

ČASOVNA ZAHTEVNOST ALGORITMOV

- Algoritem je zaporedje korakov, ki rešuje nek problem (vhodni podatki \rightarrow izhodni podatki). Potrebuje jasen opis in nujno je, da se ta konča.
- Želimo grobo oceniti "čas izvajanja" algoritma oziroma število osnovnih korakov algoritma. V najslabšem primeru isčemo zgornjo mejo števila korakov.

Časovna zahtevnost za algoritem A: $T_A: \mathbb{N} \rightarrow \mathbb{N}$

$n \mapsto$ največje št. korakov, ki ga algoritem uporabi za vhodne podatke velikosti $\leq n$

- n je liho \rightarrow popolno prirerjanje ne obstaja
- T_A je naraščajoča: $T_A(n+1) \leq T_A(n)$
- časovna zahtevnost pogledamo asimptotično

DEFINICIJA

$f, g: \mathbb{N} \rightarrow \mathbb{N}$ ali $R > 0$

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

PRIMER

$$\frac{3n+8}{f(n)} = O(n) \iff 3n+8 \leq \frac{11n}{c} \quad \forall n \geq n_0 = 7$$

$$4n^2 + 11n + 8 = O(n^2), \text{ saj je za } \forall n \in \mathbb{N}: 4n^2 + 11n + 8 \leq 23n^2$$

↓
glejamo kaj je relevantno v levem delu za velike n

$$4n^2 + 11n + 8 = O(n^3), \text{ saj je za } \forall n \in \mathbb{N}: 4n^2 + 11n + 8 \leq 23n^3 \rightarrow \text{ni lesna ocena}$$

f je reda krecjemu g $f(n) = O(g(n))$

$$\text{če } \exists \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)}, \text{ potem } f(n) = O(g(n)) \iff \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} \leq \text{konstanta} \rightarrow \text{notacija za zgornjo mejo}$$

limita je lahko enaka } 0, \text{ važno, da obstaja}

DEFINICIJA

$$f(n) = \Omega(g(n)) \text{ oziroma } f(n) \text{ je usaj reda } g(n) \text{ natanko tedaj ko } g(n) = O(f(n)) \ \exists c > 0, \exists n_0 \in \mathbb{N}: \forall n \geq n_0 \ f(n) \geq c \cdot g(n)$$

→ notacija za spodnjo mejo

PRIMER

$$4n^2 + 11n + 8 = \Omega(n^2), \ \Omega(n^3) \text{ ni pravilno, lahko damo } \Omega(n)$$

DEFINICIJA

$$f(n) = \Theta(g(n)) \text{ natanko tedaj ko } f(n) = O(g(n)) \text{ in } f(n) = \Omega(g(n))$$

$$\text{PRIMER: } 4n^2 + 11n + 8 = \Theta(n^2)$$

$$\begin{aligned} \text{ZGLED: } & \sum_{i=1}^n i^2 \leq n \cdot n^2 = n^3 \rightarrow O(n^3) \\ & \geq \frac{n}{2} \left(\frac{n}{2}\right)^2 = \frac{n^3}{8} \rightarrow \Omega(n^3) \end{aligned}$$

$\hookrightarrow i \geq \frac{n}{2}$

$$4 \log(n) \in O(\sqrt{n}) \subset O(n) \subset O(n^2) \subset O(2^{\frac{n}{\log 2}})$$

$O(n)$... linearno

$O(n^2)$... kvadratично

$O(2^n)$... eksponentialno

4 TIPIENE ČASOVNE ZAHTEVNOSTI

• $O(n)$ iskanje max v tabeli števil

• $O(n \cdot \log(n))$ algoritem za sortiranje števil (urejanje z zlivanjem (merge sort), pričakovana časovna zahtevnost za Quicksort)

• $O(n^3)$ množenje dveh $n \times n$ matrik (na klasičen način)

$$[A][B] = []$$

• $O(\log(n))$ binarno iskanje oz. bisekcija v urejeni tabeli števil

4 MOTIVACIJA

časovna zahtevnost	povečanje velikosti	povečanje časa
n	$\times 4$	$\times 4$
n^2	$\times 4$	$\times 16$
n^3	$\times 4$	$\times 64$
2^n	$\times 4$	$\times 16$ ($2^{n+4} = 2^n \cdot 2^4$)

4 OPOMBA: $4n = O(n) \rightarrow$ konstanta ni pomembna

~~$3^n = O(2^n)$~~ → baza je zelo pomembna

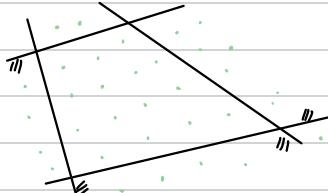
$$4^n = 2^{2n} = O(2^{2n})$$

CELOŠTEVILSKO LINEARNO PROGRAMIRANJE

$$\begin{array}{l} \hookrightarrow \max c^T x \\ \text{p.p. } Ax \leq b \\ x \geq 0 \text{ (vsestevki)} \\ x \in \mathbb{Z}^n \text{ NOVO} \end{array}$$

CLP

$$\hookrightarrow \text{MLP: mixed oz. mestano : } x_1, \dots, x_k \in \mathbb{R} \\ x_{k+1}, \dots, x_n \in \mathbb{Z}$$



$$\hookrightarrow V = \max c^T x \dots \text{optimalna vrednost CLP}$$

V_{rel} ... optimalna vrednost za LP, ko odstranimo pogoj $x \in \mathbb{Z}^n$ (relaksacija)

$V \leq V_{\text{rel}}$, ker je optimalna rešitev za CLP tudi dopustna rešitev za relaksacijo

Če optimalna rešitev za relaksacijo izpoljuje pogoj $x \in \mathbb{Z}^n$, potem je tudi optimalna rešitev za originalni problem

$$\hookrightarrow \text{ZGLED: } \max 2x_1 + 3x_2 + 5x_3$$

$$\begin{array}{l} \text{p.p.} \\ \hline \hline \end{array}$$

:

$\hookrightarrow \text{CLP:}$ • NP-težek \implies ne pričakujemo, da se lanko rešuje v pol. času

NP = nondeterministic poly

$\stackrel{?}{\text{no}} \text{ polynomial}$

• v praksi so "dobri" algoritmi

• v tem predmetu: modeliranje v CLP

• LP - MLP - CLP

\downarrow

mix LP

$x_1, \dots, x_k \in \mathbb{R}$

$x_{k+1}, \dots, x_n \in \mathbb{Z}$

ZGLED)

① POKRITJE Z MNOŽICAMI (Set Cover)

• Vhod: A končna množica

$$|A| = n$$

$$B_1, B_2, \dots, B_m \subseteq A$$

$$c_1, c_2, \dots, c_m \in \mathbb{R}_{>0} \quad [c_i \dots \text{ cena množice } B_i]$$

• Naloga: $\min \sum_{i \in I} c_i$

$$\text{p.p. } I \subseteq \{1, \dots, m\}, \quad \bigcup_{i \in I} B_i = A$$

- Zgled: $A = \{a, b, c, d, e, f\}$
- $B_1 = \{a, c, e\} \quad c_1 = 10$
- $B_2 = \{a, b, d, f\} \quad c_2 = 15$
- $B_3 = \{d, e, f\} \quad c_3 = 14$
- \vdots

B_1 in B_2 sta skupaj dopustna rešitev : $\{1, 2\}$ je dopustna rešitev cena 25
 B_1 in B_3 skupaj nista dopustna + $\{1, 3\}$ ni dopustna, saj $b \notin B_1 \cup B_3$

Zapišimo pokritje kot CLP

\rightarrow v splošnem: $\forall i \in [m] \quad x_i = \begin{cases} 1 & ; \text{če } B_i \text{ v rezult} \\ 0 & ; \text{sicer} \end{cases}$

$\boxed{\text{CLP}}$

$$\min \sum_{i=1}^m c_i x_i$$

p.p. $\forall a \in A: \sum_{\substack{i \in [m] \\ a \in B_i}} x_i \geq 1$

$\forall i \in [m]: x_i \in \{0, 1\}$

$\xrightarrow{\text{to je lepsi zapis, ker:}}$ $\begin{array}{l} x_i \geq 0 \\ x_i \leq 1 \\ x_i \in \mathbb{Z} \end{array} \begin{array}{l} \text{linearna} \\ \text{pogoja} \\ \text{celoštevki pogoji} \end{array}$

PREJŠNJI KONKRETNI PRIMER: $\min 10x_1 + 15x_2 + 14x_3 + \dots$

p.p. $x_1 + x_2 + \dots \geq 1$

$x_2 + \dots \geq 1$

$x_1 + \dots \geq 1$

$x_2 + x_3 + \dots \geq 1$

$x_1, x_2, x_3, \dots \in \{0, 1\}$

TRDITEV: I je dopustna rešitev za nalogu $\Leftrightarrow \left\{ \begin{array}{l} x_i = \begin{cases} 1 & ; i \in I \\ 0 & ; i \notin I \end{cases} \\ i = 1, \dots, n \end{array} \right\}$ je dopustna rešitev za CLP.

Cena dopustne rešitve I v nalogi = ciljnica vrednost v CLP za $x_i = \begin{cases} 1 & ; i \in I \\ 0 & ; i \notin I \end{cases}$

ŠTEVILLO OPERACI

$|A| = n$

B_1, \dots, B_m

m spremenljivk

$$\begin{aligned} &\left\{ \begin{array}{l} n \text{ pogojev } (\forall a \in A \dots) \\ + \\ O(m) \text{ pogojev } (\forall x_i \in \{0, 1\}) \end{array} \right. \\ &\implies O(n+m) \text{ pogojev} \end{aligned}$$

② HUMANITARNE KRIZE

Humanitarna kriza izbruhne na treh lokacijah: L_1, L_2, L_3 . Imamo 5 ekip, ki jih posljemo na pomoci. Ocena učinka:

št. ekip	L_1	L_2	L_3
0	0	0	0
1	45	20	30
2	70	45	70
3	90	75	80
4	105	110	100
5	120	150	130

• primer: če poslješ 2 v L_1 , 2 v L_2 , 1 v L_3 bo

skupni učinek $70+45+50$

• cilj: kako razdelimo ekipne na lokacije, da imamo max skupni učinek

max vsota učinkov

Za te podatke velja: \rightarrow za vsako lokacijo raste učinek glede na število ekip \Rightarrow opt. rešitev uporabi vse ekipne

\rightarrow učinek ni linearen na št. ekip

* POSKUSIMO ZAPISATI CLP

① $\forall i \in [3]: x_i = \text{št. ekip v lokaciji } L_i$

pogoji: $x_1 + x_2 + x_3 \leq 5$ ali $= 5$

$x_1, x_2, x_3 \in \{0, 1, 2, 3, 4, 5\}$ encaklo kot $x_i \leq 5$

$x_j \geq 0$

$x_j \in \mathbb{Z}$

ciljna funkcija: ne moremo jo zapisati. Lahko bi že bi bil učinek ekip linearen glede na št. ekip

NE DELA

② Za $i \in \{0, 1, \dots, 5\}$ in $j \in \{1, 2, 3\}$ upeljemo spremenljivke z namenom $x_{ij} = \begin{cases} 1 & \text{če } e_i \text{ posljemo načancno i.ekip} \\ 0 & \text{sicer} \end{cases}$

• CLP: $\max \sum_{\substack{i \in \{0, \dots, 5\} \\ j \in \{1, 2, 3\}}} a_{ij} x_{ij} = 0 \cdot x_{0,1} + 45 \cdot x_{1,1} + 70 \cdot x_{2,1} + \dots$
 $0 \cdot x_{0,2} + 20 \cdot x_{1,2} + 45 \cdot x_{2,2} + \dots$

...

p.p. $\forall i \in \{0, \dots, 5\} \forall j \in \{1, 2, 3\}: x_{ij} \in \{0, 1\}$

$\sum_{\substack{i \in \{0, \dots, 5\} \\ j \in \{1, 2, 3\}}} i \cdot x_{ij} \leq 5 \rightarrow$ poleg nam št. ekip

$\forall j \in \{1, 2, 3\} = \sum_{i \in \{0, \dots, 5\}} x_{ij} = 1 \rightarrow$ na vsako lokacijo posljemo samo dolozčno št. ekip (npr. neboš na 1. lokacijo poslat 1 in potem še 2.) $\rightarrow x_{0,1} + x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} + x_{5,1} = 1$
to je natancno ena spremenljivka, kjer bo preuzeela vrednost 1.

* SPLOŠNI CLP: če si imeti m ekip in n lokacij. Tabela velikosti $(m \times n) \times n: a_{ij}$

$x_{ij} \quad i \in \{0, \dots, m\} \quad j \in \{1, \dots, n\}$

$(m+n)n = O(nm)$ spremenljivk

ciljna funkcija: $\max \sum_i \sum_j a_{ij} x_{ij}$

dolžina: $O(nm)$

p.p. $\sum_j x_{ij} \leq \text{št. ekip}$

1 pogoju

$\forall j \in \{1, \dots, n\}: \sum_i x_{ij} = 1$

n pogoju

$\forall i \in \{0, \dots, m\}, \forall j \in \{1, \dots, n\}: x_{ij} \in \{0, 1\}$

$O(nm)$ pogoju

③ SUDOKU

			1					
	2		6					
			3					
				1	9			

$$x_{4,3,3} = 1$$

$$\begin{aligned} x_{ij} &\in \{1, \dots, 9\} \\ x_{11} &\neq x_{12} \quad \left\{ \begin{array}{l} x_{11} - x_{12} > 0 \\ \text{ali} \\ x_{11} - x_{12} < 0 \end{array} \right. \end{aligned}$$

Toni CLP

$$x_{ijk} = \begin{cases} 1 & ; \text{če je v položaju } (ij) \text{ matrike pravmo k} \\ 0 & ; \text{sicer} \end{cases}$$

$$x_{ijk} \in \{0, 1\} \rightarrow 9^3 \text{ spremenljivki}$$

$$\forall i,j : \sum_{k \in \{1, \dots, 9\}} x_{ijk} = 1 \quad \dots \text{v kvadratu } (ij) \text{ je eno število}$$

81 pogojev

$$\forall i,k : \sum_{j \in \{1, \dots, 9\}} x_{ijk} = 1 \quad \dots \text{na vrstici } i \text{ se pojavi število } k \text{ natanko enkrat}$$

81 pogojev

$$\forall j,k : \sum_{i \in \{1, \dots, 9\}} x_{ijk} = 1 \quad \dots \text{v stolpcu } j \text{ se pojavi število } k \text{ natanko enkrat}$$

81 pogojev

$$\forall Q \text{ } 3 \times 3 \text{ kvadrat} : \sum_{\substack{i,j \in Q \\ \text{kvadrat } 3 \times 3}} x_{ijk} = 1 \quad \dots \text{znotraj } 3 \times 3 \text{ kvadrata so vsa št.}$$

81 pogojev

$$\left[\begin{array}{l} \{(1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3)\} = \{1,2,3\} \times \{1,2,3\} = Q_1 \\ \{1,2,3\} \times \{4,5,6\} = Q_2 \end{array} \right]$$

Kaj max/min? poljubna funkcija. Dopolnilna rešitev CLP določi rešitev za sudoko

↳ saj je rešitev enolična

4×81 pogojev + pogoji tipa $x_{ijk} \in \{0, 1\}$ + vneseni podatki

RAZVEJJI IN OMESI

Ena ideja, kako se resi CLP: razvijji in omesti (branch & bound)

• CLP: $\max c^T x$

p.p. $Ax \leq b$

$x \geq 0$

$x \in \mathbb{Z}^n$



• RLP: $\max c^T x$

relaxiran p.p. $Ax \leq b$

$x \geq 0$

$x \in \mathbb{R}^n$

→ opt. rešitev: x^{*k}

opt. vrednost: v_k^*

• če $x^{*k} \in \mathbb{Z}^n$ imamo opt. rešitev

• če $x^{*k} \notin \mathbb{Z}^n$ → npr. $\left(\frac{103}{10}, 3, 10, \frac{1}{2}\right) = x^{*k}$

$\xrightarrow{x_1^{*k} \quad x_2^{*k}}$

• Pogledamo dve možnosti:

$$\begin{cases} x_1 \geq 11 \\ x_1 \leq 10 \end{cases}$$

→ [1] CLP₂: $\max c^T x$

p.p. $Ax \leq b$

$x \geq 0$

$x_1 \geq 11$

$x \in \mathbb{Z}^n \rightarrow$ ko oddorimo
dobimo RLP₂

Ker to neznano refevati
uporabljajo relaksacijo

→ [2] CLP₃: $\max c^T x$

p.p. $Ax \leq b$

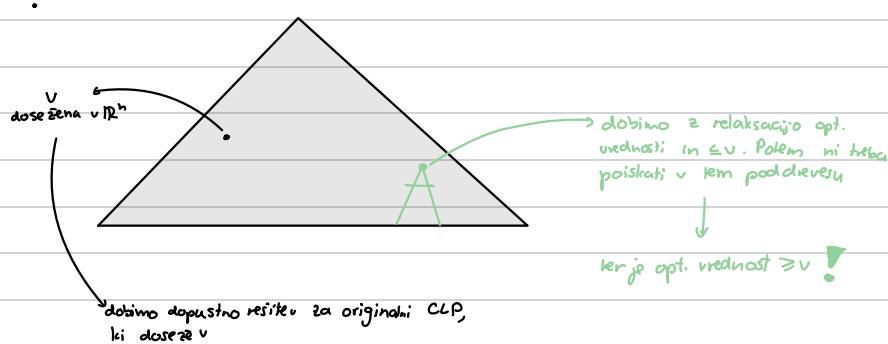
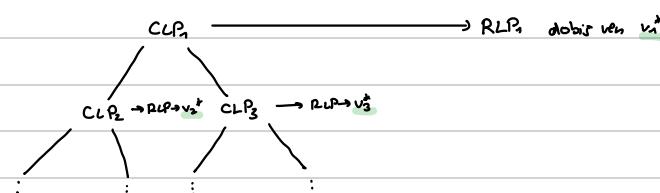
$x \geq 0$

$x_1 \leq 10$

$x \in \mathbb{Z}^n \rightarrow$ dobimo RLP₃

Optimalna rešitev za CLP je najboljša med opt. rešitvami za CLP₂ in za CLP₃

• kaj ubistva delas:



človljatna razlaga

Ideja razveji in omeji (*Branch and Bound*) pri celoštevilskem linearinem programirajuju (CLP) je iskanje optimalne rešitve, kjer morajo biti nekatere ali vse spremenljivke celoštevilske, s pomočjo postopnega razdeljevanja problema na podprobleme (razvejanje) in izločanja slabših možnosti (omejevanje), da bi se izognili nepotrebnemu preverjanju vseh kombinacij.

Omejevanje (Bounding):

Za vsak podproblem:

- Spet rešimo LP-relaksacijo.
- Če je rešitev slabša od že znane najboljše celoštevilske rešitve, ta veja ne more vsebovati boljše rešitve → jo zanemarimo.
- Če ima boljšo celoštevilsko rešitev, jo shranimo kot trenutno najboljšo.
- Če je rešitev neceloštevilska, a še vedno boljša od trenutne najboljše → ponovno razvejimo.

Ključna ideja:

Pri celoštevilskem LP imamo pogosto relaksirano (ne-celoštevilsko) rešitev, ki pa ni dovoljena. Zato:

1. Rešimo LP-relaksacijo (brez celoštevilskih omejitev).
2. Če so vse spremenljivke celoštevilske → rešitev je optimalna.
3. Če ni, izberemo neko spremenljivko x_i , ki ni celoštevilska (npr. $x_i = 4.7$), in ustvarimo dva nova problema:
 - $x_i \leq \lfloor 4.7 \rfloor = 4$
 - $x_i \geq \lceil 4.7 \rceil = 5$

Ta postopek se imenuje razvejanje (branching).

Namen metode:

- Sistematično raziskovanje možnosti.
- Izogniti se pregledu vseh kombinacij (kar bi bilo eksponentno).
- Zagotoviti globalno optimalno celoštevilsko rešitev.

2. Razveji in omeji

sistematično preverjamo vse možnosti in gradimo drevo stanj

- na vsakem koraku množico dopuščnih rešitev razdelimo v dve množici (razveji), vsaka tako množica vstreza vzdolžu v drevesu stanj
- izračunamo zgornjo in spodnjo mejo za optimalno vrednost v obih množicah (omeji)
- takoj, ko ugotovimo, da določena veja ne vodi do optimalne rešitve, jo "odrežemo"
- končamo, ko preglejamo vse veje ali sta nekje spodnja in zgornja meja enaki

SPLOŠNE METODE

① LOGIČNI IZRAZI

$p_i \equiv$ stavek "proizvajajo izdelek a_i "

$$x_i = \begin{cases} 1 & ; \text{izdelek proizvajajo} \equiv p_i = \text{TRUE} \\ 0 & ; \text{sicer} \end{cases}$$

Predpostavimo $x_i, x_j \in \{0,1\}$

operator | zapisemo v LP kot

$$p_i \Rightarrow p_j \quad x_i \leq x_j$$

$$p_i \Leftrightarrow p_j \quad x_i = x_j$$

$$p_i \vee p_j \quad x_i + x_j \geq 1 \quad [\text{ali}]$$

$$p_i \wedge p_j \quad x_i + x_j = 1 \quad \begin{array}{l} [\text{ali } x_i = 1] \\ \text{ali } x_j = 1 \end{array}$$

$$\neg p_i \quad x_i = 0$$

$$p_i \Leftrightarrow \neg p_j \quad : x_i = 1 - x_j$$

④ PRIMER:

• humanitarna kriza: $x_{ij} = \begin{cases} 1 & ; \text{če natančno i ekipo v } b_j \\ 0 & ; \text{sicer} \end{cases}$

• sudoku: $x_{ijk} = \begin{cases} 1 & ; \text{če je na položaju } (i,j) \text{ število } k \\ 0 & ; \text{sicer} \end{cases}$

⑤ ZGLED: $(p_1 \vee p_2) \Rightarrow (p_3 \vee p_4 \vee p_5)$

~~$x_1 + x_2 \leq x_3 + x_4 + x_5$~~

Lahko zapisemo kot $\begin{cases} x_1 \leq x_3 + x_4 + x_5 \\ x_2 \leq x_3 + x_4 + x_5 \end{cases} \quad \checkmark$

$$\left. \begin{array}{l} x_1 + x_2 \leq 2x \\ x \leq x_3 + x_4 + x_5 \end{array} \right\} \frac{x_1 + x_2}{2} \leq x_3 + x_4 + x_5$$

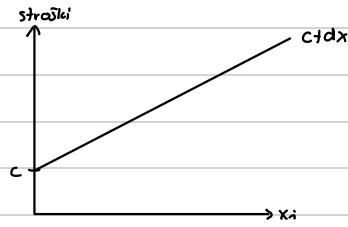
② ZAČETNI STROŠKI (MEJNI STROŠKI)

x_i = količina proizvodnje produkta a_i

$$\underline{\text{stroški}}: f(x_i) = \begin{cases} 0 & \text{če } x_i = 0 \\ cx_i & \text{če } x_i > 0 \end{cases}$$

ideja: uporabimo novo spremenljivko y_i , tako da

$$\begin{cases} y_i \in \{0,1\} \\ f(x_i) = cy_i + dx_i \\ x_i > 0 \Rightarrow y_i > 0 \end{cases}$$



Opozis problem in najdeš zgornjo mejo M_i za x_i : vedno velja $x_i \leq M_i$

$$\Rightarrow \text{naj pogoj: } y_i \geq \frac{x_i}{M_i}$$

• če $x_i = 0$, potem od naravnega problema, kjer minimiziramo stroške, bomo dobili $y_i = 0$

• če $x_i > 0$, potem $\frac{x_i}{M_i} \in (0,1]$ in $y_i = 1$.

4 ZGLED

$$f(x_1) = \begin{cases} 0 & \text{če } x_1 = 0 \\ 100 + 6x_1 & \text{če } x_1 > 0 \end{cases}$$

$$g(x_2, x_3) = \begin{cases} 0 & \text{če } x_2 = x_3 = 0 \\ 200 + 2x_2 + 4x_3 & \text{če } x_2 > 0 \text{ ali } x_3 > 0 \end{cases}$$

$$\min f(x_1) + g(x_2, x_3)$$

$$\text{p.p. } 1000x_1 + 400x_2 \geq 10^4$$

$$500x_1 + 400x_2 + 600x_3 \geq 10^4$$

$$x_1, x_2, x_3 \geq 0$$



$$\min 100y_1 + 6x_1 + 200y_2 + 2x_2 + 4x_3$$

$$\text{p.p. } x_1 + 2x_2 \geq 10 \quad \rightarrow \text{takej kdo } x_1 = 10, \text{ neodvisno od } x_2, x_3 \geq 0, \text{ pogoj velja}$$

$$5x_1 + 4x_2 + 6x_3 \geq 100 \quad \rightarrow \text{takej kdo } x_1 = 20, \text{ neodvisno od } x_2, x_3 \geq 0, \text{ pogoj velja}$$

$$y_1 \geq \frac{x_1}{20}$$

$$y_2 \geq \frac{x_2}{25} \quad \text{ali} \quad \underbrace{y_2 \geq \frac{x_2}{25} \quad y_2 \geq \frac{x_3}{50}}_{\text{boljše}}$$

$$x_1, x_2, x_3 \geq 0$$

$$y_1, y_2 \in \{0,1\}$$

v optimalni
rešitvi

$$x_2 = 5 \text{ izpoljuje 1. pogoj}$$

$$x_2 = 25 \text{ izpoljuje 2. pogoj}$$

$$x_3 = \frac{100}{6} = \frac{50}{3} \implies x_3 = 50 \text{ izpoljuje 3. pogoj}$$

$$\vee \text{optimalni rešitvi: } x_2 \leq 25, x_3 \leq 50$$

OD MAŠE - od lani razлага

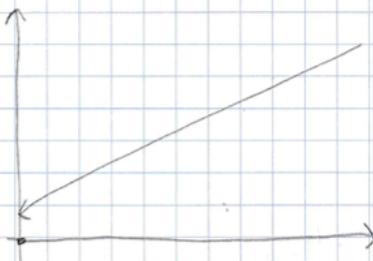
stroški niso nujno linearni

primer: če proizvajamo izdelek a_i , moramo postaviti proizvodno linijo
zanj.

x_i ... količina izdelka a_i (x_i m nujno celo števlo, običajno zahtivamo $x \geq 0$)

$$f(x_i) = \begin{cases} 0 & ; x_i=0 \\ c+dx_i & ; x_i>0 \end{cases}$$

začetni stroški



Idja: uvedemo novo spremenljivko y_i , $y_i = \begin{cases} 1 & ; x_i>0 \\ 0 & ; x_i=0 \end{cases}$

$$f(x_i) = c \cdot y_i + d \cdot x_i$$

$$\text{želimo: } x_i > 0 \Leftrightarrow y_i = 1$$

poskrbeti moramo:

$$1) x_i > 0 \Rightarrow y_i = 1$$

$$y_i \geq \frac{x_i}{M} \quad \text{za } x_i \leq M \text{ da } \frac{x_i}{M} \leq 1$$

M je zgornja meja za x_i (oz. "dovolj velik")
→ iz ostalih pogojev

$$2) x_i = 0 \Rightarrow y_i = 0$$

$$y_i \leq x_i \cdot M \quad ; M \text{ dovolj velik, da velja } x_i \cdot M \geq 1$$

$$\text{Iz } x_i \geq \frac{1}{M} \Rightarrow \frac{1}{M} \text{ je spodnja meja za } x$$

$$y_i \in \{0, 1\}$$

Ta pogoj zaradi naravnega problema (če isčemo min)
običajno ni potreben.
Avtar (če isčemo min, bo $y_i=0$, če $x_i=0$)

③ DISKRETE VREDNOSTI

$$x \in \{2, 3, 5, 7, 11, 13, 17, 19\} \equiv \begin{cases} a \leq x \leq b & \text{takže zapis ni vreden} \\ y \in \mathbb{Z} & \end{cases}$$

$$x \in A = \{a_1, a_2, a_3, \dots, a_k\} \equiv \begin{cases} a_1, a_2, a_3, \dots, a_k & \\ y_1, y_2, y_3, \dots, y_k & \text{nove sprem.} \\ y_1 + y_2 + \dots + y_k = 1 & \end{cases} \begin{cases} x = a_1 y_1 + a_2 y_2 + \dots + a_k y_k \\ y_1, \dots, y_k \in \{0, 1\} \\ y_1 + y_2 + \dots + y_k = 1 \end{cases}$$

DODATNA RAZLAGA OD LANCI

-DISKRETE VREDNOSTI SPREMENLJIVK

$x \in A$, A končna množica

$$\text{npr.: } A = \{0, 7, 15, 24, 45\}$$

$$A = \{a_1, a_2, \dots, a_k\}$$

$$\text{Uvedemo } x_i = \begin{cases} 1 & ; x_i = a_i \\ 0 & ; x_i \neq a_i \end{cases}$$

$$x \in A \equiv \begin{aligned} x_1 + x_2 + \dots + x_k = 1 & \quad (\text{x zavzame точно eno od vrednosti}) \\ x_1, \dots, x_k \in \{0, 1\} & \\ x = a_1 x_1 + a_2 x_2 + \dots + a_k x_k & \quad (\text{x zavzame pravo vrednost}) \end{aligned}$$

$$x \in \{0, 1\} \equiv \begin{aligned} x &\geq 0 \\ x &\leq 1 \\ x &\in \mathbb{Z} \end{aligned}$$

$$\text{Zajed: } \max x + y$$

$$\text{p.p. } 2x + 3y \leq 80$$

$$3x + 5y \leq 100$$

$$x \in \{1, 2, 4, 8, 16, 32, 64\} = A$$

$$y \in \{1, \frac{3}{2}, \frac{9}{2}, \frac{27}{2}, \frac{81}{2}, \frac{243}{2}\} = B$$

$$\max x + y$$

$$\text{p.p. } \begin{aligned} 2x + 3y &\leq 80 \\ 3x + 5y &\leq 100 \end{aligned}$$

$$x_1 + x_2 + \dots + x_7 = 1$$

$$x = x_1 \cdot 1 + x_2 \cdot 2 + x_3 \cdot 4 + \dots + x_7 \cdot 64$$

$$x_1, \dots, x_7 \in \{0, 1\}$$

$$y_1 + y_2 + \dots + y_6 = 1$$

$$y = y_1 + \frac{3}{2} y_2 + \dots + \frac{243}{2} y_6$$

$$y_1, \dots, y_6 \in \{0, 1\}$$

note spremenljivke:

$$x_i = \begin{cases} 1 & ; x = a_i \\ 0 & ; \text{micer} \end{cases}$$

$$y_n = \begin{cases} 1 & ; y = b_n \\ 0 & ; \text{sicer} \end{cases}$$

D.N.: lineariziraj posoj

oblic:

$$x_1 x_2 = x_3 \leftarrow \text{m. lineariz.}$$

$$x_1, x_2, x_3 \in \{0, 1\}$$

④ ZADOŠČANJE K-POGOJEM

$$\max c^T x$$

$$\begin{array}{l} p_1: a_1^T x \leq b_1 \\ p_2: a_2^T x \leq b_2 \\ \vdots \\ p_n: a_n^T x \leq b_n \end{array} \quad \left[\begin{array}{l} \text{zadoščati moramo vsaj} \\ k \text{ pogojev} \\ \text{ili} \\ m-k \text{ pogojev po zeffi} \\ \text{lanko odstranit} \end{array} \right]$$

$$\begin{pmatrix} x \geq 0 \\ x \in \mathbb{Z}^n \end{pmatrix}$$

14.3

↳ Upeljemo novo spremenljivko

$$y_i = \begin{cases} 1 & ; \text{pogoj } p_i \text{ ni treba izpolniti} \\ 0 & ; \text{sicer} \end{cases}$$

$$a_1^T x \leq b_1 + y_1 \cdot M$$

kjer je M dovolj velik in odvisen od problema

$$a_2^T x \leq b_2 + y_2 \cdot M$$

:

$$a_m^T x \leq b_m + y_m \cdot M$$

$$y_1 + y_2 + \dots + y_m = m-k$$

$$\{y_1, \dots, y_m\} \subseteq \{0, 1\}$$

$$\begin{pmatrix} x \geq 0 \\ x \in \mathbb{Z}^n \end{pmatrix}$$

↳ ZGLED

$$\max : 3x_1 + 5x_2 + 7x_3$$

$$\begin{array}{l} p_1 : p_1 : 2x_1 + 6x_2 + 5x_3 \leq 1000 \\ p_2 : 4x_1 + 2x_2 + 7x_3 \leq 2000 \\ p_3 : 3x_1 + 3x_2 - 4x_3 \leq 100 \\ p_4 : 2x_1 + x_3 \leq 2000 \end{array} \left. \begin{array}{l} \text{vsaj 2 morata} \\ \text{veljati} \end{array} \right\}$$

$$3x_1 + 3x_2 \leq 100 + 4x_3 \leq 100 + 4 \cdot 2000 = 8100$$

$$p_5 : \begin{array}{l} x_1, x_2, x_3 \geq 0 \\ x_i \in \mathbb{Z} \end{array}$$

($x_3 \leq 200$) bi bilo tuči uvedu, vendar isče najvišjo mimo

če p_1 velja (p_5) $\Rightarrow x_1 \leq 500, x_2 \leq 500, x_3 \leq 500$

če p_2 velja (p_5) $\Rightarrow x_1 \leq 500, x_2 \leq 1000, x_3 \leq 500$

če p_3 velja (p_5) $\Rightarrow ?$

če p_4 velja (p_5) $\Rightarrow x_1 \leq 1000, x_3 \leq 2000, x_2 \leq ?$

če vsaj 2 od p_1, p_2, p_3, p_4 veljata \Rightarrow vsaj ena izmed p_1, p_2, p_4 veljaca

(p_5) $\Rightarrow x_1 \leq 1000, x_3 \leq 2000$

$\Rightarrow (\text{ker } x_3 \leq 2000) \text{ od } p_3 (3x_1 + 3x_2 \leq 100 + 4x_3 \leq 100 + 8000 = 8100) \text{ sklepamo, da } x_1 \leq 3000 \text{ in } x_2 \leq 3000$

Torej če vsaj 2 izmed p_1, p_2, p_3, p_4 veljata, potem $x_1 \leq 1000, x_2 \leq 2000, x_3 \leq 3000$

na lev strani p_1 dobimo najvec : $2 \cdot 1000 + 2 \cdot 3000 + 5 \cdot 2000 = 18.000$

na lev strani p_2 dobimo najvec : $4 \cdot 1000 + 2 \cdot 3000 + 7 \cdot 2000 = 24.000$

na lev strani p_3 dobimo najvec : $3 \cdot 1000 + 3 \cdot 3000 + 0 = 12.000$

na lev strani p_4 dobimo najvec : $2 \cdot 1000 + 1 \cdot 2000 = 4.000$

Vzamemo $M=50.000$ [zbiral st. ki je \geq od 30000, 24.000, 12.000, 4.000]

$$\text{Nov problem: } \max : 3x_1 + 5x_2 + 7x_3$$

$$\text{p.p. } 2x_1 + 6x_2 + 5x_3 \leq 1000 + M \cdot y_1 \quad 2x_1, 2x_2$$

$$4x_1 + 2x_2 + 7x_3 \leq 2000 + M \cdot y_2$$

$$3x_1 + 3x_2 - 4x_3 \leq 100 + M \cdot y_3$$

$$2x_1 + x_3 \leq 2000 + M \cdot y_4$$

$$y_1, y_2, y_3, y_4 \in \{0, 1\}$$

$$y_1 + y_2 + y_3 + y_4 = 2 \rightarrow \begin{array}{l} 2 \text{ originalna veljata} \\ 2 \text{ og. ne veljata} \end{array}$$

$$x_1, x_2, x_3 \geq 0$$

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

TEORIJA ODLOČANJA

4 PRIMER:

Podjetje ima neko zemljišče, kjer je morda zaloga nafta. Geologi pravijo $P[\text{je nafta}] = \frac{1}{4}$. Če je nafta, je pričakovani zaslužek 800 k.

Druga firma ponudi, da bi kupil zemljišče za 90.000. Stroški vrtanja so 100.000.

	nafta je	nafta ni
prodaj	90	90
ne prodaj	700	-100

5 RAZLICNA PRAVILA

① "maximin", izhaja iz teorije iger

Lastnosti: - verjetnosti ne uporabimo

- garancije

- "konservativni"

max
odločitev min
stanje

prodaj	90	90	—————> 90
ne prodaj	700	-100	—————> -100

max: 90

② "maximin likelihood"

Ocenimo, da je stanje tisto z največno verjetnostjo

Lastnosti: - upošteva verjetnost

- zelo občutljiv na to kako naredimo skupine za stanja

	$JE = \frac{1}{4}$	$NJ = \frac{3}{4}$	
prodaj	90	90	
ne prodaj	700	-100	\Rightarrow prodaj

③ Bayesovo pravilo

max $E[\text{izid}]$

Lastnosti: - potrebne so dobre ocene za verjetnost

- ne upošteva varianco

	JE	NJ	E
prodaj	90	90	90
ne prodaj	700	-100	$\frac{1}{4} \cdot 700 + \frac{3}{4} \cdot (-100) = 100$

\Rightarrow ne prodaj

↳ 26 LED

Trgovina prodaja stvari, ki so nastopili dan neuporabne. Nabavna cena za eno stvar je 5, prodajna cena je 6.

Iz zgodovine ocevujemo : st. prodanih stvari na dan povpraševanje

Koliko stvari nabavimo	≤ 10	0
da max. profit?	11	0.1
	12	0.4
	13	0.3
	14	0.2
	≥ 15	0

POVPRASEVANJE

		0	0.1	0.4	0.3	0.2	0
		10	11	12	13	14	15
NABAVINO	10	10	10	10	10	10	
	11	11	11	11	11	11	
	12	6	12	12	12	12	
	13	1	7	13	13	13	
	14	-4	2	8	14		
	15	-9	-3	3	9		

E - profit

$$\begin{aligned} E &= 10 \cdot 0 + 11 \cdot 0.1 + 12 \cdot 0.4 + 13 \cdot 0.3 + 14 \cdot 0.2 + 15 \cdot 0 \\ &= 6 \cdot \frac{1}{10} + 12 \cdot \frac{4}{10} + 13 \cdot \frac{3}{10} + 14 \cdot \frac{2}{10} = 11.4 \end{aligned}$$

maxlikelihood : nabavimo 12

maximin : 11

Bayesovo : 12

↳ NADALJEVANJE PRIMERA Z NAFTO

Firma se lahko odloči za dodatno testiranje oz. geološke preiskave. Rezultati preiskave so lahko U-ugodni ali NU-ne ugodni. Spomimo se tudi, da velja $P[\text{nafta}] = \frac{1}{4} = 0.25$, $P[\text{nič}] = \frac{3}{4} = 0.75$

Vemo tudi $P[U | \text{stanje} = \text{nafta JE}] = 0.4$

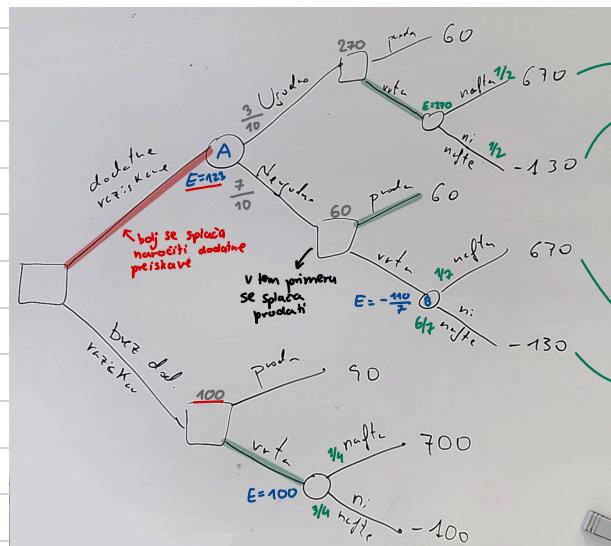
osebek (mi/firma) se odloči

$P[U | \text{stanje} = \text{nafta JE}] = 0.6$

narava odloči; [glejamo pričakovano vrn]

$P[U | \text{stanje} = \text{nafta NI}] = 0.8$

$P[U | \text{stanje} = \text{nafta NI}] = 0.2$



213

$$P[\text{nafta} | U] = \frac{P[U | \text{nafta}] \cdot P[\text{nafta}]}{P[U]} = \frac{\frac{6}{10} \cdot \frac{1}{4}}{\frac{3}{10}} = \frac{1}{2}$$

$$P[\text{nič} | U] = \frac{1}{2}$$

$$B: \frac{1}{2} \cdot 670 + \frac{6}{10} \cdot (-130) = -\frac{110}{10}$$

$$A: \frac{3}{10} \cdot 270 + \frac{7}{10} \cdot 60 = 3 \cdot 27 + 7 \cdot 6 = 123$$

$$P[\text{nafta je} | \text{NU}] = \frac{P[\text{NU} | \text{nafta}] \cdot P[\text{nafta}]}{P[\text{NU}]} = \frac{\frac{4}{10} \cdot \frac{1}{4}}{\frac{7}{10}} = \frac{1}{7}$$

$$P[\text{nafta ni} | \text{NU}] = \frac{6}{7}$$

uporabimo Bayesovo formulo

$$P[U] = P[U | \text{nafta}] \cdot P[\text{nafta}] + P[U | \text{nič}] \cdot P[\text{nič}] = \frac{3}{10}$$

$$P[\text{NU}] = \frac{7}{10}$$

UGOTOVITEV: Predlog, ki max $E[\text{dobriček}]$ je: Naročimo dodatne preiskave. Če je rezultat ugodno: vrtamo; če pa neugodno: prodamo

↳ POSTOPEK V SPLOŠNEM

Narišemo drevo, kjer "" mi dolocimo in kjer "" narava doloci.

Racunamo verjetnosti:

Racunamo $E[\text{dobriček}]$ od listov proti zacetku drevesa.

DINAMIČNO PROGRAMIRANJE

▫ Dinamično programiranje je pristop k reševanju problemov.

Značilnosti:

- problem razdelimo na manjše podprobleme

• od rešitev za podprobleme lahko sestavimo rešitev za celoten problem

(rešitev za celoten problem) \wedge (podproblem) = (rešitev za podproblem)

• isti podproblem se pojavlja večkrat. Zaradi tega dobimo rešitev enkrat in jo shranimo

↳ ZGLED 0

Fibonaccijeva številica računamo

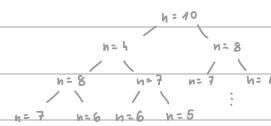
ALGORITM ZA FIB 1

vhod: $n \in \mathbb{N}$, $n \geq 1$

if $n=1$ ali 2 vrnemo 1

else vrnemo ($\text{Fib}(n-1) + \text{Fib}(n-2)$)

zelo zamudno



$\text{Fib}(1(m))$ $m \leq n$
računamo večkrat

ALGORITM FIB 2

vhod: $n \in \mathbb{N}$

nova tabela $A[1, \dots, n]$

$A[1] = A[2] = 1$

for $i=3$ to n do $A[i] = A[i-1] + A[i-2]$

vrnemo $A[n]$

To ni dinamično programiranje,
ampak bomo uporabili tak
princip shranjevanja podatkov

1 | 1 | 2 | 3 | 5 | ...

NAJDALJSÉ STROGO NARASČAJOĆE PODZAPOREDJE

▫ Podatki: zaporedje n števil $a_1, a_2, \dots, a_n \in \mathbb{R}$

▫ Naloge: isčemo najdaljše podzaporedje $a_{i_1}, a_{i_2}, \dots, a_{i_k}$ za katere velja:

→ $i_1 < i_2 < \dots < i_k$ (podzaporedje)

→ $a_{i_1} < a_{i_2} < \dots < a_{i_k}$ (strogo narasčajoće)

max k = dolžina podzaporedja

↳ ZGLED

5, 2, 8, 6, 3, 6, 9
 \bar{a}_2 \bar{a}_5 \bar{a}_6 \bar{a}_7

$i_1=2$, $i_2=6$, $i_3=7$, $i_4=8$, dolžina = 4

Želimo podzaporedje. Ponavadi je pogovarjano o njegovi dolžini:

Definiramo: $d_i = \text{dolžina najdaljšega podzaporedja za } a_1, \dots, a_i$ (prvih i elementov), ki ustreuje a_i

i	1	2	3	4	5	6	7	8
a_i	5	2	8	6	3	6	9	7
d_i	1	1	2	2	2	3	4	

$\begin{matrix} 3+1 & \text{zaradi } a_6 \\ 2+1 & \text{zaradi } a_5, a_6 \\ 1+1 & \text{zaradi } a_5, a_4 \end{matrix}$

\downarrow \downarrow \downarrow \downarrow

5 2 $(a_2, 8)$ $(2, 3, 6)$

DEFINICIJA :

Dolžina najdaljšega podzaporedja za a_1, \dots, a_n (prvih i elementov), ki vsebuje ali je definirana kot:

$$d_i = \begin{cases} 1 & ; \text{če } a_i = \min\{a_1, \dots, a_n\} \\ \max\{1 + d_j \mid j < i, a_j < a_i\} & \end{cases}$$

Zanimiva nas $\max\{d_1, d_2, \dots, d_n\}$

ALGORITEM

vhod : tabela $a[1 \dots n] // a[i] = a_i$

$d = \text{nova prazna tabela } d[1 \dots n]$

$J = \text{nova prazna tabela } J[1 \dots n]$

for $i=1$ to n do

```

    |  $d[i] = 1$ 
    | for  $j=1$  to  $i-1$  do
    |   | if  $a[j] < a[i]$  and  $d[i] < d[j] + 1$  then
    |   |   |  $d[i] = 1 + d[j]$ 
    |   |   |  $J[i] = j$ 
  
```

return $\max\{d[1 \dots n]\}$

shranimo $J[i]$ (predhodnike v podzaporedju)

$$J[i] = \begin{cases} \emptyset & ; \text{če } d[i] = 1 \\ k & ; \text{z lastnostjo } 1 + d[k] = d[i] \end{cases}$$

i	1	2	3	4	5	6	7	8		
a_i	5	2	8	6	3	6	9	7	0	3
d_i	1	1	2	2	2	3	4	4	1	2
J_i	/	/	1	1	2	5	6	6	/	2

optimum

$$\text{časovna zahtevnost : } \sum_{i=1}^n \sum_{j=1}^{i-1} \sigma(i) = \sum_{i=1}^n \sigma(i) = \sigma\left(\sum_{i=1}^n i\right) = \sigma(n^2)$$

↳ $d_i = \text{dolžina najdaljšega narasičajočega podzaporedja za } a_1, \dots, a_i$, ki vsebuje a_i (torej z a_i vred)

$$l_i = \begin{cases} 1 & ; \text{če } a_i = \min(a_1, \dots, a_i) \\ 1 + \max\{l_j \mid j < i \text{ in } a_j < a_i\} \end{cases}$$

↳ **TREITEV:** za vsak $i \in \{1, \dots, n\}$ velja $l_i = d_i$

DOKAŽI: Indukcija na i

$$\boxed{i=1}: d_1 = 1, l_1 = 1 \quad \checkmark$$

$$\boxed{i>1}: \text{Indukcijska predpostavka je } \forall j < i : l_j = d_j$$

$$d_i = \dots$$

Naj bo $\sigma = a_{i_1} a_{i_2} \dots a_{i_d}$ neko najdaljše narasičoče podzaporedje za a_1, \dots, a_n , ki se konča z a_i .

$$\text{če } a_i = \min(a_1, \dots, a_n) \Rightarrow l_i = 1 + d_i$$

Predpostavimo, sedaj da $a_i \neq \min(a_1, \dots, a_n)$

Naj bo $j = i_{d_i-1} < i$ j obstaja, ker ima σ dolžino vsaj 2.

Zaradi indukcijske predpostavke $l_j = d_j$

$$\Rightarrow l_i = 1 + \max\{l_k \mid k < i \text{ in } a_k < a_i\} \geq 1 + d_j = 1 + d_j \geq d_i$$

d_j je tukaj noter

OBRAZNO: če $l_i = 1$, potem $a_i = \max(a_1, \dots, a_n)$ in $d_i = l_i = 1$

$$\text{če } l_i > 1, \text{ potem } l_i = 1 + \max\{l_n \mid a_n < a_i ; n < i\}$$

Naj bo j indeks $\in \{n \mid a_n < a_i\}$ ja i in $a_j < a_i$ in obstaja podzaporedje σ' za a_1, \dots, a_j , ki konča pri a_j in ima dolžino $l_i - 1$.

To podzaporedje σ' lahko podaljšamo z a_i

To podaljšano podzaporedje $\sigma' a_i$: narasičoče

• konča pri $a_i \quad \Rightarrow d_i \geq l_i$

• dolžina l_i

$$\text{Skupaj } d_i \geq l_i \quad \boxed{l_i \geq d_i} \Rightarrow d_i = l_i$$

HUMANITARNA KRIZA

↳ Podatki: m ekip, n lokacij, $A \in \mathbb{R}^{m \times n}$

a_{ij} = učinek, če uporabimo i ekip v lokaciji l_j

↳ Nalog: koliko ekip uporabimo v posamezni lokaciji, da bo skupni učinek (vsota) čim večji.

Predpostavimo: $a_{0j} = 0$ za k_j

$a_{ij} \geq a_{i-n,j}$ za k_j

st ekip	L_1	L_2	L_3
0	0	0	0
1	45	20	50
2	70	45	70
3	90	75	80
4	105	110	100
5	120	150	130

Definiramo $\forall i \in \{0, \dots, m\}, j \in \{1, \dots, n\}$:

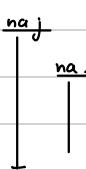
$R(i, j) = \text{največji skupni učinek (vsota), če imamo } i \text{ ekip za lokacije } L_1, \dots, L_j$

Zanima nas $R(m, n)$

↳ TRDITEV: $R(i, j) = \begin{cases} a_{i, n}, & \text{če } j=n \leftarrow \text{ker } a_{i, n} \geq a_{i, m} \\ \max \{a_{i, j} + R(i-l, j-1) \mid l \leq i\} & \text{če } j > 1 \end{cases}$

Dokazali naj bi z indukcijo

• 1. nacin



• 2. nacin \rightarrow indukcija na i, j

\rightarrow indukcija na (i, j) leksikografsko

$$(i, j) \leq (i', j')$$

①

$$i < i' \text{ ali } (i=i' \text{ in } j \leq j')$$

5 NADALJEVANJE NALOGE

$R[i,j]$	1	2	3
0	0	0	0
1	45	45	50
2	70	70	95
3	90	90	120
4	105	120	140
5	120	155	170

1.ekipa 3.ekipe 1.ekipa

$R[., 2]$

$$R[1, 2] = \max \{a_{0,2} + R(1, 1), a_{1,2} + R(0, 1)\} = \max \{0 + 45, 20 + 0\} = 45$$

$$R[2, 2] = \max \{a_{0,2} + R(2, 1), a_{1,2} + R(1, 1), a_{2,2} + R(0, 1)\} = \max \{0 + 70, 20 + 45, 45 + 0\} = 70$$

$$R[3, 2] = \max \{a_{0,2} + R(3, 1), a_{1,2} + R(2, 1), a_{2,2} + R(1, 1), a_{3,2} + R(0, 1)\} = \max \{0 + 90, 20 + 70, 45 + 45, 75 + 0\} = 90$$

$$R[4, 2] = \max \{a_{0,2} + R(4, 1), a_{1,2} + R(3, 1), a_{2,2} + R(2, 1), a_{3,2} + R(1, 1), a_{4,2} + R(0, 1)\} = \max \{105, 110, 115, 120, 100\} = 120$$

$$R[5, 2] = \max \{0 + 120, 20 + 105, 45 + 90, 75 + 70, 110 + 45, 150 + 0\} = 155$$

$$R[5, 3] = \max \{a_{0,3} + R(5, 2), a_{1,3} + R(4, 2), a_{2,3} + R(3, 2), a_{3,3} + R(2, 2), a_{4,3} + R(1, 2), a_{5,3} + R(0, 2)\} = \\ = \max \{0 + 155, 50 + 120, 70 + 90, 80 + 70, 100 + 45, 130 + 0\} = 170$$

6 ALGORITEM

• Uhodni podatki: $m, n, A[0 \dots m][1 \dots n]$

$R[0 \dots m][1 \dots n]$ nova prazna tabela // skupni učinek

$O(m) = m[0 \dots m][2 \dots n]$ nova prazna tabela // kjer je max dosežen // 1. stolpec

???

• for $i=0$ to m do $R[i, 1] = A[i, 1]$

for $j=2$ to n do

for $i=0$ to m do

$R[i, j] = A[i, j] + R[i, j-1]$

$m[i, j] = i$

for $k=0$ to $i=1$ do

if $A[k, j] + R[i-k, j-1] > R[i, j]$ then

$n \times m \times m = O(nm^2)$

$R[i, j] = A[k, j] + R[i-k, j-1]$

$m[i, j] = k$

največji skupni učinek je $R[m, n]$

ekipe[n] = $m[m, n]$

ostanek = $m - ekipe[n]$

for $j=n$ to 2 do

$ekipe[j] = m[ostanek, j]$

ostanek = $ostanek - ekipe[j]$

$ekipe[1] = ostanek$

V lokaciji l_j posljemo $ekipe[j]$ ekip.

RAZDALJA MED NIZI / UREJEVALNA RAZDALJA (edit distance)

• PODATKI: niz A[1..m] ponedeljek

niz B[1..n] četrtek

• Vprašanje: Najmanje število operacij, da iz niza A dobimo B. Operacije = $\begin{cases} \text{brisanje črke} \\ \text{ustavljanje črke} \\ \text{zamenjamo obstoječo črko} \end{cases}$

• ZGLED: • P O N E D E L J E K
 | | | | |
 C E T R T E K
 11 operacij

• P O N E D E L J E K
 | | | | |
 T R T
 7 operacij

• ALTERNATIVNI VIDIK: besede poravnamo s čim več ujemanj:

PONEDELJEK
--- C E T R T E K

• Cena = št. mest, kjer se ne ujemata (8)

• Zanima nas min cena

- $\boxed{-}$ \equiv ustavljam A
- \boxed{A} \equiv zbrisemo A
- \boxed{A} \equiv zamenjamo A z B

• Vhodni podatki: X[1..m]

Y[1..n]

$d(i,j) =$ urejena razdalja med $X[1..i]$ in $Y[1..j]$

(če $i=0$, potem $X[1..i]$ prazna beseda in podobno za $j=0$)

Zanima nas $d(m,n)$



• 3 možnosti za poravnovanje

①

$X[1..i-1]$	$x[i]$	$d(i-1, j) + 1$
$Y[1..j]$	prazen	

②

$X[1..i]$	prazen	$d(i, j-1) + 1$
$Y[1..j-1]$	$y[j]$	

③

$X[1..i-1]$	$x[i]$	$d(i-1, j-1) + \begin{cases} 1 & \text{če } x[i] \neq y[j] \\ 0 & \text{če } x[i] = y[j] \end{cases}$
$Y[1..j-1]$	$y[j]$	

④ v primeru da bi bil

--	--

 prazen NI NIKOLI OPTIMALNO

↳ 12REK

$$d(i,j) = \begin{cases} i & ; \text{če } j=0 \\ j & ; \text{če } i=0 \\ \min \{1+d(i-1,j), 1+d(i,j-1), 1+d(i-1,j-1)\} & ; \text{če } x[i] \neq y[j] \text{ in } i \neq 0, j \neq 0 \\ \min \{1+d(i-1,j), 1+d(i,j-1), d(i-1,j-1)\} & ; \text{če } x[i] = y[j] \text{ in } i \neq 0, j \neq 0 \end{cases}$$

gleđas min : ← $\square + 1$

$d(.,.)$ predstavimo s tabelo $d[.,.]$

$d(i,j)$	$j=0$	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
$i=0$	E	0	1	2	3	4
$i=1$	P	1	0	1	2	3
$i=2$	O	2	1	2	3	4
$i=3$	N	3	2	1	2	3
$i=4$	E	4	3	2	3	2
$i=5$	D	5	4	3	2	3
$i=6$	E	6	5	4	3	4
$i=7$	L	7	6	5	4	4
$i=8$	J	8	7	6	5	5
$i=9$	E	9	8	7	6	6
$i=10$	K	10	9	8	7	6

E = prazen niz / prazna množica za besede

REŠITEV:

P	O	N	E	D	E	L	J	E	K
P	-	-	E	-	-	T	-	E	K

cena je 6

↳ ALGORITEM

- Vhod: $x[1 \dots m]$, $y[1 \dots n]$
- Izhod: $d(m,n)$
- for $i=0 \dots m$ do
 - $d(i,0)=i$
 - for $j=0 \dots n$ do
 - $d(0,j)=j$
 - for $j=1 \dots n$ do
 - for $i=1 \dots m$ do
 - if $x[i]=y[j]$ then $\text{diff}=0$
 - else $\text{diff}=1$
 - $d(i,j) = \min \{d(i-1,j)+1, d(i,j-1)+1, d(i-1,j-1) + \text{diff}\}$
- return $d(m,n)$
- Časovna zahtevnost $O(m \cdot n)$

↳ OPOMBE : · Pazimo, da imamo res rekurenne enačbe

· konec rekurzije

· memoizacija (memoization)

→ uporabimo rekurenzen algoritmom, če vrednost nismo racunalni prej

→ ko računamo neko funkcijo se spomnimo rezultata

→ JA, brez R-ja je

↳ NOVA TABELA

$d[0 \dots m][0 \dots n]$ prazna

Alg 1(m,n)

Alg 1 (i,j)

if $i=0$ then return j

if $j=0$ then return i

if $d(i,j)$ je definiran, return $d(i,j)$

$$\text{diff} = \begin{cases} 1 & ; \text{če } x[i] \neq x[j] \\ 0 & ; \text{sicer} \end{cases}$$

$$d(i,j) = \min \{ \text{Alg}(i-1,j) + 1, \text{Alg}(i,j-1) + 1, \text{Alg}(i-1,j-1) + \text{diff} \}$$

return $d(i,j)$

DINAMIČNO PROGRAMIRANJE PRI OPERACIJSKIH RAZISKAVAH

Da lahko optimizacijski problem iz realnega življenja rešujemo z dinamičnim programiranjem, mora izpolnjevati določene zahteve.

1. Reševanje problema lahko predstavimo z zaporedjem etap.
2. Vsaka etapa vsebuje eno ali več : stanj.
3. Na vsaki etapi sprejmemo neko odločitev, ki pove, v katero od stanj v naslednji etapi se premaknemo. Pri tem je lahko naslednje stanje določeno:
 - deterministično (odločitev nas iz danega stanja pripelje v točno določeno stanje)
 - z dano verjetnostno porazdelitvijo (odločitev nas lahko iz danega stanja pripelje v več različnih stanj, v vsako z vnaprej določeno verjetnostjo)
4. Ko smo v nekem stanju, so nadaljnje odločitve neodvisne od teče, kako smo prišli v to stanje (pravilo optimalnosti)
5. Recimo, da pri reševanju problema nastopa n etap. Potem lahko zapisemo rekurzivne enačbe (Bellmanove enačbe), ki povezujejo zaslužek / stroške, določene v etapah $i, i+1, \dots, n$ z zaslужkom / stroški v etapah $i+1, \dots, n$ (optimalne odločitve običajno isčemo od zadnje proti prvi etapi).
oznake:
 $f_i(s) = \min/\max \{f_i(s_{ij}) | j \text{ odločitev v stanju } s\}$

- $f_i(s)$... optimalni prispevek h kriterijski funkciji (zaslužek / strošek), določen v etapah $i, i+1, \dots, n$, če začнемo v stanju s pri tem je s stanje v etapi i .
- $f_i(s_j)$... prispevek h kriterijski funkciji, če začnemo v stanju s in izberemo odločitev j , nato pa nadaljujemo optimalno (predpostavimo, da pri tem že poznamo $f_{i+1}(t)$ za stanje t v etapi $i+1$)

DINAMIČNO PROGRAMIRANJE V OR.

↳ Reševanje problemov ima etape oz. zaporedje etap. Vsaka etapa vsebuje eno ali več stanj. Na vsaki etapi spajmamo neko odločitev in ta odločitev določi stanje v naslednji etapi.

Stanje v naslednji etapi je lahko:

- deterministično

- z uporabo naključnosti

Ko smo v nekem stanju, so naslednje odločitve neodvisne od zgodovine oz. neodvisne od prejšnjih odločitev oz. neodvisne od tega kako smo prišli do stanja.

Uporabimo rekurzivne enačbe (Bellmannove enačbe), ki povezujejo zaslužek/stroške v etapi i z zaslužkom/stroški v etapi $i+1, i+2, \dots, n$ (in je zadnja etapa).

- $f_i(n)$ = optimalni prispevek k kriterijski funkciji dobjen v etapah $i, i+1, \dots, n$ ce zainemo v stanju s .
- $g_i(n, j)$ = prispevek k kriterijski funkciji, dobjen v etapah $i, i+1, \dots, n$ ce v etapi i spajmamo odločitev j in v etape $i=1, \dots, n$ določimo optimalno
- $f_i(n) = \max / \min \{ g_i(n, j) \mid j \text{ so možne odločitve v stanju } s \text{ v etapi } i \}$

2 GLED: Investicije

6000 € za investirat, 3 možne investicije (1, 2, 3), vsakr z vrednostjo

$$v_1(d_1) = 2 + 7d_1 \quad \text{ce } d_1 \geq 0$$

$$v_2(d_2) = 7 + 3d_2 \quad \text{ce } d_2 \geq 0$$

$$v_3(d_3) = 5 + 4d_3 \quad \text{ce } d_3 \geq 0$$

d_i je v 1000 €

$d_1, d_2, d_3 \in \mathbb{Z}_{\geq 0}$

$$v_i(0) = 0 \quad \text{za } i \in \{1, 2, 3\}$$

$$v_1(d_1) = \begin{cases} 0 & ; \text{ce } d_1 = 0 \\ 2 + 7d_1 & ; \text{ce } d_1 \geq 0 \end{cases}$$

$$\max v_1(d_1) + v_2(d_2) + v_3(d_3)$$

$$\text{p.p. } d_1, d_2, d_3 \geq 0$$

$$d_1, d_2, d_3 \in \mathbb{Z}$$

$$d_1 + d_2 + d_3 = 6$$

sezeden se sporeča investirati vse

Etapa i : investiramo v $i, i+1, \dots, 3$

stanje: v vsaki etapi moramo vedeti koliko nam ostane za investiranje

v etapan $i, i+1, \dots, 3$ so možna stanja: 0, 1, 2, ..., 6

$f_i(d)$... optimalna (nejvečja) neto vrednost, ki jo lahko dobimo, ce imamo d evrov za investicije v etapah $i, i+1, \dots, 3$
 \uparrow
 stanje

$x_i(d)$... kolicina denarja, ki jo investiramo v etapi i , da dosežemo $f_i(d)$

$g_i(d, x_i) = \text{opt. neto vrednost, ce investiramo } x_i \text{ v etapi } i \text{ in ostanek } d - x_i \text{ v etapah } i+1, i+2, \dots, 3$

$$f_i(d) = \max_{x_i} g_i(d, x_i)$$

$\lambda=3$

d	0	1	2	3	4	5	6
$f_3(d)$	0	9	13	17	21	25	29
$x_3(d)$	0	1	2	3	4	5	6

$$f_3(d) = v_3(d)$$

$$\begin{aligned} f_i(d) &= \max \{ g(d, x_i) \mid x_i \in \mathbb{Z}, 0 \leq x_i \leq d \} \\ &= \max \{ v_i(x_i) + f_{i+1}(d-x_i) \mid 0 \leq x_i \leq d, x_i \in \mathbb{Z} \} \end{aligned}$$

$\lambda=2$

d	0	1	2	3	4	5	6
$f_2(d)$	0	10	19	23	27	31	35
$x_2(d)$	0	1	1	1	1	1	1

$$\begin{aligned} f_2(4) &= \max \{ v_2(0) + f_3(4), v_2(1) + f_3(3), v_2(2) + f_3(2), v_2(3) + f_3(1), v_2(4) + f_3(0) \} \\ &= \max \{ 0+21, \underline{\underline{10+17}}, 13+13, 16+9, 19+0 \} \end{aligned}$$

$\lambda=1$

d	0	1	2	3	4	5	6
$f_1(d)$	0	9	16	23	30	37	44
$x_1(d)$							4

$$\begin{aligned} \text{za } \lambda=1 \text{ nema } f_1(6) &= \max \{ v_1(0) + f_2(6), v_1(1) + f_2(5), v_1(2) + f_2(4), v_1(3) + f_2(3), v_1(4) + f_2(2), v_1(5) + f_2(1), v_1(6) + f_2(0) \} \\ &= \max \{ 0+35, 9+31, 16+27, 23+23, \underline{\underline{30+19}}, 37+10, 44+0 \} \end{aligned}$$

\Rightarrow Optimalna investicija je $d=4$ za 1. investiciju (ostane 3)

$d=1$ za 2. investiciju (ostane 1)

$d=0$ za 3. investiciju

• ZGLED: IZDELAVA ZAHTEVNEGA IZDELKA

· naročilo za en izdelek

· izdelava je zahtevna. Verjetnost, da je izdelava uspešna je $\frac{2}{3}$ neodvisno od drugih poskusov.

· Izdelujemo v serijah in imamo čas za največ 3 serije

· Cena serije je $300 + 100x$ (st. izdelkov)

· Na koncu posamezne serije vemo, ali je nek izdelek OK.

· Če na koncu nismo izdelka pridemo kazen 1600.

· kako organizirati proizvodnjo, da bodo pričakovani stroški čim manjši

• ZGLED

→ nič ne delamo → ostane 1600

→ 1 serija, ker delamo 1 izdelek : $(300 + 100) + \frac{2}{3} \cdot 1600 = 1200$

$$P(\text{izdelek ni OK})$$

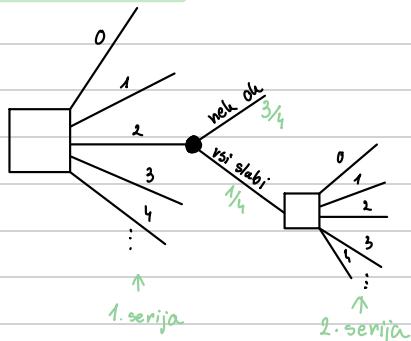
→ 1 serija, 2 izdelka : $(300 + 200) + \frac{2}{3} \cdot 1600 = 900$

$$P(\text{oba izdelka nista OK})$$

→ 1 serija z 1 izdelkom, če ni OK delamo 2. serijo z 1 izdelkom : $(300 + 100) + \frac{2}{3} (300 + 100 + \frac{2}{3} \cdot 1600) = 1000$

$$P(\text{delamo 2. serijo})$$

• ODLOČITVENO DREVO



Veliko podprtves je enakih → dinamično programiranje se spletca

· etape : serija 1, serija 2, serija 3 (sploča se tudi serijo 4, vendar smo se tu omejili na max 3)

· stanje : koliko izdelkov je rabimo $\in \{0,1\}$

· odločitve : koliko izdelkov naredimo v posamezni seriji x_1, x_2, x_3

· $C_i(s) = \text{minimum pričakovanih stroškov, če moramo v serijah } i, i+1, \dots \text{ narediti } s \text{ izdelkov } i \in \{1, 2, 3, 4\} \quad s \in \{0, 1\}$

$K(x) = \text{strošek izdelave } x \text{ izdelkov v seriji}$

$$K(x) = \begin{cases} 0 & ; \text{ če } x=0 \\ 300 + 100x & ; \text{ če } x>0 \end{cases}$$

• REKURZIVNE ZVEZE OB BELLMUNDOVE ENAČBE

$$i=4 : C_4(s) = \begin{cases} 1600 & ; \text{ če } s=1 \\ 0 & ; \text{ če } s=0 \end{cases}$$

$$i=1,2,3 : C_i(s) = \begin{cases} 0 & ; \text{ če } s=0 \\ \min \{ K(x) + \frac{2}{3} \cdot C_{i+1}(s-x), x=0,1,2, \dots \} & ; \text{ če } s=1 \end{cases}$$

Zanima nas $C_1(1)$

$i=3$	$c_3(0) = 0$	x_3	0	1	2	3	4	5	6	
	$c_3(1) = ?$	$c_3(1, x_3)$	1600	$400 + \frac{1}{2} \cdot 1600$	$500 + \frac{1}{4} \cdot 1600$	$600 + \frac{1}{8} \cdot 1600$	$700 + \frac{1}{16} \cdot 1600$	$800 + \frac{1}{32} \cdot 1600$	$900 + \frac{1}{64} \cdot 1600$	$\dots > 800$

\uparrow
 $E(\text{cena, ki v seriji 3 delamo } x_3 \text{ izd.})$

$$\min \text{ pri } x_3 = 3 \text{ ali } x_3 = 4 \implies c_3(1) = 800$$

$i=2$	$c_2(0) = 0$	x_2	0	1	2	3	4		
	$c_2(1) = ?$	$c_2(1, x_2)$	1600	$400 + \frac{1}{2} \cdot 800$	$500 + \frac{1}{4} \cdot 800$	$600 + \frac{1}{8} \cdot 800$	$700 + \frac{1}{16} \cdot 800$	$\dots > 700$	

\uparrow
 $c_2(1)$

$$\min \text{ pri } x_2 = 2 \text{ ali } x_2 = 3 \implies c_2(1) = 700$$

$i=1$	$c_1(0) = 0$	x_1	0	1	2	3			
	$c_1(1) = ?$	$c_1(1, x_1)$	1600	$400 + \frac{1}{2} \cdot 700$	$500 + \frac{1}{4} \cdot 700$	$600 + \frac{1}{8} \cdot 700$	675	$687,5$	$\dots > 700$

\uparrow
 $c_1(1)$

$$c_1(1) = 675 \text{ dosezen pri } x_1 = 2$$

* NACRT: Delamo $x_1=2$ izdelkov v prvi seriji. Če sta oba izdelka slaba, delamo drugo serijo za $x_2=2$ ali $x_2=3$ izdelke.

Če je tudi druga serija slaba, delamo 3. serijo z $x_3=3$ ali 4 izdelki. Če ne uspe, placamo kazen

$$E(\text{stroški}) = 675$$

* Analiziramo lahko, kaj se zgodi, če $P(\text{izdelek je napacer}) = \frac{1}{2} - \varepsilon$ za dvojno majhen $\varepsilon > 0$. Pomembno je, da so funkcije zvezne glede na verjetnost.

* Če bi morali odločiti med $x_2=2$ in $x_2=3$, bi odločili za $x_2=3$, če je ugled firme relevanten, ker s tem povečamo verjetnost, da izdamo izdelek. Podobno pri $x_3=4$.

ZGLED: MNOŽENJE MATRIK

Izmaimo zaporedje matrik A_1, A_2, \dots, A_n

Dimenzije so take, da obstaja produkt $A_1 A_2 \dots A_n$

V katerem vrstnem redu delamo produkt, da delamo čim manj operacij?

$A \in \mathbb{R}^{p \times 2}$ $B \in \mathbb{R}^{2 \times r}$

$$p \begin{bmatrix} A \\ \vdots \\ A \end{bmatrix} \begin{bmatrix} B \\ \vdots \\ r \end{bmatrix} = \boxed{\quad}$$

porabimo $\approx p \times 2 \times r$ operacij (to bo ocena za čas produkta)

zagled $A_1 \in \mathbb{R}^{10 \times 100}$

$$A_2 \in \mathbb{R}^{100 \times 5} \Rightarrow (A_1 A_2) A_3 \rightarrow 10 \times 100 \times 5 + 10 \times 5 \times 50 = 7500$$

$$A_3 \in \mathbb{R}^{5 \times 50} \Rightarrow A_1 (A_2 A_3) \rightarrow 100 \times 5 \times 50 + 10 \times 100 \times 50 = 25000 + 50000 = 75000$$

$$A_2 A_3 \in \mathbb{R}^{100 \times 50}$$

vhodni podatki

$$p_0, p_1, \dots, p_n \in \mathbb{N}$$

tako da A_i je velikosti $p_{i-1} \times p_i$ (v zgledu $p_0=10, p_1=100, p_2=5, p_3=50$)

Definirajmo: $m[i, j] =$ št. operacij za računanje produkta $A_i \dots A_j$ za $i \leq j$

$$m[i, i] = 0$$

$$\begin{array}{ccccccccc} A_i & A_{i+1} & \dots & A_{j-1} & A_j \\ () & () & & & () \\ () & () & & & () \\ () & A_k & () & & () \end{array}$$

TRDITEV

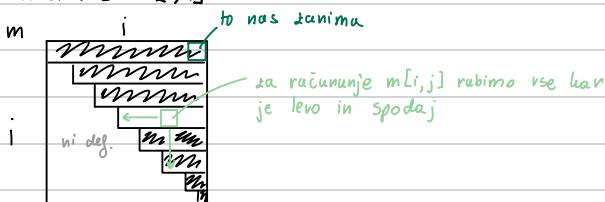
$$m[i, i+1] = p_{i-1} p_i p_{i+1}$$

$$m[i, j] = \min \{ m[i, k] + m[k+1, j] + p_{i-1} p_k p_j \mid k=i, i+1, \dots, j-1 \} \text{ za } j > i+1$$

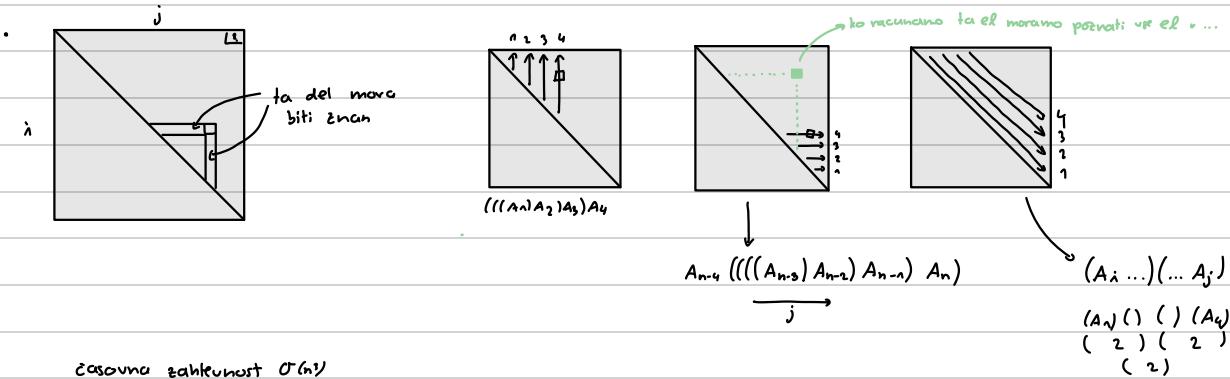
čas za $(A_i \dots A_k) (A_{k+1} \dots A_j)$

$\in \mathbb{R}^{p_{i-1} \times p_i}$ $\in \mathbb{R}^{p_k \times p_j}$

Zanima nas $m[1, n]$



18.4



ALGORITMI NA GRAFIH

1. OSNOVE

$G = (V(G), E(G))$

$V \dots$ vozilšča

$E \dots$ povezave

usmerjen / neusmerjen

enostaven / multigrat ($\equiv \exists$ vzporedne povezave)

uteži oz. cene na povezavah

$w: E(G) \rightarrow \mathbb{R}$

$w : E(G) \rightarrow \mathbb{R}_+$

skoraj vedno

NEUTEŽEN GRAP $\equiv w(e) = 1 \forall e \in E(G)$

[vsaka povezava ima utež ena, tako da sledijo povezave]

pot - zap. vozilci ali povezav (vozilci se ne smijo ponavljati)

sprehod - (lahko ponavljajo vozilca oz. povezave)

obhod - sprehod, ki se konča tam kjer se začne \equiv sklenjen sprehod



povezava graf (neusmerjen) $\equiv \forall u, v \in V(G)$ obstaja pot/sprehod od u do v .

uteži poti/sprehoda \equiv seštejemo uteži povezav v sprehodu s ponavljanjem

$w(P) = \sum_{e \in E(P)} w(e)$

2. PREDSTAVITEV GRAFA

z matriko sosednosti (adjacency matrix)

\rightarrow neusmerjen graf

- neutežen: $a_{ij} = \begin{cases} 1 & ; \text{če } ij \in E(G) \\ 0 & ; \text{če } ij \notin E(G) \end{cases}$

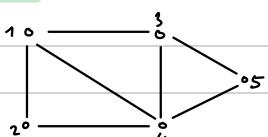
- utežen: $a_{ij} = \begin{cases} w(ij) & ; \text{če } ij \in E(G) \\ \text{NULL} & ; \text{če } ij \notin E(G) \end{cases}$

\downarrow
nesme dati 0, saj je lahko povezava
z ceno 0

- velikost matrike je st. vozilcev \times st. vozilcev: $(a_{ij})_{ij \in V(G)}$

- matrika je simetrična

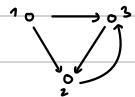
ZGLED



$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$	simetrija
---	-----------

→ usmerjen graf

$$- a_{ij} = \begin{cases} 1 & ; \text{ če } i \rightarrow j \in E(G) \\ 0 & ; \text{ sicer} \end{cases}$$



$$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

lastnosti: → uporabi $\Theta(n^2)$ prostora, neadvizno od m

$$m = |E(G)|$$

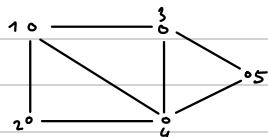
$$n = |V(G)|$$

→ lažje uporabiti:

→ V času $\Theta(1)$ lahko preverimo ali obstaja povezava

• s seznamom sosednosti (adjacency list)

→ Za vsako vozlišče $u \in V(G)$ imamo seznam ali tabelo $Adj[u]$, ki pove povezave oz. sosedje od u



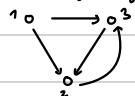
ADJ	
1	→ seznam (3, 4, 5) ali $((1, 2), (1, 4), (1, 3))$
2	→ seznam (1, 4) ali $((1, 2), (2, 4))$
3	→ seznam (1, 4, 5) ali ...
4	→ seznam (5, 2, 3) ali ...
5	→ seznam (2, 4) ali ...

Prostor $\Theta(ntm)$, ker vsaka povezava se pojavi 2x

$$\Theta = \begin{cases} O \rightarrow \\ \Omega \rightarrow \end{cases} \quad \begin{array}{l} \text{neje je ocena} \\ \text{in nekaj je spodnja} \\ \text{mogoča ??} \end{array}$$

↳ dodatna opomba

→ za usmerjene grafe



ADJ	
1	→ seznam $(1, 2), (1, 3)$
2	→ seznam $(2, 3)$
3	→ seznam $(3, 2)$

→ značilnosti:

- to bomo uporabili veliko

- prostorsko je optimalno

- preverjanje ali je posamezna povezava v grafu je počasnejše

uv $E(G)$ $Adj[u] \rightarrow$ seznam dolžine $\deg_G(u)$

$$\Theta(\deg_G(u))$$

- uporabili bomo veliko "for $v \in Adj[u]$ do"

for $i = 1, \dots$

- uporaba zgoščevalnih funkcij

Python (namesto seznamov uporabi zgoščevalne funkcije)

- ## • implicitni opisi

Zgled: Sznam tok $P \vee \mathbb{R}^2$

graf ($V(G) = P$, $E(G) = \{uv \mid \begin{matrix} u \in P \\ v \in P \end{matrix} \text{ } |uv| \leq 1\}$)

3. NAJKRAJŠA POT OZ. SPREHOD

- G graf, mogocie ukořen

$$s, t \in V(G)$$

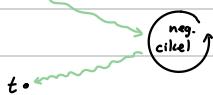
- Pogledamo use sprehode od s do t in vzamemo eno, ki min ulegi.

$$d_G(s, t) = \min \{ w(P) \mid P \text{ sprichod od } s \text{ do } t \text{ v grafu} \}$$

razdalja v G od s do t.

- OPOMBE:** $\rightarrow d_G(v, t) = -\infty$ oz. min ne obstaja natanko teče, ko v G obstaja cikel negativne uteži.

8



Prüfung

\rightarrow od zdaj naprej predpostavimo, da ne obstaja cikel negativne uteži

\rightarrow ic so alezi > 0 , negatiunih cikluu himamo

→ Lahko vidimo, da ob predpostavki $w(e) > 0$ v EEE in G neusmerjen, potem je $d_G(\cdot, \cdot) : V(G) \times V(G) \rightarrow \mathbb{R} \cup \{\infty\}$

metriku

✓ Je graf
ni povezen?

- Zanima nas računanje razdalj v grafu

→ neuterzen ④

$$\rightarrow \text{utezen} > 0 \quad (5)$$

→ akten bez neg. skrivil (6)

• 2 problema

\rightarrow SSSP = single-source shortest path

$d_G(s, t)$ za podan s in t

$\rightarrow APSP \equiv$ all-pairs shortest path

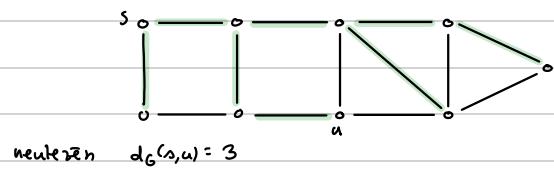
$$d_G(s, t) \quad \forall s, t \in V(G)$$

- Ponavadi hocemo najkrajšo pot

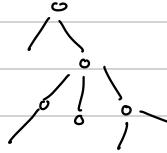
P je najkratča pot od s do $t \equiv d_G(s,t) = w(P)$ & P od s do t

- Drevo T je drevo najkrajsih poti od s , če velja $d_G(s, u) = d_T(s, u)$ $\forall u \in V(G)$

26 LED



- dveo nojkrayish poti



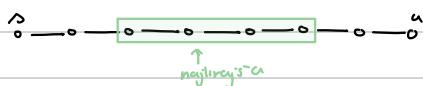
zdi se mi, da
se njemu ni dalo
narisati legu
druesa, zato
je kar samo
odčrkal drugo
v grafu

V casih obstaja več dices najkrajših poti

Ko ni negativnih ciklov vedno obstajajo dve najkrajših poti

(Lahko bi pokazal direktno z dog, bo pa automaticno sledilo kasnoje)

Glaunc idga : podpot od najkrajše poti je najkrajša pot

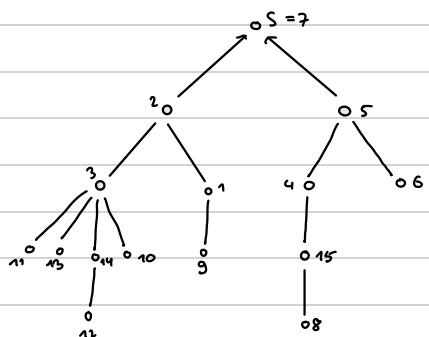


ce so neg. cikli to ni res



- konstruirali bomo drevu najkrajših poti

Predstavíkov je tabuľa súčetov /starcov



$\sqrt{1}$	1	2	3	4	5	6	7	8
odd starts	2	7	2	5	7		7	13

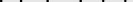
$T_j[u] =$ prvo vozlišće u drevusu na putu od u do t_{oren}

$$\text{IT}[\text{v}] = \begin{cases} \text{None} / \text{null} \\ ? \end{cases}$$

Nas zanimajojo poti od s do v , ampak to shrabimo obratno $v, \text{JT}(v), \text{JT}(\text{JT}(v)), \dots, s$

Ko nas zanima pot od s do v , konstruiramo pot od v do s in jo obrnemo.

Racunali bomo tabelo razdalij od s

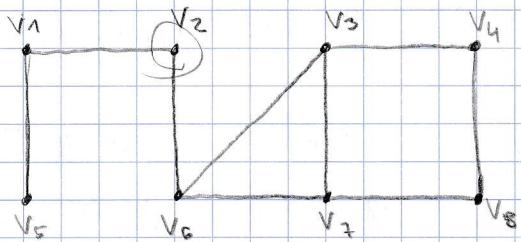
d 

$$d[v] = d_G(s, v)$$

Operacije z vrsto:

- vstavi (vrsta, u) : dodaj element u na konec vrste
- odstrani (vrsta) : odstrani prvi element iz vrste in ga vrne kot rezultat
- vstavi! (vrsta) : vstavi prazno vrsto
- prazna (vrsta) : vrne True ali False

Zagled:



Izhodisce: V_2

Vrsta: $V_2, V_1, V_6, V_5, V_3, V_7$

presekanje:

V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8
✓	✓	✓	✓	✓	✓	✓	✓

seznam
sosedov:

$V_1: V_2, V_3$

$V_2: V_1, V_6$

$V_3: V_4, V_6, V_7$

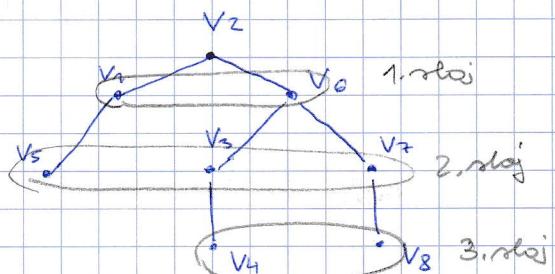
$V_4: V_3, V_8$

$V_5: V_1$

$V_6: V_2, V_3, V_7$

$V_7: V_3, V_6, V_8$

$V_8: V_4, V_7$



vozlišče	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	...
$d(V_2, V)$	1	0	2	3	2	1	2	3	
"očet"	V_2	/	V_6	V_3	V_1	V_2	V_6	V_7	

→ predstavitev
dreverja
(vsi vozlišči "kazijo"
na očeta)

najkrajša pot med V_2, V_4 : V_4, V_3, V_6, V_2

iz tabele očitamo
najkrajšo pot v
obratnem vrstnem
rednu

Pregled grafa v sirino:

VHOD: (ne)usmerjen graf $G = (V, E)$ (predstavljen s tabelo pregledano), $s \in V$

Izhod: (izpolnimo tabelo pregledano)

ustvari (vrsta)

vstavi (vrsta, s)

za vse $v \in V$ ponovi

loši čno \Downarrow

pregledano(v) = FALSE

pregledano(s) = TRUE

dokler ni (prazna(vrsta)) ponavljaj

$i = \text{odstrani}(\text{vrsta})$

za vse $j \in \text{sosedi}(i)$ ponovi

če pregledano(j) = FALSE

pregledano(j) = TRUE

vstavi(vrsta, j)

Najkrajše poti iz danega izhodisča:

VHOD: (ne)usmerjen graf $G = (V, E)$, $s \in V$

Izhod: tabela $d[i]$ dolžin najkrajsih poti od s do i , $d[i] = \infty$, če take poti n

tabela oce[i] katalcev oceta v drevesu najkrajsih poti od s do i za vse $i \in V$.

ustvari (vrsta)

vstavi (vrsta, s)

za vse $i \in V$ ponovi

$d[i] = \infty$, oce[i] = NONE

$d[s] = 0$

dokler ni (prazna(vrsta)) ponavljaj

$i = \text{odstrani}(\text{vrsta})$

za vse $j \in \text{sosedi}(i)$ ponovi

če $d[j] = \infty$, potem "rozličče se mi pregledano"

$d[j] = d[i] + 1$

oce[i] = i

vrni d, oce

④ NAJKRAJŠE POTI ZA NEUTEŽENI GRAFE

· vnos : neutezeni graf G

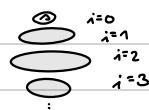
začetna točka $s \in V(G)$ [s=source]

· izhod : razdalja od s do vsakega izhodišča v obliku tabeli $d[\cdot]$: $d[v] = d_G(s, v) \forall v \in V(G)$

Drevo najkrajših poti s tabelo staršev $\pi[\cdot]$

· ideja : for $i=0$ to ...

racunamo vozilce na razdalji i



za to, da gre pot do vozilca v , ki je na razdalji i od s moramo iti skoz vozilce na razdalji $i-1$.

algoritem

$L[0]$ = seznam, ki vsebuje le s

for $v \in V(G)$ do

$d(v) = \infty$

$\pi(v) = \text{undefined} / \text{null} / \text{none}$

$d[s] = 0$

$i = 0$

 while $L[i]$ je neprazna do

$L[i+1]$ nov prazen seznam

 for $v \in L[i]$ do

 for $u \in \text{Adj}[v]$ do

 if $d[u] = \infty$ then

$d[u] = i+1$

$\pi[u] = v$

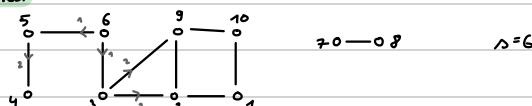
 dodamo u v seznam $L[i+1]$

skupaj
 $\sum_{\text{vervi}} \deg_G(u) = 2m$

$O(n)$

return d, π

zagled



$i=1: 0 \ 1 \ 2 \ 3 \ 4$

$d: 1$	$\infty \ \infty$	3		
2	$\infty \ \infty$	$2 \ 2$		
3	$\infty \ 1$	$1 \ 1$		
4	$\infty \ \infty$	$2 \ 2$		
5	$\infty \ 1$	$1 \ 1$		
6	$0 \ 0$	$0 \ 0$		
7	$\infty \ \infty$	$\infty \ \infty$		
8	$\infty \ \infty$	$\infty \ \infty$		
9	$\infty \ \infty$	$2 \ 2$		
10	$\infty \ \infty$	3		

$i=1: 0 \ 1 \ 2 \ 3$

π	1			2
2			3	3
3		6		6
4			5	5
5		6		6
6	6			6
7				/
8				/
9		3	3	
10				9

$L[0] = \{6\}$

za $i=1$: $L[1] = \{5, 3\}$

za $i=2$: $L[2] = \{4, 9, 2\}$

za $i=3$: $L[3] = \{10, 1\}$

za $i=4$: \emptyset

↓ je vi več nova, ostane
vr. enaka kot priva na levu

lastnosti:

- znotraj zanke spremenimo vrednost $d[u]$ največ enkrat
- vsako vozlišče u dodamo največ enkrat v nek seznam $L[i]$
- vsako povezavo uv obravnavamo največ dva krat: enkrat v " $v \in \text{Adj}[u]$ " in enkrat v " $u \in \text{Adj}[v]$ "
- Algoritem porabi $O(|V(G)| + |E(G)|) = O(n+m)$
- Ali je res, da na koncu velja: $d[v] = d_G(s, v) \quad \forall v \in V(G)$
- $L[i] = \text{seznam vozlišč na razdalji } i \text{ od } s$
- $J[i]$ je prava tabelica ocetov
- Invarianta zanke while (dokazujemo z indukcijo)
- zagled: $s=0$
- for $i=1$ to n do
- $s=s+1$
- na koncu iteracije i velja $s=1+2+\dots+i$

TRDITEV: Na začetku vsake iteracije while zanke velja:

- $\forall j \leq i : L[i] = \{v \in V(G) \mid d_G(s, v) = j\} =: w_j$
- $\forall v \in \bigcup_{j \leq i} L[j] : d[v] = d_G(s, v)$
- $\forall v \in V(G) \setminus \left(\bigcup_{j \leq i} L[j] \right) : d[v] = \infty, \quad d_G(s, v) > i$

DOKAZ: Indukcija na i

→ $i=0$ oz. pred začetkom iteracije $L[0] = \{s\} = \{v \mid d_G(s, v) = 0\} = w_0$

$$d[s] = 0$$

$$d[v] = \infty \quad \forall v \neq s$$

→ $i \mapsto i+1$ Naj bo u vozlišče za katerega naredimo " $d[u] = i+1$ " v iteraciji zanke

$$\begin{array}{c} u \in \text{Adj}[v], \quad v \in L[i], \quad d[u] = \infty \\ \downarrow \quad \downarrow \quad \downarrow \\ u, v \in E(G) \quad d_G(s, v) = i \quad d_G(s, u) > i \\ \downarrow \quad \downarrow \\ d_G(s, u) = i+1 \end{array}$$

S tem smo pokazali, da za vsako vozlišče u za katerega naredimo " $d[u] = i+1$ " v iteraciji i res velja

$$d[u] = i+1 = d_G(s, u). \text{ To n} \rightarrow \text{dovolj. To kaže } L[i+1] \subseteq w_{i+1}$$

Naj bo $u \in w_{i+1} \Rightarrow \exists$ pot dolzine $i+1$ od s do u



Naj bo v predzadnje vozlišče v tej poti. $d_G(s, v) \leq i$

Zaradi IP velja $v \in L[i]$. Torej v obravnavamo v "for $v \in L[i]$ ". Zaradi IP $d[u] = \infty \Rightarrow u$ bomo obravnavali znotraj

"if $d[u] = \infty$ " in bomo naredili $d[u] = i+1$ in dodamo u v $L[i+1] \Rightarrow u \in L[i+1]$

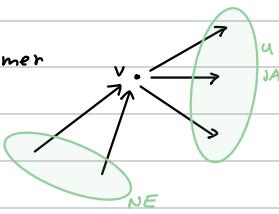
■

• IZREK: Algoritem pravilno racuna razdalje v neutrenjem grafu od danega vozlicja v casu $O(nm)$

OPOMBE

→ Algoritem dela za neusmerjene grafe in za usmerjene grafe

 povezave hodimo v pravo smer
 "for $u \in \text{Adj}[v]$ "

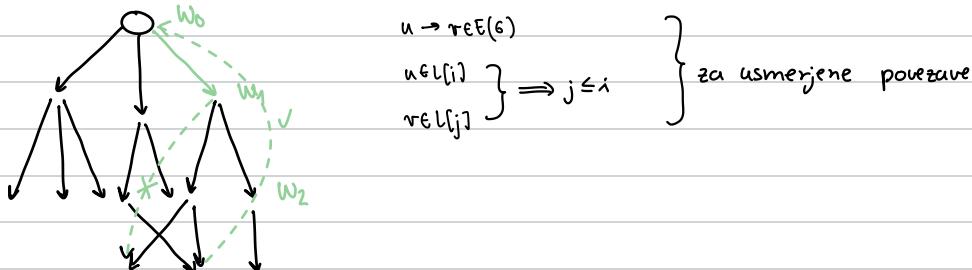
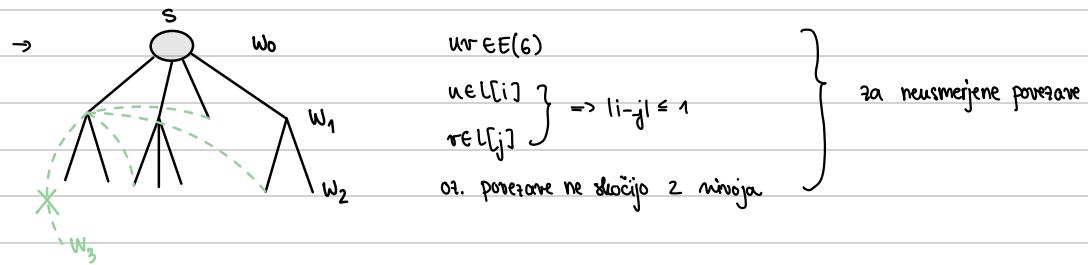


→ Algoritem se imenuje **PREGLED V ŠIRINO** (breadth-first search, BFS), če v "for $v \in L[i]$ " obravnavamo vozlicja v $L[i]$ v istem vrstnem redu, kot smo jih dodajali.

Opis je večkrat z uporabo **vrste LIFO** X FIFO - first in first out

→ Druge uporabe:

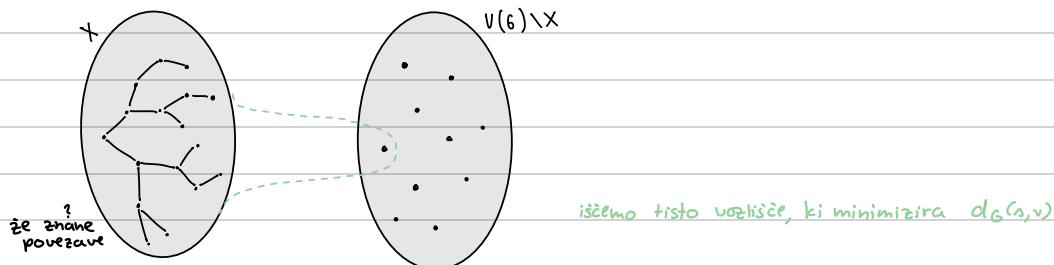
- povezanost grafa
- dvoednost grafa
- ⋮



DIJKSTROV ALGORITEM

↳ Algoritam za računanje najkrajšje poti v uteženem grafu, ki ima " > 0 " uteži. Razčlaneno vozlišča urejene po $d_G(s, v)$

↳ IDEJA:



Pogledamo poti, ki so znotraj X in potem ena povezava, ki gre od X do $V(G) \ X$

7.5

↳ Vhod: graf G = učeni $w: E(G) \rightarrow \mathbb{R}_{>0}$, se $s \in V(G)$

↳ Izvod: Drevo najkrajših poti v G od s predstavljeno s tabelo očeta π^* in razdalje $d_G(s, v) \forall v \in V(G)$ predstavljene s tabelo d

↳ Algoritam: for $u \in V(G)$ do

$d[u] = \infty$
 $\pi^*[u] = \text{nico}$ (nico izberes eno ali drugo)
 $Q = V(G) \setminus X = \emptyset$

$d[s] = 0$
while $Q \neq \emptyset$ (oz. $X \neq V(G)$) do

$u :=$ element iz Q z najmanjšo vrednostjo $d[u]$

oz. $\in \arg \min_{u \in Q} \{d[u]\}$

$Q = Q \setminus \{u\} \quad // \quad X = X \cup \{u\}$

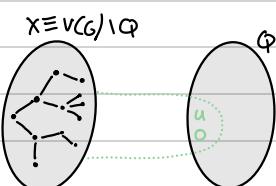
for $v \in \text{Adj}[u]$ do

if $d[v] > d[u] + w(u, v)$ then

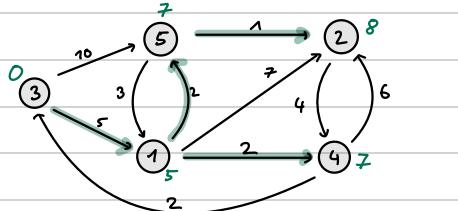
$d[v] = d[u] + w(u, v)$

$\pi^*[v] = u$

$$\text{Arg} \min_{x \in A} f(x) = \{x \in A \mid f(x) = \min_{y \in A} f(y)\}$$



↳ ZGLED



OP: zadnji korak mora biti vedno brez sprememb, sicer nismo naredili prav

smer
reverzija

	1	2	3	4	5	Q	stanje na začetku iteracije
1	∞ / nico	$\{1, 2, 3, 4, 5\}$					
2	$5/3$				$10/3$	$\{1, 2, 4, 5\}$	$u=3$
3		$5+7/1$		$5+2/1$	$5+2/1$	$\{2, 4, 5\}$	$u=1$
4				<i>ni sprememb</i>		$\{2, 5\}$	$u=4$
5				$8/5$		$\{2\}$	$u=5$
6				<i>ni sprememb</i>		$\{\}$	$u=2$

\downarrow zapisimo le vr. v graf

* bolje je da gremo $3 \rightarrow 1 \rightarrow 5$, kot iz $3 \rightarrow 5$ (črna: 10)
 $u=4=5$, vendar vzamemo 4, saj gremo po vrsti, če so vr. enake
 $u=2$
 $u=1$
 $u=0$

U=tački ki je trenutno stanje najnižje za ole

↳ Invarianta while zanke

Na koncu iteracije velja: $\forall v \in X = V(G) \setminus Q : d[v] = d_G(v, y)$

$\forall v \in Q : d[v] \text{ je dolžina najkrajše poti, ki je znotraj } X + \text{ ena dodatna povezava od } X \text{ do } v$

$$d[v] = \min_{u \in X} (d[u] + w(u, v))$$

$\forall u \in X \text{ in } V \setminus Q : d_G(u, v) \leq d_G(u, y)$

[dokazi z indukcijo]

↳ ČASOVNA ZAHTEVNOST

kako hitro delamo $u = \text{element od arg min}_{u \in Q} d[u]$?

$d[u]$ pogledamo za vsak $u \in Q$ in vzamemo min

$O(n)$ na vsaki iteraciji

$O(n^2)$ vse skupaj

Uporabimo podatkovno strukturo z imenom prioritetska vrsta.

- shranimo elemente

- vsak el. e ima nelo prioriteto $\text{key}(e)$

- operacije: \rightarrow dodano (e, key)

\rightarrow min vrste

\rightarrow izbrisemo el.

\rightarrow zmanjševanje $\text{key}(e)$ za nek element e

n elementov

$O(\log n)$

$O(\log n)$

$O(\log n)$

\downarrow tradicionalna prioritetska vrsta

v primeru neke druge prioritetske vrste
 $O(1 \log n)$

$O(\log n)$

$O(\log n)$

$O(1)$!

\Downarrow to je Fibonacci heaps /Fibonaccijeva kopica

uporabimo $\text{key}(u) = d[u]$, $O((n+m) \log n)$

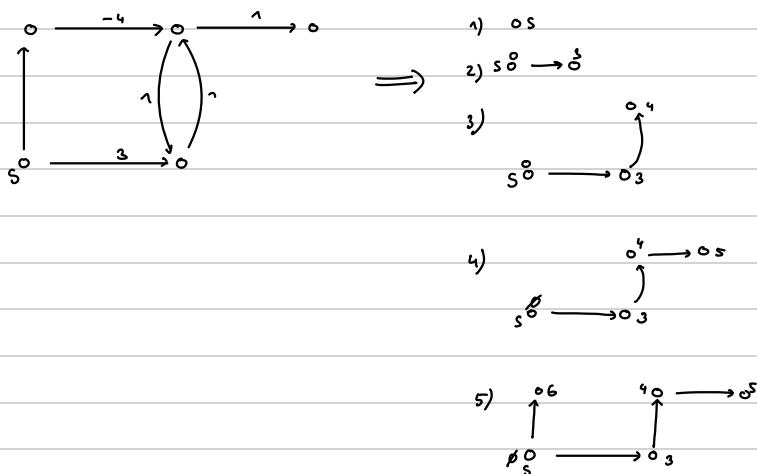
$O(n \cdot \log(n) + m)$

↳ OPOMBE

Algoritem delci, če $w : E(G) \rightarrow \mathbb{R}_{\geq 0}$

Če nas zanimala le najkrajša pot od s do t, lahko končamo, ko izberemo t iz Q

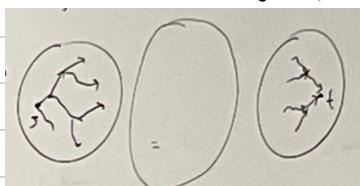
Algoritem ne delci, če so negativne uleti:



mojca opomba

V tem primeru bomo sledili pot od s navzgor šele kot zadajo povezave, saj imam največjo uleti, in v tem primeru nebomo sploh naučili povezave $s \uparrow \rightarrow \uparrow$, ker je "cenjež" krot če greš iz $s \uparrow \rightarrow \uparrow$ (taka 4) torej alg. ne deluje če imamo neg uleti

za pot od s do t v praksi je boljše, da isčemo hkrati od s in od t



6. FLOYD-WARSHELLOV ALGORITEM

↳ Računamo razdalje $d_G(u, v)$, $\forall u, v \in V(G)$. Dela ko imamo negativne uteži in nismo negativnih ciklov

Uporabili bomo dinamično programiranje

Uporabimo matrike $\Rightarrow V(G) = [n]$

$d_{ij}^{(k)} = \text{dolžina najkrajše poti od } i \text{ do } j, \text{ ki uporabi kot notranja vozlišča } \{1, \dots, k\}$

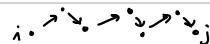
↳ TUDITEV

$$d_{ij}^{(k)} = \begin{cases} \infty & ; \text{če } i \rightarrow j \text{ ne obstaja} \\ w(i \rightarrow j) & ; \text{če } i \rightarrow j \text{ obstaja} \\ 0 & ; \text{če } i=j \end{cases} \quad \text{za } k \geq 1 : d_{ij}^{(k)} = \min \{ d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \}$$

IDEJA DOKAZA: izvedemo indukcijo na k

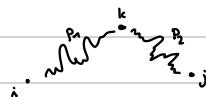
Najkrajša pot P od i do j , ki uporabi notranja vozlišča od $\{1, \dots, k\}$ lahko

• ne gre skozi k :



$$\Rightarrow \text{dolžina}(P) = d_{ij}^{(k)}$$

• gre skozi k :



$\Rightarrow P$ se razdeli na dve poti P_1 od i do k in P_2 od k do j

P_1, P_2 imata notranja vozlišča znotraj $\{1, \dots, k-1\}$

$$\text{dolžina}(P) = \text{dol}(P_1) + \text{dol}(P_2) = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$$

↳ ALGORITEM FW

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

 for $j=1, \dots, n$ do

$d_{ij}^{(0)} = w(i \rightarrow j), \text{ če } i \rightarrow j \in E(G)$

∞ ; sicer

for $k=1, \dots, n$ do

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

 for $j=1, \dots, n$ bvez $j=i$ do

$d_{ij}^{(k)} = \min \{ d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \}$

return matrika $d_{ij}^{(n)}$

v primeru da diagonale ignoriramo

↳ OPOMBE

• $O(n^3)$ - časovna zahtevnost

• počasnejši kot Dijkstrata, ampak dela, ko imamo negativne uteži in je bistveno kaj je

• prostorska zahtevnost: $O(n^3) = n$ matrik velikosti $n \times n$

Se lahko spremeni, da samo shranimo 2 matriki, ker za računanje $d^{(n)}$ le potrebujemo $d^{(n-1)}$ → prostorska zahtevnost: $O(n^2)$

• Lahko se spremeni, tako da računamo direkse najkrajše poti

9.5

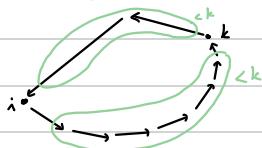
↳ Če začnemo z 0 v diagonalni, potem lahko ugotovimo, ali obstaja cikel negativnih uteži

↳ TRDITEV

Graf ima negativni cikel \iff V diagonali se pojavi negativno število

DOKAZ:

(\Rightarrow) Naj bo C negativni cikel v G . Naj bo $i \in V(C)$ najmanjši



Naj bo k največji v $V(C)$ $d_{ii}^{(k)} = \min \left\{ \underbrace{d_{ii}^{(k)}}_{\leq 0}, \underbrace{d_{ik}^{(k)} + d_{ki}^{(k)}}_{< 0} \right\} < 0$

Od takrat ostane $d_{ii}^{(k)}$ negativen, ker se nikoli ne poveča

(\Leftarrow) Podobno kot \Rightarrow

Naj bo $d_{ii}^{(k)}$ prvi k , ko je nek element v diagonali < 0

$$d_{ii}^{(k)} = \min \left\{ \underbrace{d_{ii}^{(k)}}_{= 0}, \underbrace{d_{ik}^{(k)} + d_{ki}^{(k)}}_{< 0} \right\}$$

$\Rightarrow \exists$ sprehod $i \xrightarrow{i} \dots \xrightarrow{k}$ negativne ulaz

Ta sprehod oz. obvod ima negativen cikel.

↳ POSLEDICA

V času $O(n^3)$ lahko ugotovimo, ali ima podan usmerjen graf negativni cikel.

DOKAZ: Uporabimo FW in na koncu preverimo dragonate

TOPOLOŠKO UREJANJE

[topological sorting]

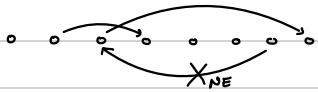
OP: Nima povezave s smerno matematike: topologijo

↪ G usmerjen graf, $|V(G)| = n$

DEFINICIJA: Preslikava $f: V(G) \rightarrow [n] = \{1, \dots, n\}$ topološko ureja graf G, če velja:

• f bijektična

• $\forall u \rightarrow v \in E(G) : f(u) < f(v)$



↪ DEFINICIJA

DAG ≡ directed acyclic graph ≡ usmerjen graf brez ciklov v smeri povezav

↪ TRDITEV

Vsek končni graf G brez usmerjenih ciklov ima vsaj eno vozlišče vhodne stopnje 0 in vsaj eno vozlišče izhodne stopnje 0.

vhodna stopnja: st. povezav iz vozlišča

izhodna stopnja: st. povezav v vozlišče

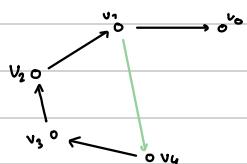


DOKAZ

• [za vhodno]: Naj bo v_0 poljubno vozlišče grafa G.

za $i=1, \dots$: → če ima v_{i-1} vhodno stopnjo 0, smo končali

→ če v_{i-1} nima vhodne stopnje 0, potem izberemo v_i tako, da $v_i \rightarrow v_{i-1} \in E(G)$



če nikoli ne končamo, ker je končno število vozlišč bo $v_i = v_j$
za i, j. Potem ima graf cikel

■

↪ IZREK

Graf G ima topološko ureditev natanko tedaj, ko G nima usmerjenih ciklov, torej G je DAG.

DOKAZ

(\Rightarrow) Počazemo (G ima usmerjen cikel) \Rightarrow (G nima topološke ureditve)

Naj bo $f: V(G) \rightarrow [n]$ poljubna bijekcija. Naj bo C cikel v G $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \dots \rightarrow v_k \rightarrow v_1$

\Rightarrow če bi bila f top. ureditev, bi bila $f(v_1) < f(v_2) < \dots < f(v_n) < f(v_1)$ \rightarrow

\Rightarrow f ni topološka ureditev

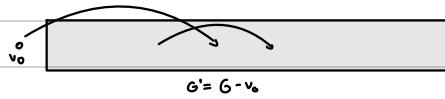
(\Leftarrow): Uporabimo indukcijo na $n = |V(G)|$

$n=1$ ✓

$n>1$: Zaradi trditve obstaja vozlišče vo vhodne stopnje 0. Graf $G' = G - v_0$ nima usmerjenih ciklov in zaradi indukcije obstaja top. ureditev $f': V(G') \rightarrow [n-1]$

Potom $f: V(G) \rightarrow [n]$ je topološka uređitev $\forall v, v':$

$$u \mapsto \begin{cases} 1 & ; \text{če } u = v_0 \\ 1 + f(v) & ; \text{če } u \neq v_0 \end{cases}$$

$$\begin{cases} \text{če } v = v_0, f(v) = 1 < f(v') \\ \text{če } v \neq v_0, f(v) = f(v) + 1 < f(v') + 1 = f(v') \end{cases}$$


■

↳ za DAGe lahko več problemov rešimo hitrejše/bolj enostavno. Skoraj vedno obraščavamo vozlišča od leve proti desni.
Kako izračunamo top. uređitev?

• TopSort(6)

→ vhod: usmerjen graf G

→ izhod: $f[]$ tabela

→ algoritam: $Q = \emptyset$ seznam/množica

```

for  $u \in V(G)$  do
    st[u] = 0
for  $u \in V(G)$  do
    for  $v \in \text{Adj}[u]$  do
        st[v] = st[u] + 1
for  $u \in V(G)$  do
    if st[u] = 0 then dodano  $u$  v  $Q$ 
    i=1
    
```

$st[u] = \text{vhodna stopnja za } u$

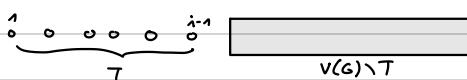
$Q = \{u \in V(G); u \text{ vhodna st. 0}\}$

```

while  $Q \neq \emptyset$  do
    u = poljubno vozlišče od Q
    f[u] = i
    i = i + 1
    for  $v \in \text{Adj}[u]$  do
        st[v] = st[u] - 1
        if st[v] = 0, then dodamo  $v$  v  $Q$ 
if i = |V(G)| + 1 then return f[]
else return "G ima cikel"
    
```

4 INVARIJANJA WHILE ZANIKE

Označimo $T = \{v \in V(G) : f[v] \text{ določen}\}$



Na začetku iteracije velja:

- $f: T \rightarrow [i-1]$ bijekcija

- $\forall u \rightarrow v, u \in T, v \in T : f(u) < f(v)$

- $Q = \{v \in V(G) \setminus T\}$ vhodna stopnja vozlišča v v grafu z vozlišči $V(G) \setminus T$ je 0

$= G[V(G) \setminus T]$ inducirani graf

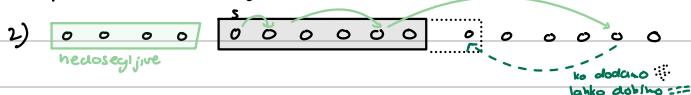
- $\forall v \in V(G) \setminus T : st[v] = \text{vhodna stopnja za } v \text{ v grafu } G[V(G) \setminus T]$

(zadnji dve točki povesta $Q = \{v \in V(G) \setminus T \mid st[v] = 0\}$)

↳ Za DAG je lahko računano drevo najkrajših poti bolj enostavno in hitrejše.

↳ Poljubne uteži:

Ideja: 1) topološko uredimo graf



Vozlišca ($v \in V(G)$) ($\pi(v) > \pi(u)$) obraunavamo v vrstnem redu, ki ga določi π

Shranimo $d[v] =$ dolžina najkrajše poti od s do v , ki uporabi kot notranja vozlišca te vozlišča, ki smo že obraunavali

16.5

↳ Časovna zahtevnost algoritma za top. ureditev je $O(nm)$, ker po vsaki povezavi hodimo $O(1)$

↳ IZREK: V času $O(nm)$ lahko ugotovimo, ali je podan usmerjen graf DAG. Če je DAG lahko v istem času računamo topološko ureditev.

↳ SP_dag (shortest path)

• vhod: usjen graf G , ki nima ciklov $\Delta V(G)$, $w: E(G) \rightarrow \mathbb{R}$ uteži

• Uporabili bomo tabele: • $\pi[\cdot]$... top. ureditev

• $\pi^{-1}[\cdot]$... inverz π

• $d[\cdot]$... razdalje

• $\pi[\cdot]$... tabela ocetov

• algoritem:

$\pi[\cdot] =$ topološka ureditev za G

(če $\not\exists$ top. ureditev, return "napaka")

for $u \in V(G)$ do

$\pi^{-1}[\pi[u]] = u$

$d[u] = \infty$

$\pi[u] = \text{nic}$

$d[n] = 0$

 do $n-1$ je tudi ok

 for $i = \pi[n]$ to $\pi^{-1}[V(G)]$ do

$u = \pi^{-1}[i]$

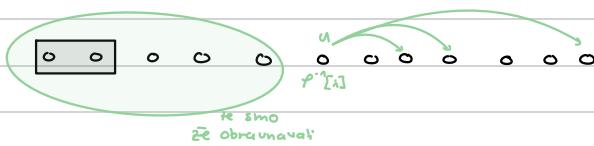
 for $v \in \text{Adj}[u]$ do

 if $d[v] > d[u] + w(u \rightarrow v)$ then

$d[v] = d[u] + w(u \rightarrow v)$

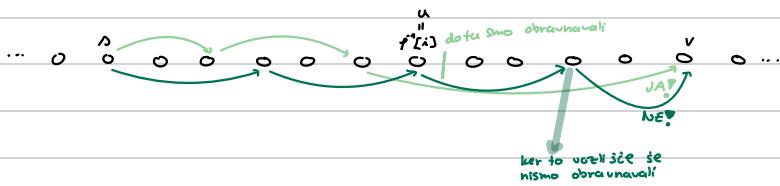
$\pi[v] = u$

return $d[\cdot]$ in $\pi[\cdot]$

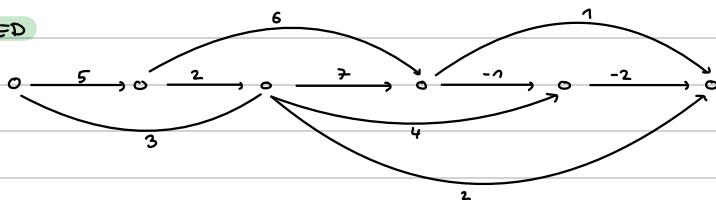


↳ Invarianta for zanke

Na koncu iteracije velja: $d[u] = \text{dolžina najkrajše poti v } G \text{ od } s \text{ do } u$, ki uporabijo kot notranja vozlišča elementi od $\{u \in V(G) \mid f[u] \leq i\}$



↳ ZGLED



$d[i]$	1	2	3	4	5	6	
na začetku iteracije i	0/nic	0/nic	0/nic	0/nic	0/nic	0/nic	
1	0/nic	5/1	3/1				$u=1=s$
2				10/2			$u=2$
3					7/3		$u=3$
4						5/3	$u=4$
5							$u=5$
6						4/5	
7	0/nic	5/1	3/1	10/3	7/3	4/5	

Časovna zahtevnost: $O(n+m)$

↳ IZREK: Drevo najkrajših poti v utreženem usmerjenem grafu brez ciklov lahko računamo v času $O(n+m) = O(|V(G)| + |E(G)|)$

NAČTOVANJE PROJEKTOV

↳ CPM - central path method

↳ Mnogičica opravil, ki sestavljajo projekt, za vsako opravilo vemo:

- trajanje

- pogoji: katera druga opravila morajo biti končana, da lahko začnemo to opravilo

↳ UPRAŠANJA

- Ali lahko opravimo vsa dela oz. celoten projekt

- kako hitro lahko končamo

- kdaj lahko najkasneje začnemo posamezno opravilo, da ne podaljšamo trajanje celotnega projekta?

↳ Predstavitev s grafoom G

- vozlišče grafa \equiv opravila + začetek s
+ konec z

- povezave: $i \rightarrow j \in E(G)$, če je opravilo i pogoj za opravilo j

- $s \rightarrow i$, če opravilo i ni nima pogojev

- $i \rightarrow z$; če opravilo i ni pogoj za nobeno opravilo

- vteži na povezavah $w(i \rightarrow j) \equiv$ trajanje opravila i

- $w(i \rightarrow z) = -\infty$

- $w(s \rightarrow i) = 0$



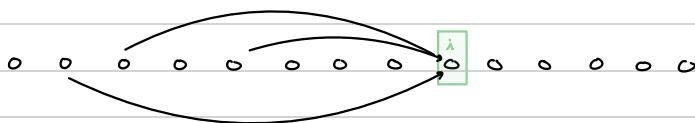
↳ Graf G topološko uredimo in ugotovimo ali ima kakšen cikel. Če ima cikel, potem je nemogoče.

↳ $t_i \equiv$ prvi čas, ob katerem lahko začnemo opravilo i

začnemo s $t_s=0$ (začetek s)

Računamo lahko t_i , ko imamo t_j za vse pogoj j za i

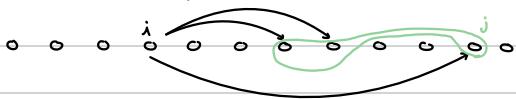
↳ Uporabimo topološko ureditev in $t_i = \max \{t_j + w(j \rightarrow i) \mid j \rightarrow i \in E(G)\}$



↳ $t_z =$ najmanjši čas za opravljanje projekta

↳ $T_1 =$ kdaj lahko začnijš začnemo opravilo i tako, da ne upliva na končni čas

T_1 računamo v obratnem uravnem redu $T_2 = t_z$



Za $i \neq z$: $T_i = \min \{T_j - w(j \rightarrow i) \mid j \rightarrow i \in E(G)\}$

↳ t_i : $m_i = T_i - t_i$ razlika/margin

- $m_i \geq 0$

- če $m_i = 0$, potem je i kritično opravilo. Kritična opravila definirajo vsaj eno pot od začetka s do konca z . [Critical-path method]

- zakaj so kritična opravila relevantna?

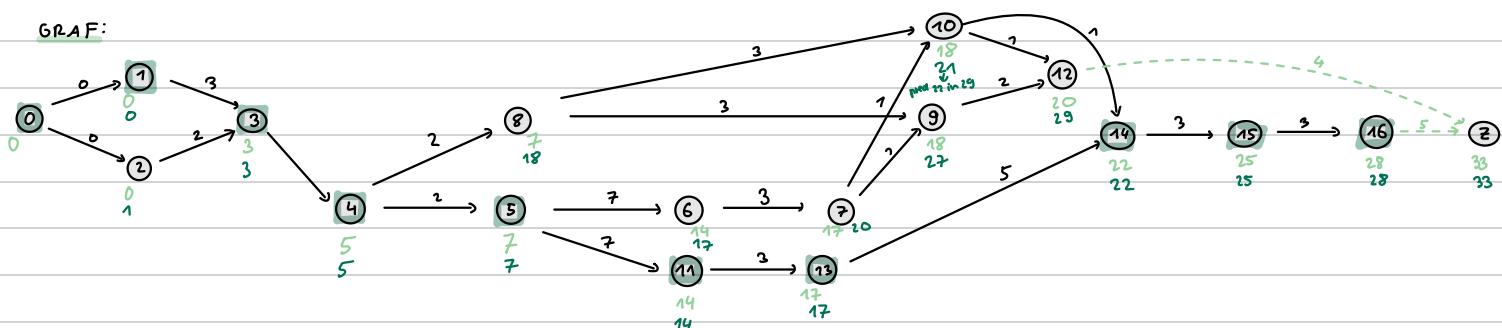
Kritična opravila so najbolj relevantna, da nebo zamud v projektu. Morebitna dodatna energija naj gre v kritična opravila

ZGLED - GRADNJA HIŠE

CAS; POGOJI

- 0 ... začetek \Leftarrow dodamo
- 1 ... priprava zemljišča ; 3 ; /
- 2 ... dostava gradbenega materiala ; 2 ; /
- 3 ... izkopljemo zatemelje ; 2 ; 1,2
- 4 ... gradnja temeljev ; 2 ; 3
- 5 ... gradnja nosilnih zidov ; 7 ; 4
- 6 ... gradnja ostresca ; 3 ; 5
- 7 ... streha ; 1 ; 6
- 8 ... zunanje instalacije ; 3 ; 4
- 9 ... zunajni omet ; 2 ; 7,8
- 10 ... okna ; 1 ; 7,8
- 11 ... predelane stene ; 3 ; 5
- 12 ... okolica ; 4 ; 9,10
- 13 ... notranje instalacije ; 5 ; 11
- 14 ... izolacija ; 3 ; 10,13
- 15 ... barvanje sten ; 3 ; 14
- 16 ... uselitev ; 5 ; 15
- 17 ... koniec = 2

GRAF:



30.5

$$t_i = \max \{ t_k + w(k \rightarrow i) \mid k \rightarrow i \in E(G) \}$$

$$T_i = \max \{ t_k - w(i \rightarrow k) \mid i \rightarrow k \in E(G) \}$$

$$m_i = T_i - t_i \geq 0$$

$m_i = 0 \Leftrightarrow$ opravilo i je kritično

	s=0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	z
t	0	0	0	3	5	7	14	17	7	18	18	14	20	17	22	25	28	33
T		0	1	3	5	7	17	20	18	27	21	14	29	17	22	25	28	33
m	0	0	1	0	0	0	3	3	9	9	3	0	9	0	0	0	0	0

od s \rightarrow z \rightarrow kritično

od z \rightarrow s \rightarrow do kraj mogoče bili dokončani, da bo z res v 33

- označka v grafu za kritična vozilista

kritična opravila morajo definirati vsaj eno pot od s do z

$t_z = T_z$ = dolžina najdaljše poti od s do z v grafu.

KOMENTAR: Opravila v kritični poti so tista, kjer ne moremo zamuditi. Če imamo dodaten dnevnik, ki bi pospešil neko opravilo, potem so opravila v kritični poti dobri kandidati. Ni nujno da je kritična pot samo ena.

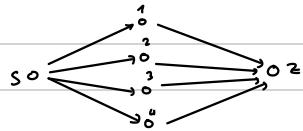
↪ Bolj splošno: usako opravilo i porabi x_i časa, ki je slučajna spremenljivka

↪ V praksi upraviščjo: → pričakovan čas opravila mi:

→ čas opravila v najslabšem primeru bi:

→ čas opravila v najboljšem primeru ar:

↪ ZGLED



$$E[\text{trajanje projekta}] = E[\max \{x_1, x_2, x_3, x_4\}]$$

H VV

$$\max \{E[x_1], \dots, E[x_4]\}$$

$$v \text{ zaporednih projektih je } E[\text{trajanje}] = \sum_i E[x_i]$$

NAJCENEJŠE VPETO DREVO

↳ Vhod: usmerjen graf G [povezan]

$$w: E(G) \rightarrow \mathbb{R}_{\geq 0}$$

↳ Naloga: min $\sum_{e \in E(T)} w(e)$

p.p. T je vpeto drevo za G

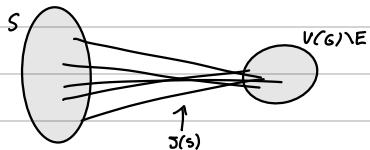
[drevo je povezan graf brez ciklov]

[vpeto graf \rightarrow usebuje vsa vozlišča, torej $V(T) = V(G)$]

↳ $S \subseteq V(G)$, $S \neq V(G)$, $S \neq \emptyset$

$J(S) = \text{množica, ki ga določi } S$

$$= \{uv \in E(G) | u \in S, v \in V(G) \setminus S\}$$



↳ Izrek

G utežen neusmerjen graf

$A \subseteq E(G)$ množica povezav, ki je usebovana v nekem najcenejšem vpetu drevesu

$$S \subseteq V(G), S \neq \emptyset \text{ in } S \neq V(G)$$

$$J(S) \cap A = \emptyset$$

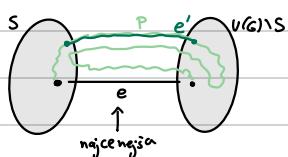
Naj bo $e \in J(S)$ najlažja povezava od $J(S)$. Polem je $A \cup \{e\}$ usebovana v nekem najcenejšem vpetu drevesu

DOKAZ:

Vemo, da je $A \subseteq E(T)$, kjer je T neko najcenejše vpeto drevo. Dre možnosti:

• $e \in E(T)$. Potem $A \cup \{e\} \subseteq E(T) \cup$

• $e \notin E(T)$



Ngi bo P pot znotraj T , ki povezuje krajišča povezave e .

$E(P) \cap J(S) \neq \emptyset$, ker P povezuje različne iz S z vozliščem iz $V(G) \setminus S \Rightarrow \exists e' \in E(P) \cap J(S)$

$$\text{Opozajemo } T' = (T - e') + e$$

• T' je vpeto drevo, saj $T' = (T - e) - e'$

$$\cdot w(T) = w(T) + \underbrace{w(e) - w(e')}_{\leq 0} \leq w(T')$$

Ker je T najcenejše drevo, je tudi T' najcenejše vpeto drevo

• $e' \notin A$, ker $e \in J(S)$ in $J(S) \cap A = \emptyset \Rightarrow A \cup \{e\} \subseteq E(T')$



KRUSKALOV ALGORITEM

↪ $e_1, \dots, e_m \leftarrow$ to so povezave urejene nepadajoče, torej velja:

$$w(e_1) \leq w(e_2) \leq \dots \leq w(e_m)$$

↪ Najprej imamo samo: $T = (V(G), \emptyset) \rightarrow$ graf brez povezav

for $i=1$ to m do

 if $T + e_i$ nima ciklov then
 $T = T + e_i$

return T

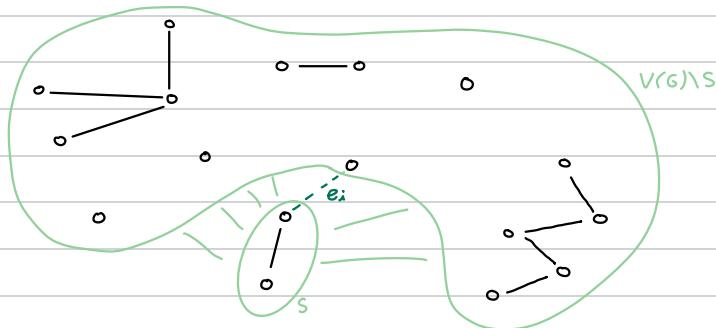
$\mathcal{O}(m)$

1. močnost:
preverimo, ali je T pot med krajnjicema
 $\mathcal{O}(n)$ iteracij, skupaj $\mathcal{O}(nm)$

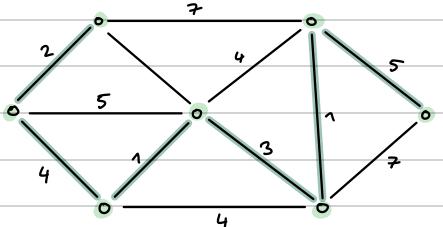
2. močnost:
"union and find" $\rightarrow \mathcal{O}(n \cdot \alpha(n))$ in boljše $\mathcal{O}(m \log(m))$

V pravilnosti za S učimoš povezano komponento v T , ki vsebuje eno krajnjico e_i .

e_i je najcenejsa povezava v $S(G)$.



↪ ZGLED



↪ časovna zahtevnost algoritma: $\mathcal{O}(m \log m)$