

# OPERACIJSKE RAZISKAVE

## VAJE



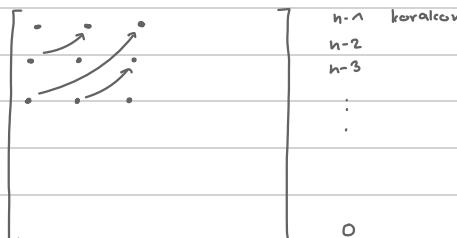
①  $A[1 \dots n][1 \dots n]$  matrika (seznam seznamov)

```

for i=1,...,n do
  for j=i+1,...,n do
    A[i][j] ← A[j][i]
  end for
end for

```

laj ta program naredi:



$$\text{število korakov: } \sum_{i=0}^{n-1} i = \frac{n(n-1)}{2} = \frac{n^2-n}{2} \in O(n^2)$$

$$f(n) \in O(g(n)) \Leftrightarrow \exists c > 0 \ \exists n_0 \in \mathbb{N} \ \forall n \geq n_0 : f(n) \leq c \cdot g(n)$$

②  $l[1, \dots, n]$  seznam, ki ima na začetku vse vrednosti nastavljene na 0

$i \leftarrow 1$

while  $i \leq n$  do

$n = 4$

$l[i] \leftarrow 1 - l[i]$

$l$	$i$
-----	-----

  if  $l[i] = 1$  then

0,0,0,0	1
---------	---

$i \leftarrow 1$

1,0,0,0	1
---------	---

  else

0,0,0,0	2
---------	---

$i \leftarrow i + 1$

0,1,0,0	1
---------	---

  end if

1,1,0,0	1
---------	---

end while

0,1,0,0	2
---------	---

0,0,1,0	3
---------	---

0,0,1,0	1
---------	---

1,0,1,0	1
---------	---

0,0,1,0	2
---------	---

0,1,1,0	1
---------	---

1,1,1,0	1
---------	---

0,1,1,0	2
---------	---

0,0,1,0	3
---------	---

0,0,0,0	4
---------	---

0,0,0,1	1
---------	---

:	
---	--

če pogledam samo tam kjer je ← vidimo

da je ta algoritem ubistven šteje v

binarnem zapisu

število korakov  $\geq 2^n$ , saj če gledam

samo tam kjer so ← bo vsaj  $2^n$  koraku

v primeru ko imamo  $n = 4$

bolj naturno št. korakov

$$2^n + 2^{n-1} + 2^{n-2} + \dots + 2^0 = 2^{n+1} - 1$$

korak, korak, se pravi, se pravi

(b) Algoritem obiše 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 potreben 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  $O(2^n)$ .

### ③ function Bubble Sort ( $l[1, \dots, n]$ )

```

 $z \leftarrow n$ 
while  $z > 1$  do
     $y \leftarrow 1$ 
    for  $i = 2, \dots, z$  do
        if  $l[i-1] > l[i]$  then
             $l[i-1], l[i] \leftarrow l[i], l[i-1]$ 
             $y \leftarrow i$ 
        end if
    end for
     $z \leftarrow y-1$ 
end while
end function

```

$$l = [7 \ 11 \ 16 \ 7 \ 5]$$

$z$	$y$	$i$	$l$
5	1	2	[7 11 16 7 5]
		3	
		4	[7 11 7 16 5]
		4	[7 11 7 5 16]
	5		
		4	
		5	
		1	
		2	
		3	[7 7 11 5 16]
		3	[7 7 5 11 16]
3	4		
		2	
		3	[7 5 7 11 16]
2	3		
		2	
		1	[5 7 7 11 16]
1	2		

St. korekciou: najmanj  $O(n)$

$$\frac{n^2-n}{2} = O(n^2)$$

### ④ MergeSort

- vhodni seznam razdeli na dva dela
- vsak del rekurzivno uredi
- zlije dobijena urejena seznamu

MergeSort ( $l[1, \dots, n]$ )  $\rightarrow$  naj bo časovna zahodnost tega  $T(n)$

if  $n \leq 1$

return  $l$   $O(1)$

$$m = \lfloor \frac{n}{2} \rfloor$$

$$l_1 = \text{mergesort}(l[1, \dots, m])$$

$$T(\frac{n}{2}) + O(n)$$

$$l_2 = \text{mergesort}(l[m+1, \dots, n])$$

$$T(\frac{n}{2}) + O(n)$$

$$l = []$$

$$i, j \leftarrow 1, 1$$

while  $i \leq m \wedge j \leq n-m$

if  $l_1[i] < l_2[j]$

$l.append(l_1[i])$

$i = i + 1$

else

$l.append(l_2[j])$

$j = j + 1$

$l.extend(l_1[i, \dots, m])$

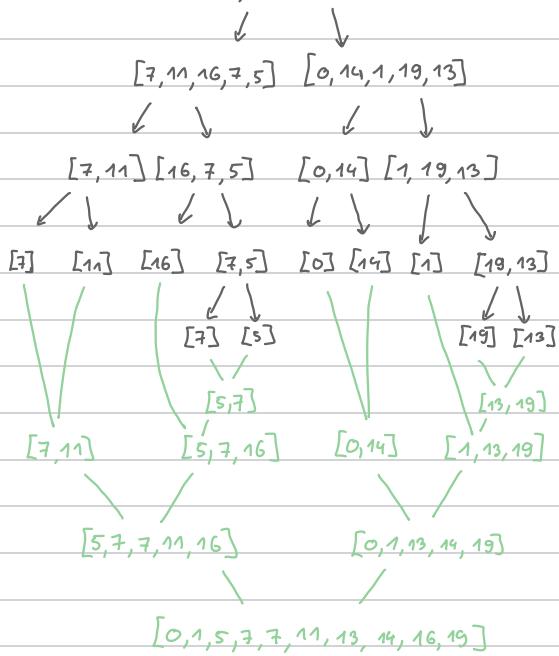
$l.extend(l_2[j, \dots, n-m])$

return  $l$

$O(n)$

$O(n)$

Rešimo primer  $\ell = [7, 11, 16, 7, 5, 0, 14, 1, 19, 13]$



časovna zahtevnost:

$$T(n) = O(n)$$

[glej zapise podleg algoritmu]

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

$$\begin{aligned} T(n) &\leq 2 \cdot T\left(\frac{n}{2}\right) + c \cdot n \\ &\leq 2 \left(2 T\left(\frac{n}{4}\right) + c \cdot \frac{n}{2}\right) + c \cdot n \\ &= 4 \cdot T\left(\frac{n}{4}\right) + 2 \cdot c \cdot n \\ &\leq 4 \left(2 \cdot T\left(\frac{n}{8}\right) + c \cdot \frac{n}{4}\right) + 2 \cdot c \cdot n \\ &= 8 \cdot T\left(\frac{n}{8}\right) + 3 \cdot c \cdot n \end{aligned}$$

$$\begin{aligned} &\leq 2^{\log_2 n} \cdot O(1) + \log_2 n \cdot c \cdot n \\ &= O(n) + O(n \cdot \log n) = O(n \cdot \log n) \end{aligned}$$

### CELOŠTEVILSKI LINEARNI PROGRAM

$$\begin{cases} \max c^T x \\ \text{p.p. } Ax \leq b \\ x_i \in \mathbb{Z} \end{cases}$$

postopek za reševanje:

- opis spremenljivk
- ciljna funkcija
- omejitve

- ① 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.

$$\mathcal{P} = \{p_1, p_2, \dots, p_n\}$$

$p_i \dots$  projekt

$M \dots$  kapital

$N \dots$  št. mest

$p_i \rightarrow$  potrebuje  $d_i \in \mathbb{E}$  finančne pomoči

$\rightarrow$  zaposli  $a_i$  ljudi

$\rightarrow$  donos  $c_i \in \mathbb{E}$

Kateri  $p_i$  naj občina financira za  
max donos?

$$x_i = \begin{cases} 0 & ; \text{se ne odloči za } p_i \\ 1 & ; \text{se odloči za } p_i \end{cases} \quad \text{za } i=1, \dots, n$$

CLP:

$$\max \sum_{i=1}^n c_i x_i$$

p.p.  $x_i \in \mathbb{Z}$

$$0 \leq x_i \leq 1 \quad \text{za } i=1, \dots, n$$

$$\sum_{i=1}^n x_i a_i \geq N$$

$$\sum_{i=1}^n x_i \cdot d_i \leq M$$

2. Obravnavajmo pospoljen scenarij iz prejšnje naloge.

- (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  $p_j$ .

$$x_i = 1 \implies x_j = 1$$

Torej bomo v CLP dodali pogoj:  $x_i \leq x_j \quad \forall (p_i, p_j) \in V$

- (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.

$$\text{Dodali bi pogoj: } \sum_{p_i \in H} x_i \leq 1 \quad \text{za } \forall H \in S$$

**3. (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.

usako skladitse lehko  
dostavlja najvec eni  
stranki, nekatere  
stranke potrebujo vec  
kot eno dobroino

h...št. skladisč

fi... strški najema i-tega skladisca

$$i=1, \dots, n$$

m ... st. strank

Cij... strošek dostave dobrine stranki; j je skladisca i

dj... št. enot, ki jih j-ta stranka potrebuje (djEZ)

kje naj hranimo za min stroške?

## Uvedemo spremenljivke

$$X_{ij} = \begin{cases} 1 & ; \text{ stranki}_j \text{ dostawiono z skladisca i} \\ 0 & ; \text{ sicer} \end{cases}$$

$$\underline{CLP}: \quad \min \quad \sum_{\lambda}^n \sum_j^m x_{ij} \cdot c_{ij} + x_{ij} \cdot f_{\lambda} = \sum_{\lambda}^n \sum_j^m x_{ij} (c_{ij} + f_{\lambda})$$

$$\begin{array}{l} \text{p.p. } x_{ij} \in \mathbb{Z} \\ 0 \leq x_{ij} \leq 1 \\ \sum_j x_{ij} = 1 \\ \sum_i x_{ij} = d_j \end{array} \quad \left[ \begin{array}{c} \text{za } i \in [n] \\ \text{za } j \in [m] \end{array} \right] \quad \begin{array}{l} \xrightarrow{\text{skladitice dostavi}} \\ \text{eni stranki} \end{array}$$

**4. (problem kombinatorične dražbe)** Dražitelj ponuja predmete iz množice  $A$  in prejme ponudbe  $\{(B_i, c_i)\}_{i=1}^k$ , pri čemer je  $c_i$  cena, ki jo udeleženec dražbe ponudi za predmete v množici  $B_i \subseteq A$ . Katere ponudbe naj dražitelj sprejme, da maksimizira dobiček, če lahko vsak predmet proda največ enkrat? Modeliraj opisani problem z linearnim programom.

množica predmetov ... A

$(B_i, c_i)$  ... ponudbe

$$\lambda = 1, \dots, k$$

Ci ... cena, k i j o u d e l e ž i e n c p o n u d i z a

$$x_i = \begin{cases} 1 & ; \text{če sprejme } (b_i, c_i) \text{ ponudbo} \\ 0 & ; \text{sicer} \end{cases}$$

Ci ... cena, ki jo udeleženeč  
medmene  $\beta_i$   $\beta_i \in A$

$$\max \sum_{i=1}^k c_i x_i$$

katere ponudbe naj dražitelj sprejme, da max dobicek

p.p.  $x_i \in \mathbb{Z}$

$f$  predmet lahko poda samo 1-krat

$$0 \leq x_i \leq 1 \quad \forall i \in [k]$$

$\forall a \in A : \sum x_i \leq 1$   $\longrightarrow$  uvačje predmet prodamo samo 1krat

⑤  $G = (V, E)$  ... neusmerjen graf

$$D \subseteq V$$

D... dominacijska množica

DEF:  $\forall u \in V \setminus D$  velja  $\exists v \in D$ , da je  $uv \in E$

Iščemo čim manjšo D

$$\min \sum_{u \in V} x_u = \text{mod } D$$

p.p.  $x_u \in \mathbb{Z}$

$$0 \leq x_u \leq 1 \quad \text{za } u \in V$$

$$\forall u \in V : x_u + \sum_{v \in E} x_v \geq 1$$

$\downarrow$   
zato ker sta lahko tudi  
oba

če vsoko vozlišče večja ali je ued ali  
pa ima sosedja v Dju

⑥  $G = (V, E)$  ... usmerjen graf

$$u, v \in V$$

Iščemo najkrajšo pot od u do v

$$x_e = \begin{cases} 1 & ; \text{ povezava } e \text{ je na iskani poti;} \\ 0 & ; \text{ sicer} \end{cases}$$

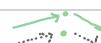
$$\min \sum_{e \in E} x_e$$

p.p.  $x_e \in \mathbb{Z}$

$$0 \leq x_e \leq 1 \quad \forall e \in E$$

$$\forall w \in V \setminus \{u, v\} : \sum_{y \in E} x_{yw} = \sum_{w \in E} x_{wy}$$

istekrat gremo v vozlišče in  
gremo izven vozlišča:



$$\sum_{y \in E} x_{yu} + 1 = \sum_{u \in E} x_{uy}$$

iz u gre ven ena povezava

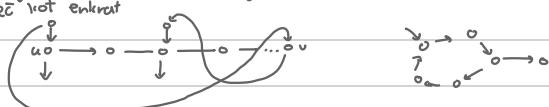
več kot noben

$$\sum_{y \in E} x_{vy} = \sum_{v \in E} x_{vy} + 1$$

iz v gre noben ena povezava

več kot ven

↓  
tukaj smo dopustili tudi da gre čez vozlišče  
več kot enkrat



dopravljamo sprehod, ki ima tudi cikel, vendar  
je ta program še vedno ustrezhen, saj pot itak  
minimiziramo (torej že avtomatsko damo cikle ven)

## 3.VAJE

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

$G = (V, E)$  usmerjen

$u \in V$

$d(u, v) ; \forall v \in V$

$x_v$  ... razdalja od  $u$  do  $v$

$$y_e = \begin{cases} 1 & \text{če je } e \in \text{drzewo} \\ 0 & \text{sicer} \end{cases}$$

$$\min \sum_{v \in V} x_v$$

$$y_e \in \mathbb{Z}$$

$$0 \leq y_e \leq 1 \quad \forall e \in E$$

$$x_v \in \mathbb{Z}$$

$$\text{za } \forall v \in V \setminus \{u\} : \sum_{w \in E} x_{vw} = 1$$

$$\sum_{w \in E} y_{wu} = 1$$

$$x_u = 0$$

$$\forall w \in E : x_v - x_w \geq 1 - n + n \cdot y_{vw}$$

2. Napiši linearni program, ki modelira določanje kromatičnega števila grafa.

$$x_{ui} = \begin{cases} 1 & \text{vozlišče } u \text{ ima barvo } i \\ 0 & \text{sicer} \end{cases}$$

$u \in V$

$$|V| = n \quad i \in \{1, \dots, n\}$$

$t$  ... število barv

$$\min t$$

$$\text{p.p. : } \begin{aligned} 0 \leq x_{ui} \leq 1 \\ x_{ui} \in \mathbb{Z} \end{aligned} \quad \forall u \in V, \forall i \in \{1, \dots, n\}$$

$$\forall u, v \in E, \forall i \in \{1, \dots, n\} : x_{ui} + x_{vi} \leq 1$$

$$\forall u \in V : \sum_{i=1}^n x_{ui} = 1$$

$$\forall u \in V, \forall i \in \{1, \dots, n\} : x_{ui} \cdot i \leq t$$

↓  
obiščivo je omogočeno z  
številko-barvne, torej bo  
najmanjše število ko bomo  
številke vozlišč jemali po  
vrsti, to pa nam itak pove  
 $\min t$

3. (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.

$n$  mest

$c_{ij}$  ... strošek potovanja  $i \rightarrow j$

vsih  $n$  mest in nazaj v začetno

min stroški

$$x_{ij} = \begin{cases} 1 & \text{iz } i \rightarrow j \\ 0 & \text{sicer} \end{cases}$$

$$\min \sum_{i=1}^n \sum_{j=1}^n c_{ij} \cdot x_{ij}$$

$$\text{p.p. : } \begin{aligned} 0 \leq x_{ij} \leq 1 \\ x_{ij} \in \mathbb{Z} \end{aligned} \quad \forall i, j \in \{1, \dots, n\}$$

$$\begin{aligned} \sum_{i=1}^n x_{ij} = 1 \\ \forall i \quad \sum_{j=1}^n x_{ij} = 1 \end{aligned} \quad \begin{array}{l} \text{v vsako vozlišče gremo} \\ \text{enkrat in ven iz vozlišča nikrat} \end{array}$$

$y_i$  ... kolico, ki se povečuje po poti

Če iz mesta  $i$  potujemo v mesto  $j > 1$ , mu to sledi v vrstnem redu:

$$\forall i \in \{1, \dots, n\} \quad \forall j \in \{2, \dots, n\} : y_i + nx_{ij} \leq y_j + n - 1$$

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  $n - 1$ .

$$y_1 = 0$$

$$\begin{aligned} \forall i \in \{1, \dots, n\} \quad \forall j \in \{2, \dots, n\} : y_j - y_i \geq 1 - n + n \cdot x_{ij} \\ \text{dodajmo razlaga} \quad \text{ker nočemo,} \\ \text{da se } y \text{ poveča,} \\ \text{to pride dom spet} \\ \forall i = 1 \end{aligned}$$

5. 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 celoštivilski linearni program, ki modelira zgoraj opisani problem.

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

$n$  ... št. asistentov

$m$  ... predmetov

$a_i, b_i$  ... najmanjše in največje št. ur, ki jih izvaja  $i$ -ti asistent

$N_i \subseteq \{1, 2, \dots, m\}$  ... množica predmetov, ki jih  $i$ -ti asistent ne želi

$c_j$  ... št. skupin za vaje pri  $j$ -tem predmetu

$u_j$  ... št. ur vaj na skupino

asistenta  $p$  in  $q$  se kregata

cilj: minimalno največje št. različnih predmetov za asistenta

$$x_{i,j} = \begin{cases} 1 & ; i\text{-ti asistent uči } j\text{-ti predmet} \\ 0 & ; \text{sicer} \end{cases}$$

$y_{ij} \in \mathbb{Z}$  ... št. skupin, ki jih  $i$ -ti asistent uči pri  $j$ -tem primeru

$t$  ... največje št. različnih predmetov

$$\min t$$

$$\forall i, \forall j : \begin{cases} 0 \leq x_{ij} \leq 1, & x_{ij} \in \mathbb{Z} \\ y_{ij} \in \mathbb{Z} \end{cases}$$

$$\forall i \in \{1, \dots, n\} : a_i \leq \sum_{j=1}^m u_j \cdot y_{ij} \leq b_i$$

$$\forall i, \forall j : y_{ij} \leq c_j \cdot x_{ij}$$

$$\forall i, \forall j \in N_i : x_{ij} = 0$$

$$\forall j : x_{pj} + x_{qj} \leq 1 \rightarrow p \text{ in } q \text{ ne učita istega predmeta}$$

$$\forall j : \sum_i x_{ij} = c_j$$

$$\forall i : \sum_j x_{ij} \leq t \rightarrow \text{omejitveni pogoj}$$

## VAJE 4

1. 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 gnezdu  $i$  preživijo, enaka  $p_{ij}$ , kjer je  $j$  število podtaknjenih jajc v gnezdu  $i$  (preživijo bodisi vsi ali noben mladič v posameznem gnezdu). 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 število preživelih mladičev.

Zapiši problem kot celoštivilski linearni program.

$$\begin{array}{cccccc} \hat{i}=1,2,\dots,12 & i=1,2 & i=3,4,5,6 & i=7,8,9 & i=10,11 & i=12 \\ 16 \text{ jajc}, 12 \text{ gnezd}, 2 \text{ taščici}, 4 \text{ penice}, 3 \text{ travniške}, 2 \text{ beli pastirici}, 1 \text{ sova} \\ \max 3 \text{ jajca} \end{array}$$

$p_{ij}$  - verjetnost, da mladiči v gnezdu  $i$  preživijo, če je  $j$  jajc

Vsaj 1 jajce pri vsaki vrsti

jajca taščice > jajca pastirice \*

če pri 1. taščici  $\geq 2$  jajci  $\Rightarrow$  pri 2. taščici nebo nis (\* \*)

$\max E$  (st. preživelih)

$$x_{ij} = \begin{cases} 1 & ; \text{ v } i\text{-tem gnezdu je } j \text{ jajc} \\ 0 & ; \text{ sicer} \end{cases}$$

$$\max \sum_{i=1}^{12} \sum_{j=1}^3 x_{ij} \cdot p_{ij}$$

$$\text{p.p. : } 0 \leq x_{ij} \leq 1, \quad k_{ij} \in \mathbb{Z}, \quad \forall i,j$$

$$\forall i : \sum_{j=1}^3 x_{ij} \leq 1$$

$$\sum_{j=1}^3 \sum_{i=1}^2 x_{ij} \geq 1$$

$$\sum_{j=1}^3 \sum_{i=3}^6 x_{ij} \geq 1$$

$$\sum_{j=1}^3 \sum_{i=7}^{11} x_{ij} \geq 1$$

$$\sum_{j=1}^3 \sum_{i=12} x_{ij} \geq 1$$

$$\sum_{j=1}^3 x_{12,j} \geq 1$$

$$\sum_{j=1}^3 \sum_{i=1}^2 j \cdot x_{ij} \geq \sum_{j=1}^3 \sum_{i=3}^{11} j \cdot x_{ij} + 1 \quad (*)$$

$$\sum_{j=1}^3 x_{12,j} + \sum_{j=1}^3 x_{11,j} \leq 1 \quad (**)$$

pri padanja  
gned  
vsaki  
vrsti

ali velja eno  
ali velja drugo

2. Vinar Janez je pridelal 2000 litrov rumenega muškata, 10000 litrov laškega rizlinga in 5000 litrov renskega 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 kupecu, in sicer v maksimalni količini (če kupec ne kupi vsega vina iste sorte, ga Janez ohrani zase). Župan pravi, da občina drugega vina kot renskega 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 celoštivilski linearni program.

2000 RM, 10000 LR, 5000 RR

občina: samo RR \*

Luka: RM, če kocka dobi LR (\*\*\*)

Kocka: Če O in L ne kupita, tudi kocka ne kupi

če A kupi  $s_A$  in B kupi  $s_B$ , potem C kupi samo C.

$\max$  zaslužek

dodatek: predpostavimo, da vsak kupi največ 1 sorto

$$x_{ij} = \begin{cases} 1 & ; \text{ kupec } i \text{ kupi vino } j \\ 0 & ; \text{ sicer} \end{cases}$$

$$\max 2000 x_{k,RM} + 10000 x_{k,LR} + 5000 x_{k,RR} +$$

$$+ 11000 x_{l,RM} + 2200 x_{l,LR} + 5500 x_{l,RR} + 5500 x_{l,RR} +$$

$$+ 750 x_{e,RM} + 750 x_{e,LR} + 750 x_{e,RR} +$$

$$+ 1800 x_{o,RM} + 1800 x_{o,LR} + 1800 x_{o,RR}$$

$$\text{p.p. : } 0 \leq x_{ij} \leq 1, \quad x_{ij} \in \mathbb{Z} \text{ za } \forall i,j$$

$$\forall i : \sum_j x_{ij} \leq 1$$

$$\forall j : \sum_i x_{ij} \leq 1$$

$$* \quad \sum_{j \neq RR} x_{e,j} = 0$$

$$(*) \quad x_{k,LR} \geq x_{k,RR}$$

$$\sum_j (x_{o,j} + x_{e,j}) \geq \sum_j x_{k,j}$$

$$x_{A,s_A} + x_{B,s_B} + \sum_{i \neq C} x_{i,s_i} \leq 2$$

3. Distributer ima  $A$  zabojev avokadov in  $B$  zabojev banan, ki jih bo prodal  $n$  trgovcem. Trgovec  $i$  ( $1 \leq i \leq n$ ) plača  $a_i$  evrov za zabolj avokadov in  $b_i$  evrov za zabolj banan, skupaj pa bo porabil največ  $c_i$  evrov. Distributer bo zabolje dostavil s tovornjaki, v katerih je lahko največ  $K$  zaboljev (ne glede na vsebino). Če nekemu trgovcu dostavi  $t$  zaboljev, 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 zabolj avokadov, če bo tudi trgovec  $r$  kupil vsaj en zabolj avokadov. Distributer želi zabolje 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 celoštenski linearni program, ki modelira zgoraj opisani problem.

$A$  zabolj avokadov,  $B$  zabolj banan

$n$  ... št. trgovcem, ki prodajamo

$a_i$  ... koliko plača  $i$ -ti trgovec za zabolj avokadov

$b_i$  ... -11- banan

$c_i$  ... največja poraba  $i$ -tega trgovca

$k$  ... največje št. zaboljev v tovornjaku

$d_i$  ... cena ene vožnje do  $i$ -tega trgovca

\* trgovec  $p$  kupi samo  $A$  ali samo  $B$

\* trgovec  $q$  kupi vsaj en zabolj  $A$ , če bo tudi trgovec \*

$r$  kupil vsaj en zabolj  $A$

\* max dobiček

$$\max \sum_i (a_i \cdot x_{Ai} + b_i \cdot x_{Bi} - d_i \cdot y_i)$$

$$\text{p.p.: } x_{Bi}, x_{Ai}, y_i \in \mathbb{Z} \text{ za } \forall i \\ y_i \geq \frac{x_{Bi} + x_{Ai}}{K} \xrightarrow[\text{zakroženo}]{\text{lepši zapis}} k \cdot y_i \geq (x_{Bi} + x_{Ai})$$

$$x_{Bi}, x_{Ai}, y_i \geq 0$$

$$c_i \geq x_{Ai} \cdot a_i + x_{Bi} \cdot b_i \text{ za } \forall i$$

$$z_{Ap} + z_{Bp} \leq 1$$

$$\text{tri, } \forall i \in \{A, B\}: x_{Si} \leq \frac{c_i}{d_i} z_{Si} *$$

$x_{Bi}$  ... št. zaboljev banan kupi  $j$ -ti trgovec

$x_{Ai}$  ... -11- Avokadu -11-

$y_i$  ... število voženj, do  $i$ -tega trgovca

$$z_{Ai} = \begin{cases} 1 & ; \text{ če } i\text{-ti trgovec kupi avocade} \\ 0 & ; \text{sicer} \end{cases}$$

## VAJE 5

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

$X$  - dobitek

$$E(X) = \frac{1}{2} \cdot 250.000 - \frac{1}{2} \cdot 100.000 = 125.000 - 50.000 = 75.000 > 0$$

2. 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 parameterja  $\alpha$ , pri katerih je po Hurwiczovem kriteriju možnih več enakovrednih odločitev.

		NE. SPREMENI	ZUŠA	ZNIŽA	PREKINE
VEZAVA ZA 5 LET	vezava za 5 let	250€	250€	250€	/
	3 leta in polem 2x po eno leto	170€	194€	154€	/
	3 leta	90€	90€	90€	90€
VEZAVA PO ENO LETU	5x po eno leto	200 €	224	184 €	/
	3x po eno leto	120 €	120 €	120 €	120 €

5 LETNA VEZAVA :	250	250	250	90	odločitvena tabela
LETNA VEZAVA :	200	224	184	120	

Optimist: 5 letna vezava

Pesimist: letna vezava [max dobitek v najslabšem primeru: 120 € > 90 €]

Laplace: [vsi primeri so enako možni]  $\rightarrow$  5 letna:  $(3 \times 250 + 90) : 4 = 210 €$   $\Rightarrow$  odloči se za 5 letno  
 $\rightarrow$  letna:  $(200 + 224 + 184 + 120) : 4 = 182 €$

Savage: [razlika med najboljši izid in najslabši izid, za možno odločitev]  $\rightarrow$  5 letna:  $30 € (120 - 90)$   $\Rightarrow$  želimo min razliko  
 $\rightarrow$  letna:  $\max(150 - 200, 250 - 224, 250 - 184, 120 - 90) = 66 €$   $\Rightarrow$  izbere 5 letno

Hurwitz: 5 letna:  $d \cdot 250 + (1-d)90 = 90 - 160d$

$$\left\{ \begin{array}{l} \text{letna: } d \cdot 224 + (1-d)120 = 120 - 104d \\ \text{presečišče: } d = \frac{30}{56} = \frac{15}{28} \end{array} \right. \begin{array}{l} \text{če je } d \text{ manjši vzameš 5 letno,} \\ \text{če je } d \text{ večji vzameš 1 letno} \end{array}$$

$d: \max(t_{\text{izid}}) \min(t_{\text{izid}})$

3. 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?

X ... dobiček (slučajna spremenljivka)

k ... število kupljenih žemelj

$$E(X|L=50) = \frac{-0.2 \cdot 50 + 0.4 \cdot 50}{kupimo so} + 0.1(10 \cdot 0.15 + 20 \cdot 0.3 + 30 \cdot 0.2 + 40 \cdot 0.15 + 50 \cdot 0.1) = 12.45$$

zadužek, ki morajo kupiti  
kupimo so  
prodano  
količina dodatnih verjetnosti

$$E(X|L=60) = -0.2 \cdot 60 + 0.4 \cdot 50 + 0.1 \cdot 0.15 \cdot 10 + 0.9 \cdot 0.4 \cdot 10 + 0.1(10 \cdot 0.3 + 20 \cdot 0.2 + 30 \cdot 0.15 + 40 \cdot 0.1) = 13.3$$

$$E(X|L=70) = -0.2 \cdot 70 + 0.4 \cdot 50 + 0.1 \cdot 0.15 + 0.4 \cdot 60 \cdot 0.15 + 0.75 \cdot 0.4 \cdot 70 + 0.1(10 \cdot 0.2 + 20 \cdot 0.15 + 30 \cdot 0.1) + 0.1 \cdot 20 \cdot 0.15 + 0.15 \cdot 10 \cdot 0.15 = 13.925$$

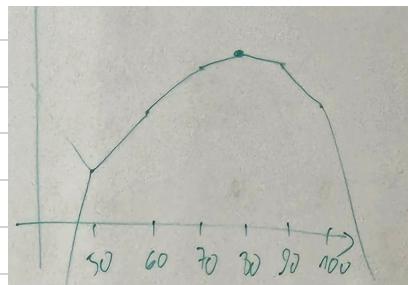
zadužek v primeru da  
vimejo v pekarni žemlje

$$E(X|L=80) = -0.2 \cdot 80 + 0.4 \cdot 50 \cdot 0.1 + 0.4 \cdot 60 \cdot 0.15 + 0.4 \cdot 70 \cdot 0.3 + 0.4 \cdot 80 \cdot 0.25 + 0.15 \cdot (30 \cdot 0.1 + 20 \cdot 0.15 + 10 \cdot 0.3) + 0.1(10 \cdot 0.15 + 20 \cdot 0.1) = 14.1$$

$$E(X|L=90) = -0.2 \cdot 90 + 0.4 \cdot (50 \cdot 0.1 + 60 \cdot 0.15 + 70 \cdot 0.3 + 80 \cdot 0.2 + 90 \cdot 0.15 + 100 \cdot 0.1) + 0.15(40 \cdot 0.1 + 30 \cdot 0.15 + 20 \cdot 0.3 + 10 \cdot 0.2) + 0.1 \cdot 10 \cdot 0.1 = 13.975$$

$$E(X|L=100) = -0.2 \cdot 100 + 0.4 \cdot (50 \cdot 0.1 + 60 \cdot 0.15 + 70 \cdot 0.3 + 80 \cdot 0.2 + 90 \cdot 0.15 + 100 \cdot 0.1) + 0.15(50 \cdot 0.1 + 40 \cdot 0.15 + 30 \cdot 0.3 + 20 \cdot 0.2 + 10 \cdot 0.1) = 13.63$$

$$E(X|L=2) = -0.2 \cdot 2 + \sum_{i \in 2} 0.4 \cdot i \cdot p_i + \sum_{i \in 2} 0.4 \cdot 2 \left(1 - \sum_{i \in 2} p_i\right) + \sum_{i \in 2} 0.15(2-i) \cdot p_i + \sum_{i \in 2} 0.1 \cdot (i=2) p_i$$



4. 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). Znane 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?

X... dobiček

k... prodane karte

$$E(X|L=100) = 100 \cdot 10 = 1000$$

$$E(X|L=101) = 101 \cdot 10 - 30 \cdot 0.2 = 1004$$

splošca se prodasti 101 karto

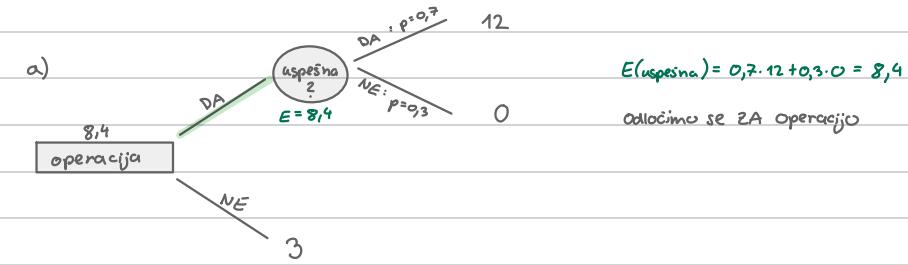
$$E(X|L=102) = 102 \cdot 10 - \underbrace{60 \cdot 0.2}_{2 \text{ ostane} \atop \text{zunaj}} - \underbrace{30 \cdot 0.3}_{1 \text{ ostane} \atop \text{zunaj}} = 999$$

$$E(X|L=103) = 103 \cdot 10 - 90 \cdot 0.2 - 60 \cdot 0.3 - 30 \cdot 0.4 = 982$$

5. Pacient ima na voljo operacijo. Brez operacije bo živel natanko 3 mesece. Z uspešno opravljenou 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 življenske dobe.

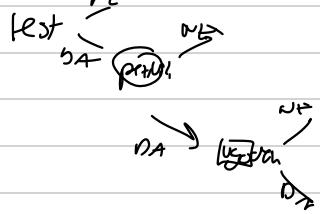
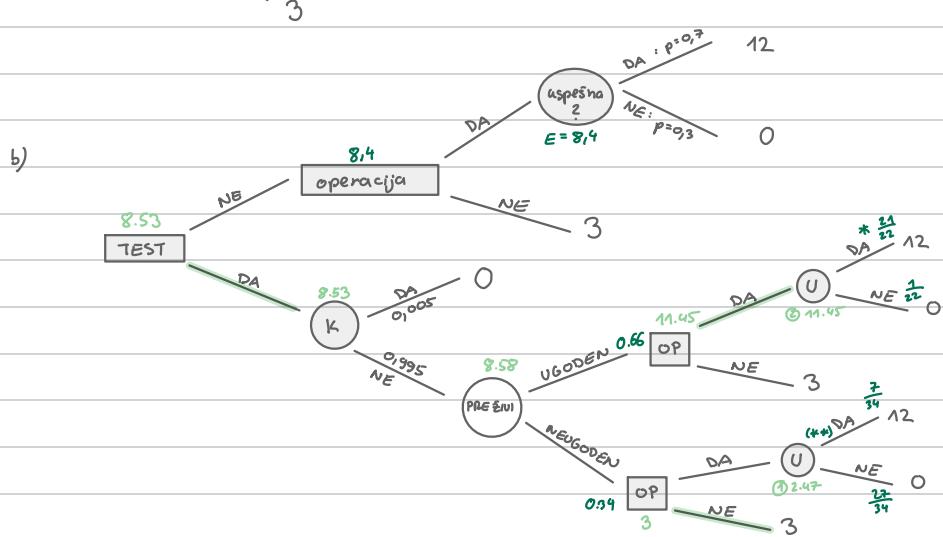
- Ali naj pacient sprejme operacijo?
- Pacient lahko opravi predhodni test, ki z zanesljivostjo 0.9 napove uspešnost operacije, vendar z verjetnostjo 0.005 pacient zaradi komplikacij med testom umre. Ali naj pacient opravi test?

Nariši odločitveno drevo in odločitve sprejmi na podlagi izračunanih verjetnosti!



rest zanesljivost  
uspešno operacijo 0,9

0,005% baret utr



34.3

$$P(\text{test = ugoden} \mid \text{operacija uspešna}) = 0.9$$

$$P(\text{test = ugoden} \mid \text{operacija neuspešna}) = 0.1$$

$$P(\text{ugoden}) = 0.9 \cdot 0.7 + 0.1 \cdot 0.3 = 0.66$$

↓  
Plagorun  
↓  
P(neuspešna)

POPOLNA VERJETNOST

$$(*) P(\text{operacija uspešna} \mid \text{test = ugoden}) = \frac{P(\text{operacija uspešna} \text{ in test = ugoden})}{P(\text{test = ugoden})} = \frac{0.9 \cdot 0.7}{0.66} = \frac{0.63}{0.66} = \frac{63}{66} = \frac{21}{22}$$

Bayesova formula

$$(**) P(\text{operacija uspešna} \mid \text{test = neugoden}) = \frac{0.1 \cdot 0.7}{0.34} = \frac{0.07}{0.34} = \frac{7}{34}$$

$$\textcircled{1} 12 \cdot \frac{7}{34} + 0 \cdot \frac{21}{34} = 2.47$$

$$\textcircled{2} \frac{21}{22} \cdot 12 + \frac{7}{22} \cdot 0 = 11.45$$

$$\textcircled{3} 0.66 \cdot 11.45 + 0.34 \cdot 3 = 8.58$$

2. 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.

- (a) Z odločitvenim drevesom opiši njegove možnosti in ugotovi, kako naj se odloči ter kakšen dobiček naj pričakuje.
- (b) 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  $P(\text{Seadovo mnenje} | \text{kakovost parcele})$ :

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

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

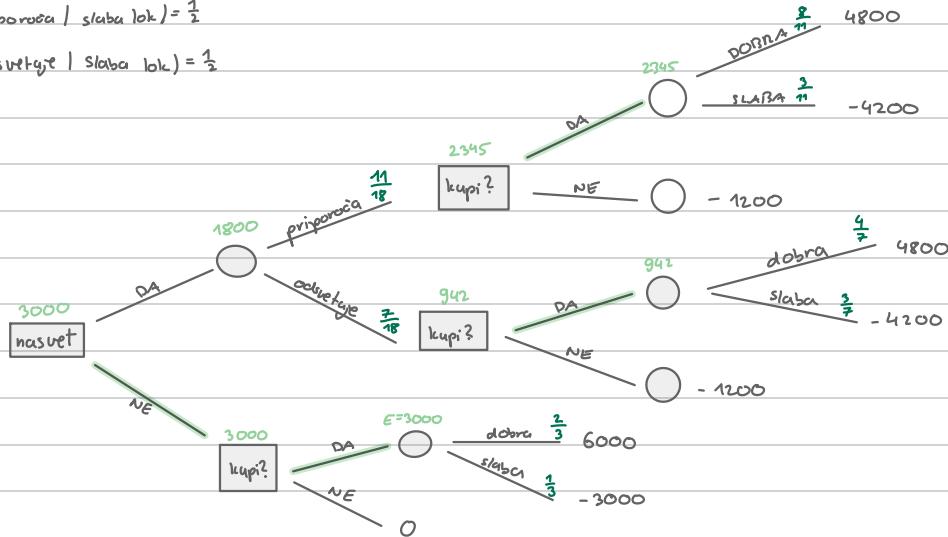
$$P(\text{dobra lok.}) = \frac{2}{3}$$

$$P(\text{priporoča} | \text{dobra lok.}) = \frac{2}{3}$$

$$P(\text{odsvetuje} | \text{dobra lok.}) = \frac{1}{3}$$

$$P(\text{priporoča} | \text{slaba lok.}) = \frac{1}{2}$$

$$P(\text{odsvetuje} | \text{slaba lok.}) = \frac{1}{2}$$



$$P(\text{priporoča}) = \frac{2}{3} \cdot \frac{2}{3} + \frac{1}{2} \cdot \frac{1}{3} = \frac{4}{9} + \frac{1}{6} = \frac{8+3}{18} = \frac{11}{18}$$

$$P(\text{dobra lok.} | \text{priporoča}) = \frac{\frac{2}{3} \cdot \frac{2}{3}}{\frac{11}{18}} = \frac{8}{11}$$

$$P(\text{dobra lok.} | \text{odsvetuje}) = \frac{\frac{2}{3} \cdot \frac{1}{3}}{\frac{7}{18}} = \frac{36}{63} = \frac{12}{21} = \frac{4}{7}$$

## DINAMIČNO PROGRAMIRANJE

4. 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 omejitve, da mora biti razdalja med oglasi vsaj  $d$  kilometrov. Oglase želimo postaviti tako, da bodo čim bolj profitabilni.

$v_i$  - naraščajoče zaporedje

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

$x_i$	1	2	8	10	12	14	17	20
$v_i$	8	8	12	10	7	5	6	10
$p_i$	8	8	20	20	20	25	26	35
$\downarrow$ max zaslužek do $x_1$ do $x_i$								$(1, 8, 12, 20)$

b)  $p_i = \max \{ p_{i-1}, v_i + p_j \mid x_i - x_j \geq d \}$  ;  $2 \leq i \leq n$ ,  $p_1 = v_1$  ;  $j$  smo označevali tisti  $x_j$ , ki je od  $x_i$  oddaljen vsaj  $d$  kilometrov  
računamo naraščajoče po indeksu  $i$

opt. vrednost :  $p^* = p_n = \max_i p_i$   
 $\text{saj je tudi } p_i \text{ nepadajoče zaporedje}$

\* MALO PREDOBLIKOVANI ZAPIS  $\Rightarrow$  dalo pogoj:  $0 \leq i \leq n$  boj pravilen

$$p_i = \max \{ p_{i-1}, v_i + p_j \mid x_i - x_j \geq d \} ; \begin{cases} 1 \leq i \leq n, & p_0 = v_0, \\ 0 \leq j \leq i-1 & x_0 = -\infty \end{cases}$$

\* časovna zahtevnost:  $O(n^2)$

c) če bi želeli "zmanjšati" časovno zahtevnost lahko uredemo novo spremenljivko, ki nam bo "shranjevala" vrednosti:

$$j_0 = 0$$

$$j_i = \max \{ j \mid j_{i-1} \leq j \leq i-1 ; x_i - x_j \geq d \}$$

$$p_i = \max \{ p_{j_{i-1}}, v_i + p_j \} \quad (1 \leq i \leq n)$$

Naraščajoče po  $i$  računamo  $j_i$  in  $p_i$

$\Rightarrow$  časovna zahtevnost  $O(n)$

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

$$a) I \subseteq \{1, \dots, n\}$$

$$\max \sum_{i \in I} v_i$$

$$\text{p.p. } \sum_{i \in I} t_i \leq M$$

pi... največja vrednost pri nosilnosti  $j \leq M$

$I_j$  ... mn. elementov, ki doseže najv. vrednost pri  $j$

$$(p_j, I_j) = \max \{(p_{j-i}, I_{j-i}) \mid i=1, \dots, n, i \notin I_{j-i}, t_i \leq j\} \quad p_0 = 0$$

$$I_0 = \emptyset$$

nimam pojma iz kje  
je on to formulo  
potegnu

$$p^* = p_M$$

$$I^* = I_M$$

$$\sigma(Mn)$$

## VAJE 7

1. 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) Reši problem za matriko

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

- (b) Napiši rekurzivne enačbe za opisani problem. → Rešili smo samo b)

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

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

pr; ... minimalna vsota poti z začetkov v  $(1,1)$  in koncem v  $(i,j)$

$$P_{1n} = Q_{n1}$$

$$p_{1j} = p_{1,j-1} + a_{ij} \quad 2 \leq j \leq n$$

$$p_{i+1} = p_{i-n,1} + d_{i,1} \quad 2 \leq i \leq m$$

Vrednosti  $p_{ij}$  računamo teksikografsko po (i,j)

$$p^k = p_{mn}$$

časovna zahtevnost:  $O(mn)$

2. Dan je niz  $S = a_1a_2\dots a_n$ , kjer so  $a_i$  ( $1 \leq i \leq n$ ) elementi neke končne abecede. Nizu  $a_ja_{j+1}\dots a_k$ , kjer je  $1 \leq j \leq k \leq n$ , pravimo *strnjen podniz* niza  $S$ . S pomočjo dinamičnega programiranja napiši algoritem, ki določi najdaljši palindromski strnjeni podniz v  $S$ .

niz, ki se isto bere naprej  
in tudi nazaj [npr. ABA]

OP: prazni nizi so palindromi in tudi z samo enim elementom

$$p_{ij} = \begin{cases} 1 & ; \text{ niz od } [a_{i+1}, \dots, a_j] \text{ je palindromski} \\ 0 & ; \text{ sicer} \end{cases}$$

$p_{ii} = 1 \rightarrow$  to je proces podniz [in je polindromski] za  $i \in [0, n-1]$

$p_{i,i} = 1 \longrightarrow$  to go next  $\neq$  current symbol  $\forall i \in [1, n]$

→ "prazni prostori med črkami so polindromi"

] robin  
pogejí

Zapišimo bolj natančen pogoj

$$p_{ij} = \begin{cases} 1 & ; p_{im,jm}=1 \wedge a_{im}=a_j \\ 0 & ; \text{otherwise} \end{cases} \quad \text{for } 0 \leq i < j \leq n, \quad j-i \geq 2$$

urednosti): p<sub>ij</sub> računamo teksikografsko po ( $j-i, j$ )

10

↳ To hi tako nujn pogoj, samo pove nam, da bomo gledali od leve proti deshi gl

$$p^* = \max \{j-i \mid 0 \leq i \leq j \leq n, p_j = 1\}$$

časovna zahtevnost:  $O(n^2)$

$$\hookrightarrow \text{če } p_{ij} = 0 \Rightarrow p_{i+1,j-1} = 0$$

$\nearrow 0$

3. 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.

$p_i \dots \max$  količina denarja, ko rupa do  $i$ -te hiše

$$p_i = \max \{p_{i-1}, p_{i-2} + c_i\} \quad i \in [2, n]$$

$$p_0 = 0, \quad p_1 = c_1$$

Računamo naravno po  $i$

$$p^* = p_n$$

časovna zahtevnost:  $O(n)$

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

- (a) Reši problem za matriko

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

- (b) Napiši rekurzivne enačbe za opisani problem.

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

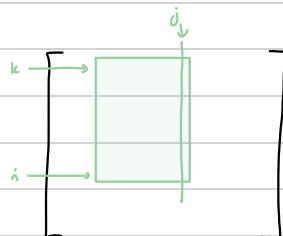
b) Če to "reduciramo" na enodimenzionalno bi gledali  $[a_1, \dots, a_m]$

$P_{k,i,j} \dots$  največja vsota od vrstice  $k$  do vrstice  $i$  v zadnjem stolpcu v  $j$

$k = \text{prva vrstica}$

$i = \text{zadnja vrstica}$

$j = \text{stolpec}$



$\Delta_{k,i,j} \dots$  vsota od vrstice  $k$  do  $i$  v  $j$ -tem stolpcu

$$P_{k,i,j} = \max \{P_{k,i,j-1}, 0\} + a_{i,j} - \Delta_{k-1,j}$$

$1 \leq k \leq i \leq m$   
 $1 \leq j \leq n$

$$\Delta_{0,i,j} = 0$$

$$\Delta_{i,j} = \Delta_{i-1,j} + a_{i,j} \quad i \in \{1, \dots, m\}$$

$\rightarrow$  čas zaht. konstantni  
skupaj  $O(mn)$

$$\Delta_{i,i,j}$$

Računamo leksikografsko po  $(i,j)$

$$P_{k,i,0} = 0$$

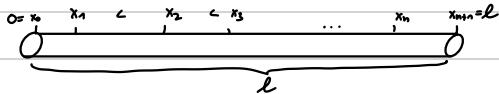
p-je računamo leksikografsko po  $(k, i, j)$   
 $\hookrightarrow$  leksikografski red po stolpcih

$$p^* = \max_{k,i,j} P_{k,i,j}$$

časovna zahtevnost  $O(mn)$

5. Imamo hlod dolžine  $\ell$ , ki bi ga radi razčagali 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) Reši problem pri podatkih  $\ell = 10$  in  $(x_i)_{i=1}^4 = (3, 5, 7, 8)$ .
- (b) S pomočjo dinamičnega programiranja reši problem v splošnem. Oceni tudi njegovo časovno zahtevnost.



Prij... min cena rezanja hloda od  $x_i$  do  $x_j$

$$p_{ij} = x_j - x_i + \min \{ p_{ik} + p_{kj} \mid i+1 \leq k \leq j-1 \}$$

$$0 \leq i < j \leq n+1$$

$$j-i \geq 2$$

$p_{i,i+1} = 0 \quad 0 \leq i \leq n$

$\downarrow$   
da je med  $i$  in  $j$ ?  
še vsaj en rez?

Računamo tekrilografsko po  $(j-i, i)$

$$p^* = p_{0,n+1}$$

Časovna zahtevnost:  $O(n^3)$

$\downarrow$   
linearno za vsakega  $i$   
vse skupaj pa je  
možnosti  $n^2$

1. Lastnik verige  $n$  trgovin z živilo 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 algoritmom, ki reši opisani problem v splošnem.

$p_{ij}$  ... pričakovani dobiček v trgovini  $j$ , če dostavimo  $i$  zabojev

$k_{ij}$  ... max zaslužek če u prvih  $j$  trgovinah dostavimo  $i$  zabojev

$$k_{ij} = \max \{ k_{l,j-1} + p_{i-l,j} ; l=0,1,\dots,i \} \quad \text{za } j=2,\dots,n \text{ in } i=0,\dots,m$$

$$(k_{0j}=0)$$

$$k_{i,n} = p_{i,n}$$

reševanje naloge

0	0	0	0	$k_{1,2} = \max \{ k_{0,1} + p_{1,2}, k_{1,1} + p_{0,2} \}$
1	5	6 <sub>l=0</sub>	6 <sub>1</sub>	$= \max \{ 6, 5 \} = 6$
2	9	11 <sub>0,1</sub>	11 <sub>2</sub>	
3	14	16 <sub>1,2</sub>	16 <sub>3</sub>	
4	17	20 <sub>1,2,2</sub>	20 <sub>4,3,2</sub>	
5	21	25 <sub>3</sub>	25 <sub>5,3</sub>	

gledati lege imenit  
največjo komb  
v prvih dveh  
stolpcih

računamo teksikografsko po  $(j, i)$

$$\text{opt vrednost : } k^* = k_{m,n}$$

lahko bi spustili računanje zadnjega stolpca, razen v zadnji vrstici

potem bi računali po  $(0 \leq j \leq m, 0 \leq i \leq m)$  in  $k^* = k_{m,n}$

$$\text{casovna zahtevnost : } O(m^2n)$$

2. 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 prepričili beg strank h konkurenči.

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  $mf_2f_3$ .

- (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

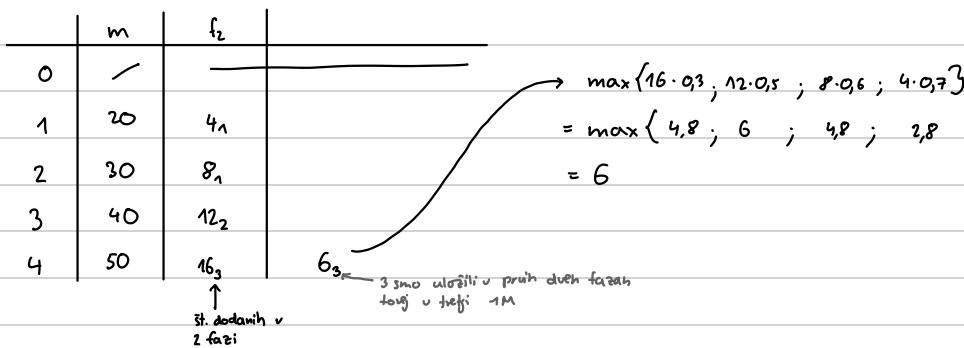
$$m = x_1(10 - x_1), \quad f_2 = 0.4 + 0.1x_2, \quad \text{in} \quad f_3 = 0.6 + 0.07x_3.$$

Kako naj sedaj razdelimo sredstva?

$p_{ij} = \max \text{ deleži, če do } j\text{-te faze porabimo } i\text{-milijonov}$

$$p_{ij} = \max \{ p_{i,j-1} \cdot f_j(i-l) \ ; \ l=0, \dots, i \}$$

$$p_{i,1} = m(i)$$



- b)  $x_i$  ~ količina denarja za  $i$ -to fazo

$$f_1(x_1) = m(x_1) = x_1(10 - x_1)$$

$$f_2(x_2) = 0.4 + 0.1x_2$$

$$f_3(x_3) = 0.6 + 0.07x_3$$

$p_{ij}(i) \dots$  od  $j$ -te faze porabimo  $i$  milijonov (max)

$$p_{ij}(x) = \max \{ p_{i,j-1}(l) \cdot f_j(x-l) \ ; \ 0 \leq l \leq x, l \in \mathbb{Z} \}$$

reševanje naloge:

$$p_3(x) = 0,6 + 0,07x$$

$$p_2(x) = \max \{ p_3(x), p_2(x-l) \} = \max \{ (0,6 + 0,07l)(0,4 + 0,1(x-l)) \mid 0 \leq l \leq x \}$$

↓

$$\text{odvojajmo, da dobimo max: } g_2(l) = 0,24 + 0,06(x-l) + 0,028l + 0,007(x-l)l$$

$$= -0,007l^2 - 0,032l + 0,007xl + 0,24 + 0,006x$$

$$g'_2(l) = -0,014l - 0,032 + 0,007x = 0$$

$$l = \frac{-0,032 + 0,007x}{0,014} = \frac{-32 + 7x}{14} = -\frac{16}{7} + \frac{1}{2}x$$

pogumno te  
velja  $0 \leq l \leq 1$

$$0 \leq -\frac{16}{7} + \frac{1}{2}x$$

$$\frac{16}{7} \leq \frac{1}{2}x \iff 4, \dots \leq x$$

ker gledamo samo za  $x \geq \max 4$ ,  
bomo dobili, da je max pri  
 $l=0$

$$\Rightarrow \max \text{ pri } l=0$$

$$p_2(x) = 0,24 + 0,06x$$

$$p_1(x) = \max \{ (0,24 + 0,06l)(10-l)(10-(x-l)) \mid 0 \leq l \leq x \}$$

$$p_1(4) = \max \{ (0,24 + 0,06l)(4-l)(6+l) \mid 0 \leq l \leq 4 \}$$

soj demo,  
da je  $l=0$ ,  
in moramo  
poravnati vse  
v M

$$\text{odvod: } g_1(l) = (0,24 + 0,06l)(24 - 2l - l^2) = -0,06l^3 - 0,24l^2 - 0,12l^2 - 0,48l + 1,44l + 5,76 =$$

$$= -0,06l^3 - 0,36l^2 + 0,96l + 5,76$$

$$g'_1(l) = -0,18l^2 - 0,72l + 0,96 = 0$$

$$l_{1,2} = \begin{array}{l} \text{D.N.} \\ \text{dokončaj} \end{array}$$

3. 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. Ocenि tudi njegovo časovno zahtevnost v odvisnosti od  $n$  in  $k$ .

$n \dots$  nadstropij

$k \dots$  lončkov

strategija: minimizirati največje št. metov, ki jih bomo potrebovali

$x_{ij} \dots$  najmanjše možno, največje št. metov pri i lončkih in j nadstropjih

$$x_{ij} = \min \left\{ \max \left\{ \underbrace{x_{i,j-l}}_{\text{se ne razbije}}, \underbrace{x_{i+1,j-1}}_{\text{se razbije}} \right\} \mid 1 \leq l \leq j \right\} + 1$$

in  
 pogledati:  
 moramo  
 j-e  
 nadstropij

$$x_{i,0} = 0$$

$$x_{0,j} = \infty \quad j > 0$$

↓  
nimamo lončkov

računali bomo teksilkografsko  $(i,j)$

$$\text{opt. rešitev: } x_{ij}^* = x_{kn}$$

časovna zahtevnost:  $O(k \cdot n^2)$

gremo po vseh  $(i,j)$ , torej  
 imamo km procesov, za vsakega  
 tega glede pa je po  $l$ -jih,  
 ki kažejo od  $i \rightarrow j$  karj od  
 $i \rightarrow n$

# ALGORITMI NA GRAFIH

$G = (V, E)$ ,  $|V| = n$ ,  $|V| + |E| = m$

Graf lahko prikažemo z  $\rightarrow$  matriko sosednosti:  $O(n^2)$

$\rightarrow$  seznam sosednosti / slovar sosednosti  $O(m)$

$$n \leq m \leq n^2$$

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

- `__init__(G)`: ustvarjanje praznega grafa
- `dodajVozlisce(G, u)`: dodajanje novega vozlišča
- `dodajPovezavo(G, u, v)`: dodajanje nove povezave
- `brisiPovezavo(G, u, v)`: brisanje povezave
- `brisiVozlisce(G, u)`: brisanje vozlišča
- `sosedi(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. Ocenji tudi prostorsko zahtevnost celotne strukture.

• `def __init__(G):`  $O(1)$

$G.A = \{ \}$   
 $\downarrow$   
 slovar

• `def dodajVozlisce(G, u):`  $O(n)$

$A[u] = \{v : 0 \text{ for } v \text{ in } G.A\}$

`for v in G.A`

$G.A[v][u] = 0$

• `def dodajPovezavo(G, u, v):`  $O(1)$

$G.A[u][v] = 1$

$G.A[v][u] = 1$   $\times \rightarrow$  ce je usmerjen graf

• `def brišiPovezavo(G, u, v):`  $O(1)$

$G.A[u][v] = 0$

$G.A[v][u] = 0$   $\times \rightarrow$  ce je usmerjen graf

• `def brišiVozlisce(G, u):`  $O(n)$

`del G.A[u]`

`for v in G.A:`

`del G.A[v][u]`

$\downarrow$  izhodni

• `def Sosedji(G, u):`  $O(n)$

`return [v for v in G.A if G.A[u][v] == 1]`

[če bi imeli usmerjen graf in bi gledali se utrdne sosedje bi samo spremenili  $G.A[v][u] = 1$

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

`def __init__(G):`

$O(1)$

—če je graf usmerjen

$G.E = \{\}$   
 $G.F = \{\}$

`def dodaj_vozilisce(G, u):`

$O(1)$

$G.E[u] = set()$   
 $G.F[u] = set()$

`def dodaj_povezavo(G, u, v):`

$O(1)$

$G.E[u].add(v)$   
 $G.E[v].add(u)$

`def briši_povezavo(G, u, v):`

$O(1)$

$G.E[u].remove(v)$   
 $G.E[v].remove(u)$

`def briši_vozilisce(G, u):`

$O(d_G(u))$

`for v in G.E[u]`

$G.E[v].remove(u)$

`del G.E[u]`

$\left[ \begin{array}{l} \text{isto za} \\ G.F \Leftrightarrow G.E \end{array} \right]$

`def sosedji(G, u):`

$O(d_G(u))$

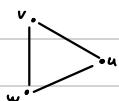
`return [v for v in G.E[u]]`

Primerjava časovne zahtevnosti za ① in ② nalogo

	mat. spos.	sez. spos.
<code>--init--</code>	$O(1)$	$O(1)$
<code>dodaj_vozilisce</code>	$O(n)$	$O(1)$
<code>dodaj_povezavo</code>	$O(1)$	$O(1)$
<code>briši_povezavo</code>	$O(1)$	$O(d_G(u)) \leq O(n)$
<code>briši_vozilisce</code>	$O(n)$	$O(d_G(u)) \leq O(n)$
<code>sosedji</code>	$O(n)$	

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

ščemo, če ima cikel dolžine 3



uporabimo sezname sosednosti

`def trikotnik(G):`

`for v in G.E:`

`for u in G.E[v]:`

`for w in G.E[u]:`

`if v in G.E[w]:`

`return True`

`return False`

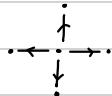
deluje tako za usmirene  
in tudi neusmirene grafe

$$\text{časovna zahtevnost : } \sum_{v \in G} \sum_{u \in E[v]} d_G(u) \leq \sum_{v \in G} \sum_{u \in E[v]} \Delta_G = \Delta_G \sum_{v \in G} d_G(v) = O(\Delta_G n)$$

$\boxed{\Delta_G - \text{max stopnja vozlišča v grafu } G}$

5. 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.

OP: DIGRAF = USMERJEN GRAF



`def zvezda(G)`

`z = None` → štel nam bo vozlišča z povezavami iz njega

`n = len(G.E)`

`for v in G.E`

`if len(G.E[v]) == 0`

`continue`

`elif len(G.E[v]) == n-1 and z is None:`

`z = v`

`else`

`return None`

`return z`

časovna zahtevnost :  $O(n)$

## ALGORITEM ZA ISKANJE V ŠIRINO

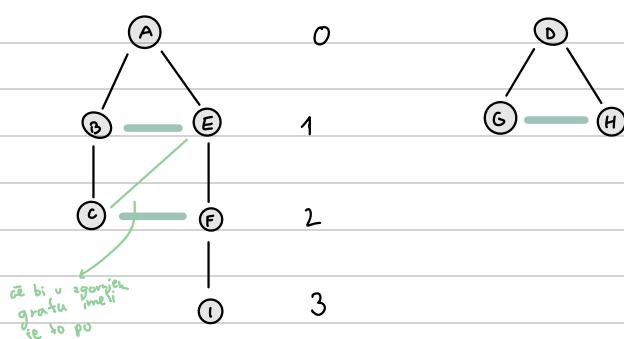
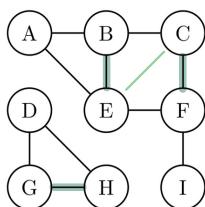
```

def BFS (G, koren = None, visit = None):
    if koren is None:
        koren = G.vozilica()
    if visit is None:
        visit = lambda u, w = None:
            predniki = {}
            globina = []
            for u in koren:
                if u in globina:
                    continue
                prednik[u] = None
                globina[u] = 0
                nivo = [u]
            while len(nivo) > 0:
                naslednji = []
                for v in nivo:
                    visit(v, prednik[v])
                    for w in G.sosed(v):
                        if w in globina:
                            continue
                        prednik[w] = v
                        globina[w] = globina[v] + 1
                        naslednji.append(w)
                nivo = naslednji
            return (prednik, globina)
    
```

časovna zahtevost:  $O(m) + O(n)$  klicev visit

IDEJA: Najprej obiščeš vsega sosedja vozilica trenutnega vozilica, preden se premakneš globlje v graf

- Na sledečem grafu 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.



```

bfs_info = {
    'a': (None, 0),
    'b': ('a', 1),
    'e': ('a', 1),
    'c': ('b', 2),
    'f': ('e', 2),
    'i': ('f', 3)
}

```

```

bfs_info_d = {
    'd': (None, 0),
    'g': ('d', 1),
    'h': ('d', 1)
}

```

## 1. Kaj je BFS?\*\*

Iskanje v širino (angl. \*Breadth-First Search\*) je algoritem za preiskovanje ali iskanje v grafih. Začne v izbranem začetnem vozlišču in najprej obišče vsa sosednja vozlišča, nato sosednje od teh sosedov itd. BFS uporablja \*\*vrsto (queue)\*\* za hranjenje vozlišč, ki jih je treba obiskati.

Ključni pojmi:

- \* `koren`: začetno vozlišče grafa.
- \* `prednik`: slovar, ki hrani predhodnike vsakega vozlišča (za rekonstrukcijo poti).
- \* `globina`: slovar, ki beleži oddaljenost vsakega vozlišča od začetka.
- \* `nivo`: seznam trenutnih vozlišč, ki jih obdelujemo.
- \* `naslednji`: nova vozlišča, ki jih bomo obiskali v naslednji iteraciji.

Zanka BFS:

```
while len(nivo) > 0:
```

```
    naslednji = []
```

```
    for u in nivo:
```

```
    ...
```

Vsako iteracijo obiščeš vsa vozlišča trenutnega nivoja, dodaš sosednja vozlišča v naslednji, če še niso bila obiskana

Glavna operacija v zanki:

```
for w in G.sosed(u):
```

```
    if w in globina:
```

```
        continue
```

```
    prednik[w] = u
```

```
    globina[w] = globina[u] + 1
```

```
    naslednji.append(w)
```

Za vsakega sosedu w:

Če še ni bil obiskan, shraniš u kot predhodnika, nastaviš globino w kot globina[u] + 1,

Dodaš w v seznam za naslednji nivo

## 4. Potek delovanja:\*\*

1. Algoritem začne pri `koren`.
2. V vsaki iteraciji obišče vsa vozlišča trenutnega nivoja (nivo).
3. Za vsakega soseda `w` preveri, če je že bil obiskan (če je v `globina`).
4. Če ni bil, ga doda v `naslednji`, nastavi prednika in globino.
5. Nadaljuje, dokler ni obiskanih vseh dosegljivih vozlišč.

Funkcija vrne:

\* `prednik`: slovar s predhodniki vsakega vozlišča,

\* `globina`: slovar z globinami (oddaljenostmi od začetka).

<i>s</i>	<i>u</i>	<i>v</i>	<i>Q</i>	množica označenih vozlišč
<i>a</i>			[ <i>a</i> ]	{ <i>a</i> }
<i>a</i>	<i>a</i>	<i>b</i>	[ <i>b</i> ]	{ <i>a</i> , <i>b</i> }
<i>a</i>	<i>a</i>	<i>e</i>	[ <i>b</i> , <i>e</i> ]	{ <i>a</i> , <i>b</i> , <i>e</i> }
<i>a</i>	<i>b</i>	<i>c</i>	[ <i>e</i> , <i>c</i> ]	{ <i>a</i> , <i>b</i> , <i>c</i> , <i>e</i> }
<i>a</i>	<i>b</i>	<i>e</i>		{ <i>a</i> , <i>b</i> , <i>c</i> , <i>e</i> }
<i>a</i>	<i>e</i>	<i>a</i>	[ <i>c</i> ]	{ <i>a</i> , <i>b</i> , <i>c</i> , <i>e</i> }
<i>a</i>	<i>e</i>	<i>b</i>		{ <i>a</i> , <i>b</i> , <i>c</i> , <i>e</i> }
<i>a</i>	<i>e</i>	<i>f</i>	[ <i>c</i> , <i>f</i> ]	{ <i>a</i> , <i>b</i> , <i>c</i> , <i>e</i> , <i>f</i> }
<i>a</i>	<i>c</i>	<i>b</i>	[ <i>f</i> ]	{ <i>a</i> , <i>b</i> , <i>c</i> , <i>e</i> , <i>f</i> }
<i>a</i>	<i>c</i>	<i>f</i>		{ <i>a</i> , <i>b</i> , <i>c</i> , <i>e</i> , <i>f</i> }
<i>a</i>	<i>f</i>	<i>c</i>		{ <i>a</i> , <i>b</i> , <i>c</i> , <i>e</i> , <i>f</i> }
<i>a</i>	<i>f</i>	<i>e</i>		{ <i>a</i> , <i>b</i> , <i>c</i> , <i>e</i> , <i>f</i> }
<i>a</i>	<i>f</i>	<i>i</i>	[ <i>i</i> ]	{ <i>a</i> , <i>b</i> , <i>c</i> , <i>e</i> , <i>f</i> , <i>i</i> }
<i>a</i>	<i>i</i>	<i>f</i>	[]	{ <i>a</i> , <i>b</i> , <i>c</i> , <i>e</i> , <i>f</i> , <i>i</i> }
<i>b</i>				{ <i>a</i> , <i>b</i> , <i>c</i> , <i>e</i> , <i>f</i> , <i>i</i> }
<i>c</i>				{ <i>a</i> , <i>b</i> , <i>c</i> , <i>e</i> , <i>f</i> , <i>i</i> }
<i>d</i>			[ <i>d</i> ]	{ <i>a</i> , <i>b</i> , <i>c</i> , <i>d</i> , <i>e</i> , <i>f</i> , <i>i</i> }
<i>d</i>	<i>d</i>	<i>g</i>	[ <i>g</i> ]	{ <i>a</i> , <i>b</i> , <i>c</i> , <i>d</i> , <i>e</i> , <i>f</i> , <i>g</i> , <i>i</i> }
<i>d</i>	<i>d</i>	<i>h</i>	[ <i>g</i> , <i>h</i> ]	{ <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>i</i> }
<i>d</i>	<i>g</i>	<i>d</i>	[ <i>h</i> ]	{ <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>i</i> }
<i>d</i>	<i>g</i>	<i>h</i>		{ <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>i</i> }
<i>d</i>	<i>h</i>	<i>d</i>		{ <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>i</i> }
<i>d</i>	<i>h</i>	<i>g</i>	[]	{ <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>i</i> }
<i>e</i>				{ <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>i</i> }
<i>f</i>				{ <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>i</i> }
<i>g</i>				{ <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>i</i> }
<i>h</i>				{ <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>i</i> }
<i>i</i>				{ <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>i</i> }

← ta tabela nam pove kako deluje alg. ki smo ga uporabili za reševanje 1) naloge

- *s* – pove iz katerega vozlišča začneš zanko
- *u-v* – pove katerega povratno gledas' sedaj
- *Q* – seznam naslednjih, katerih sosedov moraš pregledati
- zadnji stolpec pa prikazuje že obiskana vozlišča

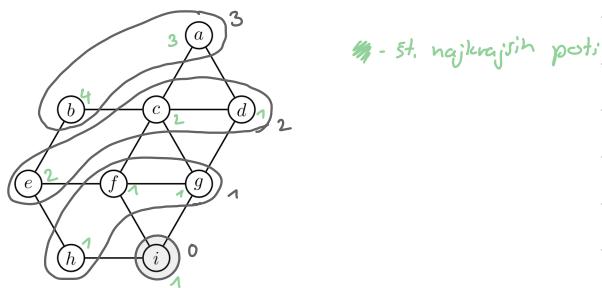
2. Zapiši algoritem, ki za vhodni graf  $G$  določi njegov premer.

```

def premer(G)
    d=0,
    for u in G:
        casovna zahitnost
        p,g = BFS(G,lu)
        if max(g.value())>d
            d=max(g.value())
    return d
    
```

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

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



```

def st.poti(G,s):
    poti = {u: 0 for u in G}
    globina = {s: 0}  to bomo potrebovali v visit, saj bomo
    def visit(v,w)   rabili pogledati katerim oglircem so na
                    nižjem nivoju. da s vedno da je globina 0
    if w is None:
        globina[v] = 0
        poti[v] = 1  stay vemo da smo
                    sedaj v s-ju
    else:
        globina[v] = globina[w] + 1
        for u in G.sosedji(v):
            if u in globina and globina[u] == globina[w]:
                poti[v] += poti[u]
    BFS(G,[s],visit)
    return poti
    
```

4. 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  $u, v \in S$ .

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  $\sum_{u \in S} c_u$ .

- (a) Denimo, da je drevo  $T$  podano kot neusmerjen graf, predstavljen s seznamimi 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  $pred[r] = \text{NULL}$ . Kako iz seznamov sosedov in slovarja *pred* ugotovimo, katera vozlišča so neposredni nasledniki danega vozlišča v drevesu?

$x_u \dots \max$  teža neodv. mn. poddrvešega s korenom v  $u$ ,

če izberemo  $u$

$y_u \dots -\text{II- ce ne izberemo } u$

$v \rightarrow u: v$  je prednodnik  $u$

$$x_u = c_u + \sum_{v \sim u} y_v$$

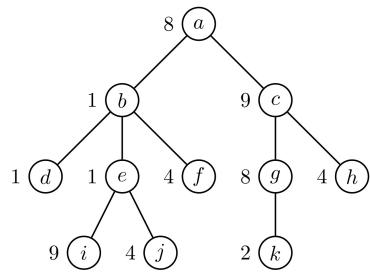
$$y_u = \sum_{v \sim u} \max \{x_v, y_v\}$$

robnih pogojev ne rabimo [vsake bodo prazne]

$$x^* = \max \{x_r, y_r\}$$

↓

$r$  je koren drevesa



## DIJKSTRA

 $G = (V, E)$  usmerjen graf $n = |V|, m = |E| + |V|$  $le \geq 0$  ( $e \in E$ ) ustrezi potreznemus  $\in V$  začetno vozliščedef dijkstra( $G, l, s$ ):    razdalja = { $s: 0$ }    prednik = { $s: \text{None}$ }    Q = vrsta ({ $u: \infty$  for  $u \in G$ })

Q[s] = 0

while len(Q) &gt; 0:

, d = Q.pop() → traja  $O(\log n)$  # vrni vozlišče in ustrezno vrednost, kjer je ta vrednost najmanjša

razdalja[u] = d

        for v in G.sosed(n): →  $O(m)$ 

r = razdalja[u] + luv

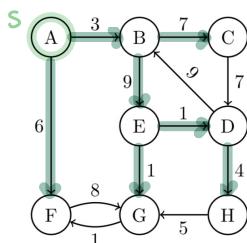
if r &lt; Q[v]

                Q[v] = r →  $O(1)$                 prednik[v] = u →  $O(\log n)$ 

return (razdalja, prednik)

Časovna zahtevnost: vrsta = slovar:  $O(n^2)$ vrsta = kopica:  $O(m \cdot \log n)$ 

1. S pomočjo Dijkstrovega algoritma določi razdalje od vozlišča  $A$  do ostalih vozlišč.



	A	B	C	D	E	F	G	H
None	*							
vzeto A	*	3/A				6/A		
najmanj		*	10/B	12/B				
			*	14/E				
				*	17/C			
					*	13/E		
						*	17/D	
							*	

# - predniki

2. 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 vhodu  $(G, \ell')$ , je tudi drevo najkrajših poti za graf  $G$  z dolžinami povezav, določenimi z  $\ell$ .

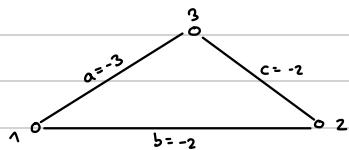
$$G = (V, E)$$

$$\ell : E \rightarrow \mathbb{R}$$

$$\ell' : E \rightarrow \mathbb{R}$$

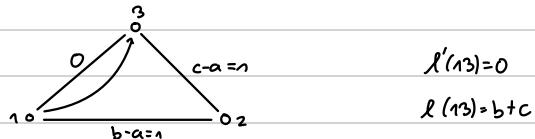
$$\forall e \in E : \ell'(e) = \ell(e) - \min\{\ell(f) \mid f \in E\}$$

OVRŽEMO s primerom



$$\cdot \text{naj bo } a = \min\{a, b, c\}$$

$$\cdot \text{če } a > b + c \quad \text{npr. } -3 > -2 - 2$$



3. 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).

Prati 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 linearinem 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.

$$G = (V, E) \text{ neusmerjen}$$

$V \dots$  mesta

$E \dots$  cesta

$\ell_e \dots$  dolžina ceste  $e$  (v km)

$\rightarrow S \cup T$  ( $S, T \subseteq V$ )

$L \dots$  št. kilometrov, ki jih lahko prepeljemo s polnim rezervoarjem, v vsakem mestu lahko natankamo

a) Algoritem :: linearen čas  $O(m)$

· nariše pot

$$1) G' = (V, \{e \in E ; \ell_e \leq L\}) \quad O(m)$$

$$2) g, p = BFS(G', \{s\}) \quad O(m)$$

$$3) \text{če pot obstaja} \Rightarrow \text{Tima globino/prednika} \quad O(m)$$

b) Izščemo najmanjše št. kilometrov, ki jih avto zmora prevoziti z enim polnim tankom

$$u, d = Q.pop()$$

#d... dolžina najkrajše povezave na poti od s do u

for v in G.sosed(u)

$$r = \max\{d, l_{uv}\}$$

if  $r < Q[v]$

$$Q[v] = r$$

$$\text{prednik}[v] = u$$

4. 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.

(a) Predlagaj čim učinkovitejši algoritmom 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  $t$  na spodnjem grafu. 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.

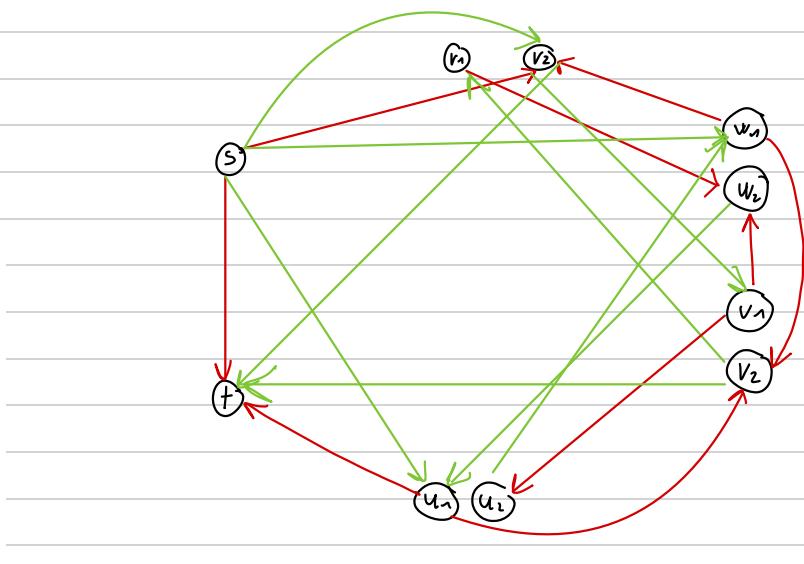
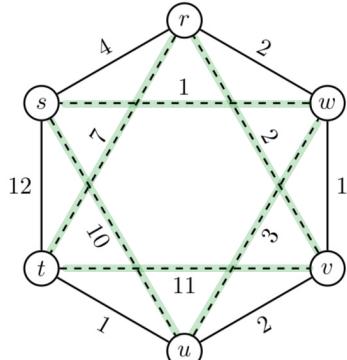
$G = (V, E)$  neusmerjen, utežen

OSele ... vozlišči povezave e

$A, B$  disjunktni množici povezav

$E = A \cup B$

najcenejša alternirajoča pot



## BELLMAN-FORD

Vhod:  $G = (V, E)$

$\det \text{BellmanFord}(G, \ell, s)$

$\ell: E \rightarrow \mathbb{R}$

$s \in V$

$n = |V|$

$m = |E|$

$\text{razdalja} = \{u: \infty \text{ for } u \in G\}$

$\text{prednik} = \{u: \text{None}\}$

$\text{nivo} = \{\}$

$\text{razdalja}[s] = 0$

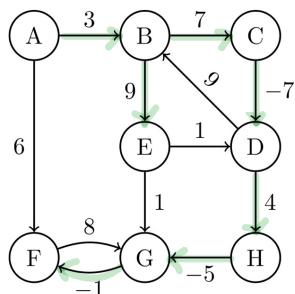
```

for _ in range(n):
    naslednji = set()
    for u in nivo:
        for v in G.sosed(u):
            r = razdalja[u] + luv
            if r < razdalja[v]:
                naslednji.add(v)
                razdalja[v] = r
                prednik[v] = u
    nivo = naslednji
    if len(nivo) == 0:
        break
else:
    # se izvede, če zanka pride do konca
    raise "V grafu je negativen cikel"
return (razdalje, predniki)

```

zanka  
for  
else

1. S pomočjo Bellman-Fordovega algoritma določi razdalje od vozlišča  $A$  do ostalih vozlišč.



časovna zahtevnost:

$$\sigma(nm)$$

$u$	$v$	$A$	$B$	$C$	$D$	$E$	$F$	$G$	$H$
$A$	BF	0	3 <sub>A</sub>				6 <sub>A</sub>		
$B$	CE			10 <sub>B</sub>		12 <sub>B</sub>			
$F$	G						14 <sub>F</sub>		
$C$	D				3 <sub>C</sub>				
$E$	GD						13 <sub>E</sub>		
$G$	F								
$D$	HB							7 <sub>D</sub>	
$G$	F								
$H$	G						2 <sub>H</sub>		
$G$	F					1 <sub>G</sub>			
$F$	G								

do  $F$  nismo zboljšali, zato pustimo prazno in  
3 zanka potem tudi ne gledamo več sosedov  
v naslednji zanki

ker smo  $G$  izboljšali, moramo spet pogledati če  
se izboljšajo tudi povezave do katerihkoli  
sosedov

ni izboljšav

najkrajše poti med vsakim parom vozlišč :

def FloydWarshall(G, l):

$$\text{razdalja} = \{(u, v) : \infty \text{ for } u \text{ in } G \text{ for } v \text{ in } G\}$$

$$\text{prednik} = \{(u, u) : \text{None for } u \text{ in } G\}$$

for  $u$  in  $G$ :

$$\text{razdalja}[u, u] = 0$$

$\sigma(m)$  for  $v$  in  $G.\text{sosed}(u)$ :

$$\text{razdalja}[u, v] = l_{uv}$$

$$\text{prednik}[u, v] = u$$

for  $w$  in  $G$ : # določimo  $w$  v notranjosti: poti

for  $u$  in  $G$ :

for  $v$  in  $G$ :

$$r = \text{razdalja}[u, w] + \text{razdalja}[w, v]$$

$\sigma(n^3)$  if  $r < \text{razdalja}[u, v]$ :

if  $u = v$ :

raise "Graf ima negativen cikel"

$$\text{razdalja}[u, v] = r$$

$$\text{prednik}[u, v] = \text{prednik}[w, v]$$

return (razdalja, prednik)

časovna zahtevnost:  $O(n^3)$

2. S pomočjo Floyd-Warshallovega algoritma poišči najkrajše poti med vsemi pari vozlišč v zgornjem grafu.

	A	B	C	D	E	F	G	H
A	0	$\frac{3}{A}$	$\frac{10}{B}$	$\frac{3}{C}$	$\frac{12}{D}$	<del><math>\frac{1}{E}</math></del>	<del><math>\frac{2}{F}</math></del>	$\frac{7}{H}$
B	0	$\frac{7}{B}$	0	$\frac{9}{B}$	<del><math>\frac{1}{E}</math></del>	<del><math>\frac{2}{F}</math></del>	$\frac{1}{H}$	$\frac{4}{D}$
C	$\frac{2}{D}$	0	$\frac{-7}{C}$	$\frac{11}{B}$	<del><math>\frac{-9}{G}</math></del>	<del><math>\frac{-8}{H}</math></del>	$\frac{-3}{D}$	
D	$\frac{9}{D}$	$\frac{16}{B}$	0	$\frac{18}{B}$	<del><math>\frac{-2}{G}</math></del>	<del><math>\frac{-1}{H}</math></del>	$\frac{4}{D}$	
E	$\frac{10}{D}$	$\frac{17}{B}$	$\frac{1}{E}$	0	<del><math>\frac{-1}{G}</math></del>	<del><math>\frac{0}{H}</math></del>	$\frac{5}{D}$	
F					0	$\frac{8}{F}$		
G					$\frac{-1}{G}$	0		
H					$\frac{-6}{G}$	$\frac{-5}{H}$	0	

- sosedji

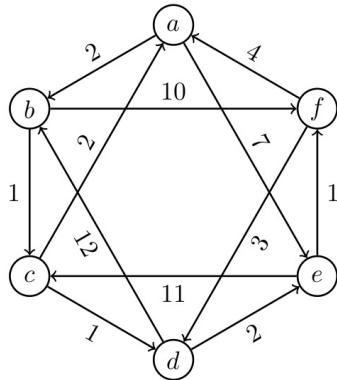
- medvezetne poti

Sprievodili smo, to  
smo oddelili to  
najprej sprememili v istem  
stolpcu, potem pa je v  
stolpcu G

Iger so prazne celice,  
ni take usmerjene  
povezave, zato je  $\infty$

3. 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? 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 spodnje slike, ki vsebujejo največ 4 povezave. Iz rešitve naj bo jasno, kako poteka izvajanje algoritma.



• usmerjen graf  $G = (V, E)$

•  $c_e \in \mathbb{R}$  (za  $e \in E$ )

•  $s \in V$

•  $0 \leq t \leq n = |V|$

Najkrajše poti od  $s$  do ostalih vozlišč, ki vsebujejo največ  $t$  povezav

• uporabimo Bellman-Ford algoritmom pri čemer globo zanko izvedemo največ  $t$ -krat

• Ne preprečujemo razdalj sproti, ampak šele, ko pregledamo celoten nivo

- (a) Uporabili bomo prirejeno različico Bellman-Fordovega algoritma, pri katerem bomo naredili le  $t$  obhodov glavne zanke.

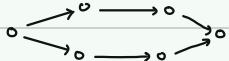
```

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 REKONSTRUIRAJPOT( $pred, u, t$ ) iz rešitve naloge 5.27.

## DAG [topološka ureditev grafov]



```
def topo(G):
```

```
    ureditev = []
```

```
    stopnja = {u: G.vhodnostopnja(u) for u in G} → slovar, ki za vsa vozlišča v G pove stopnjo vozlišča
```

```
    vrsta = {u for u, st in stopnja.items() if st == 0}
```

```
    while len(vrsta) > 0
```

vrne pare

```
        u = vrsta.pop()
```

```
        ureditev.append(u)
```

```
        for v in G.izhodniSosedji(u):
```

```
            stopnja[v] -= 1
```

```
            if stopnja[v] == 0:
```

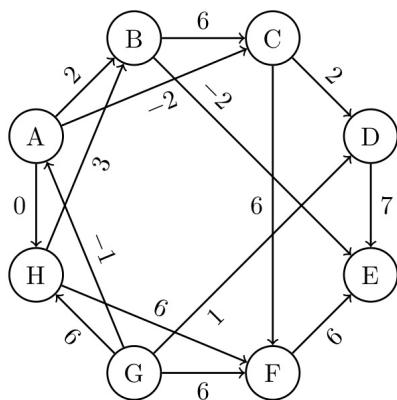
```
                vrsta.add(v)
```

```
if len(ureditev) < len(G):
```

```
    raise "Graf ni acikličen!"
```

```
return ureditev
```

1. Dan je sledeči usmerjen acikličen graf.



- Poisci topološko ureditev vozlišč zgornjega grafa.
- Poisci najkrajšo pot od vozlišča  $G$  do vozlišča  $E$ .
- Poisci najdaljšo pot od vozlišča  $G$  do vozlišča  $E$ .

a)	A	B	C	D	E	F	G	H
1	2	2	2	2	3	3	0	2
0				1		2	*	1
*	1	1						0
0						1		*
*	0		2					
	*	0			0			
		*	1					
			0	*				

← vhodne stopnje za usmerno vozlišče

← če dobimo dve 0, je vseeno kakšno utemeljeno

← ni novega nizetnega

← gledamo se, če tistega ki nam je postal

1) vhodne stopnje za vsako vozlišča

2) gledaš tisti, ki ima stopnjo 0  $\leftarrow \rightarrow$

3) izhodni vozliščem iz \* zmanjšaš stopnjo

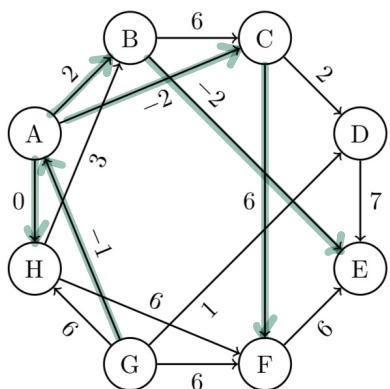
4) pogledaš naslednjega ki je niclev in spet gledas izhodne povezave

ZAPOREDJJE: GAHBCDFE

to zap je topoloska ureritev, torej je bi narisali vse povezave ~~██████████~~, bi videli da grejo samo nepej

časovna zahtevnost:  $O(m)$

b)	G	A	H	B	C	D	F	E	
	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	
*	$-1/G$	$6/G$				$1/G$	$6/G$		
*		$-1/A$	$1/A$	$-3/A$					
*			$3/H$	<del>3/A</del> iz A je izboljšano			$5/H$		
			*						
				$* -1/C$	$3/C$		$-1/B$		
					*				
							*		
									:

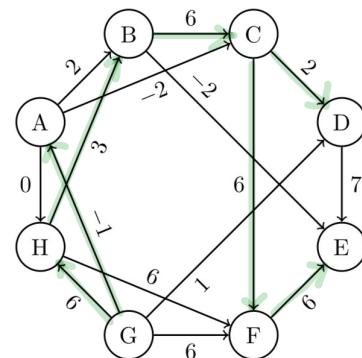


To so najkrajše poti od G do ostalih vozlišč

časovna zahtevnost:  $O(m)$

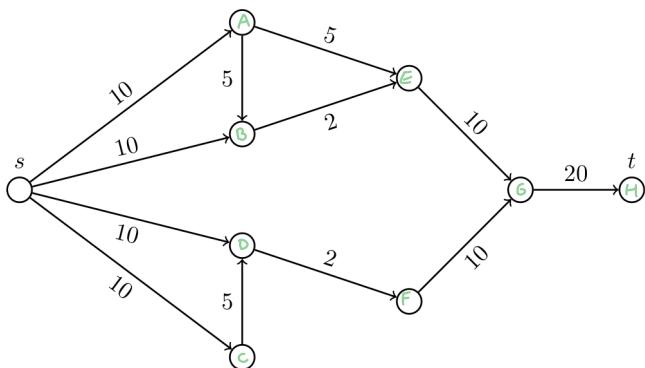
OP: Ta način bi lahko uporabili tudi iz kakšnega drugega vozlišča npr. H, vendar potem nebi mogli priti v G ali A.

c)	G	A	H	B	C	D	F	E	
	0	$-\infty$							
*	$-1/G$	$6/G$				$1/G$	$6/G$		
*			$1/A$	$-3/A$					
*			$9/H$			$12/H$			
*				$15/B$				$7/B$	
*					$17/C$	$21/C$			
*					*			$24/D$	
*						*		$27/F$	
*								*	



2. 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 utezen usmerjen graf  $G$  ter vozlišči  $s$  in  $t$ , ki predstavljata štart oziroma cilj. Uteži na povezavah nam predstavljajo, na koliko načinov jih lahko prečkamo.

(a) Reši nalogo za sledeci graf.



vsička smo označili, tako da smo ohramili običedni red v topološki vrstviti

tu bomo št. načinov množili in seznamili

primer:  
 $S \xrightarrow{10} A \xrightarrow{5} B$   
 50 načinov  
 +  
 $S \xrightarrow{10} B$   
 $\Rightarrow$  skupaj : 60

(b) Zapiši algoritem, ki reši dani problem. Kakšna je njegova časovna zahtevnost?

$s$	$A$	$B$	$C$	$D$	$E$	$F$	$G$	$t$	
1	0	0	0	0	0	0	0	0	
*	10	10	10	10					
*	60				50				
*					170				
		*		60					
			*		120				
				*		1700			
					*	1700			
						2900			
						*	58000		

KOMENTAR: Tu je bilo neko dinamično programiranje

Pv... št. načinov od s do v

$c_{uv} \rightarrow$  ulef na uv

$p_v = 1$

$$p_v = \sum_{u \neq v} p_u \cdot c_{uv}$$

$g_v$  ... dolžina najkratje poti od s do v

$$g_v = \max \{ g_u + c_{uv} \mid u \rightarrow v \}$$

$$g_s = \infty$$

## ISKANJE V GLOBINO

```

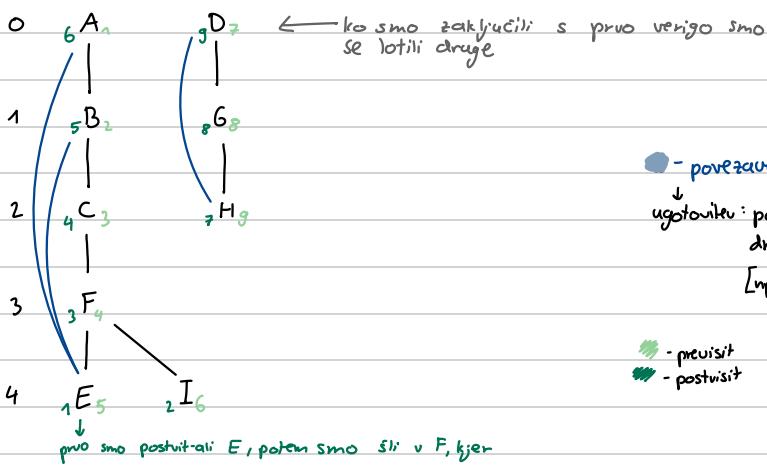
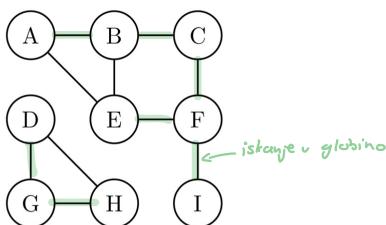
def DFS(G, koren = None, previsit = None, postvisit = None):
    if koren is None:
        koren = G.vozlisca()
    if previsit is None:
        previsit = lambda v,w = None
    if postvisit is None:
        postvisit = lambda v,w = None
    globina = { }
    predhodnik = { }

    def razisci(v,w):
        if v in globina:
            return
        predhodnik[v] = w
        globina[v] = 0 if w is None else globina[w]+1:
        previsit(v,w)
        for u in G.sosedji(v):
            razisci(u,v)
            postvisit(v,u)

    for u in koren:
        razisci(u,None)
    return (globina, predhodnik)

```

3. Na sledečem grafu 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.

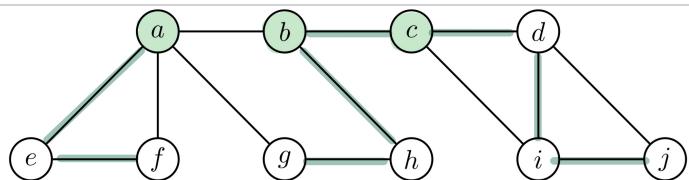


- povezave, ki jih nismo zapisali v drevo  
 ↓  
 ugotovitev: povezave iz zunanjega drevesa DFS:  $v \Rightarrow u$  je predhodnik  $v$  v tem drevesu (torej nimata različnega skupnega predhodnika)  
 [npr. nemore biti povezave med  $E$  in  $I$ ]

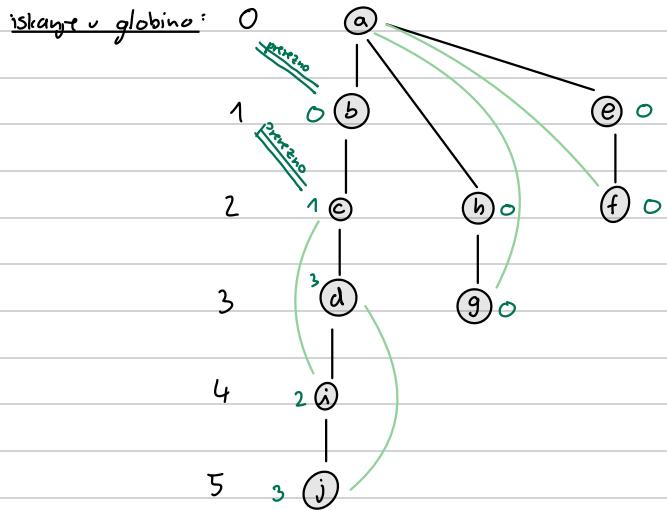
green dot - previsit  
 green square - postvisit

④ Izščemo prezena vozlišča v povezanim grafu  $G = (V, E)$

$u \in V$  je prezeno vozlišče, če je graf  $G-u$  nepovezan



- - prezena vozlišča
- - iskanje v globino



koren je prezeno vozlišče  $\Leftrightarrow$  ima več kot enega naslednika

notranje vozlišče je prezeno  $\Leftrightarrow$  obstaja veja brez povezave na višji nivo !

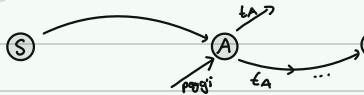
prv... najvišji nivo sosedov od u ali njegovih naslednikov

Torej u je prezeno  $\Leftrightarrow \exists v \in U : p_v \geq p_u$

## CPM

- opravila A
- trajanja za
- pogoji

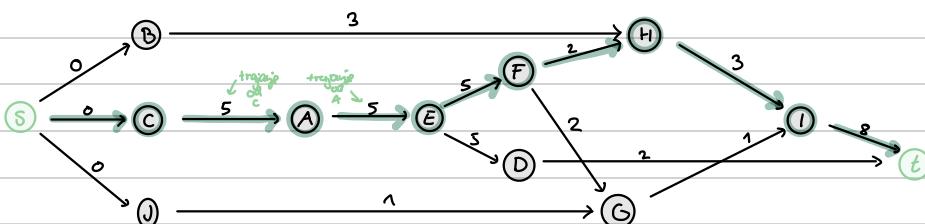
→ DAG



1. Dinamika priprave dveh palačink z dvema kuharjem je naslednja:

	aktivnost	trajanje	predhodna opravila
A	nakup moke, jajc in mleka	5 min	C
B	rezanje sira	3 min	/
C	vožnja do trgovine	5 min	/
D	čiščenje mešalnika	2 min	E
E	mešanje sestavin	5 min	A
F	pečenje prve palačinke	2 min	E
G	mazanje prve palačinke z marmelado	1 min	F, J
H	pečenje palačinke (s sirom)	3 min	B, F
I	pomivanje posode	8 min	G, H
J	odpiranje marmelade	1 min	/

- (a) Topološko uredi ustrezni graf in ga nariši.
- (b) Določi kritična opravila in kritično pot ter trajanje priprave.
- (c) Katero opravilo je najmanj kritično?
- (d) Določi razpored opravil, pri čemer en kuhan prevzame opravila na kritični poti, drugi pa naj čim kasneje začne in čim prej konča.



	S	B	C	J	A	E	F	D	H	G	I	t	
→	0	%	%	%	5/C	10/A	15/E	15/F	17/H	17/F	17/I	20/H	28/I
←	%A	14/H	%A	18/G	5/E	10/F	15/H	26/t	17/I	19/I	20/E	28	
razlika	0	14	0	18	0	0	0	11	0	2	0	0	

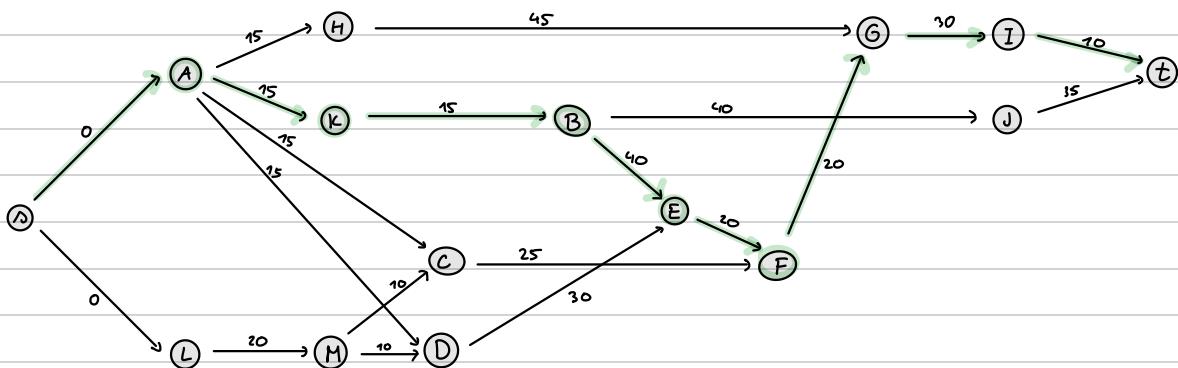
$\frac{15+2}{2} > 3$   
izčemo najdaljšo

- kjer je razlika = 0 so na kritični poti
- J je najmanj kritično opravilo (največja razlika)

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

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

- (a) Topološko uredi ustrezeni graf in ga nariši.  
 (b) Določi kritična opravila in kritično pot ter čas izdelave.  
 (c) Katero opravilo je najmanj kritično? Najmanj kritično je opravilo, katerega trajanje lahko najbolj podaljšamo, ne da bi vplivali na trajanje izdelave.



	S	A	L	H	K	M	C	D	B	E	F	G	J	I	t
→	0	%	%	15/A	15/A	20/L	30/M	30/L	30/K	70/B	90/E	110/F	70/J	140/G	150/I
←	%/A	%/K	10/M	65/G	15/B	30/D	65/F	40/E	30/E	70/F	90/G	110/I	115/t	140/E	150
razlika	0	0	10	50	0	10	35	10	0	0	0	0	45	0	0

↓ najmanj kritično opravilo

### PERT

- opravilo A:  
 - najmanje trajanje:  $a_A$   
 - najbolj verjetno trajanje:  $b_A$   
 - najdaljše trajanje:  $c_A$   
 - pogoji
- X... trajanje projekta  
 $X \sim N(\mu, \sigma)$   
 $P(X \leq t) = \Phi\left(\frac{t-\mu}{\sigma}\right)$

→ formula za pričakovano trajanje:  $\mu_A = \frac{a_A + b_A + c_A}{6}$   $\xrightarrow{\text{DAG}}$  pričakovana kritična pot  
 $\rightarrow$  pričakovano trajanje projekta

→ formula za varianco projekta:  $(\sigma_A)^2 = \left(\frac{c_A - a_A}{6}\right)^2$

3. Pri gradbenem podjetju razmišljajo, da bi se prijavili na razpis za prenovo avtocestnega viadukta.

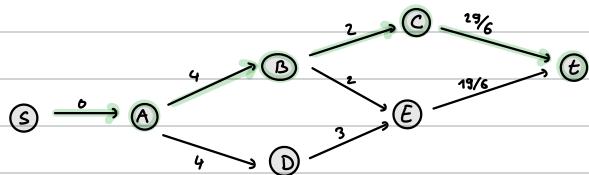
Identificirali so pet nalog:

naloga	<i>a</i> najkrajše trajanje	<i>b</i> najbolj verjetno trajanje	<i>c</i> najdaljše trajanje	predhodna opravila
A	3 tedni	4 tedni	5 tednov	/
B	2 tedna	2 tedna	2 tedna	A
C	3 tedni	5 tednov	6 tednov	B
D	1 teden	3 tedni	5 tednov	A
E	2 tedna	3 tedni	5 tednov	B, D

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€.

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

	<i>a</i>	<i>b</i>	<i>c</i>	pogoji;	pričakovano trajanje	$\sigma^2$
A	3	4	5	/	$24/6 = 4$	$4/36 = 1/9$
B	2	2	2	A	$12/6 = 2$	0
C	3	5	6	B	$25/6 \approx 4,17$	$1/4$
D	1	3	5	A	3	$16/36 = 4/9$
E	2	3	5	B,D	$19/6 \approx 3,17$	$1/4$



Odg: Na povzorce smo zapisali pričakovane vrednosti:

	S	A	B	D	E	C	t
→	0	$9/3$	$4/4$	$4/4$	$7/10$	$6/13$	$10,83/13$
←	$9/4$	$0/10$	$4/10$	$4,67/10$	$7,67/10$	$6/10$	$10,83$
razlika	0	0	0	0,67	0,67	0	0

$$\sigma^2 = \frac{1}{9} + 0 + \frac{1}{4} = \frac{4+9}{36} = \frac{13}{36}$$

$$\sigma = \sqrt{\frac{13}{36}}$$

$$d) P(X \leq 11) = \Phi\left(\frac{11 - 10,83}{\sqrt{13}}\right) = \Phi\left(\frac{11 - 10,83}{\sqrt{13}}\right) = \Phi\left(\frac{11 - 10,83}{\sqrt{13}}\right) = \Phi(0,277) = 0,6103$$

preberi iz tabele

e) Če bodo izbrani, zaslužijo 250.000€. Če ne bodo končali v 11 tednih plačajo pogodbeno kazen 500.000€

Naj se prijavijo?

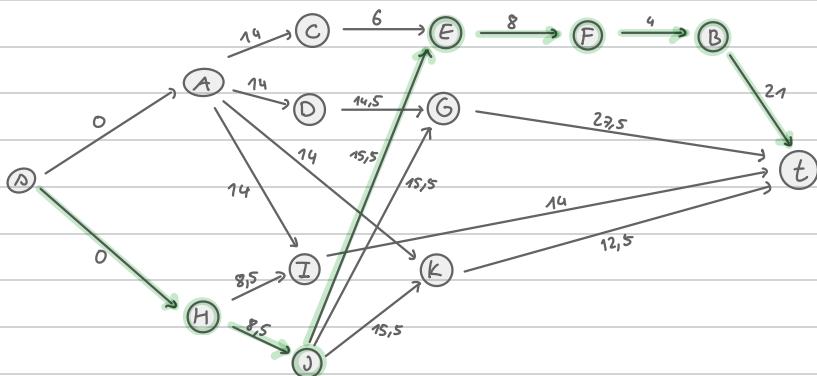
$$E(Y) = 250.000 - (1 - P(X \leq 11)) \cdot 500.000 = 250.000 - 0,3897 \cdot 500.000 > 0 \Rightarrow \text{splacata je prijaviti}$$

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

	Opravilo	Pogoji	Minimalno trajanje	Najbolj verjetno trajanje	Maksimalno trajanje	$\mu$
A	Izbira lokacije	/	10 dni	13 dni	22 dni	14
B	Rezervacija sob za goste	F	13 dni	22 dni	25 dni	21
C	Dogovarjanje za cene hotelskih sob	A	3 dni	6 dni	9 dni	6
D	Naročilo hrane in pihače	A	6 dni	15 dni	21 dni	14,5
E	Priprava letakov	C, J	5 dni	8 dni	11 dni	8
F	Pošiljanje letakov	E	4 dni	4 dni	4 dni	4
G	Priprava zbornika s povzetki	D, J	22 dni	28 dni	31 dni	27,5
H	Določitev glavnega govorca	/	5 dni	8 dni	14 dni	8,5
I	Planiranje poti za glavnega govorca	A, H	11 dni	14 dni	17 dni	14
J	Določitev ostalih govorcev	H	12 dni	15 dni	21 dni	15,5
K	Planiranje poti za ostale govorce	A, J	9 dni	12 dni	18 dni	12,5

- (a) Topološko uredi ustrezeni 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.

$$\mu_A = \frac{a_A + 4b_A + c_A}{6} = b_A + \frac{(c_A - b_A) - (b_A - a_A)}{6}$$



	S	A	H	C	D	I	J	E	G	K	F	B	t
→	0	0%	0%	14/A	14/A	14/A	8,5/H	29/J	28,5/D	24/J	32/E	36/F	57/B
←	0	100%	0%	18/E	15/G	43/I	8,5/E	24/G	29,5/I	44,5/E	32/B	36/I	57
razlika	0												

dokončaj sam doma

