Prieskumy na grafoch

Tarryho algoritmus (labyrintový algoritmus)

1° Nikdy neprechádzame dvakrát v tom istom smere po tej istej hrane

2° Nachádzajúc sa vo vrchole *v* nikdy nevyberieme hranu, ktorá nás do vrchola priviedla ako prvá, pokiaľ existuje možnosť iného výberu.

Príklad :

Využitie : – riešenie labyrintových úloh (nájdenie východu z labyrintu, nájdenie pokladu v labyrinte)

– zistenie súvislosti grafu

– zistenie počtu komponentov súvislosti v grafe

Definícia : Nech G = (V, H) je hranovoohodnotený graf, resp. digraf a u, v∈ V. Potom ***minimálnou cestou*** resp. ***dráhou*** z vrchola *u* do vrchola *v* rozumieme takú cestu, resp. dráhu C = $u, h_1, v_1, ..., v_{p-1}, h_p, v$ zo všetkých ciest, resp. dráh z vrchola *u* do *v* v grafe G, ktorá minimalizuje číslo

$$o(C)=\sum_{i=1}^{p} o(h_i)$$

Číslo o(C) nazývame ohodnotenie (dĺžkou) cesty, resp. dráhy

Algoritmus nájdenia minimálnej cesty

1° Každému vrcholu grafu v_i priradíme hodnotu $t(v_i) = \infty$ pre $1 \leq i \leq n$ a $t(v_0) = 0$.

2° V uvažovanom grafe hľadáme takú hranu (v_i, v_j), pre ktorú platí

$$t(v_j) - t(v_i) > o((v_i, v_j)$$

Ak takáto hrana (v_i, v_j) existuje, tak nahradíme hodnotu $t(v_j)$

$$t(v_j) \leftarrow t(v_i) + o((v_i, v_j)$$

a pokračujeme v uvedenom postupe, kým existuje aspoň jedna hrana v grafe, pre ktorú platí uvedená nerovnosť

3° Ak neexistuje hrana (v_i, v_j), pre ktorú by platilo $t(v_j) - t(v_i) > o((v_i, v_j)$

Potom hodnota $t(v_n)$ udáva dĺžku minimálnej cesty.

4° Minimálnu cestu zostrojíme takto :

- Určíme vrchol v_{k1} a hranu (v_{k1}, v_n), pre ktorú by platilo $t(v_n) - t(v_{k1}) = o((v_{k1}, v_n))$
- Ďalej určíme vrchol v_{k2} a hranu (v_{k2}, v_{k1}), pre ktorú by platilo $t(v_{k1}) - t(v_{k2}) = o((v_{k2}, v_{k1}))$
- Pokračujeme dovtedy, kým nenarazíme na vrchol $v_{kr} = v_0$ a hranu $(v_{kr}, v_{kr\text{-}1})$, pre ktorú platí $t(v_{kr\text{-}1}) - t(v_{kr}) = o((v_{kr}, v_{kr-1}))$

Hľadaná minimálna cesta je $v_0 = v_{kr}; (v_{kr}, v_{kr-1}); v_{kr-1}; \ldots ; v_{k2}; (v_{k2}, v_{k1}); v_{k1}; (v_{k1}, v_n); v_n$.

Algoritmus nájdenia minimálnej cesty – Dijkstrov algoritmus

1° Každému vrcholu grafu v_i priradíme hodnotu $t(v_i) = \infty$ pre $1 \leq i \leq n$ a začínajúcemu vrcholu v_0 $t(v_0) = 0$. Označme množinu V^* trvalo označených vrcholov. $V^* = \emptyset$

2° Vyberieme vrchol $v_r \in V - V^*$ (z množiny neoznačených) s vlastnosťou

$$t(v_r) = \min \{ t(v); v \in V - V^* \}$$

Ak $t(v_r) = \infty$ potom neexistuje cesta z v_0 do v_r

$t(v_r) < \infty$ potom pridáme v_r do V^*

Ak $V = V^*$ potom $t(v_r)$ je dĺžka minimálnej cesty z v_0 do v_r.

ináč pokračuj 3°

3° Pre všetky $v \in V - V^*$ ak $t(v) > t(v_r) + o((v_r,v))$

potom $t(v) \leftarrow t(v_r) + o((v_r,v))$

pokračuj 2°

Definícia : Nech G = (V, H) je hranovoohodnotený graf, resp. digraf. Potom maticou vzdialenosti rozumieme maticu D typu $n \times n$ s prvkami $d_{i,j}$

$$d_{ij} = \begin{cases} 0 & \text{ak } i = j \\ \infty & \text{ak } i \neq j \text{ a neexistuje cesta(dráha) z } v_i \text{ do } v_j \\ o(c_{ij}) & \text{ak } i \neq j \text{ a } o(c_{ij}) \text{ je dlzka minimalnej cesty z } v_i \text{ do } v_j \end{cases}$$

Floydov algoritmus

1° Položíme $d_{ij} = \begin{cases} 0, & \text{ak } i = j \\ \infty, & \text{ak neexistuje hrana } (v_i, v_j) \\ o(h) & \text{ak existuje hrana } h = (v_i, v_j) \end{cases}$

2° Položíme $k = k + 1$

3° Pre všetky $i \neq k$ a $j \neq k$ také, že $d_{ik} \neq \infty$ a $d_{kj} \neq \infty$ vypočítame

$$d_{ij} \text{ ako min } \{ d_{ij} ; (d_{ik} + d_{kj}) \}$$

4° Ak $k = p$, tak aktuálna matica D = (d_{ij}) je maticou vzdialeností daného grafu

Ak $k < p$, tak návrat na 2°.