Orientované grafy

Založeno na materiálech od pmikus

Obsah

Orientovaný graf

Orientovaný graf

je dvojice $G = (U, H)$, kde

U je konečná množina uzlů (vrcholů)
$H = \{(u, v) \mid u, v \in U\}$ je konečná množina orientovaných hran.

- mezi dvěma uzly mohou být dvě hrany v opačných směrech

Obecný orientovaný graf

může mít více hran mezi stejnými uzly
je trojice $G = (U, H, \epsilon)$, kde

U je konečná množina uzlů (vrcholů)
H je konečná množina hran
$\epsilon : H \to \{(u, v) \mid u, v \in U \wedge u \neq v\}$ je zobrazení přiřazující každé hraně dvojici uzlů

Ohodnocený orientovaný graf

orientovaný graf ve kterém je každé hraně přiřazena její cena (číslo)

Turnaj

je orientovaný graf, kde mezi každými dvěma různými uzly existuje právě jedna orientovaná hrana a každý uzol má smyčku.

Výstupní stupeň uzlu

je počet hran, které z uzlu vystupují, $deg_-(u)$.

Vstupní stupeň uzlu

je počet hran, které do uzlu vstupují, $deg_+(u)$.

Koncový uzel

Pokud $deg_-(u) = 0$, jedná se o koncový uzel.

Počáteční uzel

Pokud $deg_+(u) = 0$, jedná se o počáteční uzel.

Eulerovský graf

je orientovaný graf, kde existuje uzavřený **tah**, který obsahuje všechny jeho orientované hrany.

- Souvislý orientovaný graf je Eulerovský, pokud platí: $\forall u \in U : \ deg_+(u) = deg_-(u)$

Symetrizace/orientace grafu

- převody mezi orientovaným a neorientovaným grafem

Symetrická orientace grafu

je nahrazení každé hrany v obyčejném grafu dvěma orientovanými hranami opačného směru. Vytvoří se tak jediný orientovaný graf.

Orientace grafu

je nahrazení každé hrany v obyčejném grafu jedinou orientovanou hranou. Na rozdíl od symetrické orientace je takto možno vytvořit více orientovaných grafů.

Symetrizace grafu

je nahrazení všech hran spojujících dané dva uzly jedinou neorientovanou hranou. Vytvoří se tak jediný obyčejný graf.

Souvislost

Orientovaný sled, tah, cesta a kružnice jsou definovány analogicky k obyčejným.

Souvislost

Orientovaný graf je souvislý, pokud symetrizací vznikne souvislý obyčejný graf.

Silná souvislost

Orientovaný graf je silně souvislý, pokud pro libovolné dva uzly existuje orientovaná cesta.

Cesty

Délka hrany

je reálné číslo $l(h)$ přiřazené ke každé hraně grafu.

Délka cesty

je součet všech délek hran na dané cestě.

Minimální délka cesty

je nejmenší možná délka cesty mezi uzly u a v, značíme $d(u,v)$, pokud cesta neexistuje, pak $d(u,v) = \infty$.

Dijkstrův algoritmus pro nalezení minimální cesty

Počáteční uzel s

je uzel, od kterého se délky cesty počítají.

Horní odhad vzdálenosti

z uzlu s do uzlu v je číslo $D(v) \geq d(s,v)$, které je horním uzávěrem všech délek možných cest.

Předchozí uzel

$\pi(v)$ je uzel, který uzlu v bezprostředně předchází na aktuálně počítané cestě. Pro zatím neexistující cestu platí $\pi(v) = \emptyset$.

Následné uzly

jsou množina uzlů $N(v) = \{w \in U \mid (v,w) \in H\}$, tedy uzlů, které jsou přímo spojeny hranou počínající v uzlu v.

Další pojmy

$p(s,v)$ - aktuální cesta,

$d(s.v)$ - délka aktuální cesty,

$S \subseteq U$ - množina uzlů, pro které již cesta je spočítána,

$Q = U - S$ - dosud neprozkoumané uzly.

Algoritmus

- **1) Inicializace**

  $\forall u \in U: \ \pi(u) = \emptyset$ *(tj. všechny prohledávané cesty jsou na začátku prázdné)*
  $D(s) = 0, D(u) = \infty$ *(tj. počáteční uzel má horní odhad vzdálenosti 0, ostatní uzly nekonečno)*
  $S = \emptyset$ *(tj. zatím jsme žádné uzly neprozkoumali)*

- **2) Test konce**

Pokud $S = U$, přechod na 5. *(tj. končíme pokud jsme prozkoumali všechny uzly)*

- **3) Nalezení uzlu s definitivní cestou**

  Z Q přesuneme do S uzel v s nejnižší hodnotou $D(v)$. *(tj. vybereme neprozkoumaný uzel s nejnižším horním dohadem vzdálenosti a přesuneme ho mezi prozkoumané)*
  Pokud pro všechny $u \in Q$ platí $D(u) = \infty$, přechod na 5. *(tj. pokud všechny neprozkoumané uzly mají horní odhad vzdálenosti nekonečno jdeme na 5)*

- **4) Zlepšení horních odhadů**

  $\forall w \in N(v) \cap Q$ takový, že $D(w) > D(v) + l((v,w))$, :: *(tj. vezmeme všechny následné uzly vybraného uzlu, které ještě nebyly prozkoumány)*
  položíme $D(w) = D(v) + l((v,w))$ a $\pi(w) = v$, přechod na 2. *(tj. pokud celková dékla cesty do těchto uzlů přes uzel v je kratší nastavíme odhad vzdálenosti na tuto délku)*
  (tj.)

- **5) Konstrukce výstupu**

  Do uzlů, které zůstaly ve Q žádná cesta z s neexistuje.
  Pro ostatní uzly položíme $d(s,v) = D(v)$
  cestu minimální délky sestrojíme obrácením cesty $v \rightarrow \pi(v) \rightarrow \pi(\pi(v)) \rightarrow \ldots \rightarrow s$.
  (tj. cestu sestavíme tak, že jdeme z koncového po uzlech s nejmenším odhadem)

- **Dijkstrův algoritmus nefunguje, pokud v grafu jsou hrany se zápornou délkou.**

Floyd-Warshallův algoritmus pro nalezení minimální cesty

- Asymptotická časová složitost algoritmu je O(N^3) a paměťová je O(N^2).
- Tento algoritmus funguje i se zápornými hranami. Při nalezení kružnice záporné délky tuto odhalí (diagonální prvek matice bude záporný).
- Vstup (délky hran) je zadán maticí A (sloupec a řádek označuje počáteční/koncový uzel hrany).
- Dále máme matici posloupností P, na počátku obsahuje každý prvek pouze číslo svého sloupce.
- Postupnými iteracemi (n iterací, kde $n = |U|$) se vypočítávají další kroky matic až se dojde do koncového tvaru a matice vzdáleností obsahuje minimální vzdálenosti.
- V každé iteraci se prvky matic A^j přepočítají:
  - pokud $a_{ik}^{j-1} \leq a_{ij}^{j-1} + a_{jk}^{j-1}$ pak $a_{ik}^{j} = a_{ik}^{j-1}, p_{ik}^{j} = p_{ik}^{j-1}$
  - pokud $a_{ik}^{j-1} > a_{ij}^{j-1} + a_{jk}^{j-1}$ pak $a_{ik}^{j} = a_{ij}^{j-1} + a_{jk}^{j-1}, p_{ik}^{j} = p_{ij}^{j-1}$

Jinak 1

Pro iteraci j sledujeme matici A^{j-1} a to **pouze** její j-tý řádek a j-tý sloupec (tedy takový kříž). Pro všechny prvky z A^{j-1} porovnáváme jejich hodnotu s **průmětem** na tento kříž (tedy se součtem s odpovídajícími prvky na stejném řádku a sloupci v kříži). Pokud je součet větší než hodnota prvku, opíšeme hodnoty obou matic. Pokud je součet menší, v matici A^j zapíšeme součet a v matici P^j zapíšeme hodnotu $j-1$.

Jinak 2

Máme matici A (řádky a sloupce popsané uzly) určující nejkratší existující cestu mezi příslušnými uzly. Na začátku ji inicializujeme tak, aby na příslušných místech byly zapsány délky hran mezi danými uzly. Pak matici aktualizujeme tolikrát kolik je uzlů grafu. Každá aktualizace probíhá tak, že jedeme postupně po jednotlivých uzlech v pořadí v jakém je jimi popsána matice. Pro všechny dvojice uzlů i, j pak zkontrolujeme zda cesta přes k-tý uzel není kratší než aktuálně zapsaná hodnota. Tj. porovnáme současnou hodnotu a součet cest i do k a k do j a zapíšeme minimum.

Jinak 3

```
1 // Předpokládáme funkci cenaHrany(i, j) vracející cenu hrany z i do j
2 // (pokud hrana neexistuje, cenaHrany = nekonečno)
3 // Dále, N je počet vrcholů a cenaHrany(i, i) = 0
4
```

```
 5 int cesta[][]; // Dvourozměrné pole. V každém kroku algoritmu je cesta[i][j]
 6                // nejkratší cesta z i do j použitím 1. až k-té hrany.
 7                // Všechny hrany cesta[i][j] jsou inicializovány funkcí
 8                // cenaHrany(i,j);
 9
10 procedure FloydWarshall ()
11    for k := 1 to N
12    begin
13       foreach (i,j) in (1..N)
14       begin
15          cesta[i][j] = min(cesta[i][j], cesta[i][k] + cesta[k][j]);
16       end
17    end
18 endproc
```

Kategorie: Státnice MAT | Státnice 2011 | Matematické struktury v informatice

Stránka byla naposledy editována 29. 5. 2011 v 10:02.