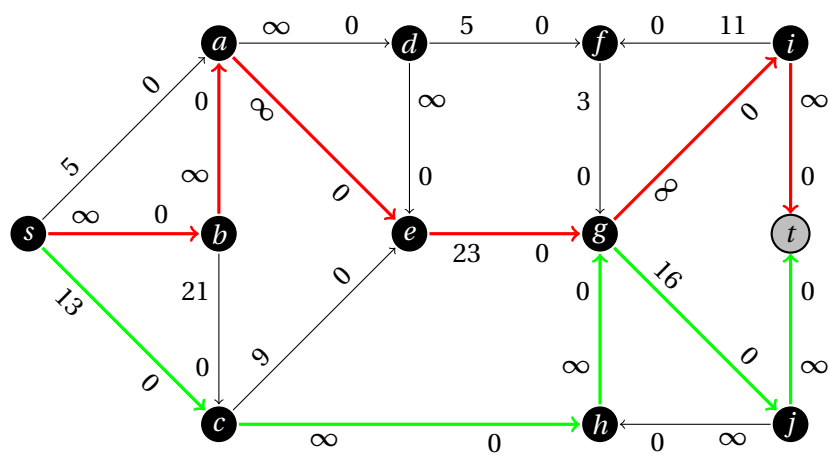




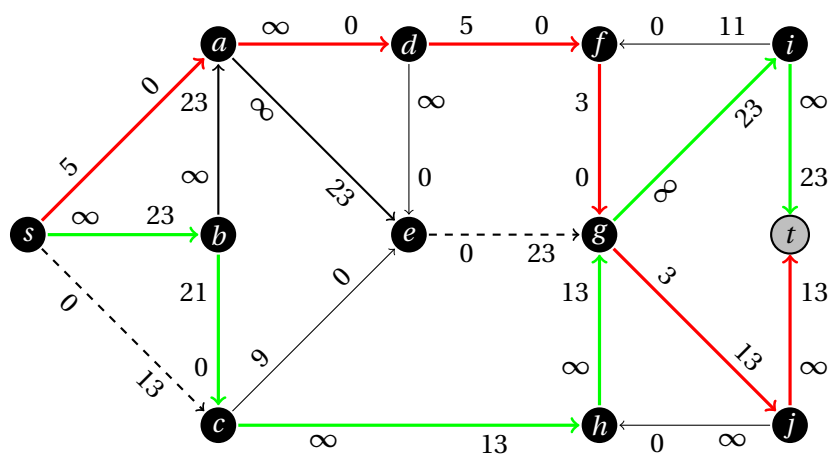
Slika 157: Tretji korak za nalogo 8.18.



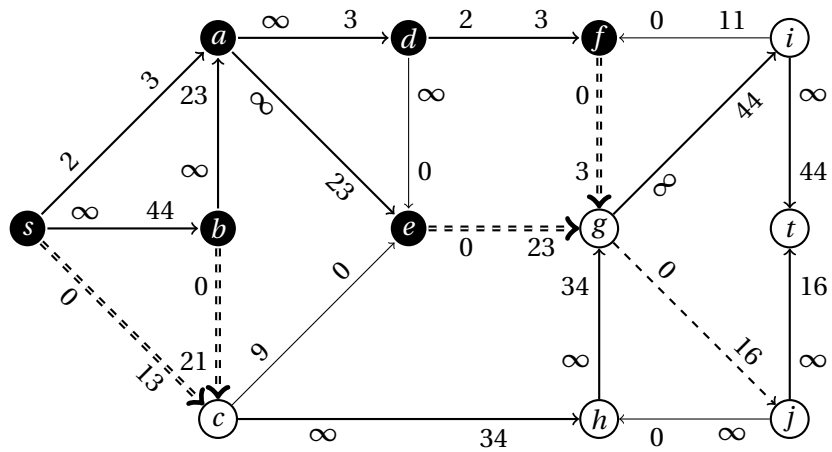
Slika 158: Maksimalni pretok in minimalni prerez za nalogo 8.18.



Slika 159: Prvi korak za nalogo 8.19.



Slika 160: Drugi korak za nalogo 8.19.



Slika 161: Maksimalni pretok in minimalni prerez za nalogo 8.19.



## 2.9 Razdelitve

### Naloga 9.1.

Celoto predstavimo z intervalom  $[0, 1]$ . Prvi igralec izbere vrednosti  $a_1, a_2, a_3$  ( $0 =: a_0 < a_1 < a_2 < a_3 < a_4 := 1$ ) tako, da intervali  $[0, a_1)$ ,  $[a_1, a_2)$ ,  $[a_2, a_3)$  in  $[a_3, 1]$  po njegovi ceni predstavljajo četrtno celote. Drugi igralec izbere indeks  $i \in \{1, 2, 3\}$ , tako da po njegovi ceni eden od intervalov  $[a_{i-1}, a_i)$  in  $[a_i, a_{i+1})$  predstavlja vsaj četrtno celote, drugi pa največ četrtno celote. Prvi igralec sedaj določi nepadajoči zvezni funkciji  $f, g : [0, 1] \rightarrow \mathbb{R}$ , za kateri velja  $f(0) = a_{i-1}$ ,  $f(1) = g(0) = a_i$  in  $g(1) = a_{i+1}$ , ter interval  $[f(x), g(x))$  za vsak  $x \in [0, 1]$  predstavlja četrtno celote po ceni prvega igralca. Drugi igralec nato izbere tak  $x \in [0, 1]$ , da interval  $[f(x), g(x))$  predstavlja četrtno celote tudi po njegovi ceni (tak  $x$  obstaja po izreku o vmesnih vrednostih). Prvi igralec tako dobi  $[f(x), g(x))$ , drugi igralec pa  $[0, f(x)) \cup [g(x), 1]$ .

### Naloga 9.2.

Najprej z madžarsko metodo z utežmi poiščemo tako razdelitev predmetov, da bo skupna cena čim večja.

$$\begin{array}{c}
 \begin{bmatrix} 11 & 9 & 19 & 14 & 11 \\ 4 & 10 & 15 & 20 & 12 \\ 6 & 5 & 5 & 17 & 9 \\ 20 & 23 & 6 & 3 & 18 \\ 10 & 18 & 23 & 9 & 10 \end{bmatrix} \rightarrow \begin{bmatrix} 8 & 10 & 0 & 5 & 8 \\ 16 & 10 & 5 & 0 & 8 \\ 11 & 12 & 12 & 0 & 8 \\ 3 & 0 & 17 & 20 & 5 \\ 13 & 5 & 0 & 14 & 13 \end{bmatrix} \rightarrow \begin{bmatrix} 5 & 10 & 0 & 5 & 3 \\ 13 & 10 & 5 & 0 & 3 \\ 8 & 12 & 12 & 0 & 3 \\ 0 & 0 & 17 & 20 & 0 \\ 10 & 5 & 0 & 14 & 8 \end{bmatrix} \rightarrow \\
 \rightarrow \begin{bmatrix} 2 & 7 & 0 & 5 & 0 \\ 10 & 7 & 5 & 0 & 0 \\ 5 & 9 & 12 & 0 & 0 \\ 0 & 0 & 20 & 23 & 0 \\ 7 & 2 & 0 & 14 & 5 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 5 & 0 & 5 & 0 \\ 8 & 5 & 5 & 0 & 0 \\ 3 & 7 & 12 & 0 & 0 \\ 0 & 0 & 22 & 25 & 2 \\ 5 & 0 & 0 & 14 & 5 \end{bmatrix} \Rightarrow \begin{bmatrix} 11 & 9 & 19 & 14 & 11 \\ 4 & 10 & 15 & 20 & 12 \\ 6 & 5 & 5 & 17 & 9 \\ 20 & 23 & 6 & 3 & 18 \\ 10 & 18 & 23 & 9 & 10 \end{bmatrix}
 \end{array}$$

Dobimo razdelitev predmetov s skupno ceno 86.

Da bomo predmete razdelili brez zavisti, bomo osebi  $j$  ( $1 \leq j \leq 5$ ) dodelili popust  $d_j$ . Smatramo, da oseba  $i$  zavida osebi  $j$  ( $1 \leq i, j \leq 5$ ), če je razlika med vrednostjo dodeljenega predmeta po ceni osebe  $i$  in plačano vrednostjo večja pri osebi  $j$  kot pri osebi  $i$ . Definirajmo torej

$$a_{ij} = b_{ij} - b_{jj} + d_j \quad (1 \leq i, j \leq 5),$$

kjer je  $b_{ij}$  cenitev predmeta, dodeljenega osebi  $j$ , s strani osebe  $i$ . Opazimo, da velja  $a_{jj} = d_j$  ( $1 \leq j \leq 5$ ). Iščemo matriko  $A = (a_{ij})_{i,j=1}^5$  z minimalno vsoto diagonalnih elementov, za katero velja  $a_{ij} \leq a_{ii}$  ( $1 \leq i, j \leq 5$ ).

Postopali bomo tako, da bomo na začetku popuste nastavili na 0, nato pa za osebi  $i, j$  z  $a_{ij} > a_{ii}$  povečali popust osebe  $i$  za  $a_{ij} - a_{ii}$  in postopek ponavljali,

dokler zavisti več ne bo.

$$\begin{aligned}
 & \begin{bmatrix} \mathbf{0} & -6 & \underline{2} & -14 & -4 \\ -7 & \mathbf{0} & \underline{3} & -13 & -8 \\ -5 & -3 & \mathbf{0} & -18 & -18 \\ \underline{9} & -17 & \underline{9} & \mathbf{0} & -17 \\ -1 & -11 & \underline{1} & -5 & \mathbf{0} \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{2} & -3 & 2 & -5 & -3 \\ -5 & \mathbf{3} & 3 & -4 & -7 \\ -3 & 0 & \mathbf{0} & -9 & -17 \\ \underline{11} & -14 & 9 & \mathbf{9} & -16 \\ 1 & -8 & 1 & \underline{4} & \mathbf{1} \end{bmatrix} \rightarrow \\
 & \rightarrow \begin{bmatrix} \mathbf{2} & -3 & 2 & -3 & 0 \\ -5 & \mathbf{3} & 3 & -2 & -4 \\ -3 & 0 & \mathbf{0} & -7 & -14 \\ 11 & -14 & 9 & \mathbf{11} & -13 \\ 1 & -8 & 1 & \underline{6} & \mathbf{4} \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{2} & -3 & 2 & -3 & 2 \\ -5 & \mathbf{3} & 3 & -2 & -2 \\ -3 & 0 & \mathbf{0} & -7 & -12 \\ 11 & -14 & 9 & \mathbf{11} & -11 \\ 1 & -8 & 1 & 6 & \mathbf{6} \end{bmatrix}
 \end{aligned}$$

Prvi osebi bomo torej dali prvi predmet po ceni  $11 - 2 = 9$ , drugi osebi četrti predmet po ceni  $20 - 3 = 17$ , tretji osebi peti predmet po ceni  $9 - 0 = 9$ , četrti osebi drugi predmet po ceni  $23 - 11 = 12$ , peti osebi pa tretji predmet po ceni  $23 - 6 = 17$ . Skupni zaslužek iz nadomestil je tako 64.

### Naloga 9.3.

- (a) Naj bo  $v[i][j]$  ( $1 \leq i \leq 2n$ ,  $1 \leq j \leq n$ )  $j$ -ta preferenca  $i$ -tega vodiča,  $d[j][i]$  ( $1 \leq i \leq 2n$ ,  $1 \leq j \leq n$ ) rang  $i$ -tega vodiča pri preferencah agencije za  $j$ -to destinacijo. Uporabili bomo varianto Gale-Shapleyevega algoritma, ki bo vsaki destinaciji dodelila dva vodiča.

**function** STABILNADODELITEV( $v[1 \dots 2n][1 \dots n]$ ,  $d[1 \dots n][1 \dots 2n]$ )

$Q \leftarrow [1, 2, \dots, 2n]$

$k \leftarrow$  seznam dolžine  $2n$  z vrednostmi 1

$M \leftarrow$  matrika velikosti  $n \times 2$  z vrednostmi NULL

**while**  $\neg Q.\text{isEmpty}()$  **do**

$i \leftarrow Q.\text{pop}()$

$j \leftarrow v[k[i]]$

$k[i] \leftarrow k[i] + 1$

$i' \leftarrow M[j][2]$

**if**  $i' = \text{NULL} \vee d[i] < d[i']$  **then**

**if**  $d[i] < d[M[j][1]]$  **then**

$M[j][2] \leftarrow M[j][1]$

$M[j][1] \leftarrow i$

**else**

$M[j][2] \leftarrow i$

**end if**

**if**  $i' \neq \text{NULL}$  **then**

$Q.\text{append}(i')$

**end if**

**else**

$Q.\text{append}(i)$

**end if**

**end while**

Q	k						M		
	A	B	C	Č	D	E	Budimpešta	Győr	Szeged
[A, B, C, Č, D, E]	1	1	1	1	1	1	[NULL, NULL]	[NULL, NULL]	[NULL, NULL]
[B, C, Č, D, E]	2	1	1	1	1	1	[NULL, NULL]	[NULL, NULL]	[A, NULL]
[C, Č, D, E]	2	2	1	1	1	1	[B, NULL]	[NULL, NULL]	[A, NULL]
[Č, D, E]	2	2	2	1	1	1	[B, NULL]	[C, NULL]	[A, NULL]
[D, E]	2	2	2	2	1	1	[B, Č]	[C, NULL]	[A, NULL]
[E, D]	2	2	2	2	2	1	[B, Č]	[C, NULL]	[A, NULL]
[D]	2	2	2	2	2	2	[B, Č]	[C, E]	[A, NULL]
[]	2	2	2	2	3	2	[B, Č]	[C, E]	[A, D]

Tabela 56: Potek izvajanja algoritma za nalogo 9.3(b).

**return M**  
**end function**

Ker se vsak od  $2n$  vodičev v vrsto Q doda največ  $n$ -krat, algoritem teče v času  $O(n^2)$ .

- (b) Izvedba algoritma STABILNADODELITEV pri danih podatkih je prikazana v tabeli 56. Budimpešto torej dodelimo Beli in Črtu, Győr Corini in Emi, Szeged pa Anisi in Dani.

#### Naloga 9.4.

Stabilno prirejanje, v katerem Mujo ni šef območja Hrvaške in BiH, ne obstaja. To lahko dokažemo tako, da predpostavimo, da imamo tako prirejanje, ter analiziramo možne primere in izpeljemo protislovje.

Denimo, da je Mujo šef območja Slovenije in Madžarske. Ker bi pri agenciji na tem mestu raje imeli Črta, bo ta moral biti šef območja z večjo preferenco, torej Srbije in Črne gore. V agenciji bi na tem mestu raje videli Žana, ki sta mu pa ostali le območji z nižjo preferenco. Prišli smo do protislovja.

Mujo torej ni šef območja Slovenije in Madžarske, pač pa območja z nižjo preferenco. Šef območja Slovenije in Madžarske je tako nekdo, ki ima pri agenciji višjo preferenco kot Mujo – tak je edino Črt. Za šefa območja Makedonije in Kosova bi na agenciji raje kot Muja imeli Šimeta – če je Mujo šef tega območja, je Šime šef območja z nižjo preferenco, protislovje. Mujo je torej šef območja Srbije in Črne gore, a bi v agenciji na tem mestu raje videli Žana, ki pa je potem šef območja z nižjo preferenco. Tako spet pridemo do protislovja, iz česar sklepamo, da stabilno prirejanje, v katerem Mujo ni šef območja Hrvaške in BiH, ne obstaja.





## Literatura

- [1] V. Batagelj in M. Kaufman, *Zbirka (deloma) rešenih nalog iz operacijskih raziskav. Druga poskusna izdaja*, 1997. <http://vlado.fmf.uni-lj.si/vlado/or/ORvaje.pdf>.
- [2] S. Dasgupta, C.H. Papadimitriou in U.V. Vazirani, *Algorithms*. McGraw-Hill Education, 2006.
- [3] F.S. Hillier in G.J. Lieberman, *Introduction to Operations Research. Seventh Edition*. McGraw-Hill Higher Education, 2001.
- [4] J. Povh in S. Pustavrh, *Zbirka rešenih nalog iz operacijskih raziskav*. Visoka šola za upravljanje in poslovanje, Novo mesto, 2001.
- [5] W.L. Winston, *Operations Research: Applications and Algorithms. Fourth Edition*. Thomson Learning, 2004.