KIV/PRO
Topologické řazení
30. listopadu 2023

Zadání

Prostudujte pro zvolený problém existující metody řešení. Vyberte jednu z nich nebo navrhněte vlastní, implementujte a ověřte na experimentech. Postup a výsledky popište ve zprávě.

Popis problému

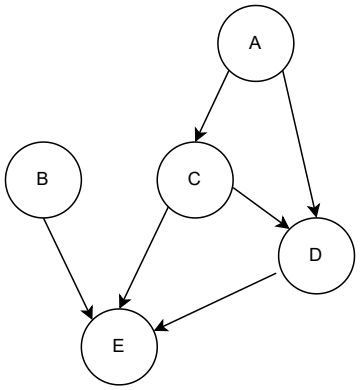
Je dán orientovaný acyklický graf $G = (V, E)$. Úkolem je najít uspořádání vrcholů množiny V tak, že pro každou hranu (i, j) patřící do množiny E bude platit, že vrchol i je nalevo od vrcholu j. Jinak řečeno topologické řazení seřadí vrcholy vstupního grafu tak, že pokud vrchol má nějakého předka, tak je vždy nalevo od daného vrcholu v seřazeném poli vrcholů [5].

Existující algoritmy

Pro topologické řazení jsou nejznámější dva algoritmy a to Kahnův algoritmus [3] a DFS upravené pro topologické řazení [2]. Oba zmíněné algoritmy jsou relativně jednoduché na implementaci. DFS je implementován pomocí zásobníku a Kahnův algoritmus pomocí fronty. Časová složitost obou algoritmů je $O(V+E)$ kde V je počet vrcholů a E je počet hran a paměťová složitost je taky u těchto algoritmů stejná a to $O(V)$ [4]. V této semestrální práci bude implementován Kahnův algoritmus. Důvod k tomuto rozdělení je jednoduší a přehlednější implementace, pokud se chceme vyhnout rekurzi.

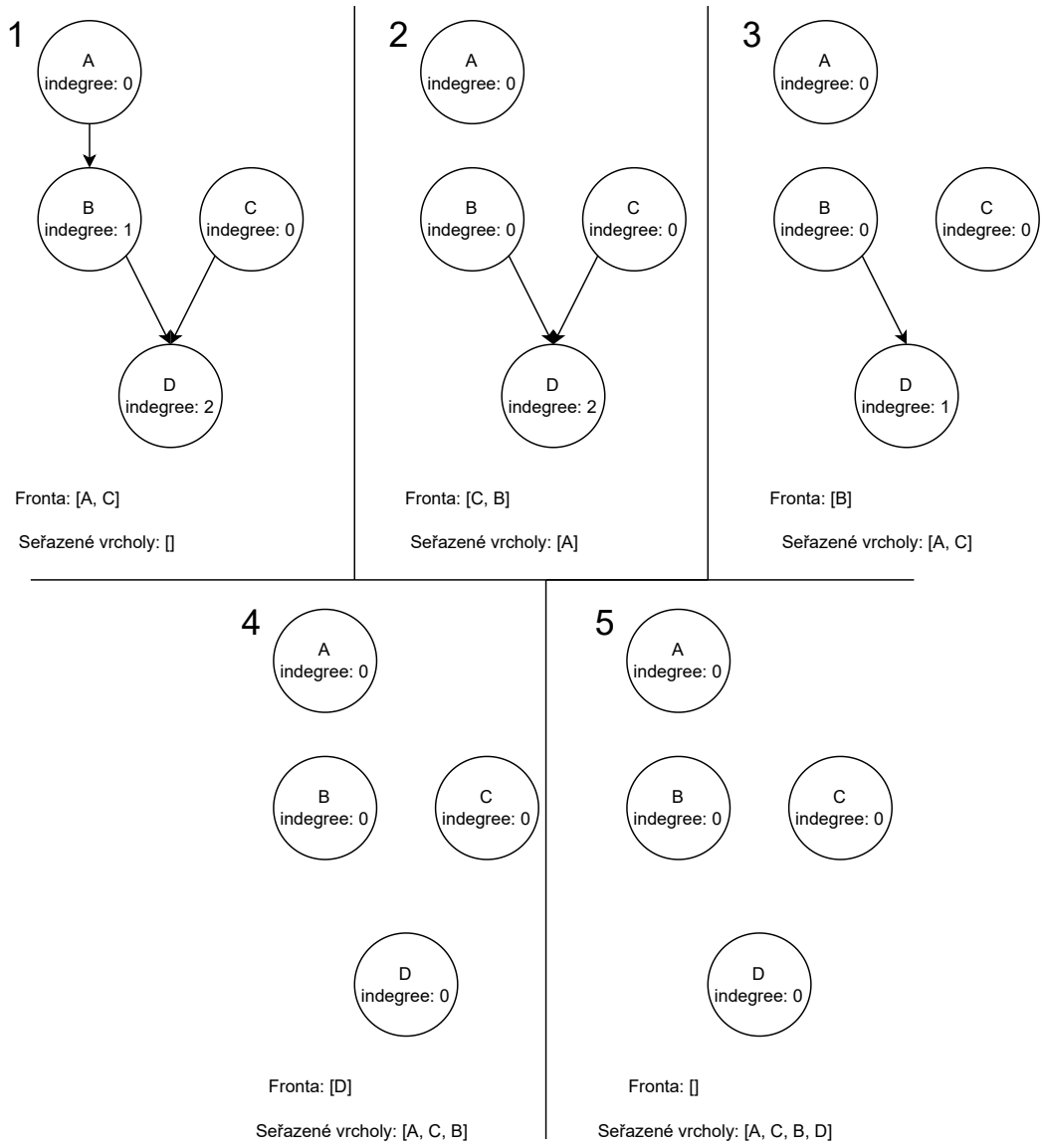
Kahnův algoritmus

Vstup Kahnova algoritmu je orientovaný acyklický graf, to znamená že všechny hrany grafu mají orientaci a zároveň graf neobsahuje žádný cyklus. Další požadavek na vstup je že graf obsahuje aspoň jeden vrchol, který má vstupní stupeň vrcholu nula, jinak řečeno to znamená že existuje aspoň jeden vrchol, který nemá žádného předka. Orientovaný acyklický graf je ukázán na obrázku 1.



Obrázek 1: orientovaný acyklický graf

Algoritmus při spuštění projde přes celý graf a uloží si do fronty všechny vrcholy, které mají vstupní stupeň nula a následně inicializuje pole pro topologicky seřazené vrcholy. Po úvodní části algoritmus vždy vezme vrchol z fronty, odstraní všechny výstupní hrany, které vycházejí z daného vrcholu. Při odstraňování hrany je kontrolováno zda vrchol, který byl před odstraněním hrany propojen s kontrolovaným vrcholem, nenabývá vstupního stupně nula, pokud ano, vrchol je vložen do fronty. Po odstranění všech výstupní hran daného vrcholu je vrchol vložen na první prázdné místo v poli určené pro výsledný topologicky seřazený graf. Celá smyčka se opakuje dokud není prázdná fronta. Na obrázku 2 je ukázán průběh algoritmu. Indegree znázorňuje vstupní stupeň vrcholu.



Obrázek 2: Ukázka Kahnova algoritmu

Pseoudokód Kahnova algoritmu

Algorithm 1 Kahnův Algoritmus pro topologické řazení

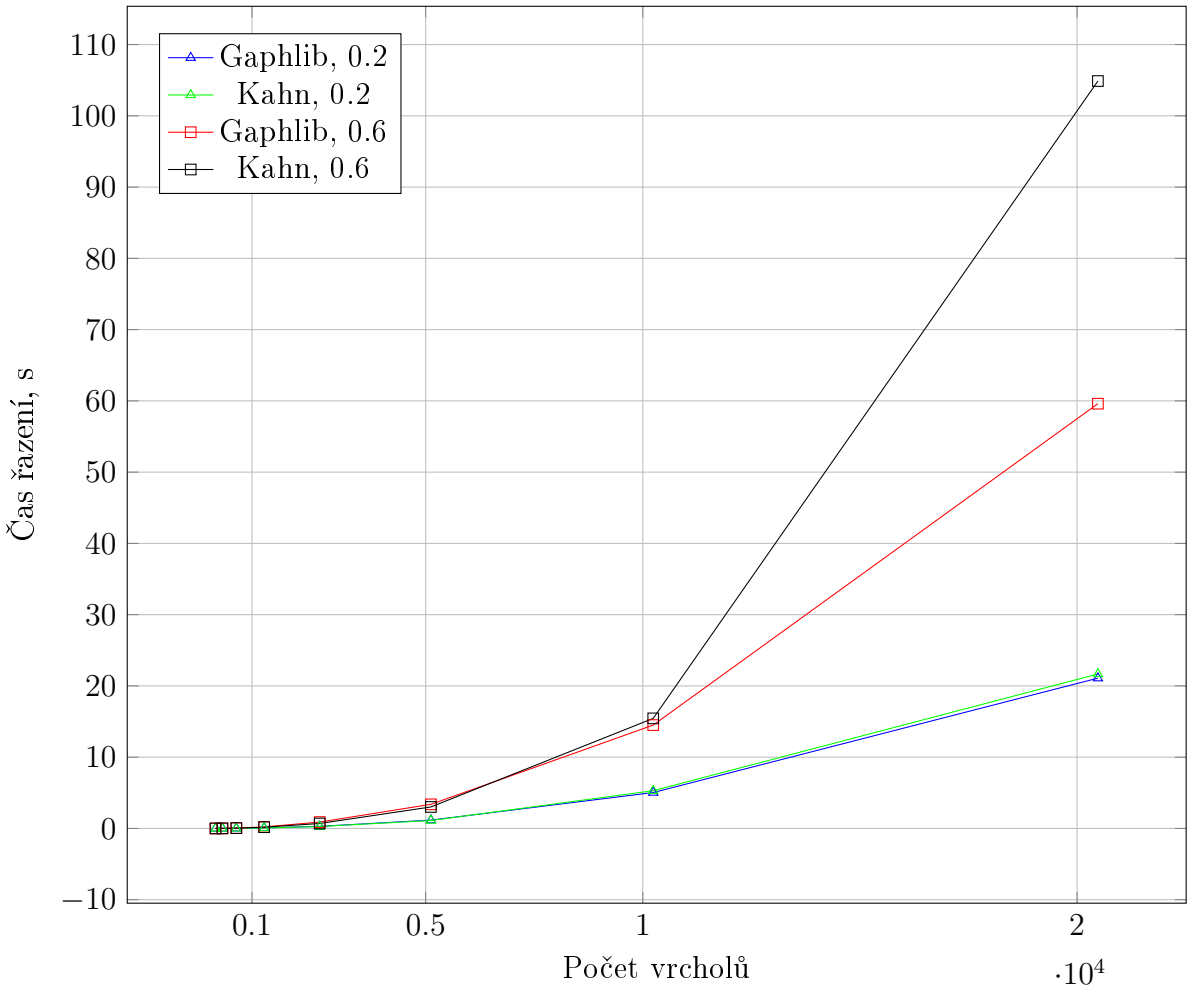
```
1:  procedure TopologicalSort(G)
2:      Input: orientovaný acyklický graf G
3:      Output: List L (topologicky seřazené vrcholy grafu)
4:      Inicializace prázdného listu L
5:      Inicializace fronty S, který bude obsahovat vrcholy se vstupním stupněm 0
6:      for each vrchol v in G do
7:          if vstupní stupeň vrcholu v == 0 then
8:              Vlož v do S
9:          end if
10:     end for
11:     while S is not empty do
12:         Odstraň vrchol n z S
13:         for each soused m of n do
14:             Odstraň hranu mezi m a n
15:             if vstupní stupeň vrcholu m == 0 then
16:                 Vlož m do S
17:             end if
18:         end for
19:         Vlož n do L
20:     end while
21:     if L obsahuje všechny vrcholy G then
22:         return L                          ▷ L je topologicky seřazený graf
23:     else
24:         return "Graf není acyklický"
25:     end if
26: end procedure
```

Experimenty a výsledky

Prezentovaný algoritmus byl pro účel testování implementován mnou v pythonu. Vlastní implementace byla porovnána s již existující knihovnou Graphlib [1]. Obě řazení vždy dostanou stejný graf, který bude vždy předem vygenerován. Graf s určitým počtem vrcholů je vždy generován s určitou pravděpodobnostní toho že z vrcholu povede hrana tzn. že čím větší pravděpodobnost tím více bude graf zahuštěn. Pro testování byla zvolena pravděpodobnost 0.2 a 0.6. V tabulce 1 jsou výsledky testování a na obrázku 3 jsou zobrazena data z tabulky v grafu.

Implementace	Počet vrcholů	Pravděpodobnost výskytu hran	Čas řazení, s
Gaphlib	160	0.2	0.0019
Kahn	160	0.2	0.0010
Gaphlib	160	0.6	0.0029
Kahn	160	0.6	0.0029
Gaphlib	320	0.2	0.0040
Kahn	320	0.2	0.0040
Gaphlib	320	0.6	0.0129
Kahn	320	0.6	0.0100
Gaphlib	640	0.2	0.0190
Kahn	640	0.2	0.0150
Gaphlib	640	0.6	0.0529
Kahn	640	0.6	0.0419
Gaphlib	1280	0.2	0.0820
Kahn	1280	0.2	0.0609
Gaphlib	1280	0.6	0.2119
Kahn	1280	0.6	0.1739
Gaphlib	2560	0.2	0.2950
Kahn	2560	0.2	0.2740
Gaphlib	2560	0.6	0.8949
Kahn	2560	0.6	0.6956
Gaphlib	5120	0.2	1.1670
Kahn	5120	0.2	1.1130
Gaphlib	5120	0.6	3.3829
Kahn	5120	0.6	3.0150
Gaphlib	10240	0.2	5.0363
Kahn	10240	0.2	5.2941
Gaphlib	10240	0.6	14.5039
Kahn	10240	0.6	15.4444
Gaphlib	20480	0.2	21.0900
Kahn	20480	0.2	21.6754
Gaphlib	20480	0.6	59.6032
Kahn	20480	0.6	104.8810

Tabulka 1: Porovnání implementací topologického řazení.



Obrázek 3: Porovnání implementací topologického řazení.

Z výsledků je vidět že pro malé grafy (počet vrcholů přibližně 2 - 5000) je implementovaný Kahnův algoritmus zanedbatelně rychlejší v obou různě zahuštěných grafech, ale u větších grafů je implementovaný algoritmus o proti knihovně Graphlib pomalejší a to hlavně u více zahuštěného grafu.

Závěr

Implementovaný Kahnův algoritmus je při malých velikostech grafu zanedbatelně rychlejší než topologické řazení z knihovny Graphlib. Při větších velikostech grafu a malé hustotě je Kahnův algoritmus o něco pomalejší než Graphlib a při zvýšení hustoty grafu se rozdíl mezi rychlosti ještě zvětšuje. Pozitivní aspekty navrhované implementace jsou: jednoduchá implementace, při malých velikostech grafu je implementace rychlejší a při velikých velikostech grafu s nízkou hustotou je podobně rychlá jako funkce z knihovny Graphlib, ale tato implementace Kahnova algoritmu není vhodná pro veliké grafy s vysokou hustotou.

Literatura

[1] Graphlib. https://docs.python.org/3/library/graphlib.html/.

[2] GeeksforGeeks. Topological sorting. 2023. https://www.geeksforgeeks.org/topological-sorting//.

[3] Arthur B Kahn. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562, 1962. https://doi.org/10.1145/368996.369025.

[4] Virtual Labs. Topological sorting time and space complexity. 2023. https://ds2-iiith.vlabs.ac.in/exp/topo-sort/analysis/time-and-space-complexity.html/.

[5] Steven S Skiena. The algorithm design manual. pages 481–483, 2008. https://mimoza.marmara.edu.tr/~msakalli/cse706_12/SkienaTheAlgorithmDesignManual.pdf.