

DAG A DP V NĚM

Petr Ryšavý

16. září 2024

Katedra počítačů, FEL, ČVUT

TOPOLOGICKÉ OČÍSLOVÁNÍ

Definice (Topologické očíslování) *Topologické očíslování*

orientovaného grafu $G = (V, E)$ je bijekce $f : V \mapsto 1, 2, \dots, n$ taková, že

$$(u, v) \in E \quad \Rightarrow \quad f(u) < f(v).$$

Proč nás zajímá topologické očíslování?

- V jakém pořadí je třeba vystudovat předměty, abychom měli splněné prerekvizity.
- V jakém pořadí skládat výrobek.
- Aplikace v mnoha algoritmech.

Věta *Graf má topologické očíslování právě tehdy, když neobsahuje orientovaný cyklus.*

Grafy bez orientovaných cyklů se nazývají acyklické, angličtině *directed acyclic* (DAG).

Naivní přístup

- Každý DAG musí obsahovat vrchol, ze kterého nevedou hrany
- Tento vrchol odstraníme (s odpovídajícími hranami) a opakujeme
- Všechny dosažitelné vrcholy musí být v číslování, abychom mohli vrchol odstranit

function DUMMY-TOPSORT(*graph*) **returns** topological sort f

$v \leftarrow$ sink vertex

$f(v) = n$

DUMMY-TOPSORT(*graph* $\setminus$ $\{v\}$)

end function

Prohledáváním do hloubky

function **TOPOLOGICAL-ORDERING**(*graph*) **returns** topological ordering f of the graph

$f \leftarrow$ empty ordering

current-label = VERTICES-COUNT(*graph*)

for all *node* $\in$ *graph* **do**

if UNVISITED(*node*) **then**

 DEPTH-FIRST-SEARCH(*graph*, *node*)

end if

end for

end function

function **DEPTH-FIRST-SEARCH**(*graph*, *s*)

 MARK-VISITED(*s*)

for all edges (*s*, *v*) **do**

if UNVISITED(*v*) **then**

 DEPTH-FIRST-SEARCH(*graph*, *v*)

end if

end for

$f(s) = \textit{current-label}$

$\textit{current-label} = \textit{current-label} - 1$

end function

- Běží v $\mathcal{O}(m + n)$.

NEJKRATŠÍ CESTY V DAG

Hledání nejkratší cesty v DAG

Nechť $G = (V, E)$ je vážený orientovaný acyklický graf a s je vrchol z V .
Nalezněte délky nejkratších cest L z s do každého $v \in V$.

- Definujeme jak řešit větší problém pomocí známých řešení podproblémů.
- Podobné rekurzi, ale
 - jdeme odspodu, ne shora a
 - řešení podproblémů máme první.

Známe-li nejkratší cesty do předchůdců vrcholu v , pak stačí vybrat

$$\min_{(u,v) \in E} L(u) + c_{u,v}.$$

Vrcholy projdeme v topologickém pořadí.

function SSSP-DAG(*graph*, *s*) **returns** shortest path to each *v*

$$\text{dist}[v] = \begin{cases} \infty, & v \neq s, \\ 0, & v = s. \end{cases}$$

spočti topologické očíslování grafu

for all $v \in V$ v topologickém pořadí **do**

$$\text{dist}[v] = \min_{(u,v) \in E} \{ \text{dist}[u] + c_{u,v} \}$$

end for

end function

- Hodnoty můžeme „tlačit“ vpřed

function SSSP-DAG(*graph*, *s*) **returns** shortest path to each *v*

$$\text{dist}[v] = \begin{cases} \infty, & v \neq s, \\ 0, & v = s. \end{cases}$$

spočti topologické očíslování grafu

for all $(v, w) \in E$ v topologickém pořadí **do**

$$\text{dist}[w] = \min\{\text{dist}[w], \text{dist}[v] + c_{v,w}\}$$

end for

end function

- Nejkratší cesta mezi dvěma vrcholy s a t

- Nejkratší cesta mezi dvěma vrcholy s a t
- Nejdelší cesta v DAG
 - Dva přístupy

- Nejkratší cesta mezi dvěma vrcholy s a t
- Nejdelší cesta v DAG
 - Dva přístupy
 - min nahradíme za max
 - ceny vah vynásobíme -1

- Nejkratší cesta mezi dvěma vrcholy s a t
- Nejdelší cesta v DAG
 - Dva přístupy
 - min nahradíme za max
 - ceny vah vynásobíme -1
- Počet cest v grafu

Nejdelší neklesající podposloupnost v $\mathcal{O}(n^2)$

Mějme posloupnost čísel $a_1, a_2, \dots, a_n$. Nalezněte nejdelší posloupnost indexů $i_1 < i_2 < \dots < i_k$, že platí

$$a_{i_1} \leq a_{i_2} \leq \dots \leq a_{i_k}.$$

5, 1, 6, 4, 8, 3, 2, 1, 5, 6, 8

[Ale kde je ten DAG?]

Mějme dvě slova $A = a_1a_2 \cdots a_n$ a $B = b_1b_2 \cdots b_m$. Nalezněte minimální počet operací INSERT, DELETE a REPLACE pro převedení A na B .

- Čas běhu je $\mathcal{O}(m + n)$ (pro porovnání Dijkstra $\mathcal{O}((m + n) \log n)$)
- Umíme řešit i jinak NP-úplné problémy nejkratších/nejdelsích cest.

- Zkuste nejprve naimplementovat hledání nejkratších/nejdélších cest sami.
- <https://www.geeksforgeeks.org/find-longest-path-directed-acyclic-graph/>
- <https://www.geeksforgeeks.org/shortest-path-for-directed-acyclic-graphs/>
- <https://cs.stackexchange.com/questions/3078/algorithm-that-finds-the-number-of-simple-paths-from-s-to-t-in-g>

- heavily inspired by Tim Roughgarden's online courses,
<http://theory.stanford.edu/~tim/videos.html>
- Robert Sedgewick and Kevin Wayne, Algorithms,
<http://algs4.cs.princeton.edu/home/>, namely
<http://algs4.cs.princeton.edu/44sp/>
- Halim, S., Halim, F., Skiena, S. S., & Revilla, M. A. (2013).
Competitive Programming 3. Lulu Independent Publish.

DĚKUJI ZA POZORNOST.
ČAS NA OTÁZKY!