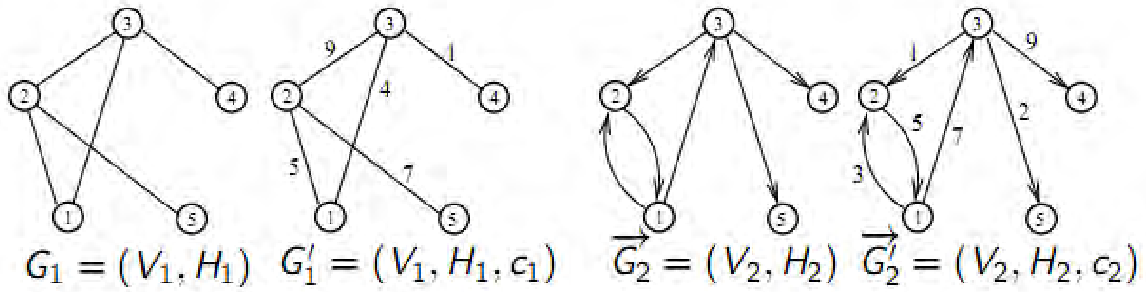
1. Základné pojmy teórie grafov. Graf, digraf, ďalšie štruktúry TG, podgraf, úplný graf, stupeň vrchola, počet vrcholov nepárneho stupňa, izomorfizmus, komplementárnosť, reprezentácia grafov.

- **Graf** - nazveme usporiadanú dvojicu vrcholov a hrán $G = (V, H)$, kde V je neprázdna konečná množina a H je množina neusporiadaných dvojíc typu $\{u, v\}$ takých, že $u,v \in V$ a u != v
- **Digraf** - nazveme usporiadanú dvojicu vrcholov a o. hrán $\overrightarrow{G} = (V, H)$, kde V je neprázdna konečná množina a H je množina **usporiadaných** dvojíc typu (u, v) takých, že $u,v \in V$ a u != v
  Prvky množiny V nazývame vrcholmi a prvky množiny H hranami grafu G / orientovanými hranami digrafu $\overrightarrow{G}$.
- **Diagram** - grafická reprezentácia grafu v priestore (Vrchol - bod; Hrana - čiara)
- **Podgraf** - Graf $G' = (V', H')$ je podgrafom grafu $G =(V, H)$, ak platí, že jeho $\mathbf{V' \subseteq V}$ **a** $\mathbf{H' \subseteq H}$. V tomto prípade budeme písať $\mathbf{G' \subseteq G}$.
  - Digraf ma podgraf rovnako
- **Faktorový podgraf -** graf $G' = (V', H')$ je faktorovým podgrafom grafu $G = (V, H)$, ak platí $\mathbf{V' = V}$ a $H' \subseteq H$.
  - Analogicky definujeme faktorový podgraf digrafu $\overrightarrow{G}$.
- **Úplný graf -** Graf nazveme úplným grafom, **ak množina H obsahuje všetky možné dvojice** typu $\{u, v\}$, kde $u, v \in V$ a u != v.
  - Úplný graf o n vrcholoch budeme značiť **Kn**.
- **Stupeň vrchola *v*** (deg(*v*)) **-** V grafe $G = (V, H)$ **je počet incidentných hrán s vrcholom *v***.
  Digraf - odeg(*v*) / ideg(*v*) - počet hran vychádzajúcich / uchádzajúcich do/z vrchola *v*.
- Počet vrcholov nepárneho stupňa v ľubovoľnom grafe $G = (V, H)$ je párny.
- **Izomorfizmus grafov -** Graf $G = (V, H)$ je izomorfný s grafom $G' = (V', H')$, **ak existuje také vzájomne jednoznačné zobrazenie** $f : V \leftrightarrow V'$, že pre každú dvojicu vrcholov $u, v \in V$ platí: $\{u, v\} \in H$ práve vtedy, keď $\{f(u), f(v)\} \in H'$. {**Grafy majú rovnaký počet vrcholov a aj ich prepojenie hrán**}
- **Pravidelný graf stupňa k -** je taký graf $G = (V, H)$, v ktorom má každý vrchol $v \in V$ **stupeň k**.
- **Komplementárne -** Grafy nazveme komplementárne, ak **V = V** a pre každú dvojicu vrcholov $u, v \in V$ takých, že u != v, **platí: $\{u, v\} \in H$** práve vtedy, keď **$\{u, v\}$!$\in$ H**. {**Nesmú obsahovať spoločné hrany**}

Reprezentácia grafov a digrafov

- **Reprezentácia diagramom grafu**



- **Reprezentácia množinami vrcholov a hrán**

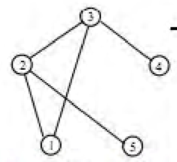
  Nexh $V_1 = \{1,2,3,4,5\}$, $H_1 = \{ \{1,2\}, \{1,3\}, \{2,3\}. \{2,5\}, \{3,4\}\}$.
  Množinami V_1 a H_1 je jednoznačne určený graf $G_1 = (V_1 ,H_1)$.

  Podobne nech $V_2 = \{1,2,3,4,5\}$ a
  $H_2 = \{ (1,2), (1,3), (2,1), (3,2), (3,4), (3,5) \}$,
  Potom množinami V_2, H_2 je jednoznačne určený digraf $G_2 = (V_2, H_2)$.

- **Maticou priľahlosti**

Matica priľahlosti $\mathbf{M} = (m_{ij})$ je štvorcová matica typu $n \times n$, kde $n = |V|$ je počet vrcholov grafu, resp. digrafu G, ktorej prvky sú definované nasledovne:

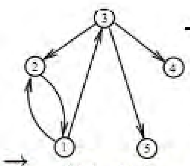
$$m_{ij} = \begin{cases} 1 & \text{ak } \{i,j\} \in H \\ 0 & \text{inak} \end{cases} \qquad m_{ij} = \begin{cases} 1 & \text{ak } (i,j) \in H \\ 0 & \text{inak} \end{cases} \qquad (8)$$



$G_1 = (V_1, H_1)$

	1	2	3	4	5
1	-	1	1	-	-
2	1	-	1	-	1
3	1	1	-	1	-
4	-	-	1	-	-
5	-	1	-	-	-

Matica priľahlosti grafu G_1.



$\vec{G_2} = (V_2, H_2)$

	1	2	3	4	5
1	-	1	1	-	-
2	1	-	-	-	-
3	-	1	-	1	1
4	-	-	-	-	-
5	-	-	-	-	-

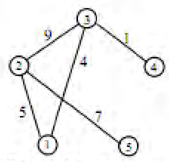
Matica priľahlosti digrafu $\vec{G}_2$.

- **Maticou ohodnotení hrán** **podobne ako Matica priľahlosti len sa namiesto <u>1</u> píše cena**

Matica $\mathbf{M}$ ohodnotení hrán grafu, resp. digrafu je štvorcová matica typu $n \times n$, kde $n = |V|$ je počet vrcholov grafu, resp. digrafu a prvky ktorej sú definované nasledovne:

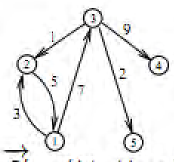
$$m_{ij} = \begin{cases} c(\{i,j\}) & \text{ak } \{i,j\} \in H \\ \infty & \text{inak} \end{cases} \qquad m_{ij} = \begin{cases} c((i,j)) & \text{ak } (i,j) \in H \\ \infty & \text{inak} \end{cases}$$

(9)



$G_1' = (V_1, H_1, c_1)$

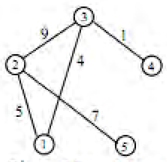
	1	2	3	4	5
1	-	5	4	-	-
2	5	-	9	-	7
3	4	9	-	1	-
4	-	-	1	-	-
5	-	7	-	-	-



$\vec{G_2'} = (V_2, H_2, c_2)$

	1	2	3	4	5
1	-	3	7	-	-
2	5	-	-	-	-
3	-	1	-	9	2
4	-	-	-	-	-
5	-	-	-	-	-

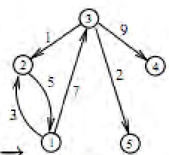
- Zoznamom vrcholov okolia každého vrchola



$G_1' = (V_1, H_1, c_1)$

$V(1)$	2	3	-
$V(2)$	1	3	5
$V(3)$	1	2	4
$V(4)$	3	-	-
$V(5)$	2	-	-

Vrcholy okolí pre graf G_1'.



$\vec{G_2'} = (V_2, H_2, c_2)$

$V^+(1)$	2	3	-
$V^+(2)$	1	-	-
$V^+(3)$	2	4	5
$V^+(4)$	-	-	-
$V^+(5)$	-	-	-

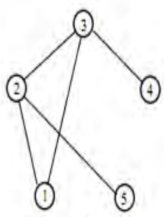
Vrcholy výstupných hviezd pre digraf $\vec{G_2'}$.

- Incidenčnou maticou vrcholov a hrán

Incidenčná matica vrcholov a hrán je matica $\mathbf{B}$ typu $n \times m$, kde n je počet vrcholov a m počet hrán reprezentovaného grafu alebo digrafu. Každý prvok b_{ij} matice $\mathbf{B}$ hovorí o spôsobe incidencie vrchola i s hranou j nasledovne:

<u>**Graf**</u> $$b_{ij} = \begin{cases} 1 & \text{ak vrchol } i \text{ je incidentný s hranou } j \text{ v grafe } G \\ 0 & \text{inak} \end{cases}$$

<u>**Digraf**</u> $$b_{ij} = \begin{cases} 1 & \text{ak vrchol } i \text{ je začiatočným vrcholom hrany } j \text{ v digrafe } \vec{G} \\ -1 & \text{ak vrchol } i \text{ je koncovým vrcholom hrany } j \text{ v digrafe } \vec{G} \\ 0 & \text{inak} \end{cases}$$



$G_1 = (V_1, H_1)$

v	{1,2}	{1,3}	{2,3}	{2,5}	{3,4}
1	1	1			
2	1		1	1	
3		1	1		1
4					1
5				1	

Tabuľka: Incidenčná matica grafu $G_1 = (V_1, H_1)$

2. Cesty v grafoch. Sled, ťah, cesta, cyklus, polosled, poloťah, polocesta, polocyklus a ichanalógie v digrafoch. Súvislosť grafov, Komponent grafu. Mosty a artikulácie. Tarryuho prieskum grafov.

- **Sled** v grafe G j**e ľubovoľná** alternujúca (**striedavá**) **postupnosť vrcholov a hrán tvaru** $\mu(v1, vk) = (v1,\{v1, v2\}, v2,\{v2, v3\}, v3, . . . ,\{vk-1, vk\}, vk)$.
- **Ťah**v grafe G je taký sled v grafe G, v ktorom sa **žiadna hrana neopakuje**.
- **Cesta** v grafe G je taký sled v grafe G, v ktorom sa **žiaden vrchol neopakuje**.
  Pripúšťame aj tzv. **triviálny sled**, pre k = 1, t. j. sled tvaru (v1). **Keď je len jeden vrchol**
- **Cyklus** (orientovaný cyklus, polocyklus) je netriviálny uzavretý ťah (orientovaný ťah, poloťah), v ktorom sa **okrem prvého a posledného vrchola** žiaden vrchol nevyskytuje viac než **raz**.
- **Polosled, poloťah, polocesta (digraf)** -(ako sled,tah,cesta) hrana má začiatočný a a koncový vrchol
- **Súvislosť grafov -** Graf je súvislý, ak pre každú dvojicu vrcholov u, v ∈ V existuje u–v cesta. Inak hovoríme, že graf G je nesúvislý.
  Digraf **- Slabo súvislý** ak existuje polosled. **Jednostranne súvislý** ak existuje u-v sled.
  Silno súvislý ako existuje u-v a aj v-u s orientovaný sled.
- **Komponent grafu** G = (V, H) je jeho ľubovoľný maximálny súvislý podgraf.
- **Mostom** v grafe G = (V, H) nazveme takú **hranu** grafu G, po vylúčení ktorej vzrastie počet komponentov.
- **Artikuláciou** v grafe G nazveme taký **vrchol**, po vylúčení ktorého spolu s **incidentnými hranami** vzrastie počet komponentov.
- **Kružnica** je **pravidelný súvislý graf 2. stupňa**. Kružnicu o n vrcholoch budeme označovať Cn.
- **Tarryho algoritmus (Tarryho sled)** na konštrukciu takého sledu v grafe G = (V, H), ktorý začína v **ľubovoľnom vrchole s ∈ V** , prejde **všetkými hranami komponentu** grafu G a **skončí vo vrchole s**.
  - Krok 1. Začni z **ľubovoľného vrchola s** ∈ V , **polož u := s**, T = (u). {T je inicializačne **triviálny sled**, obsahujúci jediný vrchol.}
  - Krok2. Ak môžeš, **pridaj** k **poslednému vrcholu** u sledu T **ďalšiu incidentnú hranu** {u, v} **podľa** nižšie uvedených **pravidiel T1, T2** a zaraď ju **do sledu T** . **Zaznač si smer použitia** hrany {u, v}. **Ak** doteraz **vrchol** *v* ešte **nebol zaradený** do sledu T , označ hranu {u, v} ako **hranu prvého príchodu**.
    T1: Každú hranu možno v jednom smere použiť iba raz
  - **T2**: Hranu prvého príchodu možno použiť, iba ak niet inej možnosti
    Krok 3. Ak taká hrana neexistuje – STOP. Inak polož u := v a pokračuj Krokom 2.

3. **Najkratšia cesta. Základný algoritmus, Dijkstrov a Floydov algoritmus. Label set a label correct implementácie algoritmov najkratšej cesty.**
NOTE: Floydov algoritmus kuknúť z prednášok

Sled (v1-vk sled) v grafe G je ľubovoľná alternujúca postupnosť vrcholov a hrán tvaru:
mikro(v1, vk) = (v1, {v1,v2}, v2, {v2,v3}, v3 .... {vk-1,vk}, vk)

- **Ťah** v grafe G je taký v1-vk sled,v ktorom sa źiadna hrana neopakuje
- **Cesta** v grafe G je taký v1-vk sled, v ktorom sa žiaden vrchol neopakuje
- **triviálny sled** - sled tvaru (v1)
- **Orientovaný sled, ťah, cesta** - to isté ako hore len sú tam jednoduché zátvorky miesto grafu je použitý digraf

- **Polosled, tvar**: mikro(v1,vk) = (v1,h1,v2,h2,....vk-1,hk-1,vk), v ktorej je každá hrana hi incidentná s obomi susednmi vrcholmi tak , že jeden je začiatočný a druhý je koncovým vrcholom hrany h
- **Poloťah, polocesta** - v digrafe taký polosled ...
- **uzavretý sled** - ak $v_1=v_k$, inak nazveme otvorený
- **súvislý graf** - ak pre každú dvojicu vrcholov u,v existuje u-v cesta. Inak nesúvislý.
- **Komponent grafu** - ľubovoľný max podgraf
- **Most** - taká hrana po ktorej vylúčení sa zvýši počet komponentov
- **Artikulácia** - taký vrchol v grafe po ktorého vylúčení spolu s incidentnými hranami vzrastie počet komponentov
- **neorientovane súvislý digraf** - ak pre každú dvojicu vrcholov existuje v G u-v polosled. Inak je nesúvislý.
- **orientovane súvislý digraf** - ak pre každú dvojicu vrcholov u,v existuje aj orientovaný u-v sled alebo orientovaný v-u sled.

- Tieto 3 algoritmy slúžia na hľadanie najkratších orientovaných u–v ciest z pevného vrchola **u** ∈ V do všetkých ostatných vrcholov v ∈ V v hranovo ohodnotenom digrafe −→G =(V, H, c) s nezápornou cenou orientovanej hrany c(h).
- **Základný algoritmus** na hľadanie najkratších orientovaných ciest z vrchola do vrchola:
  - Krok 1. Inicializácia.
    -Pre každý vrchol i ∈ V priraď dve značky t(i) a x(i). {Značka **t(i) predstavuje horný odhad dĺžky** doteraz nájdenej **najlepšej u–i cesty** a **x(i) jej predposledný vrchol**.}
    -Polož t(u) := 0, t(i) := ∞ pre i ∈ V, i != u a x(i) := 0 pre každé i ∈V
  - Krok 2. **Zisti, či existuje orientovaná hrana (i , j) ∈ H, pre ktorú platí** t(j) > t(i) + c(i,j) { **t koncového vrchola > t začiatočného vrchola + cena hrany}**
    -Ak existuje, **potom polož t(j) := t(i) + c(i , j), x(j) := i** a **opakuj Krok 2**.
  - Krok 3. **Ak** taká orientovaná hrana (z kroku 2.) **neexistuje**, **STOP**.
- **Dijkstrov algoritmus:**
  - Krok 1. Inicializácia. **Pre každý vrchol i ∈ V priraď dve značky t(i) a x(i). Značky t(i) budú dvojakého druhu**, a to **dočasné** (ktoré sa ešte v priebehu výpočtu môžu zmeniť) a **definitívne** (ktoré sa už nemôžu zmeniť), **x(i) jej predposledný vrchol**
    - Polož t(u) = 0, t(i) = ∞ pre i ∈ V, i != u a x(i) = 0 pre každé i ∈ V. **Zvoľ riadiaci vrchol r := u** a značku t( ) pri vrchole r = u **prehlás za definitívnu, ostatné značky za dočasné**.
  - Krok 2.
    Ak je r = v {riad = ciel. vrchol}, STOP. Ak t(v) < ∞, značka t(v) predstavuje **dĺžku najkratšej u–v cesty, ktorú zostroj spätne z v pomocou smerníkov x(i).** Inak pre všetky hrany tvaru (r , j) ∈ H, kde j je vrchol s dočasnou značkou, urob:
    Zober všetky vychádzajúce hrany z riadiaceho vrchola a
    Ak {t(j) > t(r) + c(r , j) **- t dočasného koncového vrchola > t definitívneho riadiaceho vrchola + cena hrany}**, potom **t(j) := t(r) + c(r , j), x(j) := r** Ponechaj zmenené značky ako dočasné.
  - Krok 3. **Zo všetkých dočasne označených vrcholov nájdi ten** vrchol i **, ktorý má značku t(i) najmenšiu**. **Značku** pri tomto vrchole i **prehlás za definitívnu a zvoľ za nový riadiaci vrchol r := i** .
  - {Pokiaľ existuje viac vrcholov s rovnakou minimálnou značkou, akú má vrchol i, za definitívnu značku môžeme prehlásiť len značku pri jednom z týchto vrcholov – ten potom berieme za riadiaci. Na tie ďalšie dôjde v nasledujúcich krokoch výpočtu.} GOTO Krok 2.
- Algoritmus 3.6. **Label-set a Label-correct**
  - Krok 1: Inicializácia. Polož t(u) := 0, t(i) := ∞ pre i ∈ V, i != u a x(i) := 0 pre každé i ∈ V . Polož E := {u}. **{značky t, x. Do Epsilon E sa pridávajú vrcholy na spracovanie}**
  - Krok 2: Vyber r ∈ E, polož E := E − {r}.**{Vyber a aj odstráň vrchol z Epsilon (r)}**
    Pre všetky orientované hrany z r tvaru (r, j) ∈ H **urob: Ak** t(j) > t(r)+c(r, j), potom t(j) := t(r)+c(r, j), x(j) := r, E := E ∪{j}.
    {t koncového vrcholu > t z vrcholu r + cena hrany. Potom t(j) := t(r)+c(r, j), x(j) := r **a pridaj koncový vrchol do Epsilon}**
  - Krok 3: **Ak E != ∅, choď na Krok 2. inak**, potom **t(i) predstavuje dĺžku najkratšej orientovanej u–i cesty pre každý vrchol i**. Najkratšiu orientovanú u–i cestu zostroj potom spätne pomocou značiek x(i) ako v predchádzajúcich dvoch algoritmoch

Ak z Epsilon berieme náhodne - Label correct. Ak berieme podľa **najmenšej značky t - Label set(Dijkstrov)**

4. Stromy a ich vlastnosti. Veta o ekvivalentných výrokoch s výrokom \\\"graf G je stromom\\\". Prehľadávanie grafu do šírky a do hĺbky.

- **Cyklus** - je netriviálny uzavretý ťah v ktorom sa okrem prvého a posledného vrchola žiaden neopakuje
- **Kružnica** - je súvislý pravidelný graf druhého stupňa
- **Acyklický graf** - je graf, ktorého podgraf neobsahuje kružnicu
- **Strom** - súvislý acyklycký graf
- **Veta -** Nasledujúce tvrdenia sú ekvivalentné:
  a) **G = (V,H) je strom**.
  b) **V grafe** G = (V,H) **existuje pre každé u, v ∈ V jediná u–v cesta**.
  c) **Graf** G = (V,H) **je súvislý a každá hrana množiny H je mostom**.
  d) **Graf** G = (V,H) je súvislý a |H| = |V| − 1. (**počet hrán = vrcholy - 1**)
  e) V grafe G = (V,H) platí |H| = |V| − 1 a G je **acyklický**.
- Graf G=(V,H,k) budeme nazývať **koreňovým stromom**, kde **k je** pevne **vybraný vrchol**, ktorý nazývame **koreň**.
- **úroveň vrchola** v koreňovom strome je dĺžka - počet hrán jedinej k-u cesty
- **výška koreňového stromu** je maximálna úroveň vrchola zo všetkých úrovní vrcholov koreňového stromu
- prehľadávanie grafu G=(V,H) do hĺbky (**Depth-First Search**) { **prehľadáva tak, že zoberie vrchol, vyberie jeho incidentnú hranu aby sa dostal na neznámy vrchol a v prípade keď už vrchol nemá takúto hranu tak rekurziou sa vracia k nájdeniu ďalších vrcholov**}.
  - Krok 1(ini): **Nech strom T je triviálny strom** obsahujúci 1 vrchol, polož **p(v)** := 1, k := 1
  - Krok 2: **ak ešte T neobsahuje všetky vrcholy grafu, GOTO3**, inak **STOP**
  - Krok 3: v grafe G so stromom T nájdi hraničnú hranu h = {u, v} s **maximálnou značkou** p(v) zaradeného vrchola u.
  - Krok 4: polož T := T U{h} U {v}, k := k + 1, p(v) := k, GOTO 2
- **algoritmus na prehľadávanie do šírky** je taký istý, len sa nehľadá maximálna značka ale minimálna (**Breadth-First Search**) - {**najskôr prehľadá okolie vrchola a potom prejde na ďalší vrchol**}

5. Kostra grafu, najlacnejšie a najdrahšia kostra grafu, Kruskalov algoritmus I. a II. Využitie pre hľadanie cesty maximálnej priepustnosti v grafe. Využitie Kruskalovho algoritmu na určenie komponentov grafu.

- **Kostra** súvislého grafu G=V,H je taký jeho faktorový podgraf, ktorý je stromom
- Keď G=(V,H,c) je **hranovo ohodnotený graf** a K **je** jeho **kostra**. **Cena** c(K) **kostry je súčet ohodnotení jej hrán**.
- **Najlacnejšia kostra** v grafe G je kostra s najmenšou cenou
- **Najdrahšia kostra** v grafe G je kostra s najväčšou cenou
- na hľadanie najlacnejšej (najdrahšej) kostry používame **Kruskalov algoritmus I**:
  - **1. Zoradíme hrany** podľa ich **ohodnotenia vzostupne/zostupne do postupnosti** P
  - **2.** Nech prvá hrana postupnosti je {u,v}. **Vyber a vylúč hranu {u, v} z postupnosti P a ak už s vybranými hranami netvorí cyklus, zaraď ju do kostry.**
  - **3. Ak** je **počet hrán vybraných rovný V-1** alebo ak je **postupnosť P prázdna -STOP**, inak GOTO2
- na hľadanie najlacnejšej (najdrahšej) kostry používame **Kruskalov algoritmus II:**
  - **1. Zoraď hrany** podla ich **ohodnotenia vzostupne/zostupne do postupnosti P**.
  - **2.** Pre každý vrchol i ∈ V poloz k(i) = i . {pole **k** o veľkosti V **naplnené všetkými V**}
  - **3.** Nech prvá hrana v postupnosti P je hrana {u, v}. **Vyber a vylúč hranu {u, v} z postupnosti P**. **Ak k(u) != k(v)** tak sprav: {ak k na indexe u = k na indexe v}
    -zaraď hranu {u, v} do kostry
    -pre každé i ∈ k (i ∈ V), **pre ktoré platí k(i) = k(v)**, sprav **k(i) := k(u)**
  - **4. Ak** je **počet hrán** vybraných **rovný V-1** alebo ak je **postupnosť P prázdna - STOP**, inak GOTO3

Cesta maximálnej priepustnosti

- Nech G=V,H,c je **hranovo ohodnotený graf**, **v ktorom cena hrany** h ∈ H c(h) **> 0 znamená priepustnosť**
- priepustnosť c(μ(u,v)) u-v cesty (sledu, polosledu ...) mikro(u,v) definujeme ako: c(μ(u,v)) = min { c(h) | h e μ(u,v)}
- u,v cesta μ(u,v) v grafe G=V,H,c je u-v **cesta maximálnej priepustnosti**, ak má **najväčšiu priepustnosť zo všetkých u-v ciest v G**

- **Algoritmus na hľadanie u-v cesty maximálnej priepustnosti** v súvislom hranovo ohodnotenom grafe G=(V,H,c):
    - **1. V grafe** G **zostroj najdrahšiu kostru K**
    - **2. V kostre** K **nájdi jedinú u-v cestu**. **Táto** jediná u-v **cesta** v kostre K **je** u-v **cestou maximálnej priepustnosti v grafe** G

  Algoritmus nájde cestu maximálnej priepustnosti, ale táto cesta **nie je z hľadiska prejdenej vzdialenosti optimálna**
- **Algoritmus na hľadanie najkratšej u-v cesty s maximálnou priepustnosťou** v súvislom hranovo ohodnotenom grafe **G=(V,H,c,d) kde c**(h) j**e priepustnosť** a **d**(h) je **dĺžka hrany** h ∈ H
    - **1. v grafe** G nájdi **cestu maximálnej priepustnosti** vzhľadom na ohodnotenia hrán c. Nech **C je priepustnosť cesty** μ(u,v)
    - **2. vytvor graf G'= (V,H', d) kde** H'= {h|h patri H, c(h) >= C}. **H' obsahuje len tie hrany ktoré majú priepustnosť väčšiu alebo rovnú než C.**
    - **3. V grafe G' nájdi najkratšiu u-v cestu** vzhľadom na **ohodnotenie** hrán **d**
- Poznámka. **Kruskalov algoritmus II** (algoritmus 4.4) môžeme s výhodou použiť na zistenie **komponentov grafu**. Ak totiž spustíme tento algoritmus na nesúvislý graf, po skončení práce algoritmu značky k( ) vrcholov z V budú určovať komponenty nasledovne: Dva vrcholy u ∈ V , v ∈ V sú v tom istom komponente grafu G práve vtedy, keď k(u) = k(v) Pre zisťovanie komponentov grafu nie je potrebné usporiadať hrany podľa ich ohodnotenia a krok 1 algoritmu 4.4 možno preformulovať nasledovne: Zoraď hrany grafu G do postupnosti P v ľubovoľnom poradí.{**takto je to myslené: -pole k = {1,1,1,1,1,1} - 1 komponent**
  -pole k = {1,1,1,4,4,4}- 2 alebo viac komponentov}

6. Acyklické digrafy a ich vlastnosti. Typy súvislosti v digrafoch orientovaná, neorientovaná a silná súvislosť. Monotónne očíslovanie vrcholov grafu. Algoritmy na hľadanie najkratšej a najdlhšej cesty v acyklických digrafoch

- **Acyklický digraf** je taký digraf, ktorý neobsahuje orientovaný cyklus
- **Strom** je suvisly acyklicky graf
- **Orientovaný strom** je neorientovane súvislý digraf, ktorý neobsahuje polocyklus
- Nasledujúce tvrdenia sú ekvivalentné:
  a) G = (V,H) je strom.
  b) V grafe G = (V,H) existuje pre každé u, v ∈ V jediná u–v cesta.
  c) Graf G = (V,H) je súvislý a každá hrana množiny H je mostom.
  d) Graf G = (V,H) je súvislý a |H| = |V| − 1. (počet hrán = vrcholy - 1)
  e) V grafe G = (V,H) platí |H| = |V| − 1 a G je acyklický.
- Vlastnosti orientovaných stromov:
    - v digrafe G existuje pre každé u,v patri V jedina u-v polocesta
    - digraf G je neorientovane súvislý a každá orientovaná hrana H je mostom
    - digraf G je neorientovane súvislý a H = V - 1
    - v digrafe platí H = V -1 a G neobsahuje polocyklus
- Nech G = V,H je **acyklický digraf**. Potom V **obsahuje aspoň jeden vrchol z** taký že **ideg(z) = 0 a** aspoň **jeden vrchol u** taký že **odeg(u) = 0**
- **Očíslovanie vrcholov v1, v2, ...** vn acyklického digrafu G = V,H platí: **ak (vi, vk)** ∈ **H, potom i < k**, nazveme **monotónnym očíslovaním v**rcholov acyklického digrafu
- **Monotónne očíslovanie acyklického digrafu**
    - Krok 1. **Polož i=1**
    - Krok 2. **Vyber a vylúč** taký **vrchol v ∈ V** , že **ideg(v) = 0** a **polož vi := v**.
    - Krok 3. Ak V - {v} = prázdna - **STOP** {**ak už je V prázdna**}, inak G=G - {v}, i=i +1 a GOTO 2 {**i++**}
- **Algoritmus na výpočet najkratšej u-v cesty v neorientovane súvislom acyklickom hranovo ohodnotenom digrafe** −→G = (V, H, c).
    - Krok 1. Monotónne očísluj vrcholy digrafu −→G, nech P = v1, v2, . . . , vk postupnosť vrcholov digrafu −→G zoradená podľa monotónneho očíslovania. Zisti index vrchola u v postupnosti P. Nech i je index taký, že u = vi.
    - Krok 2. **Pre každý vrchol v ∈ V** priraď značky **t(v), x(v)**. Polož **t(u) := 0**, **t(j) :=** ∞ pre všetky j != u, j ∈ V . Polož x(j) := 0 pre všetky j ∈ V .

- Krok 3. Pre všetky vrcholy **w výstupnej hviezdy** vrchola vi také, že **w != vi**,
  urob: **Ak t(w) > t(vi) + c(vi, w), potom t(w) = t(vi) + c(vi, w)**, a **x(w) := vi**.
- Krok 4. i := **i + 1**. Ak i = n STOP, inak GOTO Krok 3.

7. Časová analýza projektov - metóda CPM. Dve možné reprezentácie (vrcholovo ohodnoteným resp.- hranovo ohodnoteným digrafom). Najskôr možný začiatok vykonávania činnosti a najneskôr nutný koniec vykonávania činnosti. Trvanie projektu. Kritické činnosti a kritická cesta.

- **NOTE: pozri si čo je to relácia precedencie**
- **Najskôr možný začiatok z(A)- začiatok** vykonávania **projektu** začne **v čase 0**. **Je prvý okamih** od začiatku projektu **od ktorého začína činnosť A** pri dodržaní relácie precedencie. Z celého projektu zistíme **trvanie projektu T** tak, že nájdeme takú **činnosť A**, **ktorá má najväčšiu hodnotu zo začiatku A + doba trvania A** od **všetkých činností.**
  $$\mathbf{T} = \mathrm{MAX}(z(A) + p(A) \mid A \in E)$$
- **Trvanie projektu T** je celková doba trvania projektu.
- **Najneskôr nutný koniec k(A) - Musíme poznať trvanie projektu T**. **Je posledný okamih** od začiatku vykonávania projektu, **po ktorom sa môže činnosť A oneskoriť bez** vzniku **predĺženia.**
- **Časová rezerva** - činnosti A je **R(A) = k(A) − z(A) − p(A)**.
- **Kritická činnosť** - Taká **činnosť** pre **ktorú platí R(A) = 0.**
  -Taká orientovaná cesta, ktorá má **maximálny súčet ohodnotení vrcholov**
  -Činnosti ležiace na **kritickej ceste** sa nazývajú **kritické činnosti**.
  -**Súčet ohodnotení činností** v **kritickej cesty** nám dá dobu **trvania projektu** T
- Algoritmus 5.6. **Algoritmus II. na určenie najskôr možných začiatkov z(v)** elementárnych činností v digrafe −→G≺≺ = (V, H, c).
  - Krok 1. **Vytvor monotónne očíslovanie** v1, v2, . . . , vn **vrcholov** digrafu −→G≺≺
  - Krok 2. **Každému vrcholu v ∈ V** priraď **dve značky z(v), x(v)**.
    Pre každé v ∈ V inicializačne polož **x(v) := 0**, **z(v) := 0**.
  - Krok 3. Postupne pre k = 1,2, . . . , n − 1 urob:
    Pre všetky také **vrcholy w** z **výstupnej hviezdy vrchola vk**, že **w != vk**, urob:
    Ak **z(w) < z(vk) + c(vk)**, potom **z(w) := z(vk) + c(vk)** a **x(w) := vk**.
  - Krok 4. **Vypočítaj trvanie projektu**
    T := max{z(w) + c(w) | w ∈ V, odeg(w) = 0}
- **Algoritmus II. na určenie najneskôr nutných koncov k(v)** elementárnych činností v digrafe −→G≺≺ = (V, H, c). - **Skoro podobne ako pri začiatkov**
  - Krok 1. **Vytvor monotónne očíslovanie** v1, v2, . . . , vn **vrcholov** digrafu −→G≺≺ .
  - Krok 2. **Každému vrcholu** v ∈ V priraď dve značky **k(v), y(v)**. Nech **T je trvanie projektu**. Pre každé v ∈ V inicializačne polož **k(v) := T** , **y(v) := 0** .
  - Krok 3. Postupne pre i = n − 1, n − 2, . . . ,1 urob:
    Pre všetky **vrcholy w výstupnej hviezdy vrchola vi** také, že **w != vi**, urob:
    Ak **k(vi) > k(w) − c(w)**, potom **k(vi) := k(w) − c(w)** a **y(vi) := w**

Pre každý vrchol i sieťového digrafu vypočítame Ti, t. j. najskôr možný začiatok činností vychádzajúcich z vrchola i, ako Ti = dmax(1, i) a T′i najneskôr nutný koniec činností vchádzajúcich do vrchola i ako T′i = Tn − dmax(i, n) kde dmax(x, y) je dĺžka najdlhšej orientovanej x–y cesty. Pre vrchol i ležiaci na nejakej kritickej ceste je Ti = T′ i, pre ostatné vrcholy je Ti < T ′i. **?**

8. Eulerovský ťah v grafe. Eulerovský graf. Kritérium pre to, aby bol graf eulerovský. Algoritmy na zostrojenie uzavretého eulerovského ťahu v grafe (Fleuryho, labyritnový, postupným rozširovaním uzavretého ťahu).

- **Eulerovský sled** - sled s(u, v) v **súvislom** grafe G = (V,H) je eulerovský, ak **obsahuje všetky hrany grafu** G.
- Keďže ťah obsahuje každú hranu grafu G práve raz, postupnosť vrcholov a hrán ťahu t(u, v) **predstavuje postup, ako nakresliť diagram grafu G "jedným ťahom"**.
- **Eulerovský ťah** - graf G = (V,H) je eulerovský, ak v nom **existuje uzavretý** eulerovsky ťah.
- Súvislý **graf je eulerovsky** práve vtedy, **keď stupne všetkých vrcholov grafu G sú parne**.
- Algoritmus na konstrukciu eulerovskeho tahu :
    - **1. Začni z ľubovolného vrchola z**, poloz T = (z) a **postupne predlžuj ťah** T **pokiaľ sa dá**. **Ukončíš vo vrchole z**.
    - **2. Nájdi prvý vrchol v ťahu** T , ktory ma este **aspoň jednu hranu nepoužitú** v tahu T . **Ak** taky vrchol v **neexistuje**, **STOP**.
      Tah T je hladanym uzavretym eulerovskym tahom.
    - **3. Vytvor ťah S** takto: Poloz S = (***v***) a **postupne predlžuj ťah** S doteraz **nepoužitými hranami**, pokial sa da **až skončíš vo vrchole *v*** (prídeš do slepej uličky).
    - **4.** Rozdel tah T na z–v tah T1 a v–z tah T2, t. j. T = T1 + T2. Poloz T = T1 + S + T2. **GOTO Krok 2**. {**v ťahu T sa mieste *v* vloží ťah S** .T1 je časť už komplet prejdených vrcholov a T2 je ešte neprejedná časť. T = T1 + S + T2 (hamburger)}
- Fleuryho algoritmus na hladanie eulerovskeho tahu:
    - **NOTE: tento "algoritmus" je uskutočniteľný na základe ľudskej <u>intuitívy</u>, čiže stačí povedať to ako by si ručne nakreslil jeden ťah. Počítačovo by sa musel riešiť rôznymi algoritmami lebo nerieši to keby sa PC dostal do slepej uličky.**
    - **Zacni v lubovolnom vrchole** a do tahu T zarad lubovolnu s nım incidentnu hranu.
    - **Ak su** do tahu T **zaradene vsetky hrany** grafu G, **STOP**.
    - Ako dalsiu hranu **zarad** do tahu T taku **hranu** incidentnu s jeho **poslednym vrcholom**, **po vybratı ktorej sa podgraf** grafu G pozostavajuci z nevybratych hran a s nimi incidentnych vrcholov **nerozpadne na**
      - **dva** netrivialne **komponenty**
      - netrivialny komponent a izolovany zaciatok tahu T .
    - **GOTO krok 2**.
- Labyrintovy algoritmus na hladanie uzavreteho eulerovskeho tahu:
    - **Zacni z lubovolneho vrchola u** $\in$ V. Nech **<u>sled</u> S** inicializacne pozostava z **jedineho vrchola u**. Poloz w := u – vrchol **w je posledny vrchol** doteraz vytvoreneho **sledu S**.
    - Ako dalsiu hranu vyber podla nizsie uvedenych pravidiel do sledu S hranu {w, *v*}. **Zaznac si smer pouzitia hrany {w, *v*}**. **Ak** doteraz **vrchol *v*** este **<u>nebol</u> zaradeny** do sledu S, **oznac hranu** {w, *v*} **<u>ako hranu prveho príchodu</u>**, inak zaraď ako **použitú hranu**.
      Ak narazíš na slepú uličku:
      -Dalej **zaznamenaj** tzv. **<u>spätnu postupnost</u>** — poradie **hran**, v ktorom sa v slede S vyskytuju po **druhykrat.** Pri vybere hrany **dodrzuj** nasledujuce **pravidla:**
        (L1): **Kazdu hranu** mozno v jednom smere **pouzit iba raz**
        (L2): **Poradie zaradovania** hran:
          - **nepouzite** hrany
          - hrany **pouzite raz**
          - hrana **prveho prıchodu** (ak niet inej moznosti)

        – Ak taka hrana neexistuje – STOP.
        <u>Spätna postupnost</u> urcuje hladany eulerovsky tah.
    - Inak poloz w := v a pokracuj krokom 2.

9. Úloha čínskeho poštára. Párenie v grafe. Edmondsov algoritmus na riešenie úlohy čínskeho poštára.

- slovne: Poštár má výjsť z pošty, prejsť všetky ulice svojho regiónu a vratiť sa späť tak aby sa čo najmenej nachodil
- matematicky: **V súvislom hranovo ohodnotenom grafe nájsť uzavretý eulerovský sled najmenšej dĺžky**
- ak má graf vrcholy s **párnym stupňom**, stačí zostrojiť uzavretý **eulerov ťah**
- ak má vrcholy s **nepárnym stupňom**, potom ich tam je párny počet **2t**
- **pridaním fiktívnych hrán** typu {**nepárny, nepárny**} s dĺžkou rovnajúcou sa vzdialenosti príslušných vrcholov v G možno dostať **eulerovský graf**
- **čím menší** súčet dĺžok pridaných **fiktívnych hrán,** tým **lepšie** riešenie
- -eulerovský sled: sled s(u,v) v súvislom grafe G, ak obsahuje všetky hrany grafu G
- -eulerov ťah je taký ťah t(u,v) v súvislom grafe G, ktorý obsahuje všetky hrany grafu G
- -eulerov graf je taký graf, v ktorom existuje uzavretý eulerovský ťah
- **párenie je taký podgraf** P grafu, **ktorého každý stupeň vrchola je práve 1**
- **cena párenia je súčet ohodnotení hrán párenia**
- **maximálne párenie** - hovoríme, že párenie *P* je maximálne párenie v grafe *G*, **ak *P* nie je podgrafom žiadneho iného párenia v *G***
- **najpočetnejšie párenie** - párenie *P* je najpočetnejšie párenie v grafe *G* **ak *P* má zo všetkých párení najväčší počet hrán**
- **úplné párenie je párenie P, ktoré obsahuje všetky vrcholy pôvodného grafu G** (P je **faktorový podgraf** grafu G)
- na riešenie úlohy používame Edmondsov algoritmus:
  - **1. nájdeme vrcholy s nepárnym počtom stupňov** - párny počet
    Z týchto **vrcholov spravíme úplný graf G'** a **ohodnotíme vzdialenosťami koncových vrcholov** hrany v G
  - 2.v grafe **G' spravíme úplné párenie s najnižšou cenou**
  - 3.**hrany párenia pridáme** do hranovej množiny **pôvodného grafu** - dostaneme **multigraf**, v ktorom majú **všetky vrcholy párny počet stupňov**
    Zostrojíme najkratší eulerovský tah T
  - 4.**hrany párenia** v ťahu T **nahraď najkratšími cestami** v G a **označ** ich ako **prejdené na prázdno**. Dostaneme najkratší eulerovský uzavretý sled v G

10. Úloha obchodného cestujúceho. Hamiltonovský cyklus a hamiltonovský graf. Postačujúce podmienky pre to, aby graf bol hamiltonovský. Metóda zdvojenia kostry a metóda kostry a párenia. Vytváracie a zlepšujúce heuristiky.

- slovne: **Obchodný cestujúci** navštíviť **všetkých** svojich **zákazníkov** a **vrátiť sa domov** tak, aby sa **čo najmenej nachodil**
- matematicky:
  - ak dovolujeme navštíviť viackrát: V súvislom hranovo ohodnotenom grafe nájsť najkratší uzavretý hamiltonovský sled. Hamiltovovský sled je taký sled s(u,v), ktorý obsahuje každý vrchol grafu G.
  - ak nedovoľujeme: V súvislom hranovo ohodnotenom grafe G hľadáme najkratší hamiltovovský cyklus. Hamiltonovský cyklus je špeciálny prípad hamiltonovského sledu.
- **hamiltonovský sled - Sled v grafe**, ktorý **obsahuje všetky vrcholy grafu G.**
- **hamiltonovský graf -** je taký **graf**, ktorý **obsahuje hamiltonovský cyklus**
- **hamiltonovský graf -** je hamiltonovský ak:- **má aspoň 3** vrcholy
  -pre **každé 2 vrcholy**, ktoré **nie sú susedné** platí: **deg(u) + deg(v) >= |počet V|**
  -pre každý vrchol platí **deg(v) >= 1/2 * |počet V|**
- **v praxi sa hamiltonovský cyklus vyskytuje** veľmi **málo a tak** sa sústredíme na **hľadanie hamiltonovského sledu v úplnom grafe G'**. Zakazovať návštevy aj tak nemá zmysel.
- v grafe **G' platí trojuholníková nerovnosť d(u,v) <= d(u,w) + d(w,v)**
- **na riešenie máme** suboptimálne **algoritmy**, ktoré **dávajú dostatočne** dobré **riešenia ale nie dostatočne efektívne**

- Greedy Method (pažravá metda)
  - **1. Zacni v lubovol'nom vrchole** a do (buduceho) hamiltonovskeho cyklu **vloz najlacnejsiu hranu incidentnu** s tymto **vrcholom**.
  - **2. ak je vybraných n-1 hrán, uzavri cyklus**
  - **3. Inak vyber** taku **najlacnejsiu nevybranu hranu incidentnu s poslednym vrcholom** doteraz vybranej postupnosti, **ktora nie je incidentna** so **ziadnym inym vrcholom vybranej postupnosti**. Goto2
  - 4. každú hrany cyklu nahradíme príslušnou najkratšou cestou v pôvodnom grafe G
- Metóda zdvojenia kostry
  - 1. v grafe G **zostrojíme najlacnejšiu kostru**
  - 2. v kostre K zostrojíme **uzavretý sled S**, ktorý obsahuje **každú hranu práve 2x**. (použijeme **napríklad Tarryho** algoritmus)
  - 3. z uzavretého sledu vytvoríme hamiltonovský cyklus takto: **prechádzaj sledom S** a **keď** narazíš na **vrchol** alebo **úsek vrcholov** za sebou, **ktoré sa opakujú**, **premosti ich priamou hranou**.(premostiť - z hamburgera vyberieme mäso)
- Algoritmus kostry a párenia
  NOTE: Krok 1 a 7 je podobný kroky ako u zdvojenia kostry. Krok 2-6 sú podobné ako u Edmonsovho algoritmu. (*double big mac*)
  - 1. v grafe G zostroj **najlacnejšiu kostru K**
  - 2. v kostre K nájdi všetky **vrcholy nepárneho stupňa** - párny počet 2t
  - 3. z týchto vrcholov zostroj **úplný graf G'** - jeho **hrany ohodnoť podľa pôvodného grafu**
  - 4. v grafe **G'** najdi **úplné párenie s minimálnou cenou**
  - **5. hrany párenia pridaj** do najlacnejšej **kostry** K. Dostanes **multigraf** s **párnym stupňom vrcholov**
  - 6. v **grafe** (multigrafe) zostroj **uzavretý eulerovský ťah T**
  - 7. hamiltonovský cyklus takto: **prechádzaj ťahom T** a ked **narazíš na opakujúce sa vrcholy** alebo **úsek vrcholov**, **premosti ich priamou hranou**

11. Algoritmy a ich zložitosť. Definícia symbolu $O(f(n))$, polynomiálne algoritmy. Polynomiálna redukcia a polynomiálna transformácia. Polynomiálne riešiteľné úlohy a NP-ťažké úlohy.

- Pod pojmom **algoritmus rozumieme postupnosť krokov**, **ktorá nás dovedie k žiadanému riešeniu** daného **problému**. Žiada sa, aby algoritmus mal tieto **vlastnosti**:
  - **determinovanosť** – má byť **zadaný konečným** počtom **jednoznačných pravidiel**
  - **efektívnosť – má zaručiť vyriešenie úlohy** po konečnom počte krokov
  - **hromadnosť** – má byť **použiteľný na celú triedu prípadov** úlohy svojho typu
- Pre **ohodnotenie výpočtovej zložitosti algoritmu** nás však viac ako jeden konkrétny prípad **zaujíma** závislosť **počtu elementárnych krokov** algoritmu na veľkosti resp. rozsahu počítanej úlohy. Budeme definovať dĺžku úlohy ako množstvo vstupných dát príslušnej úlohy.

Príklad

Pre graf G = (V, H) alebo digraf $\longrightarrow$G = (V, H) s n vrcholmi a m hranami, t.j. kde |V| = n, |V| = m, bude dĺžka úlohy (m + n). Niekedy sa udáva len závislosť času výpočtu len na počte vrcholov n, pričom sa berie do úvahy, že $m \leq n(n-1)/2 \leq n2$ pre grafy, resp. $m \leq n(n-1) \leq n2$ pre digrafy.

- **Počet krokov algoritmu môže** totiž závisieť nielen od množstva **vstupných dát**, ale aj od ich **vzájomnej konfigurácie**. Preto **funkcia T** (n) môže **vyjadrovať** iba **hornú hranicu počtu krokov algoritmu** pre **najhorší prípad úlohy** s dĺžkou n (worst case analysis).
- Nech g(n), h(n) sú dve kladné funkcie definované na množine prirodzených čísel. Budeme písať g(n) = O(h(n)) a hovoriť, že funkcia h(n) **asymptoticky dominuje funkcii** g(n), ak existuje konštanta K a prirodzené číslo n0 také, že $g(n) \leq K.h(n)\ \forall n \geq n_0$.
  Hovoríme, že **algoritmus A má zložitosť** O(f(n)), ak pre horný odhad T (n) počtu krokov algoritmu A pre úlohu dĺžky n platí T (n) = O(f(n)).

Skrátene hovoríme o O(f(n)) algoritme. Špeciálne ak $f(n) \le n^k$ pre nejaké konštantné k, hovoríme, že A je **polynomiálny algoritmus**.
Hovoríme, že problém má zložitosť nanajvýš O(f(n)), ak preň existuje O(f(n)) algoritmus.

- **Polynomiálna transformácia** - Nech je daný problém P_1 so vstupnými dátami D_1. Problém P_1 môžeme riešiť aj tak, že ho pretransformujeme na iný problém P_2 tak, že dáta D_1 prerobíme na dáta D_2 problému P_2. Ak riešenie R_2 problému P_2 vieme prerobiť na riešenie R_1 problému P_1, transformácia je hotová. Ak prerobenie dát D_1 na D_2 a riešenia R_2 na R_1 vyžaduje len polynomiálny počet elementárnych operácií, nazveme túto transformáciu **polynomiálnou transformáciou**. **{(Problem1 + Data1) pretransformovanie -> (Problem2 + Data2) z toho ak môžeme dostať Riešenie2 na Riešenie1 tak sme spravili Problem1}** (pretransformovanie - prevod polynomiálnym počtom krokov)
- **Polynomiálna redukcia** - Ak prevody dát a riešení vyžadujú len polynomiálny počet elementárnych operácií a výpočet vyžaduje polynomiálny počet výpočtov problému P2, hovoríme, že sme problém P1 **polynomiálne redukovali na problém** P2. **{ak je P1 zložený z mnoho P2}**
- Ak problém **P1 možno polynomiálne redukovať na problém P2 a naopak**, problém P2 možno polynomiálne redukovať na P1 hovoríme, že problémy P1, P2 sú **polynomiálne ekvivalentné**.
- Ako etalón ťažkých úloh bola vybratá úloha bivalentného lineárneho programovania (BLP)Problém, ktorý možno polynomiálne redukovať na úlohu BLP, je ľahší alebo rovnako ťažký ako úloha BLP. Problém, na ktorý možno polynomiálne redukovať úlohu BLP je ťažší alebo rovnako ťažký ako úloha BLP.
- Hovoríme, že **problém P je NP–ťažký** , ak úlohu bivalentného lineárneho programovania možno polynomiálne redukovať na P. **{BLP na P}**
- Hovoríme, že **problém P je NP–ľahký** , ak problém P možno polynomiálne redukovať na úlohu bivalentného lineárneho programovania. **{P na BLP}**
- Hovoríme, že **problém P je NP–ekvivalentný** ,**{BPL rovný P}**

12. Alokačné úlohy - depá a havarijné strediská. Vážený p-medián a vážené p-centrum. Heuristický výmenný algoriotmus. Atrakčné obvody.

- (***intro***)Pri zásobovaní územia nejakým tovarom (uhlím, nábytkom, potravinami, liekmi atď.) často potrebujeme rozhodnúť, koľko skladov a v ktorých lokalitách máme postaviť. Podobný problém môžeme riešiť pri rozhodovaní koľko stredísk zdravotnej záchrannej služby a v ktorých miestach zriadiť. Tieto problémy spadajú pod tzv. lokačné problémy. Lokačné problémy sa líšia typom kriteriálnej funkcie a modelom prostredia, v ktorom ich riešime. Existuje niekoľko spôsobov na modelovanie a riešenie lokačných problémov. Najdôležitejšími z nich sú metódy celočíselného lineárneho programovania a metódy teórie grafov. My sa, pochopiteľne, budeme zaoberať metódami teórie grafov v najjednoduchšom prípade, kedy je už množstvo skladov či záchranných stredísk známe. Modelom prostredia, v ktorom budeme tieto úlohy riešiť, bude súvislý hranovo a vrcholovo ohodnotený graf G = (V, H, c, w), v ktorom vrcholy predstavujú križovatky a dôležité body, hrany modelujú ulice, resp. priame cesty medzi vrcholmi, ohodnotenie c(h) hrany h ∈ H predstavuje dĺžku hrany h, ohodnotenia w(v) vrchola v ∈ V – váha vrchola v (predstavuje relatívnu dôležitosť vrchola v). Všetky vrcholy v grafe G = (V, H, c, w) potrebujú obsluhu. Ich náročnosť na obsluhu je vyjadrená ich váhou. Niektoré vrcholy v grafe G môžu naviac slúžiť ako strediská obsluhy. Poznáme dve základné funkcie stredísk obsluhy. Prvá z nich je funkcia zásobovacia. V tomto prípade pre stredisko obsluhy používame termín depo. V depe je umiestnený sklad materiálu. Každý vrchol v grafu G = (V, H, c, w) potrebuje za jednotku času w(v) jednotiek materiálu, jednotkové náklady na dovoz materiálu sú úmerné prepravovanej vzdialenosti. Tu hľadáme také umiestnenie diep, ktoré minimalizuje celkové dopravné náklady na obsluhu všetkých vrcholov grafy G. Druhá funkcia stredísk obsluhy je záchranná. Takú funkciu plnia napríklad stanice pohotovostnej lekárskej služby, požiarne zbrojnice, strediská horskej služby atď. V tomto prípade pre stredisko obsluhy používame termín havarijné stredisko. Tu už dopravné náklady nehrajú takú dôležitú úlohu ako v predchádzajúcom prípade — kritériom je tu dostupnosť najhoršie položeného vrchola grafu G = (V, H, c, w). Chceme nájsť umiestnenia staníc záchrannej lekárskej služby tak, aby ani ten najhoršie položený pacient neumrel pre neskorý prí chod pomoci, chceme nájsť umiestnenie požiarnych zbrojníc tak, aby v prípade potreby požiarnici došli včas, aj keby požiar vznikol aj v najhoršie položenom mieste.
- **Všetko sa odohráva na grafe G = (V, H, c, w) je súvislý hranovo a vrcholovo ohodnotený graf.** + je tu D čo sa berie asi ako množina vrcholov (asi depá) - **D podmnožina** V {ďalej moc netuším čo sa deje tak možno keď si prečítaš hore intro a povieš niečo z toho tak si trochu pomôžeš}
- Nech $1 \le p < |V|$, Dp p-prvková podmnožina množiny V . Hovoríme, že Dp je vážený p-medián grafu G, ak pre ľubovoľnú p-prvkovú podmnožinu D′p množiny V platí $f(Dp) \le f(D'p)$, t.j. ak súhrnn vážená vzdialenosť všetkých vrcholov grafu G od Dp je najmenšia medzi

všetkými p-prvkovými podmnožinami množiny V . Špeciálne ak w(v) = 1 pre všetky v ∈ V , hovoríme, že Dp je p-medián.

- Nech 1 ≤ p < |V |, Dp p-prvková podmnožina množiny V .Hovoríme, že Dp je vážené p-centrum grafu G, ak pre ľubovoľnú p-prvkovú podmnožinu D′p množiny V platí ecc(Dp) ≤ ecc(D′p),t. j. ak množina Dp má najmenšiu váženú excentricitu zo všetkých p-prvkových podmnožín množiny V . Špeciálne ak w(v) = 1 pre všetky v ∈ V , hovoríme, žeDp je p-centrum.
- Heuristický algoritmus na hľadanie váženého p-mediánu v súvislom hranovo a vrcholovo ohodnotenom grafe G = (V, H, c, w) .
  - Krok 1. Náhodne vyber p-prvkovú podmnožinu množiny V .
    Nech Dp = {v1, v2, . . . , vp}, V − Dp = {u1, u2, . . . , uq}, kde q = |V | − p.
  - Krok 2. Hľadaj také i, j, 1 ≤ i ≤ p, 1 ≤ j ≤ q,
    že pre D′p(i, j) = (Dp ∪ {uj}) − {vi} je f(D′p) < f(Dp).
  - Krok 3. Ak taká dvojica indexov i, j neexistuje, STOP.
    Inak polož Dp := D′p(i, j) a GOTO Krok 2.
- Nech je daný súvislý hranovo a vrcholovo ohodnotený grafG = (V, H, c, w) a p-prvková množina diep Dp. Atrakčný obvod A(v) depa v ∈ Dp je množina všetkých takých vrcholov grafu G, ktorých vzdialenosť od depa v je menšia alebo rovná ako vzdialenosť od iných diep, t.j. A(v) = {x| x ∈ V, ∀u ∈ Dp d(v, x) ≤ d(u, x)}
- Prvotný atrakčný obvod A′(v) depa v ∈ Dp je množina všetkých takých vrcholov grafu G, ktorých vzdialenosť od depa v je menšia ako vzdialenosť od iných diep, t.j.
  A′(v) = {x| x ∈ V, ∀u ∈ Dp, u 6= v d(v, x) < d(u, x)}

13. Toky v sieťach. Dopravná sieť, zdroj, ústie. T~~ok~~definícia, veľkosť toku, maximálny tok. Rezová množina, Ford-Fulkersonova veta a algoritmus na hľadanie maximálneho toku v sieti. Algoritmus na hľadanie maximálneho toku s minimálnou cenou. Siete s viacerými zdrojmi a viacerými ústiami. NOTE:toto je lepšie si pozerať v prezentáciách

- **sieťou** nazveme **neorientovane súvislý hranovo ohodnotený digraf** ->G =(V,H,c) v ktorom **ohodnotenie c(h) > 0** každej hrany h patrí H je **celočíselné** a predstavuje **priepustnosť hrany h** a v ktorom existuje
  - **práve jeden vrchol z ideg(z) = 0**, – zdroj
  - **práve jeden vrchol u odeg(u) = 0**, - ústie
- pre každý vrchol digrafu platí značenie:
  - H+(v) - **množina hrán z vrchola v vychádzajúcich**
  - H-(v) je **množina hrán do vrchola v vchádzajúcich**
- **Tok** v sieti G=(V,H,c) j**e celočíselná funkcia y**: H->R **definovaná na množine orientovaných hrán H, kde platí:** {y - tok, c - priepust. hrany, u - ústie, z - zdroj}
  - **y(h) >= 0** pre **všetky h** ∈ H
  - **y(h) <= c(h)** pre **všetky h** ∈ H
  - ∑ (**h** ∈ **H+(v)**) **y(h)** =∑ (**h** ∈ **H-(v)**) **y(h)** pre všetky v ∈ V a **v!=u** a **v!=z**
  - ∑ (**h** ∈ **H+(z)**) **y(h)** = ∑ (**h** ∈ **H-(u)**) **y(h)**
- **Veľkosťou toku** y nazveme **číslo** F(y) = ∑ (h patri **H+(z)**) y(h), čo sa rovná ∑ (h patri H-(u)) y(h)
- **tok v sieti G je maximálny**, **ak má najväčšiu veľkos**ť zo **všetkých** možných **tokov** v sieti G
- Orientovanú **hranu** nazveme **nasítenou**, ak **c(h) = y(h)**
- **Rezerva hrany** v **polоceste** μ(v,w): **r(h) = c(h) - y(h)** ak je **hrana** h použitá **v smere orientácie**, **r(h) = y(h)** ak je **hrana** použitá **proti smeru orientácie** polocesty
- **Rezerva polocesty** μ(u,v) je **minimum rezerv hrán** tejto polocesty
- polocesta je rezervná polocesta ak má kladnú rezervu
- rezervná polocesta mikro(u,v) zo zdroja do ústia sa nazýva zväčšujúca polocesta
- Ford-Fulkerson veta: **Tok y v sieti** G=V,H,c so zdrojom z a ústím u **je maximálny** práve vtedy **keď neexistuje z-u zväčšujúca polocesta**
- Ford-Fulkerson algoritmus na hľadanie maximálneho toku v sieti:
  - 1. Zvoľ v sieti **začiatočný tok y**, napríklad nulový tok

- **2. nájdi v sieti** G s tokom y **zvačšujúcu polocestu** $\mu(z,u)$
- **3. ak zväčšujúca polocesta neexistuje, tok** y je **maximálny - STOP**
- **4.** ak **existuje** a má rezervu r **zmeň tok y nasledujúco:**
  - y(h) = y(h)   ak h **neleží** na ceste $\mu(z,u)$
  - y(h) = y(h) **+ r** ak h **leží** na ceste **v smere** svojej orientácie
  - y(h) = y(h) **- r** ak h **leží** na ceste **proti smeru** svojej orientácie
  - **GOTO2**

Algoritmus na najlacnejší tok si pozri v prezentaciach

14. Rovinné grafy. Stena rovinného diagramu, Eulerov vzorec. Maximum počtu hrán rovinného grafu. Homeomorfizmus grafov. Prototypy najjednoduchších nerovinných grafov. Kuratowského veta.

- Graf je usporiadaná dvojica G = (V,H), kde V je neprázdn konečná množina a H je množina neusporiadaných dvojíc typu {u,v} takých že u patri V, v patri V a u!=v.
- Prvky množiny V nazývame vrcholy a prvky množiny H hranami grafu G.
- Digraf - usporiadnaá dvojica - (u,v) - V - vrcholy, H - orientované hrany digrafu
- **Diagram grafu** - grafická reprezentácia grafu v priestore - jeho príslušný obrázok
- graf G=V,H nazveme **rovinný**, **ak** k nemu **existuje rovinný diagram**
- Diagram grafu nazveme **rovinný ak sa jeho hrany nepretínajú nikde inde okrem vrcholov**
- niekde aj planárny graf, nákresy ...
- **Stena** je **maximálna časť roviny**, ktorej **2 ľubovoľné body** možno spojiť **súvislou čiarou nepretínajúcou** žiadnu **hranu** rovinného diagramu.
- **2 druhy stien**, **vonkajšia** - práve **jedna stena** je **neohraničená** a **ďalšie sú vnútorné**
- **Eulerova polyedrická formula**: Nech G=V,H **je súvislý rovinný graf** a **S je množina stien** jeho rovinného diagramu, potom platí: **|S| = |H| - |V| + 2**
- **Maximum počtu hrán rovinného grafu**: Nech G=V,H je **maximálny rovinný graf s** množinou vrcholov V, **kde V >= 3**. Potom: **|H| = 3*V - 6**, teda H<=3*V-6
- úplný graf s piatimi vrcholmi K5 a úplný bipartitný graf K3,3 nie sú rovinné
- **Rozpolenie hrany - rozdelenie hrany na 2 hrany** a v bode rozdelenia **pridáme vrchol**
- grafy G=V,H a G'=V',H' sú **homeomorfné ak sú izomorfné alebo** ak konečným počtom **rozpolením hrán môžeme** dostať **izomorfné grafy**
- Kuratowski - graf G je rovinný práve vtedy ak ako podgraf neobsahuje graf homeomorfný s K5 alebo K3,3

15. Farbenie grafu, n-zafarbiteľnosť grafu, chromatické číslo grafu. Heuristiky na farbenie grafov. Praktické úlohy vedúce na riešenia úlohy farbenia grafu.

- môžeme ilustrovať na úlohe **zafarbenia štátov** na politickej mape **tak**, aby **žiadne 2 susedné štáty neboli zafarbené rovnakou farbou**.
  - **každému štátu i moru pridelíme jeden vrchol**
  - **vrcholy pospájame susedný so susedným**
  - **diagram výsledného grafu**
- **zafarbenie grafu** je **funkcia** ktorá **každému vrcholu priradí** práve jednu **farbu**
- **prípustným zafarbením** nazveme také zafarbenie, ktoré **žiadnym 2 susedným vrcholom nepriradí tú istú farbu**
- graf G = (V,H) nazveme **k-zafarbiteľným** ak jeho **vrcholy možno** prípustne **zafarbiť** k **farbami unikátne**(t. j. tak, aby ziadne dva susedne vrcholy neboli zafarbene rovnakou farbou.)
- **chromatické číslo grafu** je najmenšie prirodzené číslo **k** také, že graf G je k-zafarbiteľný. Chromatické číslo budeme značiť symbolom **X(G)**
- **Sekvenčné farbenie grafu:**
  - Nech P = v1, v2 ... vn je ľubovolná postupnosť vrcholov grafu
  - Postupne pre i=1,2 ...n urob: Zafarbi vrchol vi farbou najmenšieho čísla takou, že žiaden zo zafarbených susedov vrchola vi nie je zafarbený touto farbou.
  - algoritmus potrebuje najviac max {deg(v) | v patri V} + 1 farieb, resp (G)   1 + max{deg(v) | v 2 V}
- **Paralelné farbenie grafu:**

  - Zoraď vrcholy G do postupnosti P podľa stupňa vrchola nerastúco. Inicializuj množinu farieb F = {1}; j=1;
  - Postupne prechádzaj vrcholy vi postupnosti P a ak vrchol vi nemá suseda zafarbeného farbou j, tak ho farbou j zafarbi.
  - ak sú všetkcy vrcholy postupnosti zafarbené, STOP
  - a nie sú zvýš počet farieb, j=j+1; F=F zjednotenie {j} a goto2
- **Aplikácie**: priraďovanie rádiových frekvencií, minimalizácia počtu fáz na svetelnej križovatke, minimalizácia nákupných tašiek....