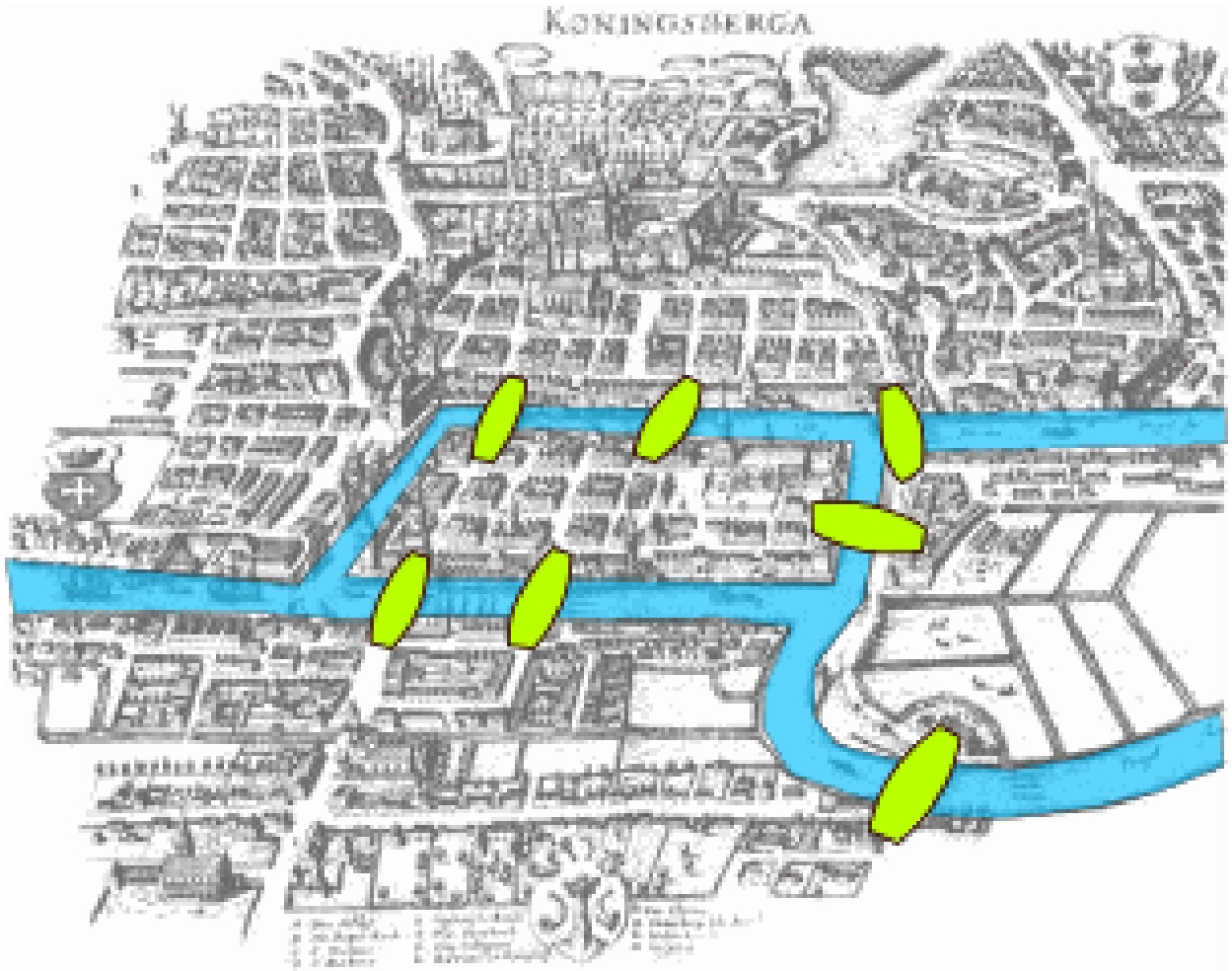
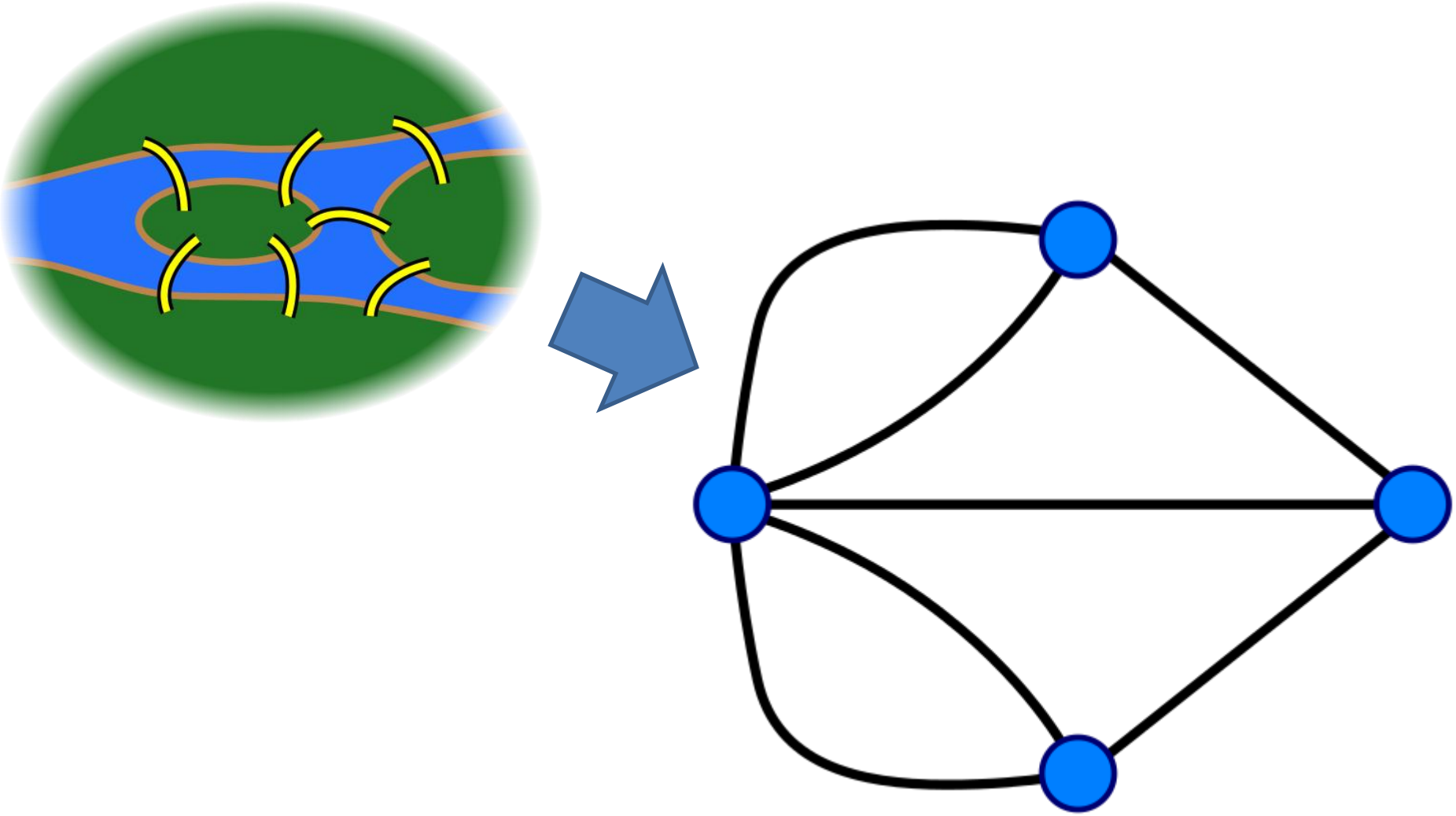
# 1. část: Základy teorie grafů

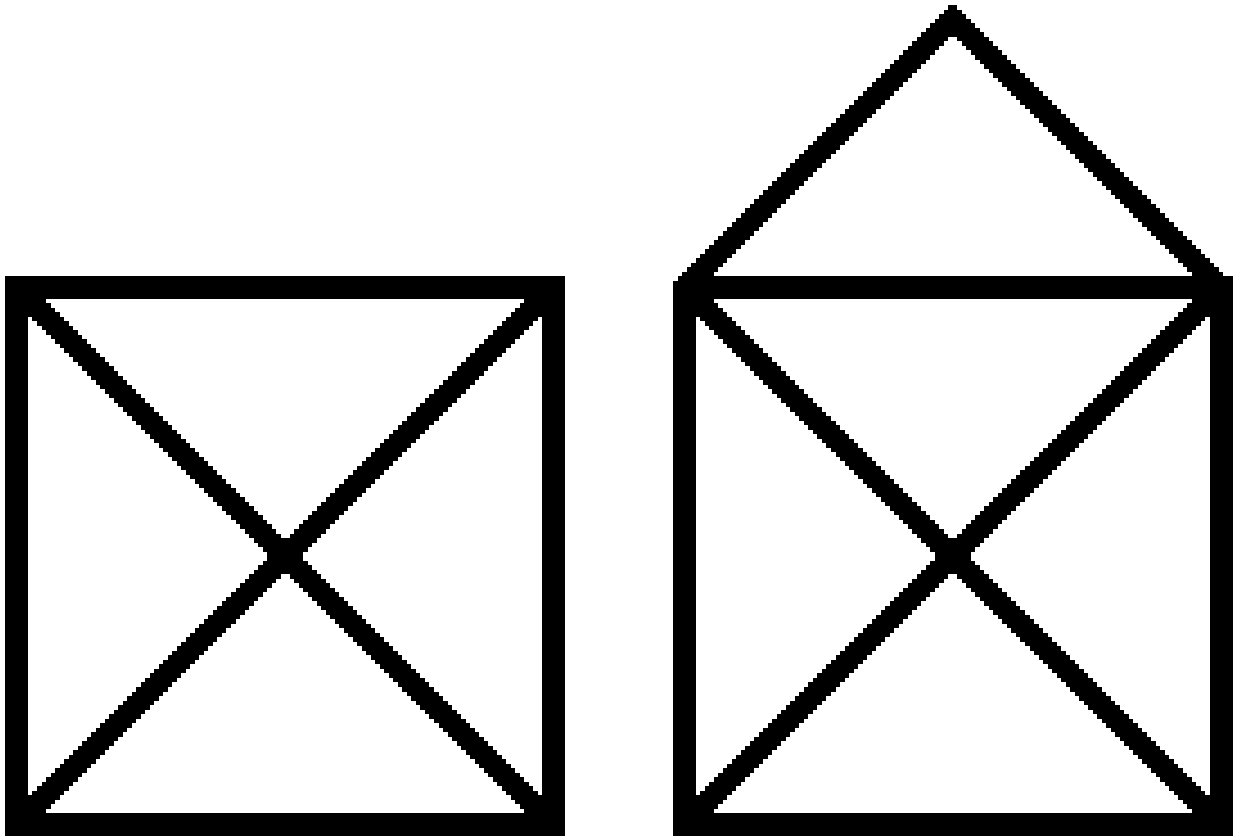
# Co je to teorie grafů?



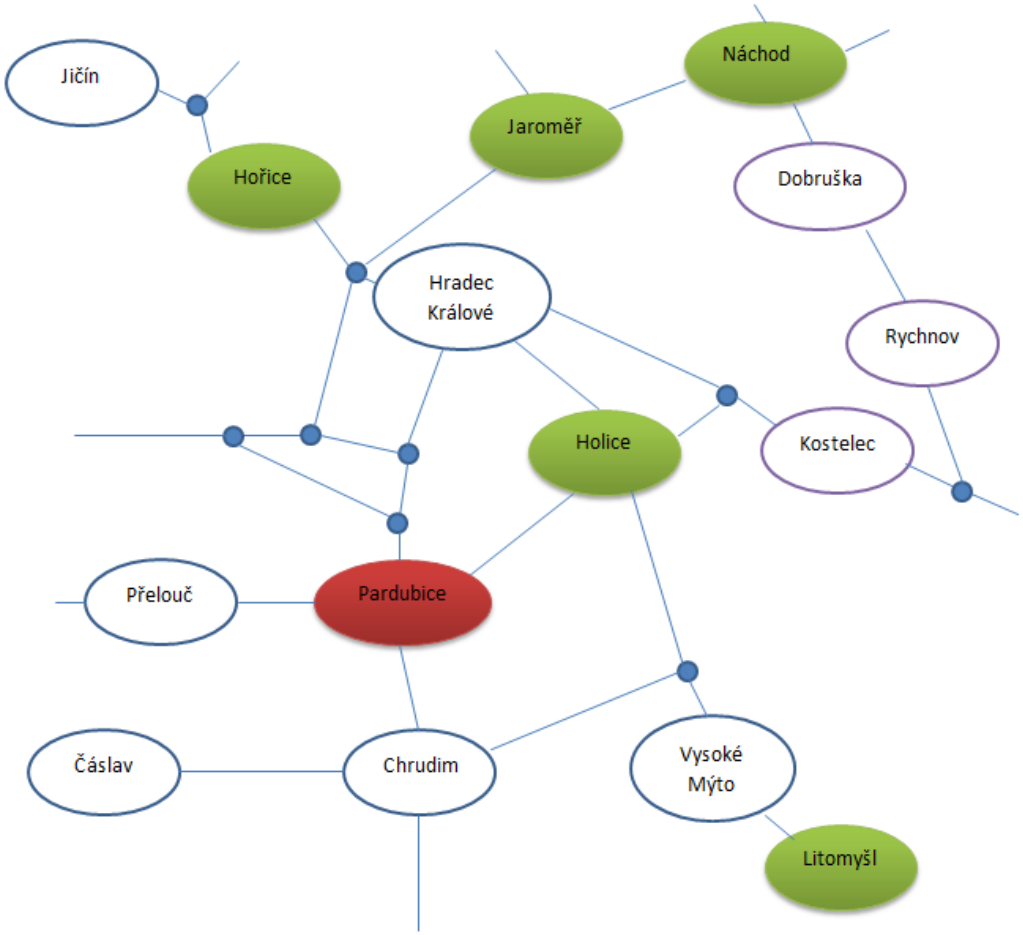
# Co je to teorie grafů?



# Co je to teorie grafů?

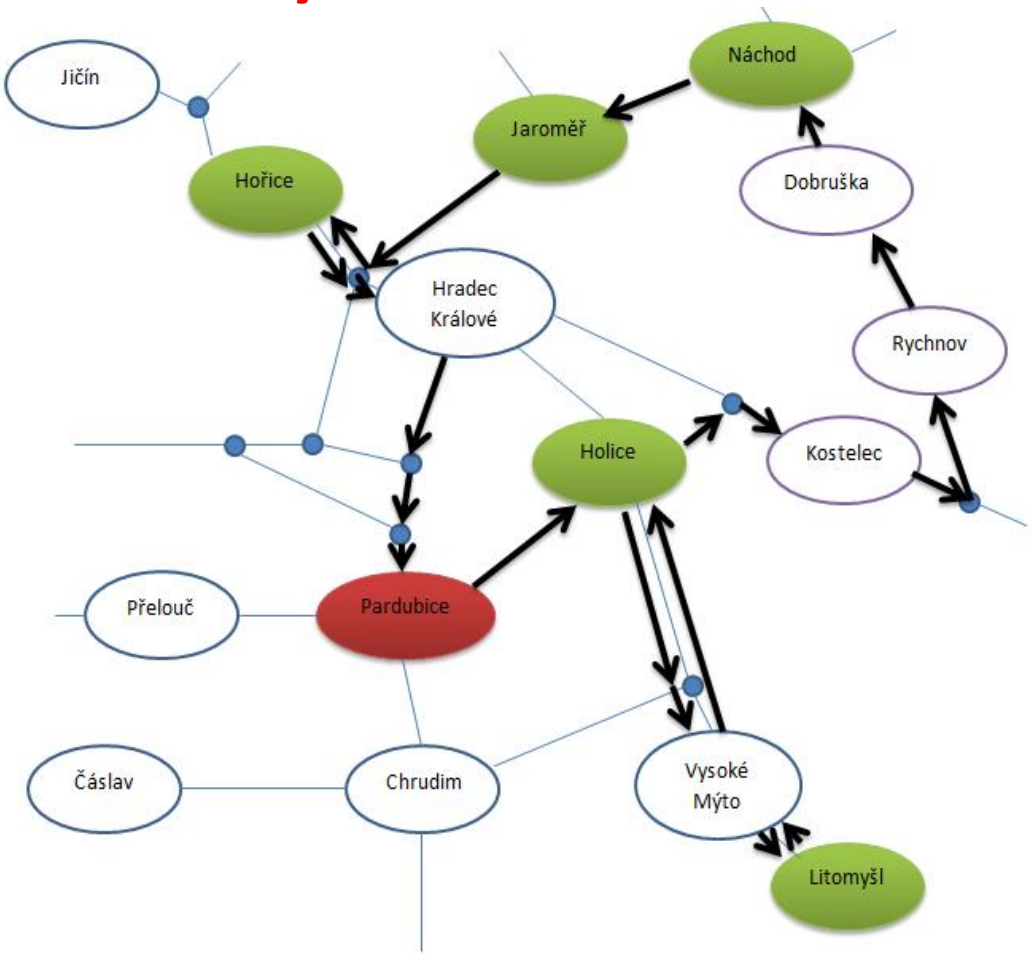


# K čemu jsou nám optimalizační metody?



# K čemu jsou nám optimalizační metody?

Ujedeme 254 km.



# K čemu jsou nám optimalizační metody?

| Počet obsluhovaných vrcholů (bez zdroje) | Počet existujících okružních jízd |
|---|---|
| **5** | 120 |
| **10** | 3 628 800 |
| **15** | 1 307 674 368 000 |
| **20** | 2 432 902 008 176 640 000 |
| **25** | 1 551 120 043 330 985 984 000 000 |
| **30** | $\approx 2{,}65 \cdot 10^{32}$ |
| **50** | $\approx 3 \cdot 10^{64}$ |

# K čemu jsou nám optimalizační metody?

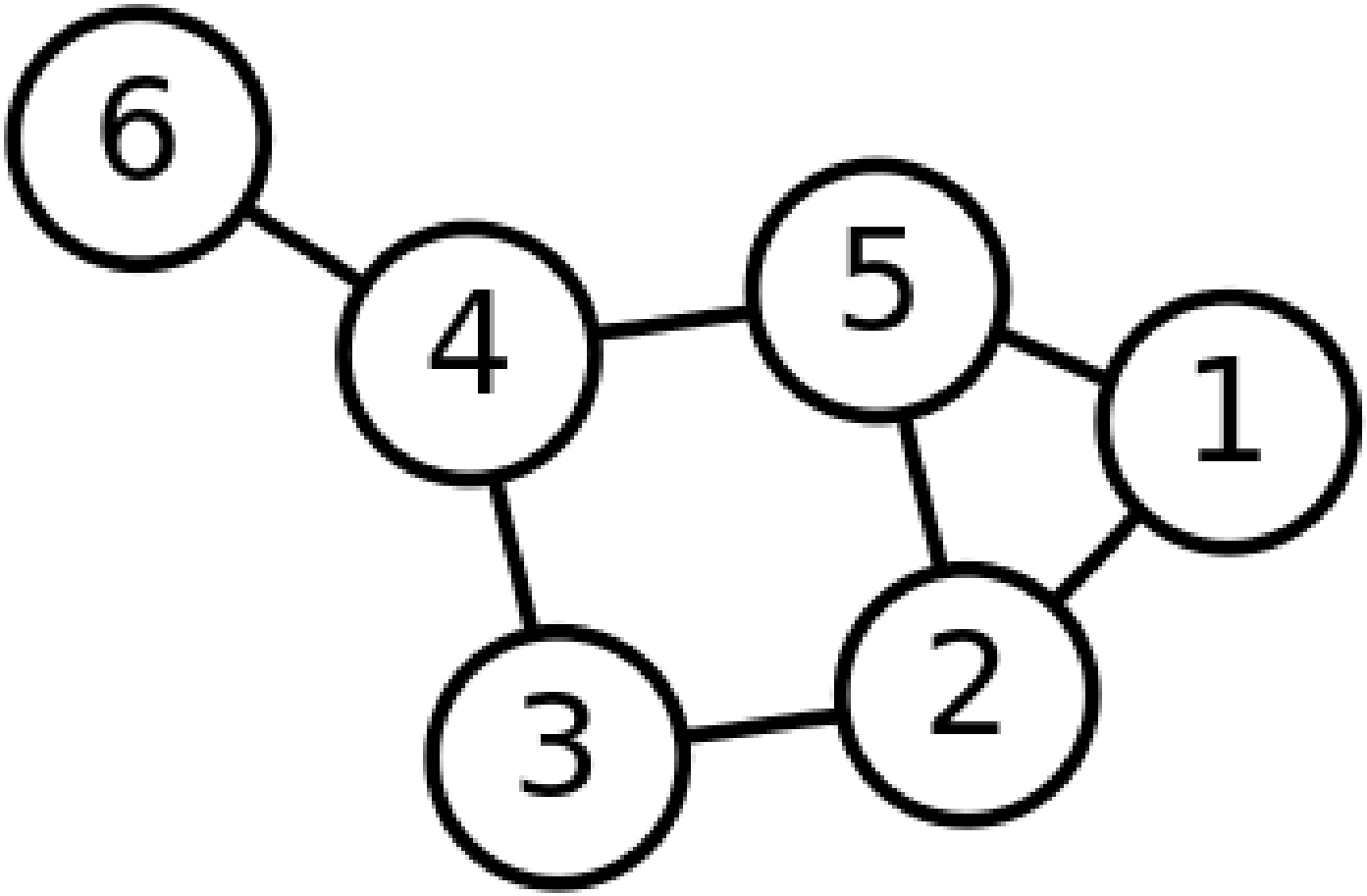
| Počet obsluhovaných vrcholů (bez zdroje) | Doba výpočtu při rychlosti prověření 1 jízdy za 1 s [roky] |
|---|---|
| **5** | $\approx 3{,}8 \cdot 10^{-6}$ (2 minuty) |
| **10** | 0,115 (42 dnů) |
| **15** | $\approx$41 466 |
| **20** | $\approx 7{,}7 \cdot 10^{10}$ |
| **25** | $\approx 4{,}9 \cdot 10^{16}$ |
| **30** | $\approx 8{,}4 \cdot 10^{24}$ |
| **50** | $\approx 9{,}5 \cdot 10^{56}$ |

Stáří Sluneční soustavy se odhaduje na $4{,}5 \cdot 10^{9}$ let.

# Graf

- **Graf** - je tvořen vrcholy a hranami. Značíme jako *G[V,H]*, kde *V* je množina vrcholů a *H* je množina hran grafu.
- Rovinný nebo prostorový útvar, který bude znázorňovat důležité vazby mezi důležitými prvky dopravního systému (vrcholy – uzly v dopravní síti, hrany – vazby mezi vrcholy).

# Graf



# Hrany grafu

- **Hrany** znázorňují vztahy mezi vrcholy.
- Rozlišujeme hrany:
  - **Neorientované** – {i,j} – nelze stanovit počáteční a koncový vrchol.
  - **Orientované** – (i,j) – lze stanovit počáteční a koncový vrchol
- **Násobné hrany** – hrany, které mají společné oba krajní vrcholy.
- **Smyčka** – hrana, která má stejné oba krajní vrcholy.

# Ohodnocení prvků grafu

- **Ohodnocení vrcholů** – číslo přiřazené vrcholu, může např. představovat počet obsluh vrcholu za jednotku času.
- **Ohodnocení hran** – číslo přiřazené hraně, může např. představovat délku hrany, počet obsluh hrany apod.

# Rozdělení grafů

- Podle toho, jaké hrany graf obsahuje, můžeme grafy dělit na:
  - **Neorientované grafy** – obsahují pouze neorientované hrany.
  - **Orientované grafy (digrafy)** – obsahují pouze orientované hrany.
  - **Smíšené grafy (migrafy)** – obsahují jak neorientované, tak orientované hrany

# Neorientované grafy

- Neorientované grafy můžeme dále dělit na:
  - **Obyčejné (jednoduché) grafy** – neobsahují žádné násobné hrany ani smyčky.
  - **Prosté grafy** – obsahují smyčky, ale neobsahují násobné hrany.
  - **Multigrafy** – obsahují násobné hrany, ale neobsahují smyčky.
  - **Pseudografy** – obsahují jak násobné hrany, tak i smyčky.

# Orientované grafy

- Orientované grafy můžeme dále dělit na:
  - **Obyčejné (jednoduché) digrafy** – neobsahují žádné násobné hrany ani smyčky.
  - **Prosté digrafy** – obsahují smyčky, ale neobsahují násobné hrany.
  - **Multidigrafy** – obsahují násobné hrany, ale neobsahují smyčky.
  - **Pseudodigrafy** – obsahují jak násobné hrany, tak i smyčky.

# Smíšené grafy

- Smíšené grafy můžeme dále dělit na:
  - **Obyčejné (jednoduché) migrafy** – neobsahují žádné násobné hrany ani smyčky.
  - **Prosté migrafy** – obsahují smyčky, ale neobsahují násobné hrany.
  - **Multimigrafy** – obsahují násobné hrany, ale neobsahují smyčky.
  - **Pseudomigrafy** – obsahují jak násobné hrany, tak i smyčky.

# Stupeň vrcholu

- **Stupeň vrcholu** udává počet hran, které s daným vrcholem incidují.
  - V neorientovaném grafu – stupeň vrcholu v označujeme *st* (v) = (*a*).
  - V orientovaném grafu – stupeň vrcholu v označujeme jako *st* (v) = (*a*,*b*), kde *a* udává počet hran, které do vrcholu v vstupují (příchozí stupeň) a *b* udává počet hran, které z vrcholu v vystupují (odchozí stupeň).

# Způsoby reprezentace neorientovaných grafů

- Grafy lze reprezentovat více způsoby, nejčastější formy reprezentace jsou:
  - Množinový zápis.
  - **Diagram grafu**.
  - Matice sousednosti.
  - Matice ohodnocení hran.
  - Incidenční matice.

# Speciální typy grafů

- **Prázdný graf** – neobsahuje žádný vrchol, tedy ani žádnou hranu, tedy $V = \{\ \} \Rightarrow H = \{\ \}$.
- **Nulový graf** – obsahuje pouze vrcholy, neobsahuje žádnou hranu, tedy $V \neq \{\ \}, H = \{\ \}$.
- **Triviální graf** – případ nulového grafu, obsahuje pouze jeden vrchol, tedy $|V| = 1, H = \{\ \}$.
- **Pravidelný graf** – graf, ve kterém jsou všechny vrcholy stejného stupně.

# Speciální typy grafů

- **Kružnice** – pravidelný graf, ve kterém jsou všechny vrcholy druhého stupně.
- **Kompletní (úplný) graf** – graf, ve kterém jsou všechny dvojice vrcholů spojeny hranou (triviální graf se považuje za kompletní). Počet hran kompletního grafu je roven $\binom{n}{2}$, kde *n* je počet vrcholů.
- **Disjunktní grafy** – jsou grafy, jejíchž průnikem je graf prázdný.

# Speciální typy grafů

- **Komplementární (doplňkové) grafy** – grafy $G_1[V,H_1]$ a $G_2[V,H_2]$ nazveme komplementární, platí-li, že $H_1 \cap H_2 = \emptyset$, $H_1 \cup H_2 = H$ a graf $G[V,H]$ je úplný graf.

# Speciální typy grafů

- **Izomorfní grafy** – grafy, které jsou navzájem ekvivalentní, liší se pouze jiným označením vrcholů a hran a jiným způsobem zakreslení. Podmínky izomorfismu jsou následující:
  - počty vrcholů grafů musí být shodné,
  - počty hran grafů musí být shodné,
  - počty vrcholů daných stupňů musí být shodné,
  - nechť jsou vrcholy prvního grafu vzory, potom každý vzor musí mít v druhém (resp. třetím,…) grafu svůj obraz; vrchol je obrazem tehdy, pokud inciduje se stejným počtem vrcholů stejného stupně jako vzor.

# Sled, cesta a tah

- **Sled** – střídavá posloupnost vrcholů a hran začínající a končící ve vrcholu.
- **Cesta** – sled, ve kterém se neopakuje žádný vrchol.
- **Minimální cesta** mezi vrcholy *u* a *v* je cesta, jež má ze všech možných cest mezi těmito vrcholy minimální součet ohodnocení hran do cesty zařazených.
- **Tah** – sled, ve kterém se neopakuje žádná hrana.

# Souvislost grafů

- Graf nazveme ***souvislý***, pokud pro každou dvojici vrcholů existuje cesta, která je spojuje.

# Souvislost digrafů

- U digrafů rozlišujeme:
  - **Neorientovanou souvislost** – digraf je neorientovaně souvislý, existuje–li pro každou dvojici vrcholů cesta bez ohledu na orientaci hran.
  - **Slabou souvislost** – digraf je slabě souvislý, existuje–li pro každou dvojici vrcholů u a v cesta buď z u do v nebo cesta z v do u.
  - **Silnou souvislost** – digraf je silně souvislý, existuje–li pro každou dvojici vrcholů u a v cesta z u do v a současně cesta z v do u.

# Podgrafy grafů

- Graf G[V,H] nazveme **podgrafem** grafu $G_0[V_0,H_0]$ tehdy, pokud platí $V \subseteq V_0$ a $H \subseteq H_0$. Rozlišuje dva typy podgrafů:
  - **Vlastní podgrafy** – platí pro ně $V \subset V_0$ a $H \subset H_0$.
  - **Faktorové podgrafy** – platí pro ně $V = V_0$ a $H \subseteq H_0$ (každý graf je zároveň svým faktorovým podgrafem).

# Komponent grafu, most, artikulace

- **Komponentem** grafu nazveme maximální souvislý podgraf; má-li graf jeden komponent, potom je souvislý.
- **Most** je hrana, jejímž odstraněním se zvýší počet komponentů o jednu.
- **Artikulace** je vrchol, jehož odstraněním (spolu s incidujícími hranami) se zvýší počet komponentů alespoň o jeden.

# Strom

- **Strom** – souvislý graf, který v žádné své části neobsahuje kružnici; pro strom dále platí:
  - Mezi každými dvěma vrcholy existuje cesta, která je spojuje.
  - Každá hrana je mostem, každý vrchol se stupněm vyšším než 1 je artikulací,
  - Pro počet hran ve stromu platí, že je nižší o jednu než je počet vrcholů, tedy $|H| = |V| - 1$.

# Hvězda, kostra grafu

- **Hvězda** – speciální případ stromu, pro který dále platí:
  - (n-1) vrcholů je 1. stupně.
  - 1 vrchol je stupně (n-1).
- **Kostra grafu** – faktorový podgraf, který je zároveň stromem.
- **Minimální kostra grafu** – kostra grafu s minimálním součtem ohodnocení hran do kostry zařazených.

# Eulerův tah, Hamiltonova kružnice

- **Eulerův tah** – je tah, který prochází všemi hranami sítě (užití např. při obsluze všech hran sítě).
- **Hamiltonova kružnice** – kružnice, která prochází všemi vrcholy původního grafu (užití např. při obsluze všech vrcholů grafu).
- **Minimální Hamiltonova kružnice** – HK s minimálním součtem ohodnocení hran do HK zařazených.

# Princip sudosti

- V neorientovaném obyčejném konečném grafu (souvislost není podmínkou) platí, že součet stupňů všech vrcholů je roven dvojnásobku počtu hran, tedy:

$$\sum_{i=1}^{n} st\left(V_i\right) = 2\left|H\right|.$$

- **Princip sudosti**: Počet vrcholů lichého stupně v neorientovaném obyčejném konečném grafu je sudý.

# Princip sudosti

- Důkaz principu sudosti:
  - Označme $\sigma_k$ počet vrcholů *k*-tého stupně. Potom můžeme psát:
    $$\sum_{k=0}^{\infty} k \cdot \sigma_k = 2|H|.$$
  - Rozdělme výše uvedený součet na dva součty – vrcholy lichého stupně a vrcholy sudého stupně a upravme:

$$\sum_{j=0}^{\infty} 2j \cdot \sigma_{2j} + \sum_{j=0}^{\infty} (2j+1) \cdot \sigma_{2j+1} = 2|H|,$$

$$\sum_{j=0}^{\infty} 2j \cdot \sigma_{2j} + \sum_{j=0}^{\infty} 2j \cdot \sigma_{2j+1} + \sum_{j=0}^{\infty} \sigma_{2j+1} = 2|H|,$$

# Princip sudosti

$$\sum_{j=0}^{\infty} \sigma_{2j+1} = 2|H| - \sum_{j=0}^{\infty} 2j \cdot \sigma_{2j} - \sum_{j=0}^{\infty} 2j \cdot \sigma_{2j+1}.$$

Sudá čísla

Výraz na pravé straně je určitě sudý, výraz na levé straně vyjadřuje počet vrcholů lichého stupně → Počet vrcholů lichého stupně je sudý.

# Konstrukce Eulerova tahu

- Podmínky existence ET (nutné a zároveň postačující):
  - Graf musí být souvislý, obyčejný a konečný.
  - Všechny vrcholy musí mít sudý stupeň (uzavřený ET) nebo právě 2 vrcholy musí být stupně lichého (otevřený ET).
- V případě právě 2 vrcholů lichého stupně doplníme graf pomocnou hranou spojující tyto vrcholy – získáme graf se všemi vrcholy sudého stupně.

# Fleuryho algoritmus

1) Konstrukci ET započneme v libovolném uzlu sítě a do ET zařadíme libovolný úsek incidující s tímto uzlem (v případě grafu s právě dvěma vrcholy lichého stupně zařadíme do ET pomocnou hranu, kterou poté z ET vypustíme), pokračujeme na krok 2).
2) Jsou–li v tahu zařazeny všechny hrany grafu, máme ET, jinak pokračujeme na krok 3)
3) Jako další zařadíme do ET dosud nezařazenou hranu incidující s naposledy navštíveným vrcholem, dbáme na to, aby označením této hrany nedošlo k rozpadu podsítě složené ze zbývajících dosud nezařazených hran a s nimi incidujících vrcholů na dva neprázdné komponenty grafu nebo na neprázdný komponent a uzel, v němž jsme konstrukci ET začali; návrat na krok 2).

# Konstrukce kostry grafu

- Využití např. při zimních kalamitních stavech pro obnovení základního spojení mezi jednotlivými vrcholy sítě (grafu).
- Pro vyhledání minimální kostry slouží **Kruskalův (hladový) algoritmus**.

# Kruskalův (hladový) algoritmus

1) Podle výše ohodnocení utvoříme vzestupnou posloupnost hran (mají-li některé hrany stejná ohodnocení, potom na vzájemném pořadí mezi nimi nezáleží) a pokračujeme krokem 2).
2) Z posloupnosti hran vybereme hranu s minimálním ohodnocením a nevytvoří-li společně s předchozími úseky zařazenými do kostry v žádné části grafu kružnici, zařadíme jej do kostry. Následuje krok 3).
3) Je-li počet hran zařazených do kostry menší než $n-1$, přičemž $n$ je počet vrcholů, vracíme se na krok 2), v opačném případě algoritmus končí, vyhledaná kostra sítě je minimální.

# Vyhledání Hamiltonovy kružnice

- Nutné podmínky: Neorientovaný graf musí být souvislý, obyčejný a konečný, musí obsahovat alespoň 3 vrcholy a nesmí obsahovat mosty ani artikulace.
- Postačující podmínky:
  - Diracova – každý vrchol stupeň alespoň $\frac{n}{2}$.
  - Oreho – součet stupňů každé dvojice nesousedních vrcholů musí být alespoň *n*.

# Vyhledání Hamiltonovy kružnice

– Posova podmínka – musí být splněno

$$\delta_k < k,$$

kde *k*…každé přirozené číslo z intervalu $\left(0, \frac{n}{2}\right)$,

$\delta_k$…počet vrcholů grafu se stupněm nejvýše *k*.

- Kompletní graf s alespoň 3 vrcholy splňuje nutné a postačující podmínky vždy, proto v něm HK vždy existuje.

# Metoda nejbližšího dosud nenavštíveného vrcholu

- Tato metoda je heuristická, nezaručuje nalezení optimálního řešení, tedy minimální Hamiltonovy kružnice!
- Nejdříve transformujeme původní graf na graf úplný (množina vrcholů je totožná a hrany reprezentují minimální cesty mezi jednotlivými vrcholy, ohodnocení hran odpovídá vzdálenostem), minimální HK se vyhledá v transformovaném grafu a získané řešení se poté zpětně transformuje do původní sítě.
- V případě, že je třeba nalézt optimální řešení, použijeme Littlův algoritmus, bude probrán později.

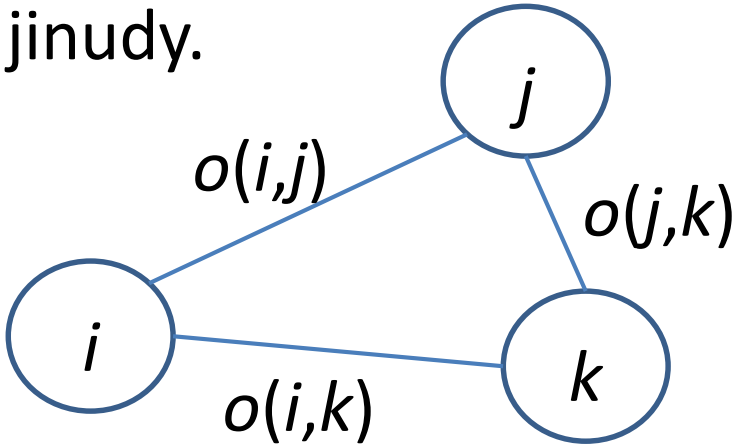
# Metoda nejbližšího dosud nenavštíveného vrcholu

1) S konstrukcí HK začneme v libovolném vrcholu a do posloupnosti hran zařadíme incidující hranu s minimálním ohodnocením (je-li takovýchto hran více, pak volíme libovolnou z nich) a pokračujeme krokem 2).
2) Je-li vybráno *n-1* hran, potom pokračujeme krokem 4), v opačném případě následuje krok 3).
3) Z naposledy navštíveného vrcholu jako další zařazujeme do posloupnosti hran hranu incidující s dosud nenavštíveným vrcholem, který má minimální ohodnocení (existuje-li takovýchto hran více, vybíráme libovolnou z nich) a vracíme se na krok 2).
4) Uzavřeme kružnici a algoritmus končí. Vyhledaná kružnice je Hamiltonova.

# Vyhledávání vzdáleností v grafech

- **Vzdálenost** – vzdáleností rozumíme délku minimální cesty.
- Algoritmus je založen na porovnání hodnot přímých a nepřímých vzdáleností:
  - Hrana *(i,j)* patří do minimální cesty tehdy, pokud nevede minimální cesta jinudy.

$$o(i,k)+o(k,j)<o(i,j)$$



# Floydův algoritmus

- 1) Sestavení matice délek přímých cest *C*, přičemž pro prvky matice platí:

$$c_{ij} = 0, \text{ je - li } i = j,$$

$$c_{ij} = o(i, j), \text{ je - li } i \neq j \text{ a hrana spojující vrcholy } u_i \text{ a } u_j \text{ existuje,}$$

$$c_{ij} = \infty, \text{ je - li } i \neq j \text{ a hrana spojující vrcholy } u_i \text{ a } u_j \text{ neexistuje.}$$

- V případě výpočtu na počítači se nekonečno nahrazuje dostatečně velkým číslem, např. $c_{\max} \cdot n$.

# Floydův algoritmus

- 2) Položíme $k = 1$; tato pomocná proměnná bude v průběhu nabývat hodnot $k = 1,2,\ldots,n$ a představuje index vrcholu, přes který provádíme přepočet.

# Floydův algoritmus

- 3) Provedeme přepočet jednotlivých prvků matice *C* podle vztahu:

$$c_{ij} = \min\{c_{ij}, c_{ik} + c_{kj}\},$$

přičemž nemusíme přepočítávat prvky, pro které platí *i* = *j* (prvky ležící na hlavní diagonále), prvky, pro které platí *i*,*j* = *k* (leží v řádku či sloupci s indexem *k*) a prvky *i* ≠ *k* a *j* ≠ *k*, pro které $c_{ik} = \infty$ a $c_{kj} = \infty$.

# Floydův algoritmus

- 4) Platí-li $k < n$, položíme $k = k + 1$ a vracíme se na krok 3), v opačném případě je naposledy získaná matice maticí vzdáleností.

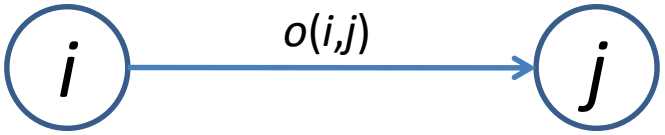
- Z uvedeného postupu je tedy zřejmé, že přepočet matice $C$ musíme provést $n$-krát.

# Metoda kritické cesty (CPM)

- Metoda kritické cesty je matematická metoda, která se užívá při řízení projektů složených z dílčích činností. Umožňuje vyhledat činnosti, u kterých, kdyby došlo k jejich prodloužení nebo posunutí, by došlo k prodloužení trvání celého projektu.
- Metoda CPM pracuje s deterministickými dobami trvání činností.

# Metoda kritické cesty (CPM)

- **Síťový graf** je prostředkem pro znázornění jednotlivých činností projektu a vazeb mezi nimi.
  - Dílčí stavy projektu zpravidla vyjadřujeme vrcholy síťového grafu.
  - Činnosti zpravidla znázorňujeme jako orientované hrany grafu.
  - Doba trvání činnosti je potom vyjádřena ohodnocením příslušné hrany.



# Metoda kritické cesty (CPM)

- **Počáteční vrchol grafu** je vrchol, ze kterého orientované hrany pouze vystupují, tento vrchol reprezentuje počáteční (výchozí) stav projektu.
- **Koncový vrchol grafu** je vrchol, do kterého orientované hrany pouze vstupují, tento vrchol reprezentuje konečný (cílový) stav projektu.

# Metoda kritické cesty (CPM)

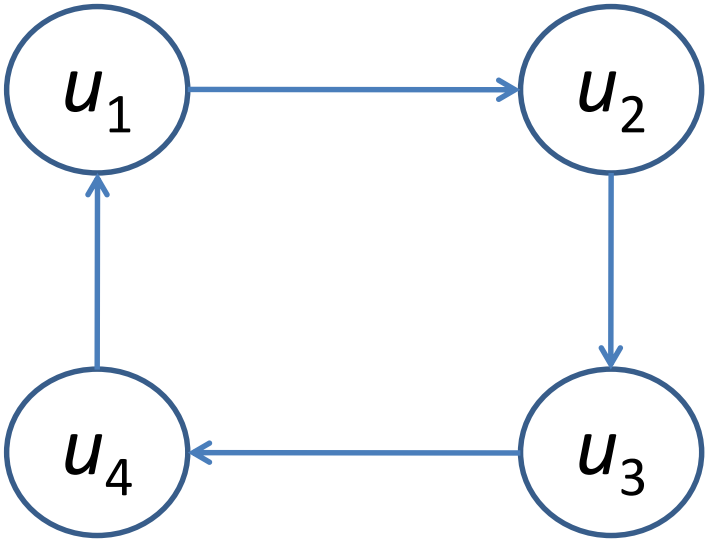
- Síťový graf musí být:
  - Konečný.
  - Souvislý.
  - Orientovaný.
  - Acyklický (v žádné své části nesmí tvořit cyklus).
  - Hranově ohodnocený.
  - Obyčejný (tj. bez násobných hran a smyček).

# Metoda kritické cesty (CPM)

- Cyklem rozumíme digraf, ve kterém platí:

$$\forall i \in \{1,2,\ldots,n\} st(u_i) = (1,1),$$

kde *n* je počet vrcholů digrafu.

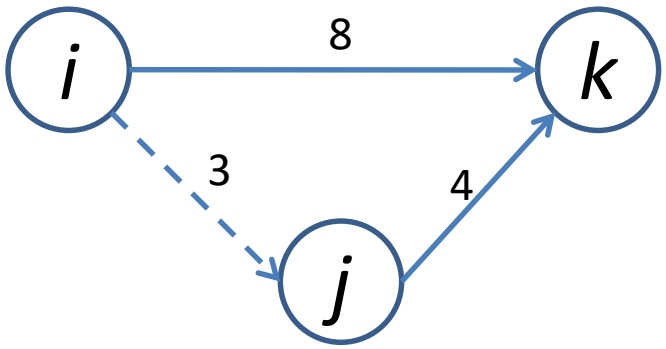


# Metoda kritické cesty (CPM)

- V síťovém grafu rozlišujeme tři druhy hran reprezentující:
  - **Reálné činnosti** – tyto hrany reprezentují skutečné činnosti, trvají tedy určitou dobu a spotřebovávají lidské nebo jiné zdroje.
  - **Fiktivní činnosti** – tyto hrany nepředstavují skutečné činnosti (mají nulovou délku trvání a ani nespotřebovávají zdroje), slouží k vyjádření např. organizačních a jiných vazeb.

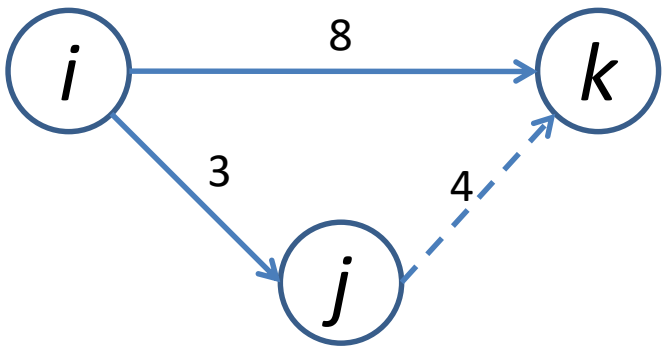
# Metoda kritické cesty (CPM)

– **Čekací činnosti** – speciální případ fiktivní hrany s nenulovým trváním, tato činnost se např. používá, je-li třeba zajistit, že určitá činnost může začínat až po uplynutí určitého času po skončení jiné činnosti.




Činnost (*i*,*j*) může začít nejdříve 3 časové jednotky po začátku činnosti (*i*,*k*).

# Metoda kritické cesty (CPM)

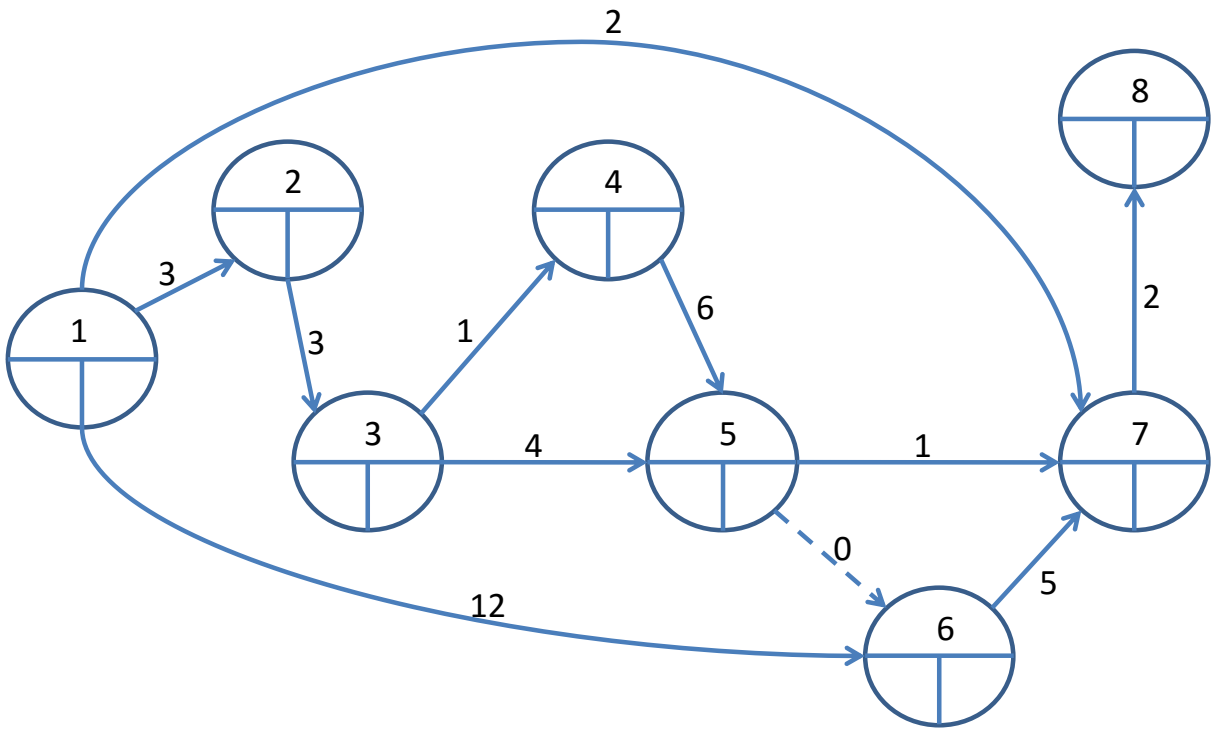



Činnost (*i*,*j*) musí skončit nejdříve 4 časové jednotky před koncem činnosti (*i*,*k*), nejpozději tedy musí začít 1 časovou jednotku od začátku činnosti (*i*,*k*).

- Síťový graf můžeme zobrazit více způsoby, nejčastěji se používá:
  - Grafický zápis (diagram síťového grafu).
  - Tabulkový zápis.

# Metoda kritické cesty (CPM)

| Činnost | *i* | *j* | *o* [*i*,*j*] [týden] |
|---|---|---|---|
| 12 | 1 | 2 | 3 |
| 16 | 1 | 6 | 12 |
| 17 | 1 | 7 | 2 |
| 23 | 2 | 3 | 3 |
| 34 | 3 | 4 | 1 |
| 35 | 3 | 5 | 4 |
| 45 | 4 | 5 | 6 |
| 56 | 5 | 6 | 0 |
| 57 | 5 | 7 | 1 |
| 67 | 6 | 7 | 5 |
| 78 | 7 | 8 | 2 |



# Metoda kritické cesty (CPM)

- Při sestavě síťového grafu je nutno znát:
  - Které činnosti bezprostředně předcházejí každé činnosti.
  - Které činnosti bezprostředně následují po každé činnosti.
  - Které činnosti probíhají paralelně.
  - Které činnosti jsou navzájem závislé.

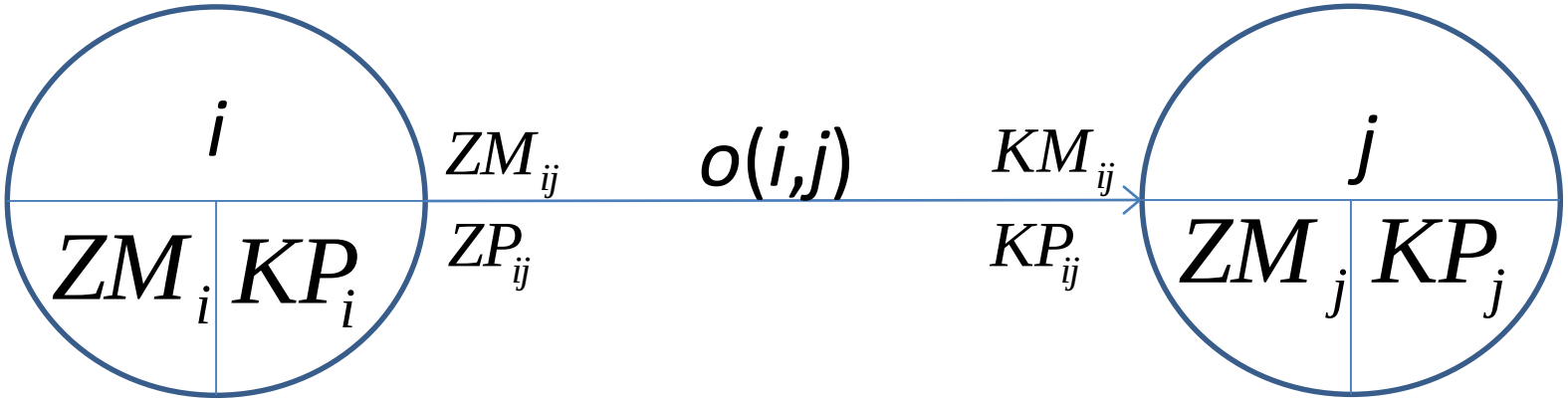
# Metoda kritické cesty (CPM)

- Pro potřeby výpočtu zavádíme pro činnosti:
  - $t_{ij}^{(0)}, ZM_{ij}$ - nejdříve možný začátek činnosti (*i,j*).
  - $t_{ij}^{(0)} + o(i, j), KM_{ij}$ - nejdříve možný konec činnosti [*i,j*].
  - $t_{ij}^{(1)}, KP_{ij}$ - nejpozději přípustný konec činnosti (*i,j*).
  - $t_{ij}^{(1)} - o(i, j), ZP_{ij}$ - nejpozději přípustný začátek činnosti [*i,j*].
  - $CR_{ij} = t_{ij}^{(1)} - \left\{t_{ij}^{(0)} + o(i, j)\right\} = KP_{ij} - KM_{ij}$ - celková rezerva činnosti (*i,j*).

# Metoda kritické cesty (CPM)

- Pro potřeby výpočtu zavádíme pro vrcholy:
  - $t_i^{(0)}, ZM_i$ - nejdříve možný začátek všech činností vycházejících z vrcholu *i*.
  - $t_j^{(1)}, KP_j$ - nejpozději přípustný konec všech činností končících ve vrcholu *j*.
- Pro celý projekt zavádíme:
  - $T_0$ – čas počátku projektu (zpravidla $T_0 = 0$).
  - $T$ – vypočítaná doba trvání projektu.

# Metoda kritické cesty (CPM)




Pro každou činnost platí následující vztahy:

$$ZM_{ij} = ZM_i \,,$$

$$KP_{ij} = KP_j \,.$$

# Metoda kritické cesty (CPM)

1) První fáze výpočtu v síťovém grafu probíhá od počátečního vrcholu sítě ke koncovému vrcholu. Počítají se při ní nejdříve možné začátky $ZM_j$ činností vycházejících z vrcholu *j* podle výrazu:

$$ZM_j = \max_i \{KM_{ij}\},$$

přičemž pro počáteční uzel sítě platí:

$$ZM_1 = T_0 \text{ (zpravidla } T_0\text{=0).}$$

# Metoda kritické cesty (CPM)

2) Druhá fáze výpočtu probíhá od koncového vrcholu sítě k počátečnímu vrcholu. Počítají se při ní nejpozději přípustné konce $KP_i$ činností končících ve vrcholu *i* podle výrazu:

$$KP_i = \min_j \{ZP_{ij}\},$$

přičemž pro koncový vrchol sítě platí:

$KP_n = ZM_n = T$, kde *T* je celková doba trvání projektu.

# Metoda kritické cesty (CPM)

3) V poslední fázi počítáme pro každou činnost celkovou rezervu činnosti podle vztahu:

$$CR_{ij} = KP_{ij} - KM_{ij} \, .$$

Pro kritické činnosti (tedy činnosti ležící na kritické cestě) platí:

$$CR_{ij} = 0.$$

# Plánování obsluhy vrcholů dopravní sítě z jednoho střediska

- Uvažujme situaci, kdy z jednoho vrcholu – **střediska (depa)** – provádíme rozvoz / svoz zásilek k *n* přepravcům / od *n* přepravců v rámci **atrakčního obvodu** – část dopravní sítě obsluhovaná z daného střediska.
- Omezujícím faktorem bude kapacita vozidla, kterým provádíme obsluhu.
- Optimalizačním kritériem bude celková vzdálenost ujetá vozidlem.

# Plánování obsluhy vrcholů dopravní sítě z jednoho střediska

- Uvažujme, že každého přepravce navštívíme právě jednou v rámci jedné jízdy, musí tedy platit:

$$\sum_{j=1}^{n} b_j \leq K,$$

kde *K* vyjadřuje kapacitu vozidla a $b_j \text{ pro } j = 1,2,\ldots,n$ jsou požadavky jednotlivých přepravců.

- Úkolem tedy bude naplánovat jízdu vozidla tak, aby se minimalizovala celková ujetá vzdálenost.

# Plánování obsluhy vrcholů dopravní sítě z jednoho střediska

- Uvedená úloha je úlohou o vyhledání minimální Hamiltonovy kružnice.
- Pro existenci HK v grafu – nutné a postačující podmínky.
- Může se stát, že v dané síti neexistuje HK, ale obsluha musí být naplánována.
- Proto se nepracuje s původním grafem, ale s grafem transformovaným.

# Plánování obsluhy vrcholů dopravní sítě z jednoho střediska

- V transformovaném grafu je množina vrcholů shodná s množinou vrcholů původního grafu, hrany spojující jednotlivé vrcholy představují minimální cesty, ohodnocení hran potom odpovídá vzdálenostem.
- Transformovaný graf je tedy grafem kompletním, HK v něm existuje vždy.
- Po vyhledání minimální HK v transformovaném grafu je nutno tuto HK zpětně transformovat do původní sítě.

# Littlův algoritmus

- Slouží k vyhledání minimální HK v zadané dopravní síti, tato úloha je rovněž nazývána **úlohou obchodního cestujícího**.
- Algoritmus je založen na Metodě větví a mezí:
  - Množina přípustných řešení se postupně dělí na menší podmnožiny.
  - Všechna řešení v dané podmnožině jsou charakterizována dolním odhadem ÚF (při minimalizaci).
  - Proces dělení množiny přípustných řešení znázorňujeme pomocí grafu zvaného strom.

# Littlův algoritmus

1) V každém řádku matice vzdáleností nalezneme minimální prvek a ten odečteme od všech prvků v daném řádku (v každém řádku získáme alespoň jeden nulový prvek). Postup na krok 2).
2) V každém sloupci matice vzdáleností, ve kterém se nenachází alespoň jeden nulový prvek, vyhledáme minimální prvek a ten odečteme od všech prvků v daném sloupci (v každém sloupci získáme alespoň jeden nulový prvek). Postup na krok 3).

# Littlův algoritmus

3) Výpočet dolního odhadu účelové funkce – DOÚF – jako součet všech hodnot, které jsme odečítali v jednotlivých řádcích, resp. sloupcích řešící matice. DOÚF udává hodnotu účelové funkce, pod kterou její hodnota určitě neklesne. Postup na krok 4).

# Littlův algoritmus

4) Ohodnotíme všechny nuly v řešící matici. Ohodnocení každé nuly je dáno součtem řádkového a sloupcového minima ze zbylých prvků (právě ohodnocovanou nulu tedy neuvažujeme). Nelze-li provést ohodnocení nul, postupujeme na krok 10), v opačném případě postupujeme na krok 5). Ohodnocení nuly představuje hodnotu, o kterou se zvýší DOÚF, nezařadíme-li uvedený prvek do HK.

# Littlův algoritmus

5) Z ohodnocených nul vybereme nulu s maximálním ohodnocením – větvící prvek. Existuje-li více nul s maximálním ohodnocením, volíme libovolnou z nich. Výběr prvku znamená zařazení příslušného úseku do HK. Současně začínáme pěstovat strom řešení, resp. pokračujeme ve stromu řešení a jdeme na krok 6).

# Littlův algoritmus

6) V aktuální řešící matici zrušíme ohodnocení nul, postup na krok 7).
7) Redukce aktuální řešící matice o příslušný řádek a sloupec (výběr prvku $c_{ij}$ znamená realizaci spojení vrcholu $u_i$ do vrcholu $u_j$, je tedy zřejmé, že z vrcholu $u_i$ už nemůžeme jít do jiného vrcholu než $u_j$ a naopak do vrcholu $u_j$ se nelze dostat z jiného vrcholu než $u_i$.

# Littlův algoritmus

8) Zákaz prvků v aktuální řešící matici, které umožňují vznik nepřípustného řešení (výběrem těchto prvků do HK by došlo k předčasnému uzavření kružnice).
9) Kontrola, zda máme v každém řádku, resp. Sloupci aktuální řešící matice alespoň jednu nulu. Pokud ne, odečteme v příslušném řádku, resp. sloupci hodnotu řádkového, resp. sloupcového minima a o tuto hodnotu zároveň zvýšíme DOÚF. Návrat na krok 4).

# Littlův algoritmus

10) Zbylé prvky v aktuální řešící matici určují úseky uzavírající HK. Vyskytuje-li se v úloze větev s nižší hodnotou DOÚF než je hodnota DOÚF naposledy vytvořeného řešení, vytvoříme novou řešící matici zohledňující podmínky v příslušné větvi řešícího stromu (zákaz příslušných prvků) a vracíme se na krok 1), v opačném případě postupujeme na krok 11).

# Littlův algoritmus

11) Naposledy vytvořené řešení je optimálním řešením, tedy minimální HK.

- HK jsme nalezli v transformovaném grafu, získané řešení je nutno transformovat do původní sítě. Po zpětné transformaci se může stát, že některé vrcholy v rámci obsluhy navštívíme víckrát. Dojde-li k tomu, že středisko v rámci naplánované jízdy navštívíme víckrát, nemusí být striktně splněn požadavek $\sum_{j=1}^{n} b_j \leq K$.

# Lokalizace havarijních středisek

- Je dána dopravní síť. Vrcholy v síti reprezentují místa výskytu negativních událostí, hrany budou reprezentovat komunikace.
- Graf je hranově a vrcholově ohodnocen, ohodnocení vrcholů představuje četnost (intenzitu, pravděpodobnost) vzniku negativních událostí, ohodnocení hran představuje délku komunikací.

# Lokalizace havarijních středisek

- Úkolem je umístit v této síti obslužné středisko (depo), ze kterého budou vyjíždět servisní týmy za účelem likvidace následků negativních událostí.
- Cílem řešení je rozhodnout o umístění střediska tak, aby náročnost obsluhy místa nejnáročnějšího na obsluhu z tohoto místa byla co nejnižší.

# Lokalizace havarijních středisek

- **Excentricita vrcholu *u*** (maximální obslužná vzdálenost, resp. vzdálenost k nejvzdálenějšímu vrcholu od vrcholu u):

$$ex(u) = \max_{v \in V} \{d(u,v)\}.$$

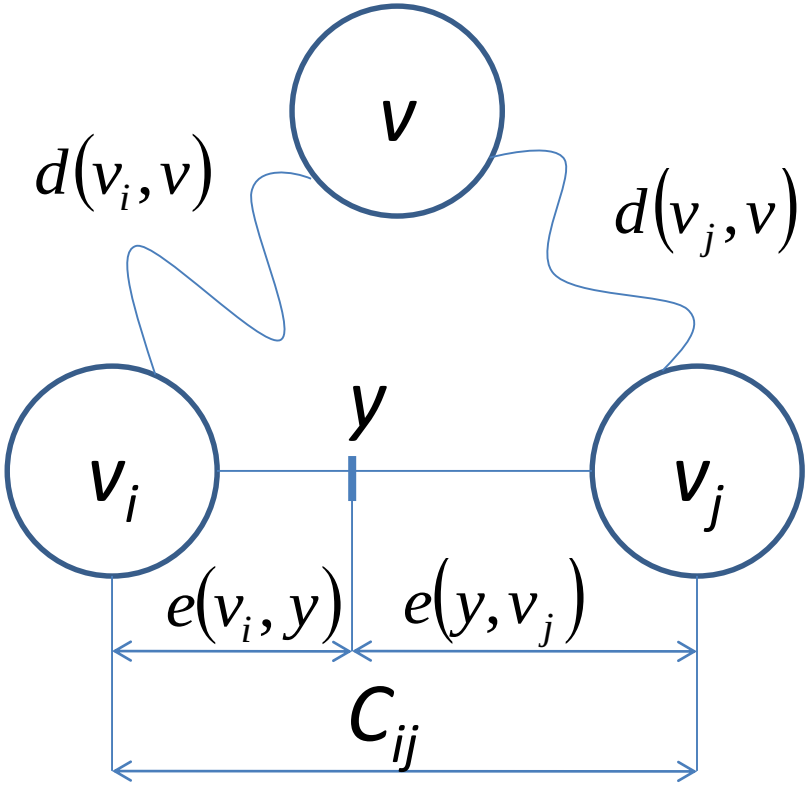
- **Vážená excentricita vrcholu *u*** (maximální obslužná náročnost):

$$ec(u) = \max_{v \in V} \{w(v) \cdot d(u,v)\}.$$

Vážená excentricita bude optimalizačním kritériem.

# Lokalizace havarijních středisek

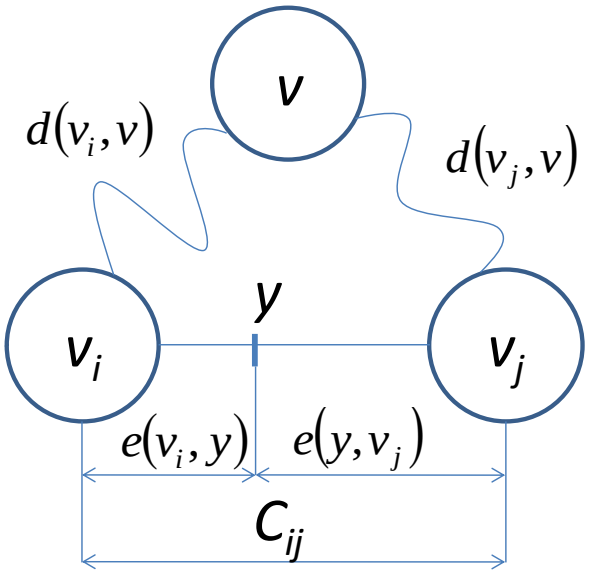
- **Vážená excentricita obecného místa *y* v síti**:




$$C_{ij} = e(v_i, y) + e(y, v_j)$$

# Lokalizace havarijních středisek

- **Vážená excentricita obecného místa *y* v síti**:




$$d(y,v) = \min\{e(v_i, y) + d(v_i, v), e(y, v_j) + d(v_j, v)\}$$

$$ec(y) = \max_{v \in V}\{w(v) \cdot d(y,v)\} =$$

$$= \max_{v \in V}\{w(v) \cdot \min\{e(v_i, y) + d(v_i, v), e(y, v_j) + d(v_j, v)\}\}$$

# Lokalizace havarijních středisek

- **Absolutně vzdálenostně optimálně umístěné depo (absolutní depo)** – depo považujeme za absolutně vzdálenostně optimálně umístěné v případě, leží–li v takovém místě sítě $y^* \in G$, pro jehož váženou excentricitu platí:

$$ec(y^*) = \min_{y \in G} \{ec(y)\}.$$

- Absolutní depo může tedy ležet jak ve vrcholu grafu, tak i na hraně grafu.

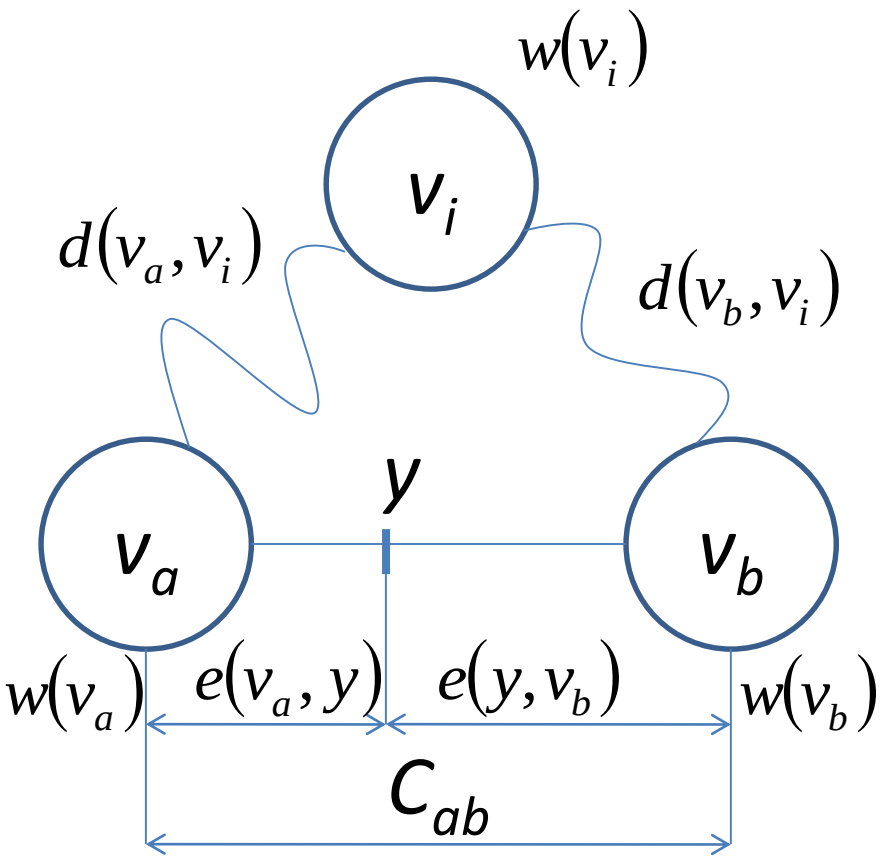
# Hakimiho algoritmus

- K vyhledání absolutního depa slouží Hakimiho algoritmus.
- Algoritmus vyhledává na každé hraně grafu, na které je možno depo umístit, místo (resp. místa) s minimální váženou excentricitou a ze všech těchto nalezených míst vybereme to místo, pro které bude vážená excentricita minimální, v tomto místě umístíme absolutní depo.

# Hakimiho algoritmus

- Hakimiho algoritmus zavádí $T_i$ a $T_i'$, což jsou funkční zápisy obslužných náročností. $T_i$ je zápisem obslužné náročnosti vrcholu $v_i$ při obsluze přes vrchol $v_b$ , $T_i'$ je zápisem obslužné náročnosti vrcholu $v_i$ při obsluze přes vrchol $v_a$.

# Hakimiho algoritmus




$$T_i = w(v_i) \cdot [e(y, v_b) + d(v_b, v_i)]$$

$$T_i' = w(v_i) \cdot [C_{ab} - e(y, v_b) + d(v_a, v_i)]$$

Zavedeme $e = e(y, v_b)$,

$e(v_a, y) = C_{ab} - e$:

$$T_i = w(v_i) \cdot [e + d(v_b, v_i)]$$

$$T_i' = w(v_i) \cdot [C_{ab} - e + d(v_a, v_i)]$$

# Hakimiho algoritmus

- Postup výpočtu:
  1. Výpočet provádíme samostatně pro každou hranu, na které je možno umístit depo. Pro všechny vrcholy, jejichž obsluhu provádíme, stanovíme funkční předpisy $T_i$ a $T_i'$, jež znázorníme graficky. Pro každou dvojici $T_i$ a $T_i'$ vybereme dolní průběh obslužných náročností (tedy stanovíme, pro jaká umístění $y$ je vhodnější obsluhovat vrchol $v_i$ přes vrchol $v_a$, resp. $v_b$.

# Hakimiho algoritmus

2. Ze všech dolních průběhů nalezneme maximální horní ohraničení, což odpovídá průběhu vážené excentricity (tedy průběhu optimalizačního kritéria) pro depo umístěné na hraně ($v_a$,$v_b$). Na této křivce vyhledáme nejníže položené body, tedy kandidáty na umístění depa ležící na právě vyšetřované hraně s nejnižší hodnotou optimalizačního kritéria.

# Hakimiho algoritmus

3. Vyšetříme-li tímto postupem všechny hrany, na které lze umístit depo, vybereme ze všech kandidátů na umístění depa toho s nejnižší hodnotou optimalizačního kritéria a do tohoto bodu umístíme absolutní depo. Existuje-li více bodů s nejnižší hodnotou optimalizačního kritéria, můžeme vybrat libovolný z nich.