STU FIIT

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Dátové štruktúry a algoritmy

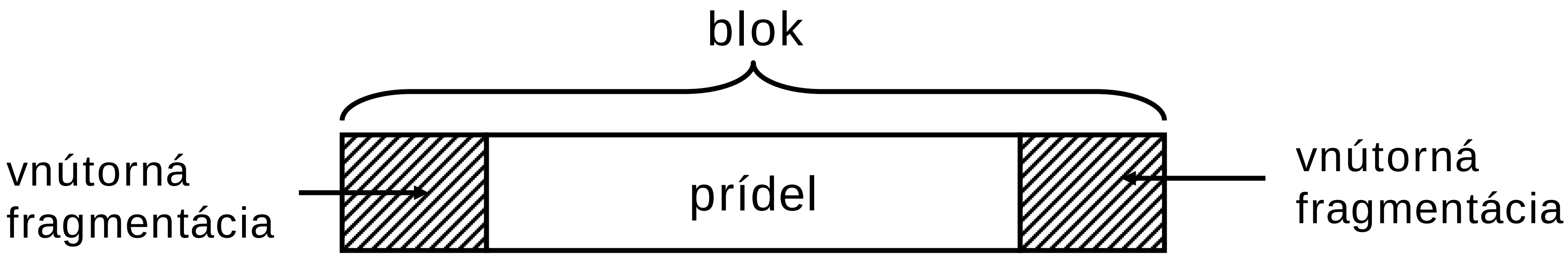
Priebežný test

14. 11. 2017

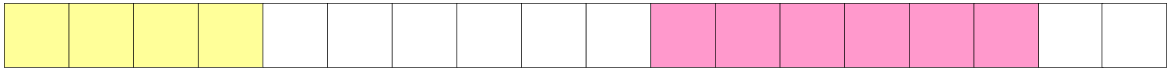
zimný semester 2017/2018

Úloha A – Fragmentácia

- Čo je to **vnútorná** fragmentácia?
  - je rozdiel medzi veľkosťou bloku a veľkosťou prídelu



- Čo je to **vonkajšia** fragmentácia?
  - nastáva, keď je síce dosť voľnej pamäti spolu (agregátne), ale žiadny voľný blok nie je dostatočne veľký



```
p4 = malloc(7*sizeof(int))
           Hopla!
```

Úloha B – Špecifikácia operácií

- Uveďte špecifikáciu operácií (nemusíte slovne opisovať) pre abstraktnú dátovú štruktúru **binárny strom**.
  - **Nie: binárny vyhľadávací strom.**

CREATE() → bintree

MAKE(item,bintree,item) → bintree

LCHILD(bintree) → bintree

DATA(bintree) → item

RCHILD(bintree) → bintree

ISEMPTY(bintree) → boolean

Úloha B – Špecifikácia operácií

- Uveďte špecifikáciu operácií (nemusíte slovne opisovať) pre abstraktnú dátovú štruktúru **zásobník**.
  - **Nie: zásobník (resp. heap) pri prideľovaní pamäti.**

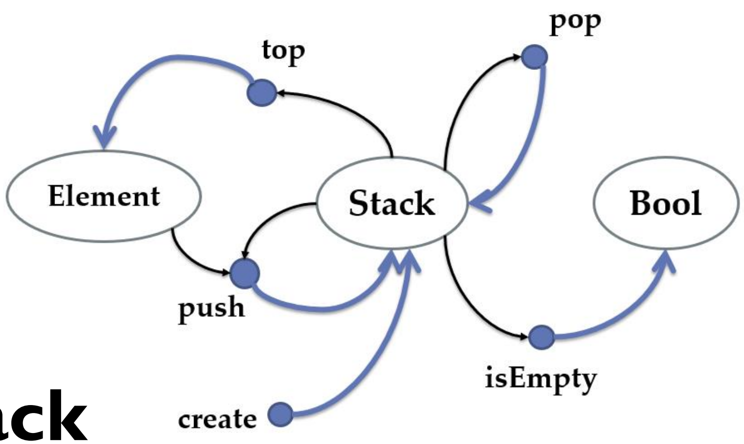
CREATE() → stack

POP(stack) → stack

TOP (stack) → element

PUSH(stack,element) → stack

ISEMPTY(stack) → boolean



Úloha C

- Dané je číslo X a postupnosť N celých čísel. Navrhnite algoritmus, ktorý rozhodne v lineárnom čase O(N), či súčet niektorých dvoch čísel v postupnosti je rovný X. Stačí slovný opis alebo pseudokód. Napr. X=60, N=5 a čísla 50 20 30 10 40, výsledok ÁNO (napr.20+40)
- **Nie: usporiadať čísla, binárny vyhľadávací strom**
  - **(radix nie je O(N) ak nie je vopred daný rozsah).**
- **Riešenie: hashovanie**
  - **Prechádzam každé číslo a[i] práve raz:**
  - **Ak sa v tabuľke nachádza x-a[i], tak výsledok je ÁNO.**
  - **Inak, výsledok je NIE**

Úloha C

- Daná je binárna min-halda obsahujuca N prvkov zapísaná vo vektore A[1..N]. Navrhnite algoritmus, ktorý ju transformuje na binárnu max-haldu s rovnakými prvkami. Stačí slovný opis alebo pseudokód.
- **Nie: heuristiky, ktoré stransformujú min-haldu na max-haldu (min-halda nemá z pohľadu max-haldy žiadne zaujímavé vlastnosti)**
- **Konštrukcia haldy (metódou heapify) je O(N) pre N neusporiadaných prvkov, lepšie sa to nedá...**

Úloha D

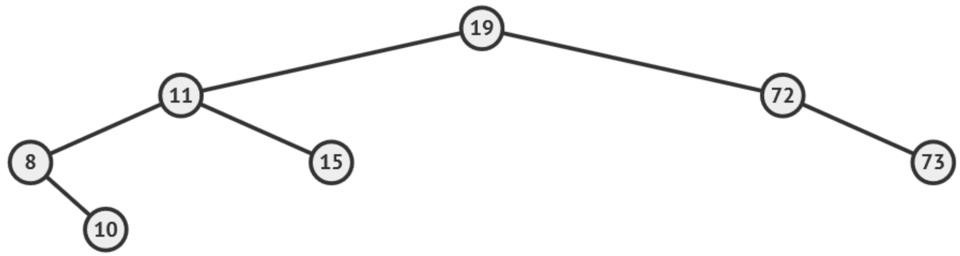
- Ktoré z polí reprezentuje binárnu min-haldu? Haldu nakreslite v tvare stromu.

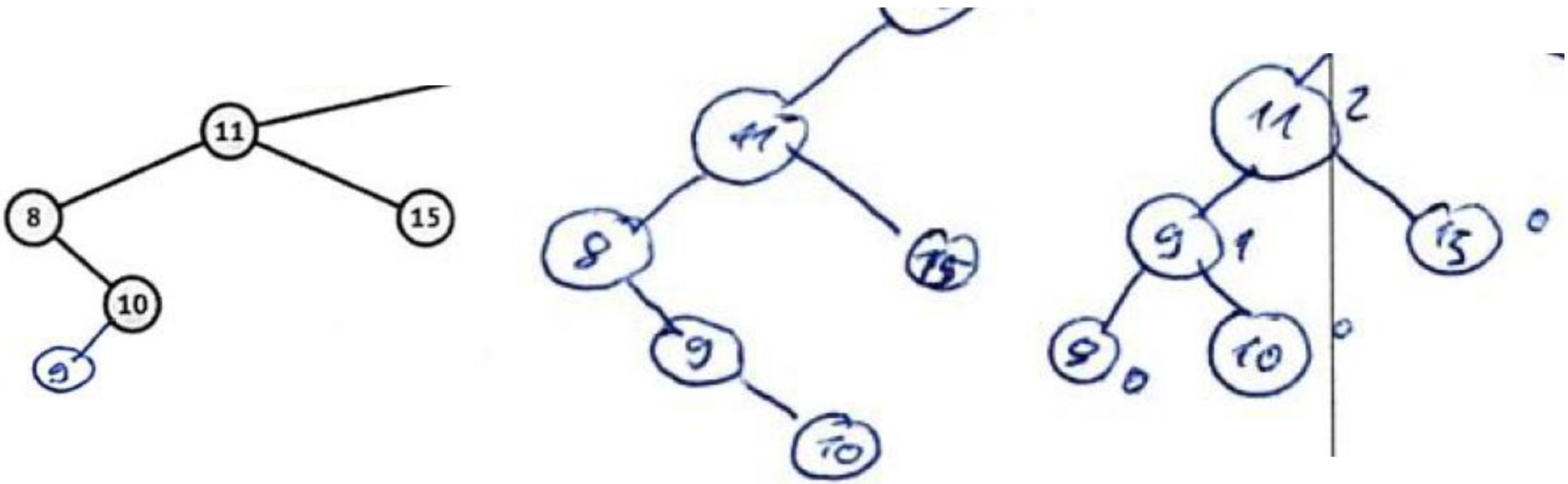
	1	2	3	4	5	6	7	8	9	10	11	12
A =	1	4	54	8	45	76	65	44	11	47	57	
B =	1	8	45	13	43	47	44	65	23	76	57	
C =	1	13	47	23	65	54	67	45	32	76	57	

	1	2	3	4	5	6	7	8	9	10	11
A =	4	6	15	7	18	22	35	30	9	17	
B =	4	6	18	15	7	22	35	30	9	17	
C =	4	6	18	7	15	22	35	30	9	17	

Úloha E

- Vložte prvok 9 do nasledovného AVL stromu, nakreslite strom po vložení a každej rotácií.





Úloha F

- Napíšte pseudokód operácie predchodca v binárnom vyhľadávacom strome.
- Dva prípady:
  - má ľavého nasledovníka: maximálny prvok v ľavom podstrome
  - Nemá: postupujem smerom hore (do koreňa), ak prídem do rodiča sprava, tak ten rodič je predchodca.
- Napíšte pseudokód operácie odstránenia z binárneho vyhľadávacieho stromu v prípade, že vrchol na odstránenie má dva podstromy.
  - Odstraňujeme vrchol v, nájdem predchodcu/nasledovníka (označme x), nahradím vrchol v hodnotou vo vrchole x, a vrchol x triviálne odstránim.

Úloha G

- V hashovacej tabuľke máte N=3200 prvkov. Určte faktor naplnenia **α** ak chcete dosiahnuť nízky (do 3) očakávaný počet pokusov pri **vyhľadaní** / **vkladaní** v prípade riešenia kolízií: a) lineárnym skúšaním, b) dvojitým skúšaním, c) reťazením.

Očakávaný počet pokusov		Faktor naplnenie α			
		50%	66%	75%	90%
Lineárne skúšanie	search	1.5	2.0	3.0	5.5
	insert	2.5	5.0	8.5	**55.5**
Dvojité rozptýlenie	search	1.4	1.6	1.8	2.6
	insert	1.5	2.0	3.0	**5.5**

Opravný – náhradný test

- Na budúcej prednáške:
- Test bude 21.11. od 11:55 do 12:50
- Témy rovnaké, úlohy/otázky iné ...
- **Opravný test: úlohy sú nastavené na 15 bodov, ale bodový zisk je zhora ohraničený 5 bodov.**
- Náhradný test: úlohy sú nastavené na 15 bodov, môžete získať 15 bodov.

- Prednáška bude od 11:00 – 11:50

STU FIIT
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Dátové štruktúry a algoritmy

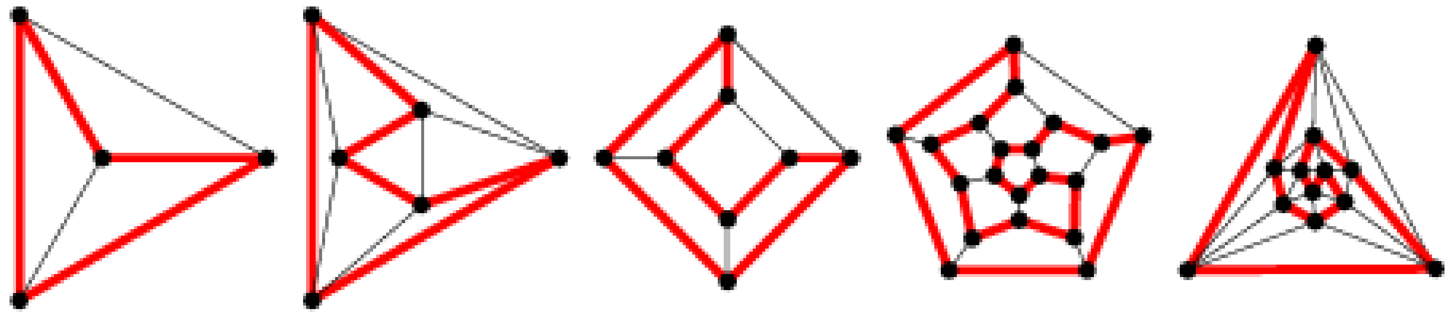
Grafové algoritmy (Hamiltonovské grafy, NP)

14. 11. 2017

zimný semester 2017/2018

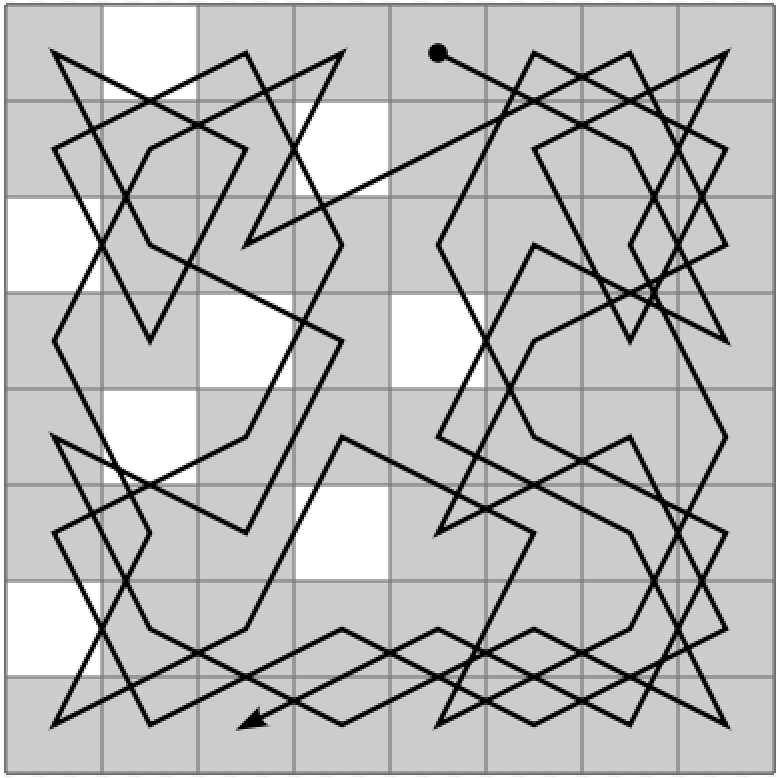
Teória grafov – Hamiltonovský sled, cesta, cyklus

- **Hamiltonovský sled** v grafe G je taký sled, ktorý obsahuje všetky vrcholy grafu G.
- **Hamiltonovská cesta** je taký hamiltonovský sled, ktorý neobsahuje rovnaké hrany
- **Hamiltonovský cyklus** je taký hamiltonovský sled, v ktorom sa okrem prvého a posledného vrcholu žiaden vrchol nevyskytuje viac než raz



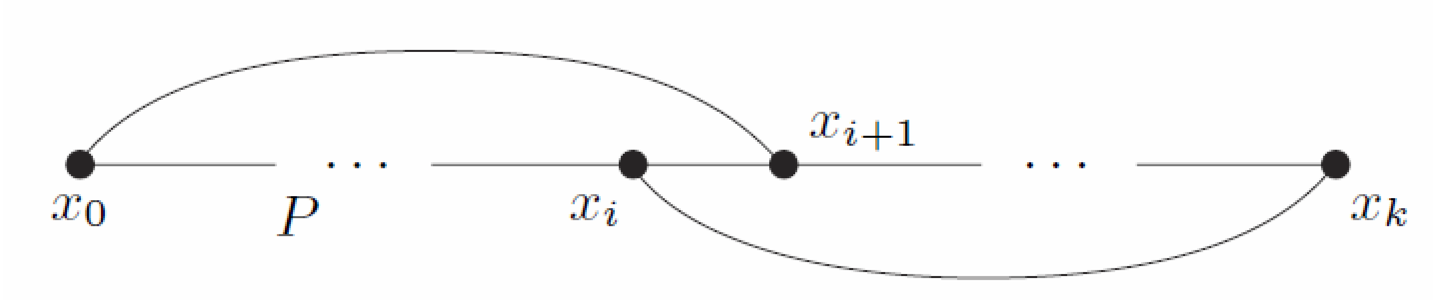
Hamiltonovská cesta – Ukážka

- Prechod koňom po šachovnici



Teória grafov – Hamiltonovský graf

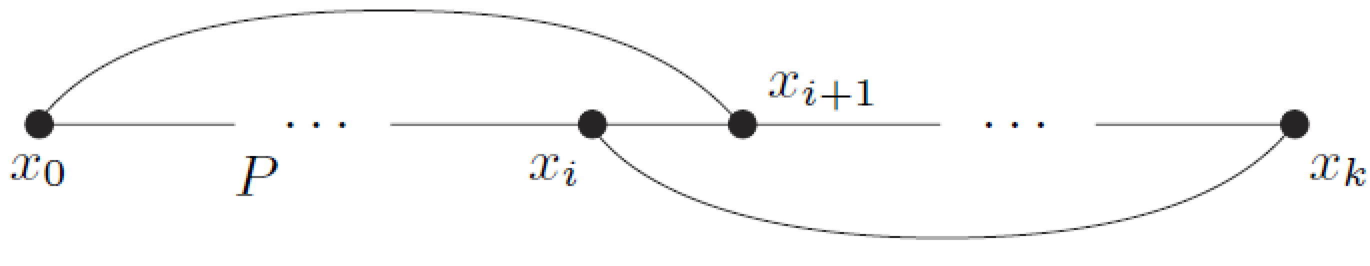
- Graf je hamiltonovský, ak obsahuje hamiltonovský cyklus.
- Dirac 1952:
  (Každý) graf s $N \geq 3$ vrcholmi so stupňom (každého vrcholu) aspoň $N/2$ obsahuje hamiltonovský cyklus.
- Zjavne tento graf je súvislý. (inak v najmenšom súvislom komponente C, by stupne vrcholov boli $|C| \leq N/2$.
- Uvažujme najdlhšiu cestu $P = x_0, \ldots, x_k$:



- Všetci susedia x_0 a x_k musia byť na tejto ceste.

Teória grafov – Hamiltonovský graf (2)

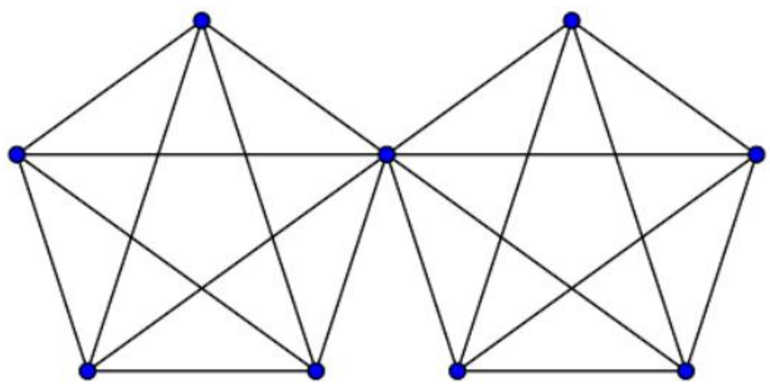
- Uvažujme najdlhšiu cestu P = $x_0, \ldots, x_k$:



- Všetci susedia x_0 a x_k musia byť na tejto ceste.
- Teda, aspoň N/2 vrcholov z $x_0, \ldots, x_{k-1}$ je susedných s x_k a súčasne aspoň N/2 vrcholov z týchto k < N vrcholov x_i je takých, že $(x_0, x_{i+1}) \in E$
- Z Dirichletovho princípu vyplýva, že existuje aspoň jeden vrchol x_i s oboma vlastnosťami: $(x_0, x_{i+1}) \in E$, $(x_i, x_k) \in E$
- Potom sled $x_0 x_{i+1} P x_k x_i P x_0$ je hamiltonovský cyklus.

Teória grafov – Hamiltonovský graf (3)

- (Každý) graf s N $\geq$ 3 vrcholmi so stupňom (každého vrcholu) aspoň N/2 obsahuje hamiltonovský cyklus.
- Tento odhad je „tesný“.
- Pre N nepárne, a stupeň $\lfloor N/2 \rfloor$ graf nemusí byť hamiltonovský (napr. N=9):

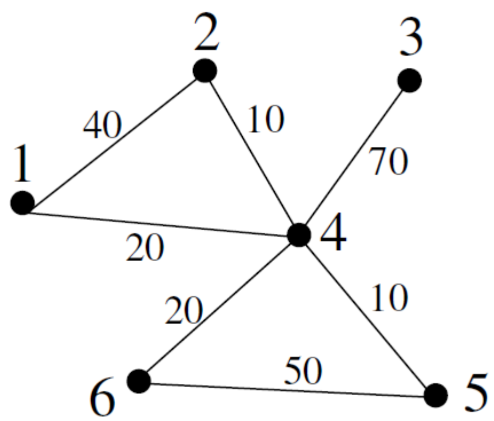


- Ak pre (všetky) nesusedné vrcholy u a v platí: deg(u) + deg(v) ≥ N, tak graf je hamiltonovský.

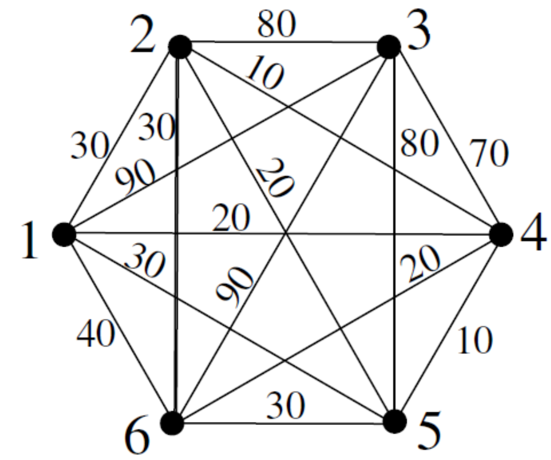
Úloha obchodného cestujúceho

- Travelling Salesman Problem (TSP):
  Obchodný cestujúci má navštíviť všetkých svojich zákazníkov a vrátiť
- Teória grafov: V súvislom ohodnotenom grafe nájsť najkratší hamiltonovský cyklus.
- Ak pripustíme navštíviť to isté miesto viac krát:
  V súvislom ohodnotenom grafe nájsť najkratší hamiltonovský sled.
- V praxi nie je dôvod zakazovať prechod cez jeden vrchol viac krát.

Úloha obchodného cestujúceho – Ukážka



Hamiltonovský cyklus neexistuje



Doplníme do úplneho grafu hranami ohodnotenými vzdialenosťou

- V úplnom grafe G’ každá permutácia vrcholov zodpovedá cyklu v G a teda aj hamiltonovskému sledu v pôvodnom grafe
- Koľko je rôznych hamiltonovských cyklov v G?
  - (N-1)! … (začneme v niektorom vrchole, a uvažujeme všetky usporiadania ostatných)
- Ako ľudstvo zatiaľ nepoznáme podstatne lepší algoritmus ako systematické prehľadanie všetkých (N-1)! permutácií...

Trieda zložitosti P

- Množina problémov, ktoré je možné vyriešiť v najhoršom prípade v polynomiálnom čase
  - Všetky problémy, pre ktoré máme algoritmus bežiaci v čase $O(N^k)$ pre nejaké konštantné k.
- Príklady problémov v P:
  - Prehľadávanie stromu
  - Usporadúvanie
  - Najkratšia cesta v grafe
  - Eulerov ťah
  - ...

Trieda zložitosti NP

- Množina problémov, ktoré je možné vyriešiť v polynomiálnom čase na nedeterministickom Turingovom stroji.
- Iná intuitívnejšia predstava:
  Množina problémov, pre ktoré je možné overiť predložené (možné) riešenie v polynomiálnom čase
  (na deterministickom stroji)
- Príklad problému v NP:
  - Nájsť Hamiltonovský cyklus.
  - **Prečo je to v NP?**
  - Pre dané riešenie (postupnosť vrcholov) vieme v polynomiálnom (lineárnom) čase overiť, či je to naozaj hamiltonovský cyklus – skontrolujeme, či sa tam vrcholy nachádzajú práve raz okrem začiatku a konca.

Trieda zložitosti NP (2)

- Postup výpočtu na nedeterministickom Turingovom stroji:
  - úhadneme riešenie (nedeterministicky),
  - skontrolujeme riešenie (deterministicky)
  - Princíp je ten: že, ak existuje také uhádnutie, ktoré umožní pri kontrole prísť do akceptačného stavu, tak sa zrealizuje ... (ak by sme spravili kontrolu zle, tak sa akceptuje zlé riešenie)
- Iný príklad problému v NP:
  - Usporiadanie čísel.
  - **Prečo je to v NP?**
  - Pre dané riešenie (postupnosť prvkov) vieme v polynomiálnom čase overiť, či je to permutácia vstupu a či je usporiadaná...
  - **Ešte poznáte nejaké iné úlohy v NP?**

Trieda zložitosti NP (3)

- Všetky problémy v P sú aj v NP
  - Nedetermininistický Turingov stroj predsa môže pracovať aj čisto deterministicky...
  - Teda: P ⊆ NP
    (Ak dokážem úlohu VYRIEŠIT v polynomiálnom čase, tak určite dokážem aj skontrolovať riešenie v polynomiálnom čase)
- **Million dollar question potom je:**
  - **Či NP ⊆ P ?**
  - **Nikomu sa to nepodarilo dokázať, môžeš byť prvá, možeš byť prvý!!**

NP-úplne (NP-complete) problémy

- Zaujímavé problémy, ktoré sú v NP a sú istým spôsobom najťažšie:
- Problém X je NP-úplný, ak X ∈ NP a zároveň problém Y ∈ NP dokážeme v polynomiálnom čase previesť (transformovať) na problém X.
- Teda, ak by sa vám podarilo nájsť polynomiálne riešenie pre ľubovoľný z NP-úplných problémov, tak VŠETKY problémy v NP by boli riešiteľné v polynomiálnom čase.
  - Napr. Úloha nájdenia hamiltonovského cyklu je NP-úplná.

Základný NP-úplný problém – SAT

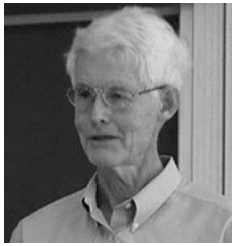
- Problém splniteľnosti (Boolean satisfiability problem)
  - Zistiť, či existuje interpretácia, ktorá splňuje boolovskú formulu (teda či existuje také dosadenie hodnôt TRUE/FALSE za premenné tak, aby bola výsledná pravdivostná hodnota TRUE)

$$(a \vee b) \wedge (\neg a \vee c) \wedge (\neg b \vee c)$$

Prečo je SAT v NP?

Prečo je problém splniteľnosti (SAT) v NP?

- Stephen Cook 1971 ukázal, že SAT môžeme použiť na simulovanie akéhokoľvek nedeterministického Turingovho stroja.

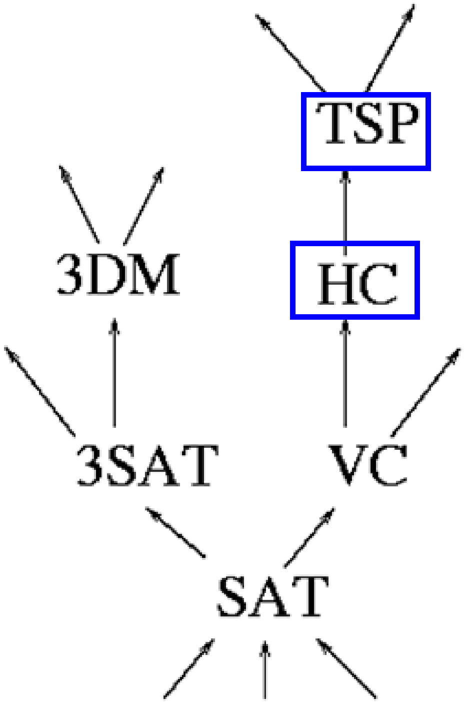


- Pre (každý) vstup X zostrojíme boolovskú formulu, ktorá je splniteľná práve vtedy, keď stroj akceptuje vstup X
- Formula overuje prechody, akceptačný stav v závislosti od stavu na páske a aktuálneho stavu
- „Uhádnuté“ premenné určujú, ktorú vetvu zvolíme
- Súčasne v ZSSR objavil Leonid Levin, 1969 ...

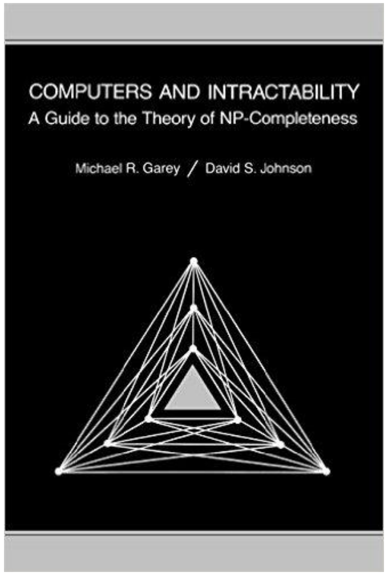
- Cook-Levinova veta

NP-úplné problémy

- Stovky problémov sa ukázali ako NP-úplné
- Ako?
  - Algoritmom, ktorý prevedie niektorý zo známych NP-úplných problém na nový problém

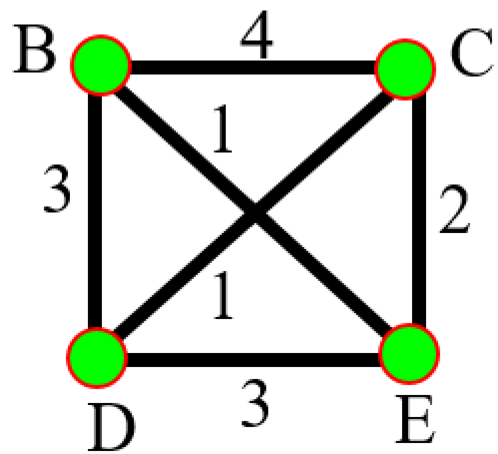


- *Computers and Intractability: A Guide to the Theory of NP-Completeness*, by Michael S. Garey and David S. Johnson



Ukážeme, že TSP je NP-úplný

- Daný je úplný súvislý ohodnotený graf, existuje cyklus, ktorý navštívi každý vrchol práve raz a celková dĺžka cyklus bude $\leq$ K ?
- Napr. Existuje cyklus s dĺžkou $\leq$ 8 ?
  Áno: BDCEB (7)

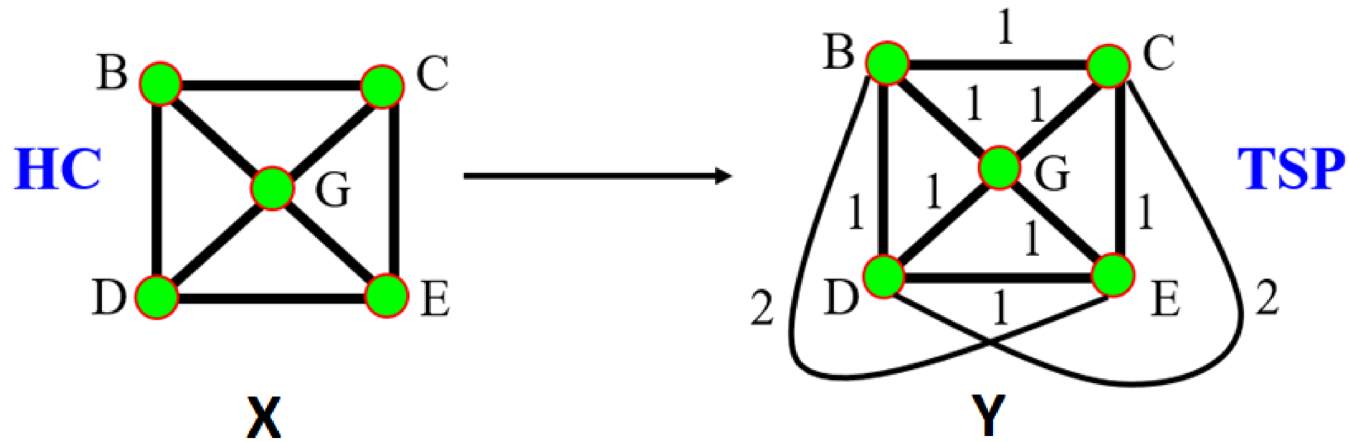


- Prečo je TSP v NP?

Ukážeme, že TSP je NP-úplný (2)

- Vychádzajúc zo skutočnosti, že úloha zistiť, či v grafe existuje hamiltonovský cyklus (HC) je v NP.
- Ukážeme že, akúkoľvek inštanciu problému HC vieme previesť na inštanciu problému TSP
- a potom sporom:
  ak by sme vedeli vyriešiť TSP deterministicky v polynomiálnom čase, tak by sme vedeli v polynomiálnom čase riešiť aj akúkoľvek inštanciu HC, čo však zatiaľ nevieme

Ukážeme, že TSP je NP-úplný (3)



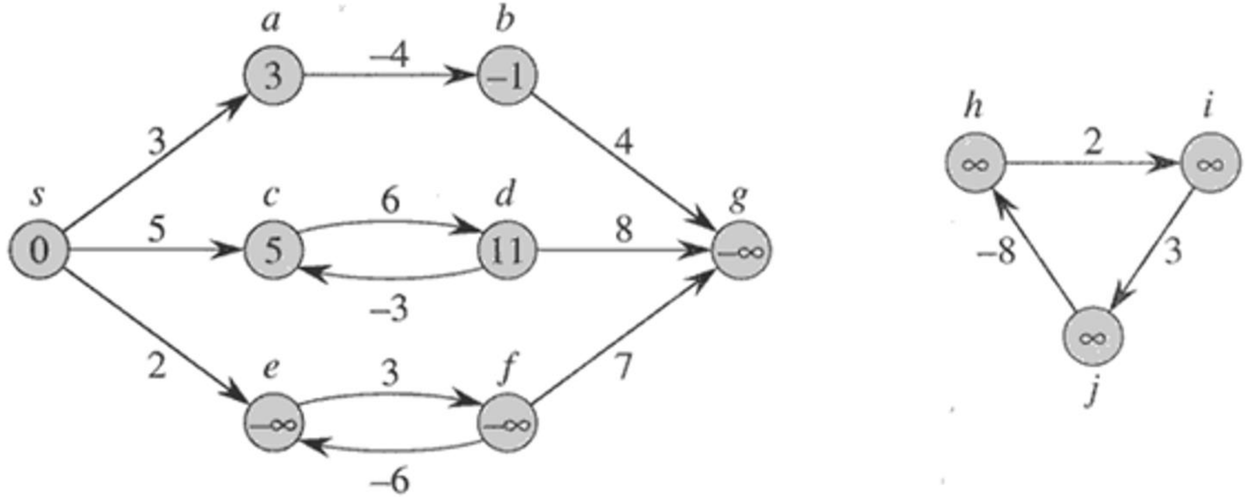
- Do grafu doplníme hrany do úplného grafu
  - Dĺžka hrany bude vzdialenosť koncových vrcholov
- Graf X obsahuje hamiltonovský cyklus práve vtedy keď úplný ohodnotený graf Y obsahuje hamiltonovský cyklus dĺžky najviac K, kde K = |V| je počet vrcholov (na obrázku K=5).

Najdlhšia cesta v grafe

- Rozhodovacia verzia: **Daný je graf G, existuje cesta medzi vrcholmi x a y, dĺžky aspoň k ?**
- Prečo to je v NP?
- Redukujeme HC:
  - Vstup: Daný graf G, je hamiltonovský?
  - (Triviálne) Riešime úlohu o najdlhšej ceste pre k=N (medzi každou dvojicou vrcholov) ...
    Transformácia na hľadanie cyklu:
  - Zdvojíme nejaký vrchol x (dvojníka označme y)
    Riešime úlohu o najdlhšej ceste medzi x a y pre k=N
    (cesta z x do y zodpovedá cyklu z x do x v pôvodnom grafe)
    teda ak by sme to vedeli určiť v polynomiálnom čase, tak by sme vedeli v pôvodnom grafe určiť, či je tam cyklus dĺžky N (hamiltonovský cyklus)

Vráťme sa späť... Najkratšia cesta v grafe

- Čo keď sú ohodnotenia záporné?
- Napr.



- Počiatočný vrchol s
- Vnútri vrcholov je vpísaná dĺžka najkratšieho sledu
  - V prípade ak v grafe nie je záporný cyklus, tak dĺžky najkratších sledov sú konečné, a rovnaké ako dĺžka najkratšej cesty
  - Ak môže byť záporný cyklus, má zmysel uvažovať najkratšiu dĺžku <u>cesty</u> (sledu, v ktorom sa neopakujú vrcholy)?

    Určite má, ale zatiaľ ľudstvo nepozná efektívny algoritmus ako to riešiť, viac na nasledujúcej prednáške...

Najkratšia cesta ak môžu byť záporné cykly

- Je tento problém NP?
- Redukcia z problému najdlhšej cesty:
  - Uvažujme graf G len s nezápornými ohodnoteniami hrán
  - Zostrojíme G':Všetkým hranám dáme opačné ohodnotenie
  - Nájdeme najkratšiu cestu v G' v polynomiálnom čase
  - Najkratšia cesta v G' zodpovedá najdlhšej ceste v G
- Inak povedané:
  Ak by existoval polynomiálny algoritmus pre najkratšiu cestu v grafe so zápornými cyklami, tak by sme vedeli (týmto spôsobom) vyriešiť problém najdlhšej cesty v ľubovoľnom grafe... (čo zatiaľ nevieme)

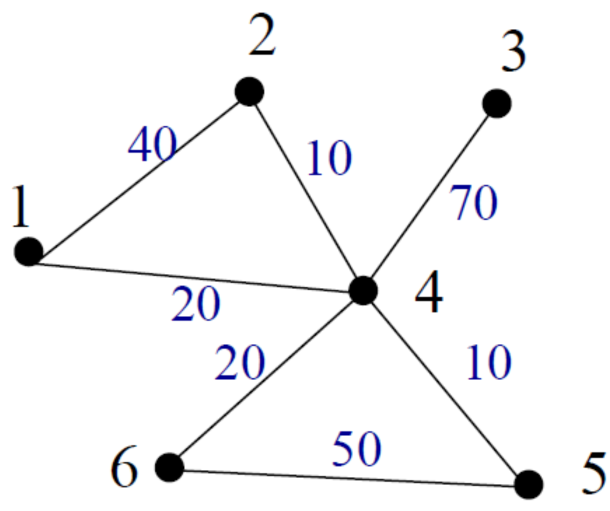
Ako sa riešia takéto ťažké problémy v praxi?

- Rýchle algoritmy, ktoré <u>určia približné riešenie</u>
  - Aproximatívne algoritmy
  - Pažravé (greedy) algoritmy
- Algoritmy ktoré sú <u>rýchle v priemernom prípade</u>, alebo <u>rýchle v špeciálnom prípade</u>
  - Dynamické programovanie
  - ...

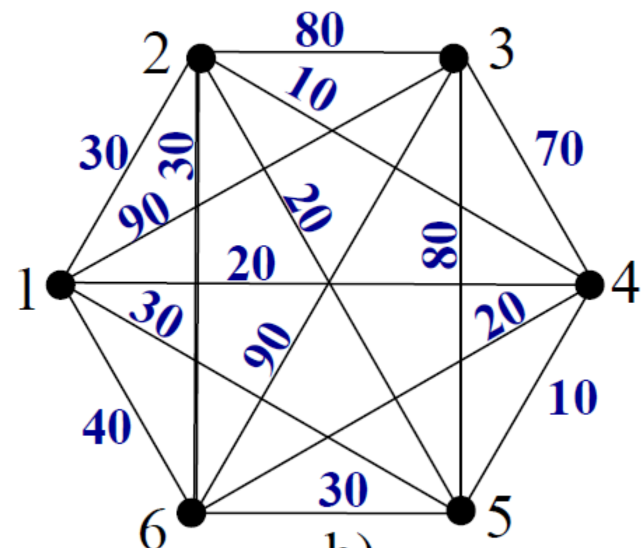
Aproximatívne riešenie TSP

- Špeciálny prípad: predpokladajme trojuholníkovú nerovnosť $d(x,y) \leq d(x,z) + d(z,y)$ pre $x,y,z \in V$.
- **Pažravý (greedy) algoritmus**
  Heuristika na hľadanie (nejakého) riešenia TSP v úplnom grafe G pre $N \geq 3$ a platí trojuholníková nerovnosť:
  1. Začni v ľubovoľnom vrchole
  2. Ak je vybratých N-1 hrán, končíme.
  3. Inak, vyber najlacnejšiu nevybranú hranu incidentnú s posledným vrcholom doteraz vybranej postupnosti takú, ktorá nie je incidentná s iným vrcholom vybratej postupnosti. Choď na Krok 2.

Pažravá heuristika pre TSP – Ukážka



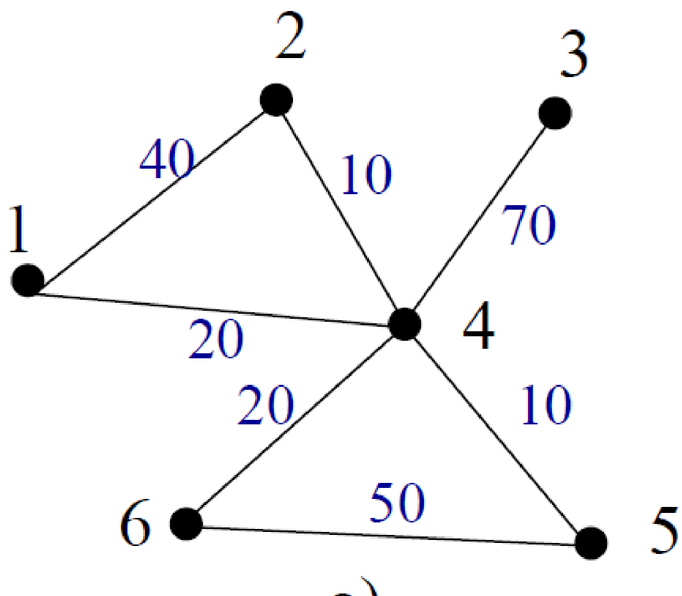
a)

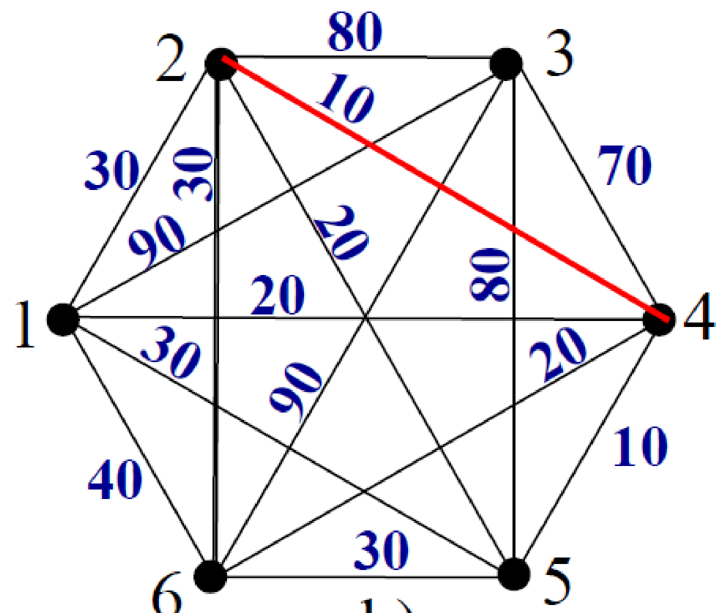


b)

$\mathcal{C} = (2)$

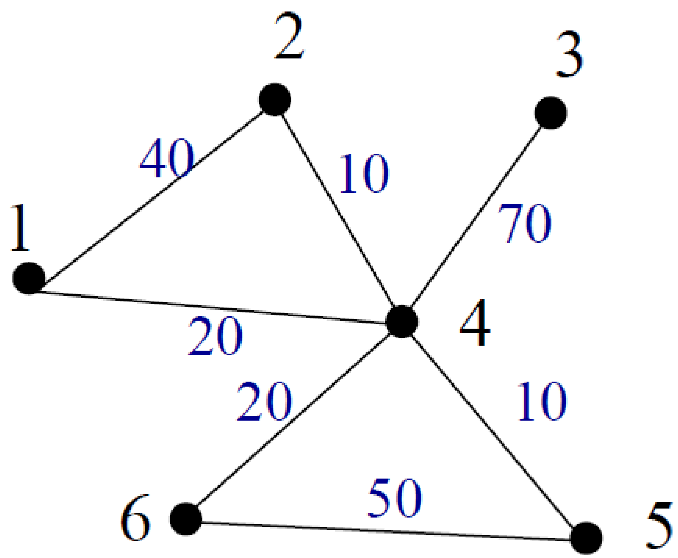
Pažravá heuristika pre TSP – Ukážka



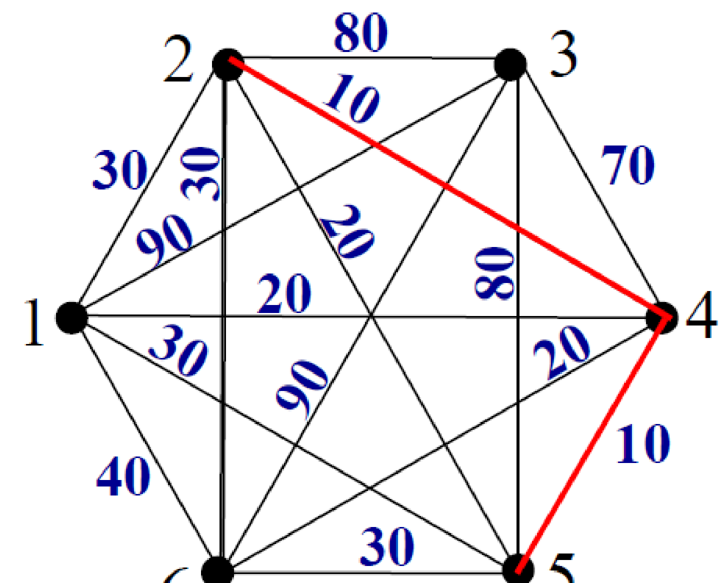


$\mathcal{C} = (2, \{2, 4\}, 4)$

Pažravá heuristika pre TSP – Ukážka



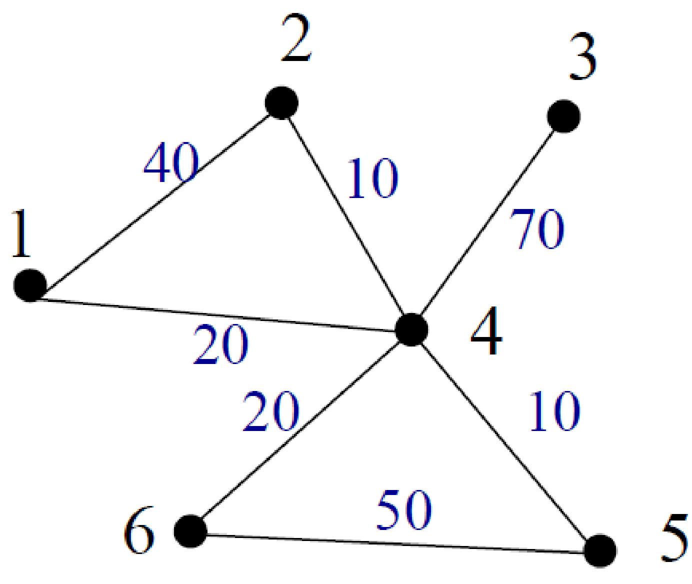
a)



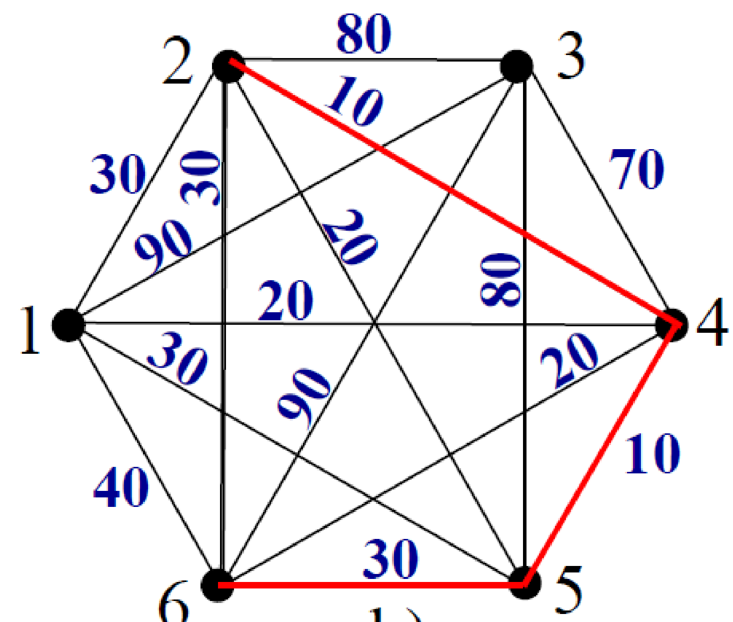
b)

$$\mathcal{C} = (2, \{2,4\}, 4, \{4,5\}, 5)$$

Pažravá heuristika pre TSP – Ukážka



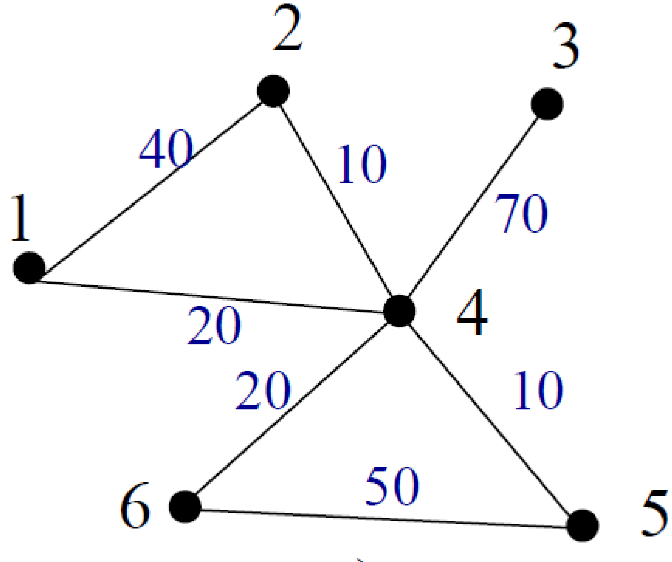
a)



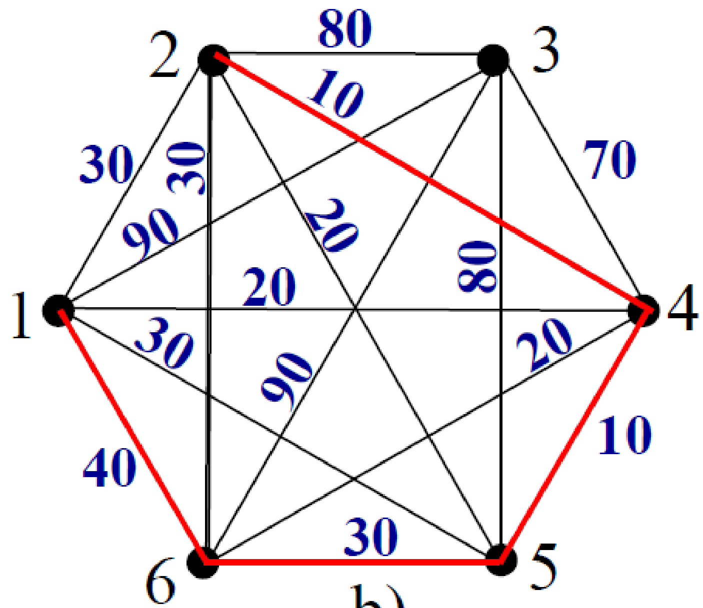
b)

$$\mathcal{C} = (2, \{2, 4\}, 4, \{4, 5\}, 5, \{5, 6\}, 6)$$

Pažravá heuristika pre TSP – Ukážka



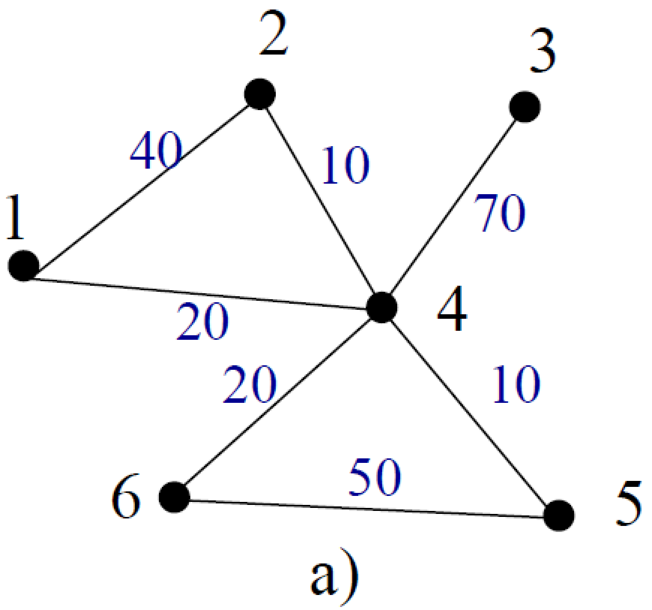
a)

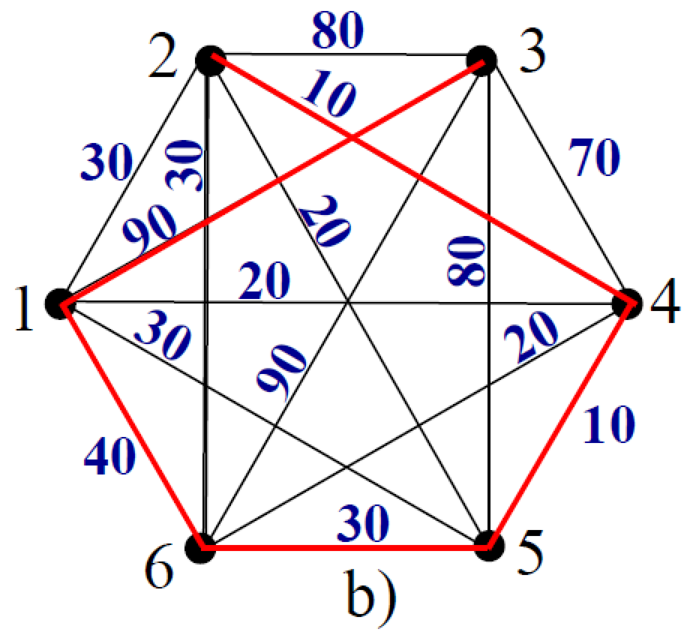


b)

$$\mathcal{C} = (2, \{2, 4\}, 4, \{4, 5\}, 5, \{5, 6\}, 6, \{6, 1\}, 1)$$

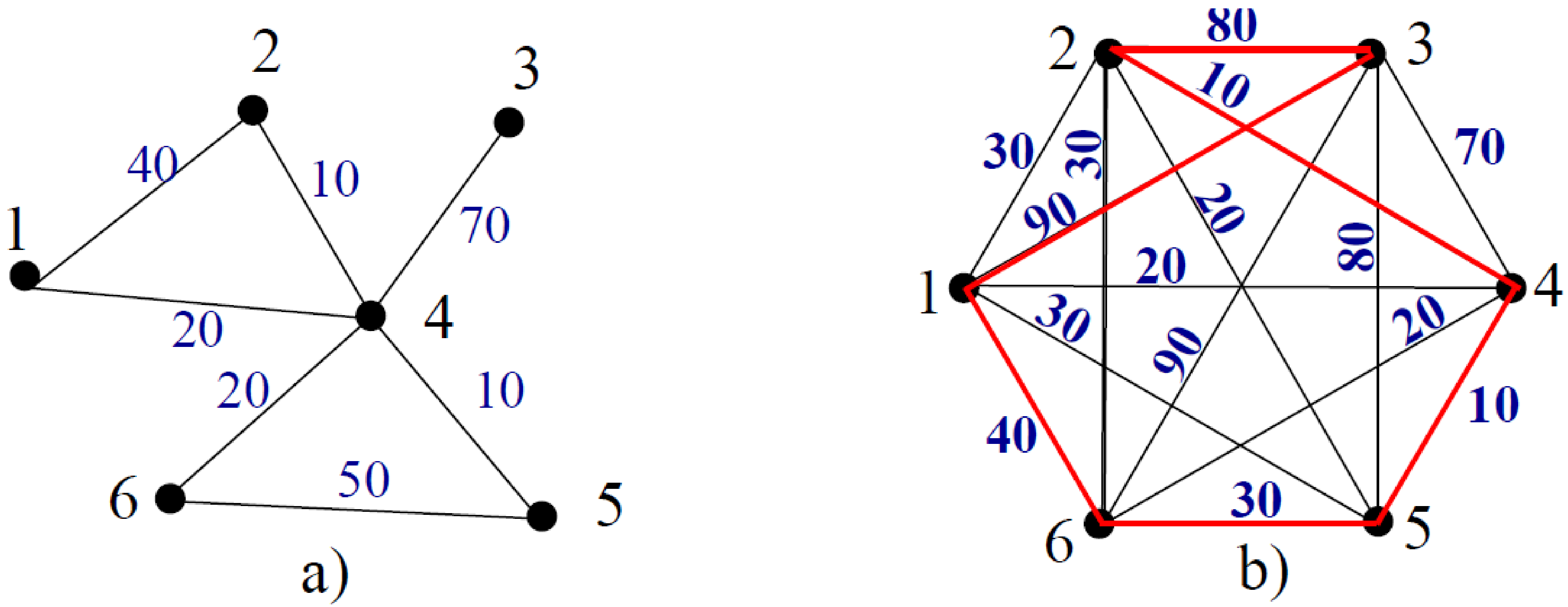
Pažravá heuristika pre TSP – Ukážka





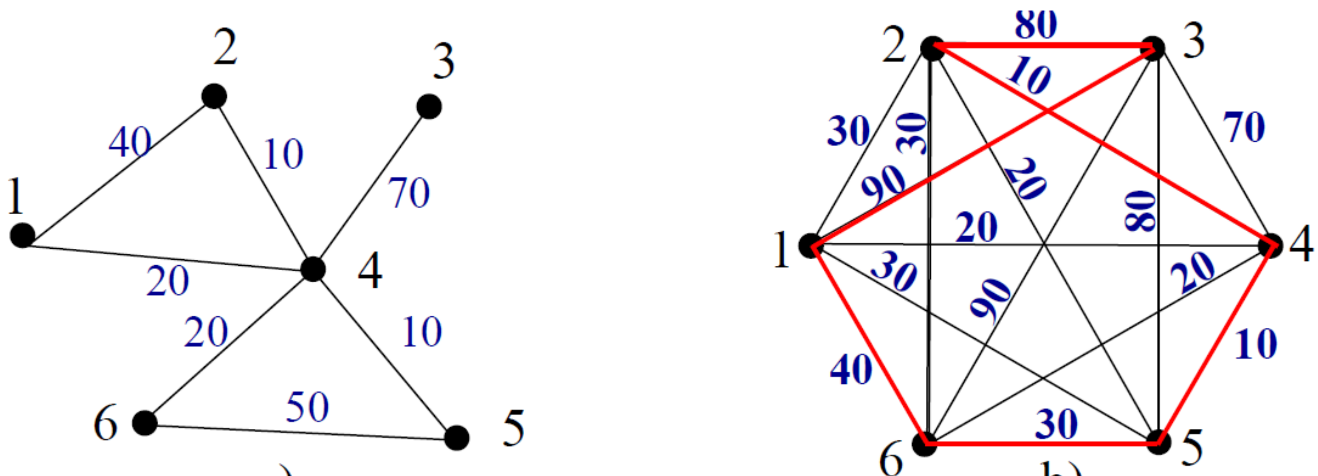
$$\mathcal{C} = (2, \{2,4\}, 4, \{4,5\}, 5, \{5,6\}, 6, \{6,1\}, 1, \{1,3\}, 3)$$

Pažravá heuristika pre TSP – Ukážka



$$\mathcal{C} = (2, \{2,4\}, 4, \{4,5\}, 5, \{5,6\}, 6, \{6,1\}, 1, \{1,3\}, 3, \{3,2\}, 2)$$

Pažravá heuristika pre TSP – Ukážka



a) b)

$$\mathcal{C} = (2, \{2,4\}, 4, \{4,5\}, 5, \{5,6\}, 6, \{6,1\}, 1, \{1,3\}, 3, \{3,2\}, 2)$$

- Každú hranu cyklu C (v úplnom grafe G') nahradíme najkratšou cestou v pôvodnom grafe G:

$$(2, \{2,4\}, 4) \rightarrow (2, \{2,4\}, 4)$$
$$(4, \{4,5\}, 5) \rightarrow (4, \{4,5\}, 5)$$
$$(5, \{5,6\}, 6) \rightarrow (5, \{5,4\}, 4, \{4,6\}, 6)$$
$$(6, \{6,1\}, 1) \rightarrow (6, \{6,4\}, 4, \{4,1\}, 1)$$
$$(1, \{1,3\}, 3) \rightarrow (1, \{1,4\}, 4, \{4,3\}, 3)$$
$$(3, \{3,2\}, 2) \rightarrow (3, \{3,4\}, 4, \{4,2\}, 2)$$

Pažravá heuristika pre TSP – Aká dlhá je cesta?

- Pre N náhodne rozmiestnených vrcholov v rovine je greedy heuristika zvyčajne cca 25% horšia ako optimálne riešenie

- Aproximačný faktor $\boldsymbol{\rho}$ (rho): $\boldsymbol{\rho} = \frac{w(alg)}{W(OPT)}$
  - w(alg) je hodnota riešenia získaného nejakým (aproximatívnym) algoritmom
  - w(OPT) je hodnota optimálneho riešenia

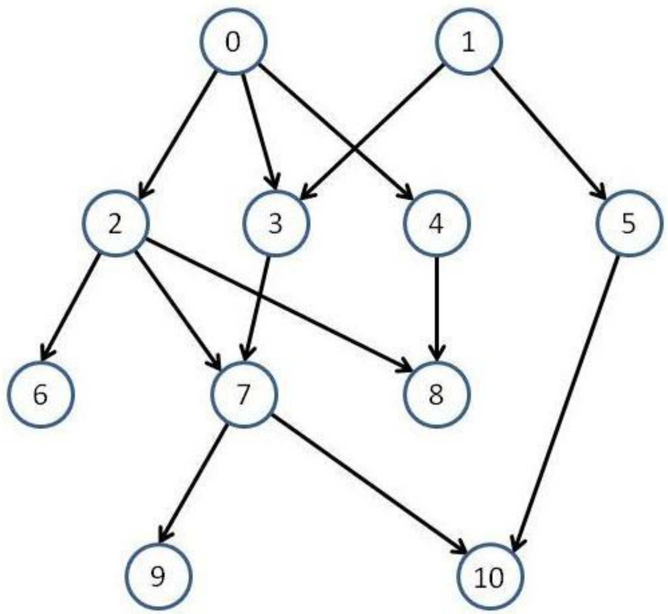
- Pažravý algoritmus v grafoch s trojuholníkovou nerovnosťou: $\boldsymbol{\rho = \theta(logN)}$

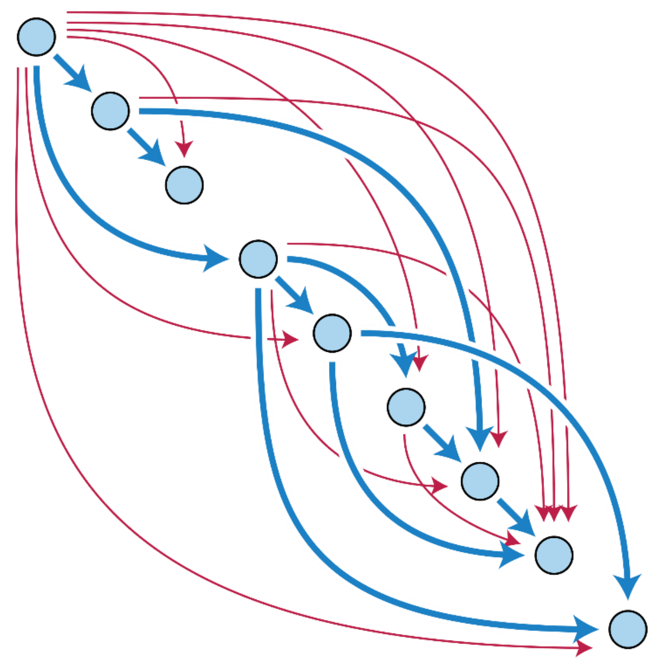
Ďalšie aproximácie TSP

- **Metóda zdvojenia kostry** (Kim 1975):
  - Nájdi minimálnu kostru
  - Zostroj uzavretý sled ako zdvojenie hrán kostry
  - Zo sledu vytvor hamiltonovský cyklus prechodom sledu, a keď narazíš na vrchol, ktorý si už navštívil, skráť úsek priamou hranou
  - $\boldsymbol{\rho} =$ 2 (keďže optimum je určite dlhé aspoň ako kostra)
- **Christofidesova metóda** (1976):
  - Nájdi minimálnu kostru
  - V kostre nájdi vrcholy nepárneho stupňa, a zostroj úplný graf K_{2t} doplnení hrany dĺžky budú vzdialenosti v pôvodnom grafe
  - **Nájdi úplné párovanie s minimálnou cenou**
  - Zostroj uzavretý eulerovský ťah
  - Z uzavretého ťahu zostroj hamiltonovský cyklus podobne ako v metóde zdvojenia kostry
  - $\boldsymbol{\rho} =$ 1,5

Najdlhšia cesta – špeciálny prípad (v DAGu)

- Orientovaný acyklický graf (DAG – directed acyclic graph)
- Reprezentácia závislostí medzi činnosťami:
  - Orientovaný graf bez (orientovaných) cyklov
  - Cyklus (v závislostiach) je problém





Najdlhšia cesta – špeciálny prípad (v DAGu)

- Ako nájdeme najdlhšiu cestu v DAGu?
- Využijeme topologické usporiadanie ...
  - Také poradie vrcholov, že žiaden vrchol nie je spracovaný skôr ako vrchol, ktorý na neho ukazuje
  - Keď spracúvame vrchol, tak „relaxujeme“ odhady dĺžok najdlhšej cesty do ešte nespracovaných vrcholov.

```
Longest-Path-in-DAG(G)
length_to = int array of |V(G)| elements (default value 0)
for each vertex v in topOrder(G) do
  for each edge (v, w) in E(G) do
    if length_to[w] <= length_to[v] + weight(G,(v,w)) then
      length_to[w] = length_to[v] + weight(G, (v,w))
return max(length_to[v] for v in V(G))
```