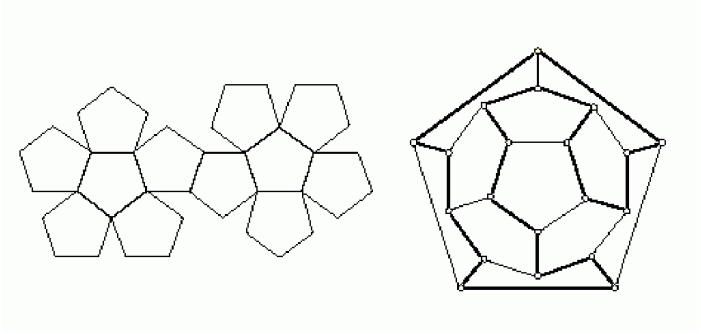
# Fundamentálne algoritmy teórie grafov pre ekonómov

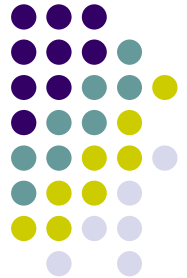
doc. Ing. Ivan Brezina, CSc.

Katedra operačného výskumu a ekonometrie

Fakulta hospodárskej informatiky EU v Bratislave



# Hľadanie minimálnej kostry grafu



*Kostra grafu* – taký strom grafu **G** = [**U**, **H**], pre ktorého podrgaf **G´** = [**U´**, **H´**] platí **U´**= **U** a **H´**$\subset$ **H** (faktor grafu).

*Kostra grafu* – každý súvislý graf má kostru.

*Minimálna kostra grafu* – taká kostra grafu, v ktorej je súčet ohodnotení hrán kostry minimálny.

# Hľadanie minimálnej kostry grafu

*Predpoklad* – graf s ohodnotenými hranami $k_{ij}$

*Cieľ* – nájsť takú kostru grafu, pre ktorú je súčet ohodnotení jej hrán minimálny

*Aplikácie* – budovanie elektrickej, resp. komunikačnej siete.

*Príklad* – vytvoriť kábelové spojenie medzi obcami tak, aby existovalo spojenie medzi všetkými obcami a celková dĺžka použitého kábelu by bolo minimálna.

# Hľadanie minimálnej kostry grafu
## Počet kostier grafu

*Počet kostier  grafu* – neplatí žiadne jednoduché univerzálne pravidlo.

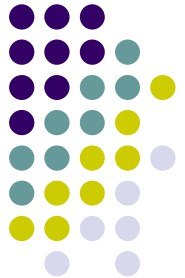
*Špeciálne prípady*:

*Počet kostier úplného grafu* – Cayleyho formula: určiť počet stromov na *n* vrcholoch - teda počet kostier grafu:

Pre každé $n \geq 3$ je počet stromov (kostier) na daných *n* vrcholoch rovný $n^{n-2}$.

*Počet kostier kružnice (cyklu)* – počet kostier grafu je rovný $|\mathbf{V}|$.

*Počet kostier stromu* – má práve jednu kostru (samotný graf je kostrou).



# Hľadanie minimálnej kostry grafu

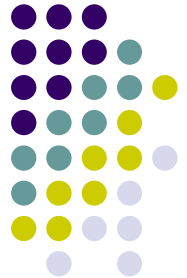
*Spôsoby riešenia:*

– riešenie úlohy lineárneho programovania,

– *špeciálne algoritmy*:

- Kruskalov algoritmus (1956) na báze Borůvkovho algoritmu (1926)
- Primov algoritmus (1957) - Jarníkov algoritmus
- Chu-Liu-Edmondsov algoritmus (1965, 1967)

# Hľadanie minimálnej kostry grafu
# Formulácia úlohy LP



*Predpoklady:*

*n* – počet všetkých uzlov

***C*** $=\{c_{ij}\}$ – matica priamych vzdialeností medzi *i*-tým a *j*-tym uzlom, pričom platí:

$$c_{ij} = \begin{cases} c_{ij}, & \text{ak existuje priama cesta medzi } i\text{-tým a } j\text{-tym uzlom,} \\ 0, & \text{ak } i = j, \\ +\infty, & \text{ak neexistuje priama cesta medzi } i\text{-tým a } j\text{-tym uzlom.} \end{cases}$$

# Hľadanie minimálnej kostry grafu
## Formulácia úlohy LP

*Premenné:*

$x_{ij} \in \{0,1\}$, $i,j = 1, 2, ..., n$

- hodnota 1, ak existuje spojenie medzi i-tým a j-tym vrcholom

$y_{ij} \in \langle 0, (n\text{-}1\rangle$, $i,j = 1, 2, ..., n$

- reprezentujú počet použití príslušnej cesty a zabezpečujú, aby existovala súvislá trasa od príslušného uzla k prvému uzlu

# Hľadanie minimálnej kostry grafu
# Formulácia úlohy LP

$$f(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \rightarrow \min$$

$$\sum_{j=2}^{n} x_{1j} = 0$$

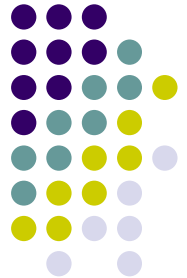
$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 2,3,\ldots n$$

$$\sum_{j=1}^{n} y_{ij} - \sum_{j=2}^{n} y_{ji} = 1, \qquad i = 2,3,\ldots n-1$$

$$0 \le y_{ij} \le (n-1)x_{ij}, \qquad i,j = 1,2,\ldots n$$

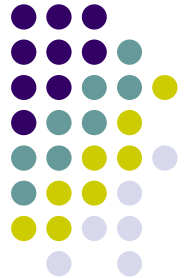
$$x_{ij} \in \{0,1\}, \qquad i,j = 1,2,\ldots n$$

# Hľadanie minimálnej kostry grafu Kruskalov algoritmus



- Kruskalov algoritmus najskôr zoradí hrany podľa ich ohodnotenia (od najmenšieho vzostupne) a následne pridáva hrany do grafu tak, aby nevznikol žiadny cyklus.
- Výsledná kostra grafu *G (n* uzlov) musí obsahovať *n*-1 hrán a nesmie obsahovať cyklus.
- Každý podgraf musí byť acyklický. Najväčší problém - testovanie acyklickosti.

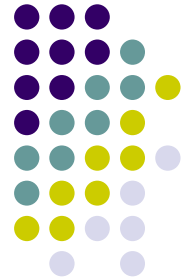
# Hľadanie minimálnej kostry grafu Kruskalov algoritmus



1. Zoradíme hrany podľa ich ohodnotenia (od najmenšieho vzostupne). Vyberieme z grafu $G$ dve hrany s minimálnym ohodnotením $k_{ij}$ a zaradíme ich do výslednej kostry.

2. Vyberieme ďalšie hrany s najnižším ohodnotením, ktoré netvoria s už vybranými cyklus (kružnicu).

3. Krok 2 opakujeme až po vybranie $n$ – 1 hrán.



# Hľadanie minimálnej kostry grafu
## Primov algoritmus

*Princíp* – konštruuje kostru postupným priberaním nových vrcholov do kostry.

Začína s prázdnou kostrou tak, že z grafu vyberie ľubovoľný vrchol $u_i$ – podgraf $G_1$.

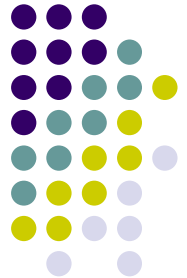
V každom kroku algoritmu vyberá hrany do kostry, ktoré spolu tvoria jeden strom, pričom zvyšné vrcholy sú navzájom izolované.

Do minimálnej kostry sa vyberá hrana s minimálnym ohodnotením, ktorá je z hrán vychádzajúcich z vrcholu $u_i$.

Vznikne podgraf, ktorý obsahuje vrchol $u_i$, minimálnu hranu a s ňou ďalší incidentný vrchol – podgraf $G_2$ (podgraf $G_i$).

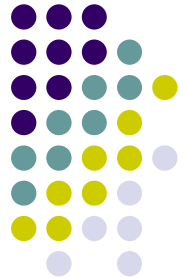
Podgraf $G_{i+1}$ sa vytvorí výberom hrany s najmenším ohodnotením vychádzajúcou z podgrafu $G_i$ a končiacou mimo neho. Pri konečných grafoch tento postup skončí pri *n* vrcholovom grafe po *n* krokoch. Výsledný podgraf je minimálna kostra grafu G.

# Hľadanie minimálnej kostry grafu
# Primov algoritmus



- Neprehľadáva systematicky všetky kostry grafu.
- Začína v ľubovoľnom vrchole a buduje strom.
- Najvyššiu prioritu majú hrany s najnižším ohodnotením.

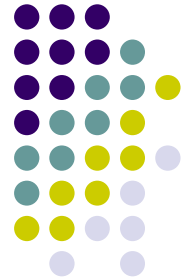
# Hľadanie minimálnej kostry grafu
# Primov algoritmus



1. Vyberieme ľubovoľný východiskový vrchol.

2. Vyberieme hranu s najnižším ohodnotením idúcou z východiskového vrcholu a zostrojíme kostru K1.

3. Prechádzame všetky vrcholy v kostre a pridáme vždy hranu s najnižším ohodnotením vychádzajúcu z niektorého z vrcholov. Ak by mala vzniknúť kružnica, hranu preskočíme.



# Hľadanie minimálnej kostry grafu
# Primov algoritmus

## Algoritmus:

Množina vrcholov $U$ rozdelená na podmnožinu $U_1$ (množina vybraných vrcholov) a $U_2$ (množina nevybraných vrcholov). Množina $H_1$ označuje množinu vybraných hrán.

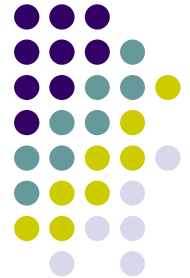
**1.** Inicializácia: súvislý ohodnotený graf **G** = [**U**, **H**]. Položíme $U_2 = U$ a $U_1 = \emptyset$. Nech $U_1 = \{u_i\}$, kde $u_i$ je ľubovoľný vrchol z $U$, $H_1 = \emptyset$.

**2.** Ak $U_1 = U$, koniec. Výstup: **G**($U_1$,$H_1$) je minimálna kostra grafu.

**3.** Výber hrany ($u_i$,$u_j$) z H s minimálnym ohodnotením tak, že $u_i$ je $U_1$ a $u_j$ nie je z $U_1$.

**4.** Pridanie $u_j$ do $U_1$ a ($u_i$,$u_j$) do $H_1$.

**5.** Návrat na krok 2.



# Hľadanie minimálnej kostry grafu
## Primov algoritmus – maticová varianta

Množina vrcholov $U$ rozdelená na podmnožinu $U_1$ (množina vybraných vrcholov) a $U_2$ (množina nevybraných vrcholov).

1. Položíme $U_2 = U$ a $U_1 = \emptyset$. Vyberieme ľubovoľný vrchol $u_0$ a zaradíme ho do $U_1 \Rightarrow U_2 = U_2 - \{u_0\}$ a $U_1 = \{u_0\}$.

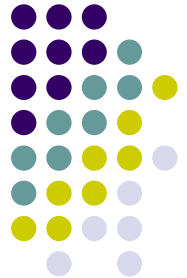
2. Vyberieme hranu s jedným vrcholom v množine $U_1$ a druhým v množine $U_2$ s minimálnym ohodnotením:

$$\min_{h_{ij}\,(u_i \in U_1, u_j \in U_2)} k_{ij} = k_{pq}$$

$\Rightarrow U_2 = U_2 - \{u_q\}$ a $U_1 = U_1 \cup \{u_q\}$.

3. Ak je $U_2 = \emptyset$, algoritmus končí nájdením minimálnej kostry grafu.

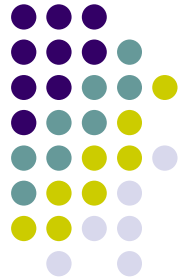
# Hľadanie najkratšej cesty



*Predpoklad* – graf s ohodnotenými hranami $k_{ij}$

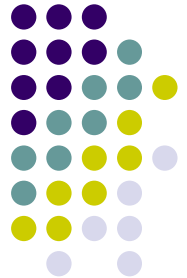
*Cieľ* – nájsť takú cestu v grafe, pre ktorú je súčet ohodnotení jej hrán minimálny

# Hľadanie najkratšej cesty



- úloha lineárneho programovania
  - špeciálny prípad priraďovacieho problému
- algoritmy na báze teórie grafov
  - Ford - Fulkersonov
  - Dantzigov
  - Dijkstrov
  - Tabourierov
- algoritmy na základe matíc susednosti
  - stromy minimálnych vzdialeností
  - operácia minimálneho sčítania
  - Floydov algoritmus

# Hľadanie najkratšej cesty



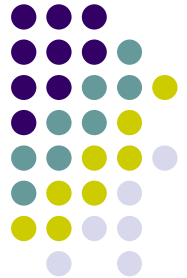
*Predpoklady:*

*n* – počet všetkých uzlov

***C*** *=*$\{c_{ij}\}$ – matica priamych vzdialeností medzi *i*-tým a *j*-tym uzlom, pričom platí:

$$c_{ij} = \begin{cases} c_{ij}, & \text{ak existuje priama cesta medzi } i\text{-tým a } j\text{-tym uzlom,} \\ 0, & \text{ak } i = j, \\ +\infty, & \text{ak neexistuje priama cesta medzi } i\text{-tým a } j\text{-tym uzlom.} \end{cases}$$

# Hľadanie najkratšej cesty Formulácia úlohy LP



*Premenné:*

$x_{ij} \in \{0,1\}$, $i,j = 1, 2, ..., n$

-hodnota 1, ak existuje spojenie medzi i-tým a j-tym vrcholom

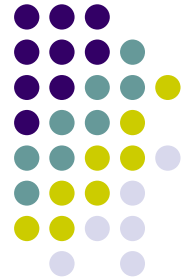
*Množiny:*

množiny $P_j$ a $R_i$, $i, j = 1, 2,...n$, obsahujúce indexy vrcholov, do ktorých (množina $P_j$), príp. z ktorých (množina $R_i$) existuje priama cesta:

$$P_j = \left\{ p \in \{1,2,...n\} / c_{pj} \neq 0 \wedge c_{pj} \neq \infty \right\}$$

$$R_i = \left\{ r \in \{1,2,...n\} / c_{ir} \neq 0 \wedge c_{ir} \neq \infty \right\}$$

# Hľadanie najkratšej cesty
# Formulácia úlohy LP



$$f(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \rightarrow \min$$

$$\sum_{i\in P_j} x_{ij} = \sum_{k\in R_j} x_{jk} \qquad j = 2,3,\ldots n-1$$

$$1+\sum_{i\in P_1} x_{i1} = \sum_{k\in R_1} x_{1k}$$

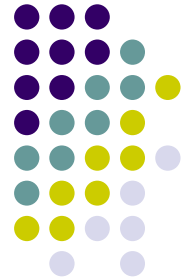
$$\sum_{i\in P_n} x_{in} = \sum_{k\in R_n} x_{nk} + 1$$

$$x_{ij} \in \{0,1\} \quad i, j = 1,2,\ldots n$$

$$P_j = \{p \in \{1,2,\ldots n\} / c_{pj} \neq 0 \wedge c_{pj} \neq \infty\}, j = 1,2,\ldots n$$

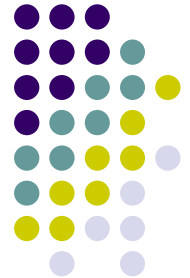
$$R_i = \{r \in \{1,2,\ldots n\} / c_{ir} \neq 0 \wedge c_{ir} \neq \infty\}, i = 1,2,\ldots n$$

# Hľadanie najkratšej cesty Dijkstrov algoritmus



1. Začiatočný vrchol $u_r$, kde $v_r = 0$.
2. Zápis vzdialeností pre priame cesty z vrcholu $u_r$ do ostatných vrcholov v sieti $d_j$, ak neexistuje, potom $M$.
3. Min $d_{jk}$, označíme ju * a zapíšeme do stĺpca min $j$ označenie príslušného stĺpca. Do riadka D zapíšeme hodnotu $d_{jk}$.
4. Ohodnotíme vrcholy, do ktorých vedie priama cesta z vrcholu $u_{jk}$, pričom aktualizujeme hodnoty na základe vzťahu:
   - ak platí $d_{jk} + c_{jk,j} < d_j$, potom $d_j$ nahradené $d_{jk} + c_{jk,j}$
   - ináč ostáva $d_j$
5. Najkratšia cesta do daného vrcholu je $v_j = v_\pi + c_{\pi j}$
6. Návrat na 3.

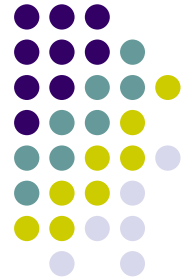
# Hľadanie najkratšej cesty Dantzigov algoritmus



1. Pre ľubovoľné $u_i$ položíme $v_i = 0$.
2. Určíme pre i a j $v_r = \min(v_i + c_{ij})$

   hranu, ktorá tvorí minimálne spojenie, výrazne označíme a z celej tabuľky vylúčime všetky ostatné hrany, ktoré smerujú do uzla $u_r$.
3. Podľa bodu 2 pokračujeme, kým neohodnotíme všetky vrcholy v sieti, t. j. kým nevypočítame aj hodnotu $v_n$.
4. Hodnota $v_n$ udáva dĺžku najkratšej cesty z východiskového vrcholu do vrcholu $u_n$.

# Hľadanie najkratšej cesty
## Floydov algoritmus



1. $k = 0$, zápis $\mathbf{D^k} = \{d_{ij}^k\}$ do tabuľky
2. $k = k + 1$, výpočet hodnôt $d_{ij}^k$

$$d_{ij}^k = \begin{cases} \text{ak } i = j & 0 \\ \text{ak } d_{ij}^k = M \text{ a aj } d_{ik}^k + d_{kj}^k = M \text{ pre } i, j \neq k & M \\ \text{ináč} & \min\left(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}\right) \text{ a označíme } * \end{cases}$$

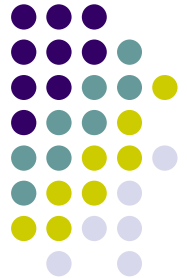
3. Opakujeme krok 2, pokiaľ $k \neq n$.
4. Matica $\mathbf{K} = \{k_{ij}\}$ :

- ak $d_{ij}^n = d_{ij}^0$ $\quad k_{ij} = 0$
- ak $i = j$ $\quad k_{ij} = -$

- ináč – najvyššia iterácia $k$, pri ktorej je v príslušnom poli * označujúca zmenu hodnoty $d_{ij}^k$

5. $\theta_{iK} + \theta_{Kj} \quad \leq d_{iK} + d_{Kj}$

# Hľadanie najkratšej cesty
## Operácia minimálneho sčítania



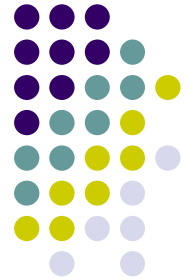
Počet iterácií *s*: $n - 1 \leq 2^s$

1. Matica susednosti $\mathbf{D^0}$, $k = 1$.
2. Matica $\mathbf{D^k}$, prvky pre $k$ na základe vzťahu

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

3. Ak platí $k < s$, $k = k + 1$ a návrat na krok 2.

# Hľadanie cesty s najväčšou pravdepodobnosťou



*Predpoklady:*

$n$ – počet všetkých uzlov

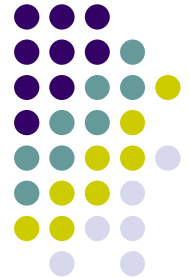
$\boldsymbol{P} = \{p_{ij}\}$ – matica pravdepodobností prechodu medzi $i$-tým a $j$-tym uzlom, pričom platí:

$$0 \leq p_{ij} \leq 1$$

resp.

$$0 < p_{ij} \leq 1$$

# Hľadanie cesty s najväčšou pravdepodobnosťou
# Upravený Dantzigov algoritmus

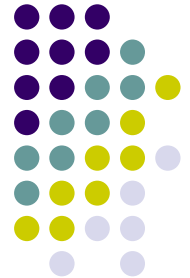


1. Pre ľubovoľné $u_i$ položíme $v_i = 1$.
2. Určíme pre i a j $v_r = \max(v_i \,.\, p_{ij})$

   hranu, ktorá tvorí spojenie, výrazne označíme a z celej tabuľky vylúčime všetky ostatné hrany, ktoré smerujú do uzla $u_r$.
3. Podľa bodu 2 pokračujeme, kým neohodnotíme všetky vrcholy v sieti, t. j. kým nevypočítame aj hodnotu $v_n$.
4. Hodnota $v_n$ udáva dĺžku cesty s najväčšou pravdepodobnosťou z východiskového vrcholu do vrcholu $u_n$.

# Hľadanie cesty s najväčšou pravdepodobnosťou

## Upravený algoritmus operácie minimálneho sčítania



Počet iterácií *s*: $n - 1 \leq 2^s$

1. Matica $\mathbf{D^0}$ s prvkami:

$$d_{ij}^0 = \begin{cases} 1 & \text{pre } i = j \\ p_{ij} & \text{pre } h_{ij} \in H \\ 0 & \text{inák} \end{cases}$$

$k = 1$.

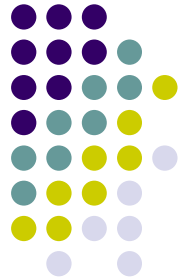
2. Matica $\mathbf{D^k}$, prvky pre $k$ na základe vzťahu

$$d_{ij}^{k} = \max_{k} \left(d_{ik}^{k-1} \,.\, d_{kj}^{k-1}\right)$$

3. Ak platí $k < s$, $k = k + 1$ a návrat na krok 2.

# Hľadanie cesty s minimálnou pravdepodobnosťou

## Upravený algoritmus operácie minimálneho sčítania



Počet iterácií *s*: $n - 1 \leq 2^s$

1. Matica $\mathbf{D^0}$ s prvkami:

$$d_{ij}^0 = \begin{cases} 1 & \text{pre } i = j \\ p_{ij} & \text{pre } h_{ij} \in H \\ 0 & \text{inák} \end{cases}$$

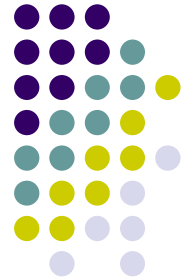
$k = 1$.

2. Matica $\mathbf{D^k}$, prvky pre $k$ na základe vzťahu

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

3. Ak platí $k < s$, $k = k + 1$ a návrat na krok 2.

# Hľadanie cesty s maximálnou priepustnosťou

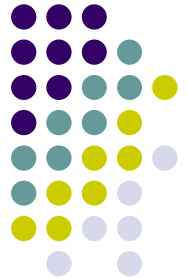


*Predpoklady:*

*n* – počet všetkých uzlov

***K*** $=\{k_{ij}\}$ – matica priechodnosti medzi *i*-tým a *j*-tym vrcholom

# Hľadanie cesty s maximálnou priepustnosťou
# Upravený Dantzigov algoritmus

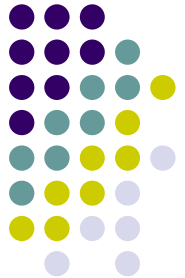


1. Pre ľubovoľné $u_i$ položíme $v_i = \max k_{1j}$.
2. Určíme pre i a j $v_r = \max\{\min(v_i ; k_{ij})\}$

   hranu, ktorá tvorí spojenie, výrazne označíme a z celej tabuľky vylúčime všetky ostatné hrany, ktoré smerujú do uzla $u_r$.
3. Podľa bodu 2 pokračujeme, kým neohodnotíme všetky vrcholy v sieti, t. j. kým nevypočítame aj hodnotu $v_n$.
4. Hodnota $v_n$ udáva dĺžku cesty s najväčšou priepustnosťou z východiskového vrcholu do vrcholu $u_n$.

# Hľadanie cesty s maximálnou priepustnosťou

## Upravený algoritmus operácie minimálneho sčítania



Počet iterácií *s*: $n - 1 \leq 2^s$

1. Matica $\mathbf{D^0}$ s prvkami:

$$d_{ij}^0 = \begin{cases} M & \text{pre } i = j \\ k_{ij} & \text{pre } h_{ij} \in H \\ 0 & \text{inák} \end{cases}$$

$k = 1$.

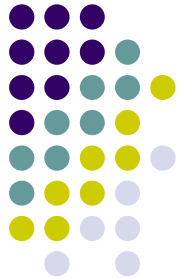
2. Matica $\mathbf{D^k}$, prvky pre $k$ na základe vzťahu

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

3. Ak platí $k < s$, $k = k + 1$ a návrat na krok 2.

# Hľadanie cesty s minimálnou priepustnosťou

## Upravený algoritmus operácie minimálneho sčítania



Počet iterácií $s$: $n - 1 \leq 2^s$

1. Matica $\mathbf{D^0}$ s prvkami:

$$d_{ij}^0 = \begin{cases} 1 & \text{pre } i = j \\ p_{ij} & \text{pre } h_{ij} \in H \\ 0 & \text{ináč} \end{cases}$$

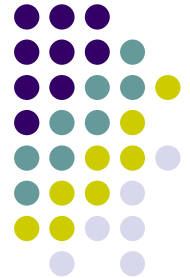
$k = 1$.

2. Matica $\mathbf{D^k}$, prvky pre $k$ na základe vzťahu

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

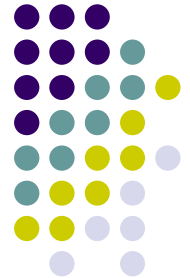
3. Ak platí $k < s$, $k = k + 1$ a návrat na krok 2.

# Hľadanie najkratšej okružnej cesty ako ÚLP



- $n$ – počet vrcholov
- $d_{ij}$ ($i, j = 1, 2,..., n$) – najkratšie vzdialenosti medzi jednotlivými vrcholmi
- $x_{ij}$ – bivalentné premenné

# Hľadanie najkratšej okružnej cesty ako ÚLP



$$f(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \rightarrow \min$$

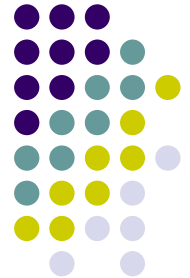
$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1,2,\ldots n, \quad i \neq j$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1,2,\ldots n, \quad i \neq j$$

$$y_i + y_j - nx_{ij} \leq n-1 \quad i = 2,3,\ldots n \quad j = 2,3,\ldots n \quad i \neq j$$

$$x_{ij} \in \{0,1\}, \quad i, j = 1,2,\ldots n$$

# Hľadanie najkratšej okružnej cesty Algoritmus najbližšieho suseda



1. $k = 1$, $I = \{1, 2, ..., n\}$, $i = i_1 = 1$ a $J = I - \{i_1\} = \{2, 3, ..., n\}$.
2. Ak $k = n$, prechod ku kroku 3.

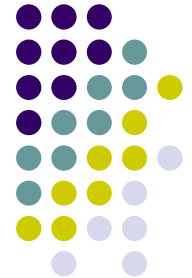
   ak $k < n$, nájdeme v $i$-tom riadku matice **C** v množine $J$ indexov najmenší prvok (označíme ako $q$), t. j. pre $c_{iq} \in$ **C**

$$\min_{q \in J} \{c_{iq}\}$$

   $k = k + 1$, $i = i_k = j$ a $J = J - \{j\}$.

   opakovanie kroku 2.
3. $i_{n+1} = 1$, riešenie je dané postupnosťou uzlov $i_1, i_2, ..., i_n, i_{n+1}$.

# Hľadanie najkratšej okružnej cesty
# Algoritmus najvýhodnejšieho suseda



1. Výpočet matice frekvencií E s prvkami pre úplnú sieť

a) nesymetrickú $e_{ij} = (n-1).d_{ij} - \sum_{k=1}^{n} d_{ik} - \sum_{q=1}^{n} d_{qj} - d_{ji}$

b) symetrickú $e_{ij} = (n-2).d_{ij} - \sum_{k=1}^{n} d_{ik} - \sum_{q=1}^{n} d_{qj}$ $\quad k \neq j \quad q \neq i$

2. $k = 1$, $I = \{1, 2, ..., n\}$, $i = i_1 = 1$ a $J = I - \{i_1\} = \{2, 3, ..., n\}$.
3. Ak $k = n$, prechod ku kroku 4.

   ak $k < n$, nájdeme v $i$-tom riadku matice **E** v množine $J$ indexov najmenší prvok (označíme ako $q$), t. j. min $e_{iq} \in$ **E**, pre $q \in J$

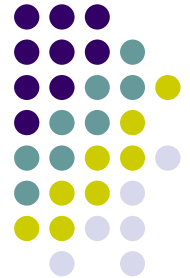
   $k = k + 1$, $i = i_k = j$ a $J = J - \{j\}$, opakovanie kroku 3.

4. $i_{n+1} = 1$, riešenie je dané postupnosťou uzlov $i_1, i_2, ..., i_n, i_{n+1}$.

# Hľadanie najkratšej okružnej cesty
## Littleho metóda

- Optimalizačná metóda – hľadá optimálne riešenie
- Metóda patrí do skupiny metód vetiev a hraníc
- Kombinatorická metóda – preskúmava každé riešenie, ktorých je n!

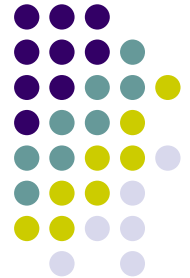
# Hľadanie najkratšej okružnej cesty
## Littleho metóda



**Littleho metóda** patrí do skupiny metód vetiev a hraníc.

Ak existuje *n* miest, cez ktoré má prejsť obchodný cestujúci a ak sú všetky mestá navzájom spojené komunikáciami (kompletný graf), potom existuje *n*! rôznych okružných ciest.

# Hľadanie najkratšej okružnej cesty
## Littleho metóda – algoritmus (1)



1. Redukcia matice vzdialeností. Od každej definovanej vzdialenosti v každom riadku odpočítame najmenšiu vzdialenosť $p_i$ ($i$ = 1, 2, ..., $m$), čím dostaneme prvú redukciu matice. Ak v jednotlivých stĺpcoch sa nenachádza aspoň jedna nula, odpočítame najmenšiu redukovanú vzdialenosť v každom stĺpci $q_j$ ($j$ = 1, 2, .., $m$). Dostaneme redukovanú maticu prepravných sadzieb (vzdialenosti medzi mestami)

$$\mathbf{D}^{(0)} = (d_{ij}^{(0)} - p_i^{(0)} - q_j^{(0)})$$

kde (0) charakterizuje krok výpočtu,

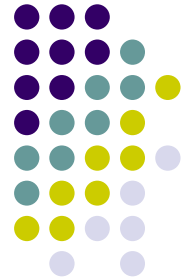
$p_i$ - minimálna prepravná sadzba v $i$-tom riadku,

$q_j$ - minimálna prepravná sadzba v $j$-tom stĺci v redukovanej matici.

Suma minimálnych čísel pre riadky a stĺpce tvorí dolnú hranicu vzdialeností pri hľadanej najkratšej okružnej cesty

$$S^{(0)} = \sum_{i=1}^{m} p_i^{(0)} + \sum_{j=1}^{m} q_j^{(0)}$$

# Hľadanie najkratšej okružnej cesty
## Littleho metóda – algoritmus (2)



2. Matica $\mathbf{D}^{(0)}$ obsahuje polia s nulovou prepravnou sadzbou. Ku každej z týchto 0 priradíme číslo $\rho_{ij}$, ktoré sa rovná súčtu najnižších vzdialeností v danom riadku a stĺpci, ktorých indexy prislúchajú k danému nulovému poľu. Určíme

$$\rho_{rs}^{(1)} = \max \rho_{ij}^{(1)}$$

Získame dve množiny okružných ciest:

$T_1^{(1)}$ podmnožina všetkých okružných ciest, ktorá neobsahuje cestu $(r, s)$

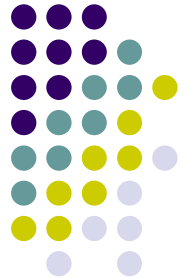
$T_2^{(1)}$ podmnožina všetkých okružných ciest, ktorá obsahuje cestu $(r, s)$.

Potom $S_1^{(1)} = S^{(0)} + \rho_{rs}^{(1)}$ je dolná hranica vzdialeností pre podmnožinu $T_1^{(1)}$.

Ak nepoužijeme cestu $(r, s)$, musíme použiť cestu $(r, j)$ j $\neq$ s a $(i, s)$ i $\neq$ r. Minimálna vzdialenosť je však určená pomocou $\rho_{rs}^{(1)}$ .

3. Zostavíme novú maticu $\mathbf{D}^{(0)\prime}$, v ktorej vyškrtneme $r$-tý riadok a $s$-tý stĺpec. Súčasne musíme uzavrieť cestu $(s, r)$, resp. iné cesty, ktoré by viedli k uzavretiu cesty a teda k vzniku cyklov.

# Hľadanie najkratšej okružnej cesty
## Littleho metóda – algoritmus (3)



4. Vykonáme redukciu tejto matice, čím dostaneme novú maticu **D(1)** a $p_i^{(1)}$ a $q_j^{(1)}$. Vypočítame

$$S^{(1)} = \sum_{i=1}^{m} p_i^{(1)} + \sum_{j=1}^{m} q_j^{(1)}$$

a určíme dolnú hranicu pre podmnožinu $T_2^{(1)}$

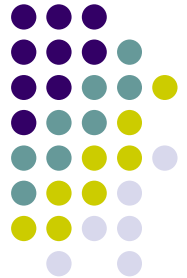
$$S_2^{(1)} = S^{(0)} + S^{(1)}$$

Vypočítame čísla $\rho_{ij}$ pre nulové polia a určíme novú hodnotu

$$\rho_{rs}^{(2)} = \max \rho_{ij}^{(2)}$$

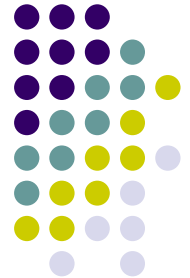
Vo výpočte pokračujeme tým smerom, kde je redukcia ($\rho$ alebo $\sum pi + \sum q_j$ ) menšia, resp. kde $S_1^{(1)}$ a $S_2^{(1)}$ sú menšie. Vrátime sa ku kroku 2 a pokračujeme vo výpočte krokom 3 a 4.

# Hľadanie najkratšej okružnej cesty
## Littleho metóda – algoritmus (4)



5. Ak pri vyškrtnutí riadku a stĺpca zostane iba jedna cesta, zaradíme ju do okružnej cesty.
Zodpovedajúca hodnota sa pripočíta k $S^{(n-1)}$ a získa sa hodnota účelovej funkcie.

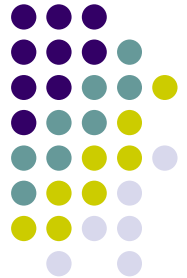
# Hľadanie najkratšej okružnej cesty hranami siete
# Úloha čínskeho poštára



- cieľ - nájsť najkratšiu cestu z východiskového uzla $u_1$ pol všetkých hranách siete s návratom do východiskového uzla $u_1$
- *Eulerov cyklus* - cyklus, ktorý prechádza po každej hrane grafu práve jedenkrát
- *Eulerovský graf* - obsahuje Eulerov cyklus, súvislý graf, párne stupne všetkých uzlov v grafe (z jedného uzla vychádza párny počet hrán ($U^+$))

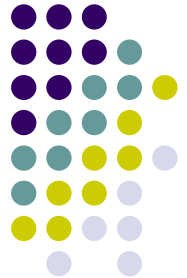
# Hľadanie najkratšej okružnej cesty hranami siete

## Algoritmus úlohy čínskeho poštára



1. Pre všetky dvojice uzlov z $U^-$ - najkratšie cesty cez všetky uzly. Nájdenie najkratšej vzdialenosti medzi všetkými vrcholmi v sieti - matica **D** = ($d_{ij}$) s rozmerom $U^-$ . $U^-$ najkratších vzdialeností $d_{ij}$ pre $u_i$, $u_j$ z množiny $U^-$.
2. Úloha minimálneho párovania na množine $U^-$ vzhľadom na hodnoty **D** = ($d_{ij}$). Zostavenie párov uzlov z množiny $U^-$, pre ktoré je súčet dĺžok najkratších vzdialeností $d_{ij}$ minimálny.
3. K pôvodnému grafu pridáme umelé hrany, paralelné k hranám, ktoré zodpovedajú hranám najkratších ciest minimálneho párovania získaných. Umelé hrany majú rovnaké ohodnotenia ako hrany paralelné. Vzniknutý graf je eulerovský.
4. V upravenom grafe nájdeme Eulerov cyklus. Dĺžka cyklu je súčtom ohodnotení všetkých hrán v grafe (aj umelých).

# Hľadanie maximálneho toku v sieti



*Predpoklady:*

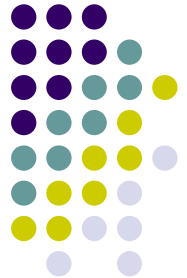
*n* – počet všetkých uzlov

*Premenné:* $x_{ij} \geq 0$, $i,j = 1, 2, ..., n$

$\boldsymbol{K} = \{k_{ij}\}$ – matica maximálneho prietoku medzi *i*-tým a *j*-tym uzlom, pričom platí:

$$x_{ij} \leq k_{ij}$$

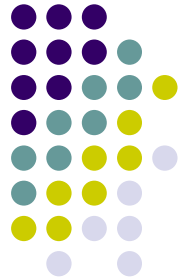
# Hľadanie maximálneho toku v sieti



*Predpoklady:*

1. V sieti sa prepravuje homogénny substrát.
2. Vrchol $u_1$ je vstupným vrcholom, vrchol $u_n$ výstupným vrcholom, ostatné vrcholy sú tranzitné.
3. Ohodnotenia $k_{ij}$ predstavujú maximálnu priepustnosť (kapacitu) z *i*-tého do *j*-teho vrcholu.
4. Tok po hrane od $u_i$ do $u_j$ je označený $x_{ij}$, opačným smerom $x_{ji}$, pričom $x_{ij} = - x_{ji}$.
5. Platí $x_{ij} \leq k_{ij}$.
6. Ak platí $x_{ij} = k_{ij}$, ide o nasýtenú hranu, ak platí $x_{ij} < k_{ij}$, ide o nenasýtenú hranu, rozdiel $k_{ij} - x_{ij}$ je nevyužitá kapacita hrany.
7. Platí: $\sum_j x_{1j} > 0;\ \sum_i x_{ni} < 0;\ \sum_j x_{zj} - \sum_i x_{iz} = 0\ (z \neq 1, n)$

# Hľadanie maximálneho toku v sieti
# Formulácia úlohy ÚLP

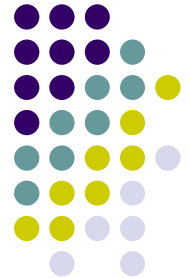


$$f(\mathbf{x}) = \sum_{j=1}^{n} x_{1j} - \sum_{j=1}^{n} x_{j1} \rightarrow \max$$

$$\sum_{j=1}^{n} x_{ij} = \sum_{j=1}^{n} x_{ji} \qquad j = 2,3,\ldots n-1$$

$$0 \le x_{ij} \le k_{ij} \quad i, j = 1,2,\ldots n$$

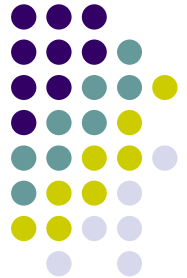
# Hľadanie maximálneho toku v sieti
# Algoritmus pre rovinné siete



1. Nájdenie „najsevernejšej“ (najvrchnejšej) cesty v sieti tak, že sú vybrané hrany „najviac vľavo“ od hrany, ktorá do vrcholu vstúpila. Nájdená cesta má vlastnosť, že ostávajúce rany a vrcholy ležia vo vnútri plochy v rovine ohraničenej nájdenou cestou. Ak takáto cesta neexistuje, výpočet končí.
2. Priepustnosť (tok) nájdenej cesty je rovná minimálnej kapacite z kapacít hrán tejto cesty, označenie k(C).
3. Redukcia siete tak, že na nájdenej ceste odstránime hrany s minimálnou kapacitou hrany a kapacity ostatných ostávajúcich hrán na tejto ceste sú znížené o túto minimálnu hodnotu.
4. Návrat na 1.

Celková hodnota maximálneho toku: súčet tokov jednotlivých ciest

# Hľadanie maximálneho toku v sieti Všeobecný (Ford-Fulkersonov) algoritmus



Tok *F* nazveme plným (kompletným), ak každá cesta medzi vstupným a výstupným vrcholom obsahuje aspoň jednu nasýtenú hranu.

Ak tok nie je plný, hľadáme nenasýtenú cestu, ktorej priepustnosť určuje priepustnosť toku v *k*-tom kroku:

$$x^k = \min_{i,j}(k_{ij} - x_{ij})$$

Celkový tok *F* v *k*+1-om kroku je

$$F^{k+1} = F^k + x^k$$

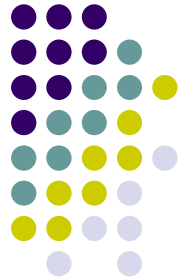
Hodnoty jednotlivých hrán:

$x_{ij}^{k+1} = x_{ij}^k + x^k$ pre nenasýtené hrany na ceste v smere od vstupu k výstupu

$x_{ij}^{k+1} = x_{ij}^k - x^k$ pre nenasýtené hrany na ceste v smere od výstupu k vstupu

$x_{ij}^{k+1} = x_{ij}^k$ pre ostatné hrany

# Hľadanie maximálneho toku v sieti Všeobecný (Ford-Fulkersonov) algoritmus (značkovací)



1. Označenie vstupného vrcholu indexom 0.
2. Označenie každého susedného vrcholu *j* už označeného vrcholu *i* na základe pravidiel:

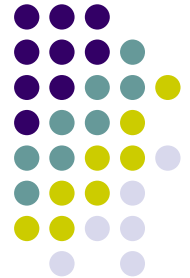
- hrana $h_{ij}$ je nenasýtená ($x_{ij} < k_{ij}$), potom označenie vrcholu *j* symbolom $+i$,
- existuje kladný tok z vrcholu *j* do vrcholu *i* ($x_{ji} > 0$), potom označenie vrcholu *j* symbolom $-i$,

3. Ak je možné podľa kroku 2 označiť výstupný vrchol, existuje nenasýtená cesta medzi vstupným a výstupným vrcholom určená postupnosťou symbolov jednotlivých uzlov, ak nie krok 5. Tok na tejto ceste možno zväčšiť:

- $x_{ij}^{k+1} = x_{ij}^{k} + x^{k}$ pre nenasýtené hrany na ceste v smere od vstupu k výstupu
- $x_{ij}^{k+1} = x_{ij}^{k} - x^{k}$ pre nenasýtené hrany na ceste v smere od výstupu k vstupu
- $x_{ij}^{k+1} = x_{ij}^{k}$ pre ostatné hrany

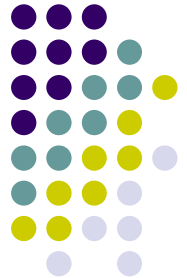
4. Návrat na krok 2.
5. Koniec, nájdený tok je maximálny.

# Hľadanie minimálneho toku v sieti



- cestná sieť - sieť, kde vrcholy $u_i$ a $u_j$ sú spojené hranami $h_{ij}$, ktoré sú ohodnotené hodnotami – minimálnymi požadovanými tokmi $k_{ij}$
- cieľ - nájsť cestu medzi začiatočným vrcholom $u_1$ a koncovým vrcholom $u_n$ s minimálnym počtom prepravných prostriedkov

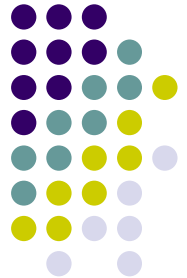
# Hľadanie minimálneho toku v sieti



*Predpoklady:*

1. V sieti sa prepravuje homogénny substrát.
2. Vrchol $u_1$ je vstupným vrcholom, vrchol $u_n$ výstupným vrcholom, ostatné vrcholy sú tranzitné.
3. Ohodnotenia $k_{ij}$ predstavujú minimálny požadovaný tok (kapacitu) z *i*-tého do *j*-teho vrcholu.
4. Tok po hrane od $u_i$ do $u_j$ je označený $x_{ij}$, opačným smerom $x_{ji}$, pričom $x_{ij} = - x_{ji}$.
5. Platí $x_{ij} \geq k_{ij}$.
6. Transformácia úlohy na úlohu hľadania maximálneho toku.

# Hľadanie minimálneho toku v sieti
# Algoritmus na nájdenie minimálneho toku v sieti



1. Nájdenie prípustného toku *P* tak, aby na každej hrane platil vzťah (nie je problém s hľadaním, pretože neexistujú horné ohraničenia hrán)

$$x_{ij}^{P} \geq k_{ij}$$

2. Určenie nových kapacít $k_{ij}' = x_{ij}^{P} - k_{ij}$.
3. Na základe kapacít $k_{ij}'$ nájdenie hodnoty maximálneho toku na základe príslušných algoritmov. Hodnota maximálneho toku nech je *M*. Potom pre toky na každej hrane musí platiť

$$x_{ij}^{M} \leq k_{ij}'$$

4. Hodnota minimálneho toku v sieti

$$F_{min} = P - M$$

a pre toky každej hrany platí

$$x_{ij} = x_{ij}^{P} - x_{ij}^{M} \geq x_{ij}^{P} - k_{ij}' = k_{ij}$$