Přehled přednášky

Kapitola 5. Algoritmizace diskrétních struktur

- generování výběrů
- generátory náhodných čísel
- kombinatorická exploze

Část II Úvod do Teorie Grafů

Kapitola 6. Pojem grafu

- motivace
- definice grafu
- stupeň vrcholu v grafu

5.3. Generování výběrů

Často narazíme na úkol projít všechny výběry některého typu. Všechna

- různá zobrazení,
- variace,
- kombinace bez opakování.

Jednoduché procházení dvojic (trojic, atd...)

Všechny uspořádané dvojice indexů `i,j` projdeme dvojí smyčkou

```
for (i=0; i<N; i++)                 // dvakrát vnořený cyklus
     for (j=0; j<N; j++) {
          // zpracujeme uspořádanou dvojici
     }
```

Všechny neuspořádané dvojice indexů `i,j` projdeme podobně

```
for (i=0; i<N; i++)                 // jen "nad diagonálou"
     for (j=i+1; j<N; j++) {
          // zpracujeme neuspořádanou dvojici
     }
```

Procházení všech zobrazení

Všech n^k zobrazení k-prvkové do n-prvkové množiny

$$\texttt{map} : \{0, 1, \ldots, K-1\} \rightarrow \{0, 1, \ldots, N-1\}$$

projdeme kódem: Nemusíme sestavovat k vnořených cyklů!

```
int i, map[K];
map[i = 0] = -1;
while (i>=0) {
     if (++map[i]>=N)              // zvyš o 1
          { i--; continue; }
     if (++i<K)                    // 'nuluj' další prvky
          { map[i] = -1; continue; }
     // zpracujeme zobrazení (map[0], ..., map[K-1])
     i--;
}
```

Pro každou volbu testujeme:

- Zda už překročila rozsah `N`, pak se vracíme na předchozí úroveň.
- Zda už byla poslední volba `K`-tého prvku, jinak na další úroveň.

Procházení všech k-prvkových variací (bez opakování) z n prvků

```
int i, j, select[K];
select[i = 0] = -1;
while (i>=0) {
     if (++select[i]>=N)          // zvyš o 1
          { i--; continue; }
     for (j=0; j<i; j++)          // opakuje se?
          if (select[i]==select[j])  break;
     if (j<i)  continue;          // opakované vynech
     if (++i<K)                   // 'nuluj' další prvky
          { select[i] = -1; continue; }
     // zpracujeme variaci (select[0],...,select[K-1])
     i--;
}
```

Pro každou volbu testujeme:

- Zda už překročila rozsah `N`, pak se vracíme na předchozí úroveň.
- Zda se neopakuje předchozí prvek volby, jinak výběr přeskočíme.
- Zda už byla poslední volba `K`-tého prvku, jinak na další úroveň.

Procházení všech kombinací

Projdeme všechny k-prvkové kombinace (bez opakování) z n prvků. Podobné předchozímu příkladu, avšak nyní všechny výběru generujeme seřazené podle velikosti. Proto každou kombinaci obdržíme jen jednou.

```
int i, j, select[K];
select[i = 0] = -1;
while (i>=0) {
    if (++select[i]>=N)             // zvyš o 1
        { i--; continue; }
    if (++i<K) {
        select[i] = select[i-1];    // zde seřazené!
        continue;
    }
    // zpracujeme kombinaci (select[0],...,select[K-1])
    i--;
}
```

(Není třeba testovat opakované výběry, protože prvky kombinace jsou ostře seřazené.)

5.4. Generátory náhodných čísel

Skutečně náhodná posloupnosti bitů v počítači.

Kde všude potřebujeme používat náhodná čísla/bity?

- Vytváření náhodných (velkých) privátních klíčů (v SSL certifikátech). Použití náhodného hesla při šifrování SSL přenosu (zde musí být heslo skutečně náhodné, jinak by mohlo být uhodnuto!).

- Řešení kolizí paketů na Ethernetu náhodně dlouhým čekáním s příštím vysíláním.

- Využití při *pravděpodobnostních algoritmech*, využívají náhodných bitů pro výrazné statistické urychlení běhu výpočtu.

- Při statistické analýza algoritmů a reálných procesů, různé modelování chaotických fyzikálních dějů, atd.

Typy náhodných generátorů

Primitivní pseudonáhodné generátory

Využívají aritmetické vzorce typu

$$x := (A \cdot x + B) \bmod C.$$

Opakovaně iterujeme a vybrané bity x vytvářejí posloupnost.

Nevýhody: *velká závislost* bitů na předchozích iteracích a *snadná předvídatelnost*.

Pseudonáhodné generátory s vnějším vstupem

Podobné vzorce jako předchozí typ, navíc přidávají vstupy z *vnějších fyzických procesů* (prodlevy stisků kláves, zpoždění operací disku, statistika síťových paketů, atd.)

Problémy: závislost na vnějších podmínkách, *možnost ovlivnění* vnějšími podmínkami, velká „cena“ každého bitu.

Hardwarové náhodné generátory

Založeny na různých *kvantových šumech* (třeba přechodové stavy polovodičů).

Problémy: převod šumu na posloupnost uniformních bitů, důvěra v kvantovou mechaniku.

5.5. Kombinatorická exploze

Při programátorském řešení problémů spjatých s diskrétní matematikou často používáme algoritmy typu:

Projdi všechny možnosti a podle nich rozhodni.

Často však narážíme na fenomén zvaný *exponenciální kombinatorická exploze*.

- rychlost růstu funkce faktoriál.
- z Indie: příběh se zrnky obilí na polích šachovnice

Pokud počet prohledávaných možností roste exponenciálně, tak i při zvětšení vstupu o 1 se potřebný čas výpočtu zmnohonásobí.
Stává se, že výpočet pro vstup o velikosti 10 zvládne stroj s procesorem 386, ale vstup o velikosti 15 nelze spočítat ani na nejvýkonnějších počítačích.

Mějte tento fenomén na paměti, až budete programovat procházení možností „hrubou silou“!

Vhodnou volbou algoritmu, datové struktury či omezujících podmínek můžeme dosáhnout výrazného zrychlení výpočtu.

Část II
Úvod do Teorie Grafů

Teorie grafů je jednou z mladších disciplin matematiky

- L. Euler: Problém mostů města Královce 1736
- První monografie 1936

Populární problémy teorie grafů

- problém čtyř barev
- nejkratší cesta v grafu
- maximální tok v grafu

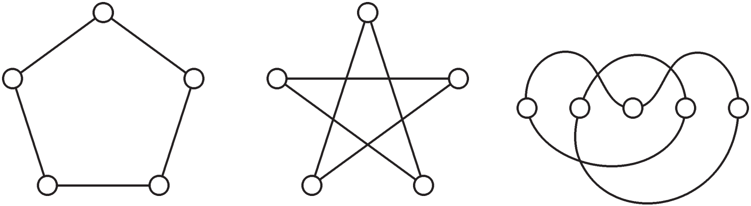
Motivace

Grafy tvoří jednu z nejdůležitějších disciplin diskrétní matematiky

- pomocí grafů můžeme dobře popsat praktické problémy
- intuitivně pochopitelné
- snadná implementace v počítači

Graf se skládá z

- vrcholů (uzlů) – „puntíky“ body v obrázku
- hran – „spojnice“ v obrázku



Obrázek: *Různá nakreslení téhož grafu.*

6.1. Definice grafu

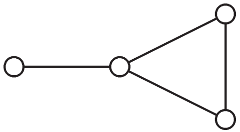
Definice 6.1. *Graf G* (*jednoduchý* graf, *obyčejný* graf)
je uspořádaná dvojice $G = (V, E)$, kde V je množina *vrcholů* a E je množina *hran* – množina (některých) dvouprvkových podmnožin množiny V.

Prvky množiny V nazýváme *vrcholy* a značíme je malými písmeny u, v, ….
Prvky množiny E nazýváme *hrany*. Hrana mezi vrcholy u a v je dvouprvková podmnožiny $\{u, v\}$, značíme ji zkráceně uv.

Máme-li dán graf G, rozumíme symbolem $V(G)$ množinu vrcholů grafu G a symbolem $E(G)$ množinu hran grafu G.

Poznámka
Graf $G(V, E)$ můžeme chápat jako speciální relaci E na množině V, kde E je *ireflexivní* a *symetrická*.

Příklad:
Graf $G(V, E)$, kde $V = \{1, 2, 3, 4\}$
a $E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}\}$
znázorníme obrázkem:



Běžné třídy grafů

Grafy můžeme zadat

- množinami V a E
- obrázkem
- jménem a parametrem (parametry)

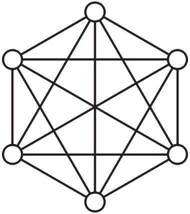
Některé typy grafů se často vyskytují a dokonce mají své jméno:

- cesty
- stromy
- housenky
- lízátka
- knihy
- ...

Úplný (kompletní) graf K_n

Graf na n vrcholech ($n \geq 1$), který obsahuje všech $\binom{n}{2}$ hran se nazývá *úplný* nebo také *kompletní* graf a značí se K_n.

$$G(V,E):\ V = \{1,2,\ldots,n\}\ \wedge\ E = \{ij : i,j = 1,2,\ldots n \wedge i \neq j\}$$

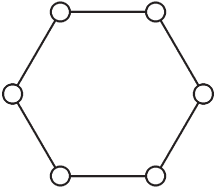


Obrázek: *Graf* K_6.

Kružnice (cyklus) C_n

Graf na n vrcholech ($n \geq 3$), které jsou spojeny hranami do jednoho cyklu se nazývá *kružnice* nebo také *cyklus* na n vrcholech a značí se C_n.

$$G(V, E): \; V = \{1, 2, \ldots, n\} \; \wedge \; E = \{i(i+1) : i = 1, 2, \ldots n-1\} \cup \{1n\}$$



Obrázek: *Graf* C_6.

Cyklu délky tři říkáme *trojúhelník*.

Cesta P_n

Graf na $n+1$ vrcholech ($n \geq 0$), které jsou spojeny po řadě hranami se nazývá *cesta* a značí se P_n.

$$G(V, E): \; V = \{1, 2, \ldots, n+1\} \; \wedge \; E = \{i(i+1) : i = 1, 2, \ldots n\}$$

Obrázek: *Graf* P_5.

Pozor na jiné (obvyklé) značení:
index = počet vrcholů

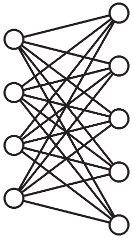
V některých knihách se používá termín *had*.

Úplný (kompletní) bipartitní graf $K_{m,n}$

Graf, který má vrcholovou množinu rozdělenu na dvě disjunktní podmnožiny M a N ($|M| = m \geq 1$, $|N| = n \geq 1$) a který obsahuje všech $m \cdot n$ hran uv takových, že $u \in M$ a $v \in N$ se nazývá *úplný bipartitní* nebo také *kompletní bipartitní* graf a značí se $K_{m,n}$.

$$G(M \cup N, E) : M = \{u_1, u_2, \ldots, u_m\} \ \wedge \ N = \{v_1, v_2, \ldots, v_n\}$$

$$E = \{u_i v_j : i = 1, 2, \ldots, m \wedge j = 1, 2, \ldots n\}$$



Obrázek: *Graf* $K_{4,5}$.

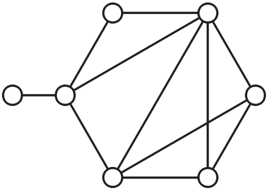
6.2. Stupeň vrcholu v grafu

Máme-li hranu uv, pak vrcholy u a v nazýváme *koncové* vrcholy hrany uv.

Také říkáme, že u a v jsou *incidentní* s hranou uv ($u \in \{u, v\}$).

Dva různé vrcholy, které jsou koncovými vrcholy téže hrany nazýváme *sousední*. Pokud taková hrana neexistuje, nazýváme vrcholy *nesousední* nebo také *nezávislé*.

Definice 6.2. *Stupeň vrcholu* v grafu G
je počet hran, se kterými je daný vrchol v incidentní. Stupeň vrcholu v v grafu G značíme $d_G(v)$ (nebo také $deg_G(v)$).



Obrázek: *Graf se stupni* $1, 2, 3, 3, 4, 4, 5$.

Věta 6.3.

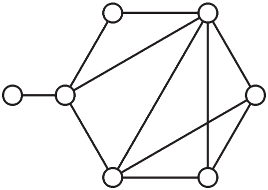
Součet stupňů v grafu je vždy sudý a je roven dvojnásobku počtu hran:

$$\sum_{v \in V(G)} deg_G(v) = 2|E(G)|.$$

Důkaz

Levou i pravou stranu rovnice spočítáme přes všechny hrany. Při sčítání stupňů vrcholů v grafu započítáme každou hranu dvakrát – jednou každý její koncový vrchol. Proto je součet stupňů vždy sudý a je roven dvojnásobku počtu hran.

□



Obrázek: *Graf se stupni* $1, 2, 3, 3, 4, 4, 5$.

Vlastnosti stupňů

Mějme graf G s vrcholy $v_1, v_2, \ldots, v_n$.
Posloupnost $\deg(v_1), \deg(v_2), \ldots, \deg(v_n)$ nazýváme *stupňovou posloupností* grafu G. Někdy značíme také $(\deg(v_1), \deg(v_2), \ldots, \deg(v_n))$.
(obvykle uspořádaná posloupnost)

Ne každá posloupnost je stupňovou posloupností nějakého grafu. Jak poznat?

Věta 6.4. Havlova–Hakimiho

Nechť $d_1 \leq d_2 \leq \ldots \leq d_n$ je posloupnost přirozených čísel. Pak graf s n vrcholy se stupňovou posloupností

$$d_1, d_2, \ldots, d_n$$

existuje právě tehdy, když existuje graf s $n-1$ vrcholy se stupňovou posloupností

$$d_1, d_2, \ldots, d_{n-d_n-1}, d_{n-d_n} - 1, \ldots, d_{n-1} - 1.$$

Protože se zajímáme jen o konečné grafy, umíme rekurzivně o každé posloupnosti v konečném čase rozhodnout, zda je stupňovou posloupností nějakého grafu. Navíc budeme umět nějaký takový graf sestrojit (nemusí být jediný!).

Příklady:

Existuje graf se stupňovou posloupností $(1, 1, 3, 3, 3)$?
Neexistuje, dle Věty 6.3., alias „Věty 1.1.“.

Existuje graf se stupňovou posloupností $(1, 3, 3, 3)$?
Ne, ukážeme užitím Věty 6.4.
Ze stupňové posloupnosti $(1, 3, 3, 3)$ dostaneme

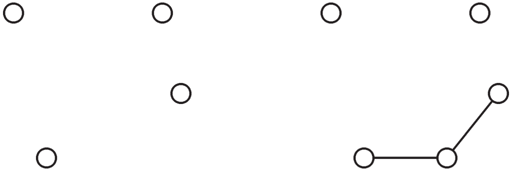
$$(1, 3, 3, 3) \stackrel{V6.4.}{\rightarrow} (0, 2, 2) \stackrel{V6.4.}{\rightarrow} (-1, 1)$$

Což evidentně není stupňová posloupnost.

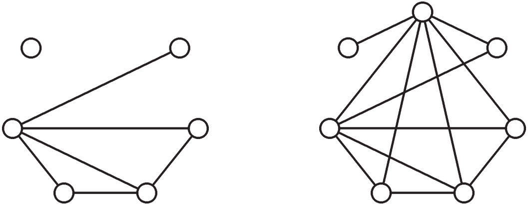
Existuje graf se stupňovou posloupností $(1, 2, 3, 3, 4, 5, 6)$?
Ano, ukážeme užitím Věty 6.4. a navíc graf zkonstruujeme. Ze stupňové posloupnosti $(1, 2, 3, 3, 4, 5, 6)$ dostaneme

$$(1, 2, 3, 3, 4, 5, 6) \stackrel{V6.4.}{\rightarrow} (0, 1, 2, 2, 3, 4) \stackrel{V6.4.}{\rightarrow} (0, 0, 1, 1, 2) \stackrel{V6.4.}{\rightarrow} (0, 0, 0, 0)$$

Konstrukce grafu: přidáváním vrcholů.



Obrázek: *Grafy se stupňovými posloupnostmi* $(0,0,0,0)$, $(0,0,1,1,2)$.



Obrázek: *Grafy se stupňovými posloupnostmi* $(0,1,2,2,3,4)$, $(1,2,3,3,4,5,6)$.

Otázky:

- Je každá neklesající posloupnost přirozených čísel grafová?
- Jak poznáme, zda posloupnost je stupňovou posloupností nějakého grafu?
- Je graf určen stupňovou posloupností jednoznačně?
- Kolik existuje grafů s danou stupňovou posloupností?

Příště

Kapitola 6. Pojem grafu

- stupeň vrcholu v grafu
- podgrafy a izomorfismus
- orientované grafy a multigrafy
- implementace grafů

Kapitola 7. Souvislost grafu

- motivace
- spojení mezi vrcholy, komponenty grafu
- prohledávání grafu
- vyšší stupně souvislosti
- Eulerovské grafy „jedním tahem“