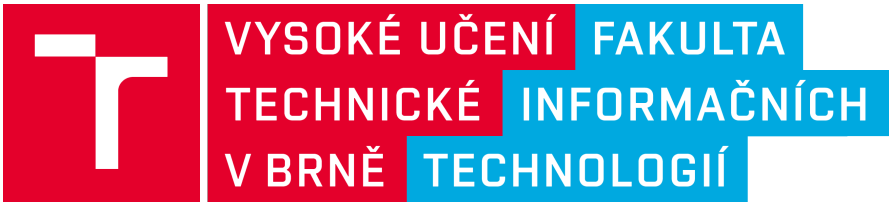

Porovnání algoritmů hledání cyklů v grafech

Bc. Jan Bíl
Bc. Michal Šedý

14. listopadu 2022

Obsah

Kapitola 1

Úvod

Orientovaný graf je struktura popisující množinu bodů (uzlů), jenž jsou mezi sebou propojeny orientovanými hranami. Cyklus v orientovaném grafu představuje takovou spojitou posloupnost uzlů, že se žádný uzel s výjimkou prvního a posledního v sekvenci neopakuje a zároveň pro dvojici sousedících uzlů v posloupnosti $\ldots u_m u_n \ldots$ platí, že existuje orientovaná hrana vedoucí z uzlu u_m do uzlu u_n. Pato práce se zabývá popisem algoritmů pro získání seznamu všech existujících cyklů v zadaném grafu.

Vyhledávání všech (výčet) cyklů v grafu je využíváno v mnoha odvětvích teorie grafů. Tato informace je používána k optimalizaci počítačových programů [1], při analýze booleovských sítí využívaných pro modelování biologických sítí nebo sítí genových regulátorů [8], při návrhu, vývoji [7] nebo ověření spolehlivosti a fault-tolerance komunikačních systému [6], atd.

Tato páce porovnávající tři algoritmy pro výčet všech cyklů v grafu byla vytvořena v rámci projektu "Porovnání - Hledání cyklů" do předmětu GAL (grafové algoritmy). Text na úvod definuje potřebné pojmy dále využívané v algoritmech. V následujících kapitolách jsou uvedeny jednotlivé implementované algoritmy. Kapitola 3 popisuje přímočarý algoritmus [2, str. 287], který postupně generuje různé kandidáty cest, a ti jsou následně ověřování. Algoritmus, který navrhl Herbert Weinblatt využívající zpětné navrácení [10] je uveden v kapitole 4. Kapitola 5 popisuje algoritmus Hongbo Liu a Jiaxin Wangův algoritmus využívající frontu [5]. Tyto algoritmy byly implementovány v jazyce Python3. Popis návrhu implementace aplikace a její používání jsou uvedeny v kapitolách 6 a 7. Experimenty porovnávající efektivitu jednotlivých postupů výčtu všech cyklů včetně grafové knihovny Networkx[1] jsou uvedeny v kapitole 8.

[1] Dostupné z https://networkx.org/

Kapitola 2

Prerekvizity

Tato kapitola poskytuje základní definice pro orientované grafy, jakými jsou základní definice grafu, sledu, cesty a cyklů. Dále jsou popsány základní algoritmy pro práci s grafy, kterými jsou prohledávání do hloubky (DFS) a topologické uspořádání, které jsou využívány pro zjednodušení výčtu cyklů grafů. Tato kapitola je převzata z [4].

2.1 Orientovaný graf

Definice 2.1.1 ***Orientovaný graf*** *je uspořádaná dvojice $G = (V, E)$, kde V je množina uzlů grafu a $E \subseteq V \times V$ je množina orientovaných hran, kde hrana $(u, v) \in E$ znamená, že v grafu G vede hrana z uzlu u do uzlu v (uzly u, v jsou incidentní).*

Orientovaný graf $G = (V, E)$ je možno v algoritmech reprezentovat dvěma způsoby. Nechť $u, v \in V$. 1) jako pole Adj seznamů sousedů, pro které platí $v \in Adj[u] \iff (u, v) \in E$. 2) jako matici souslednosti Adj_M, kde $Adj_M[u][v] = 1 \iff (u, v) \in E \wedge Adj_M[u][v] = 0 \iff (u, v) \notin E$. Pro účely této práce byl zvolen první přístup, kterým je pole seznamů sousedů.

Definice 2.1.2 *Nechť $G = (V, E)$.* ***Transponovaný graf*** *$G^T = (V, E^T)$, kde $E^T = \{(v, u) \mid (u, v) \in E\}$.*

Definice 2.1.3 ***Vstupní stupeň uzlu*** *je dán funkcí $d_+ : V \to \mathbb{N}_0$, která udává počet přechodu vstupujících do uzlu.*

Definice 2.1.4 ***Vstupní stupeň uzlu*** *je dán funkcí $d_- : V \to \mathbb{N}_0$, která udává počet přechodu vstupujících do uzlu.*

Lze snadno ukázat, že pokud má uzel $u \in V$ hodnotu $d_-(u) = 0$ nebo $d_+(u) = 0$, pak nemůže být součástí žádného cyklu, pro každý stav obsažný v cyklu musí platit, že jeho vstupní i výstupní stupeň je nenulový. Tyto uzly s nulovým stupněm mohou být v části přípravy algoritmů pro výčet cyklů zanedbány (odstraněny). Toto zanedbání uzlu může snížit hodnotu vstupních nebo vstupních stupňů uzlů incidentních s uzlem u na nulu. V takovém případě jsou dále rekurzivně zanedbány také tyto uzly.

Definice 2.1.5 ***Sled*** *je posloupnost vrcholů $\langle v_0 \dots v_n \rangle$, kde $n \in \mathbb{N}$, $v_i \in V$ pro $0 \leq i \leq n$, a $(v_{j-1}, v_j) \in E$ pro $1 \leq j \leq n$.*

Definice 2.1.6 ***Cesta*** *(otevřený cesta) je sled, ve kterém se neopakují uzly.*

Definice 2.1.7 ***Cyklus*** *je cesta, ve které shodují první a poslední uzel.*

2.2 Prohledávání do hloubky

Algoritmus prohledávání do hloubky (DFS) je základním algoritmem pro práci s grafy. DFS postupně prochází všechny uzly grafu $G = (V, E)$ a vytváří strom prohledávání do hloubky.

Definice 2.2.1 *Nechť $G = (V, E)$ a π pole předchůdců, kde $u \in \pi[v] \implies (u, v) \in E$.* ***Strom prohledávání do hloubky*** *je $G_\pi = (V, E_\pi)$, kde $E_\pi = \{(u, v) \in E \mid u = \pi[v]\}$.*

Během výpočtu se vytváří pole barev uzlů $color[u] \in \{WHITE, GRAY, BLACK\}$, pole časů prvního prozkoumání $d[u] \in \mathbb{N}$, pole časů dokončení prozkoumávání seznamu sousedů $f[u] \in \mathbb{N}$ a pole předchůdců $\pi[u] \subseteq V$.

Algorithm 1: DFS

Input: $G := (V, E)$
Output: π, d, f

```
1  Procedure DFS–VISIT(v)
2      color[u] ← GRAY
3      d[u] ← time ← time + 1
4      for v ∈ Adj do
5          if color[v] = WHITE then
6              DFS–VISIT(v)
7          end
8      end
9  end
10 for u ∈ V do
11     color[u] ← WHITE
12     π[u] ← NIL
13 end
14 time ← 0
15 for u ∈ V do
16     if color[u] = WHITE then
17         DFS–VISIT(u)
18     end
19 end
20 return π, d, f
```

Teorém 2.2.2 *Časová složitost algoritmu DFS je $\mathcal{O}(|V| + |E|)$.*

Důkaz. Inicializační část 10–13 má časovou obtížnost $\mathcal{O}(|V|)$. Hlavní cyklus 15–19 je prováděn maximálně $|V|$-krát, tedy časová obtížnost je $\mathcal{O}(|V|)$. Funkce DFS-VISIT je spouště na pouze pro bílé uzly, tedy $|V|$-krát a cyklus v proceduře 4–8 je proveden maximálně $|Adj[v]|$-krát. Protože $\sum_{v \in V} |Adj[v]| = |E|$ je časová obtížnost cyklu 4–8 $\mathcal{O}(|E|)$. Celková složitost je tedy $\mathcal{O}(|V| + |E|)$. □

2.3 Topologické uspořádání

Definice 2.3.1 ***Topologické uspořádání*** *orientovaného grafu $G = (V, E)$ je lineární uspořádání všech uzlů tak, že pokud $(u, v) \in E$, pak u předchází v v daném uspořádání.*

Pokud graf G obsahuje cykly, poté není možné určit topologické uspořádání. Nicméně algoritmus lze spustit. Výsledkem bude *pseudo-topologické uspořádání*, ve kterém bude platit, že pokud $(u, v) \in E$ a zároveň se u nenachází v žádném cyklu, pak u předchází v v daném uspořádání.

Algorithm 2: Topological-sort

Input: $G := (V, E)$
Output: L

zavolej DFS(G) pro výpočet hodnot $f[v]$
každý dokončený uzel zařaď na začátek seznamu uzlů L
return L

Teorém 2.3.2 *Protože výpočet topologického uspořádání využívá pouze DFS v časovou složitostí $\mathcal{O}(|V| + |E|)$ a operaci vložení na začátek seznamu, která má konstantní časovou složitost, je časová složitost topologického uspořádání $\mathcal{O}(|V| + |E|)$.*

2.4 Zanedbání stavů

Algorithm 3: Zanedbání stavů

Input: $G := (V, E)$
Output: G_{simply}

```
Procedure Pruning(G_p := (V_p, E_p))
    for u ∈ Topological-sort(G_p) do
        if d_{p+}[u] = 0 then
            for v ∈ Adj_p[u] do
                d_{p+}[v] ← d_{p+}[v] − 1
            end
            V_p.remove(u)
        end
    end
end
G_t ← G^T                                   // Transponujeme graf G
Pruning(G_t)                                // Smažeme zanedbatelné stavy v grafu G_t
G_simply ← G_t^T                            // Transponujeme graf G_t
Pruning(G_simply)                           // Smažeme zanedbatelné stavy v grafu G_simply
return G_simply
```

Jak již bylo dříve řečeno, stavy, jejichž vstupní, nebo výstupní stupeň je nulový nemohou být součástí žádného cyklu, a proto mohou být při výčtu všech cyklů grafu zanedbány (odstraněny). Při zanedbání těchto uzlů se ale mohou změnit hodnoty funkcí d_- a d_+ tak, že budou objeveny nové stavy s nulovým vstupním nebo výstupním stupněm. K jejich kompletní eliminaci slouží následující algoritmus.

Teorém 2.4.1 *Časová složitost algoritmu zanedbání stavů je* $\mathcal{O}(|V| + |E|)$.

Důkaz. Transponování grafu má časovou složitost $\mathcal{O}(|V|+|E|)$. Topologické uspořádání má časovou složitost $\mathcal{O}(|V| + |E|)$. V proceduře Pruning se hlavní cyklus 2–9 prochází $|V|$-krát a vnitřní cyklus 4–6 se prochází $|Adj[u]|$-krát. Časová složitost procedury Pruning je tedy $\mathcal{O}(|V|+|E|)$, z čehož plyne, že časová složitost algoritmu zanedbání stavů je $\mathcal{O}(|V|+|E|)$. □

Kapitola 3

Brute-force

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

3.1 Popis algoritmu

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque.

Algorithm 4: Dummy Algorithm.

```
1  Do something.;
2  for k = 0, 1, 2, 3, ... do
3  |  Do something.;
4  |  if x ≥ y then
5  |  |  Do something.;
6  |  else
7  |  |  Do something.;
8  |  |  for j = 1, ..., 10 do
9  |  |  |  Do something.;
10 |  |  end
11 |  |  THIS IS THE SPOT WHERE I NEED THE PAGE BREAK.;
12 |  |  Do something.;
13 |  end
14 end
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

3.2 Časová složitost

Teorém 3.2.1 *Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque.*

Důkaz. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. □

3.3 Prostorová složitost

Teorém 3.3.1 *Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque.*

Důkaz. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. □

Kapitola 4

Herbert Weinbaltt

Tento algoritmus byl publikován v roce 1972 Herbertem Weinblattem [10]. Jeho základem je Tiernanův algoritmus [9] publikovaný o dva roky dříve, který vyhodnocuje každý cyklus pouze jednou, jednalo se tedy o teoreticky nejefektivnější algoritmus, ovšem za cenu vyšších paměťových nároků. Sam Tiernan ale naznačil, že pro průměrně husté grafy s více než 100 hranami by bylo použití tohoto algoritmus neprakticky pomalé. Herbert Weinblatt ve svém článku popisuje nový přístup, který stejně jako Tiernanův vyhodnocuje každý cyklus pouze jedno, ale nově také minimalizuje množství prozkoumaných hran nutných k objevení cyklu. V důsledku toho dokázal algoritmus implementovaný v experimentálním jazyce Snobol3 na počítači IBM 7094 objevit všech 44 cyklů v grafu s 194 uzly a 294 hranami za méně než sedm sekund.

4.1 Popis algoritmu

Před samotným popisem algoritmu se potřeba definovat pomocné funkce *END* a *TAIL*.

Definice 4.1.1 ***END*** *je unární funkce, které pro cestu* $\langle v_0 v_1 \dots v_n \rangle$*, vrací poslední uzel cesty* v_n*.*

Definice 4.1.2 ***TAIL*** *je binární funkce, která pro dvojici uzlu* v_k *a otevřenou cestu* $\langle v_0 v_1 \dots v_k v_{k+1} \dots v_n \rangle$*, respektive cyklus* $\langle v_0 v_1 \dots v_k v_{k+1} \dots v_0 \rangle$ *vrací podcestu* $\langle v_{k+1} \dots v_n \rangle$*, respektive* $\langle v_{k+1} \dots v_0 \rangle$ *s respektem k uzlu* v_k*. V případě cyklu* $\langle v_k v_{k+1} \dots v_k \rangle$ *vrací prázdnou cestu* $\langle\rangle$ *délky 0.*

V průběhu výpočtu využívá algoritmus seznam TT, který reprezentuje aktuální zkoumanou cestu. Dále jsou udržovány dvě pomocné struktury S_v a S_e. Kde S_v je pomocné pole udržující informaci, zda již byl uzel $v \in V$ v seznamu TT. $S_v[v]$ může nabývat hodnot 0, 1, nebo 2, což indikuje, že uzel v ještě nabyl v seznamu TT, uzel v se momentálně nachází v seznamu TT, nebo že se uzel v již nenachází v TT. Obdobná informace je udržována pro hrany. S_e je matice informující, zda již byla hrana $(u, v) \in E$ v seznamu TT. $S_e[u][v]$ může nabývat pouze dvou hodnot, a to 0, respektive 2, což indikuje, že hrana (u, v) ještě nebyla v TT, respektive že hrana (u, v) již byla v TT a stále může být.

Pro sjednocení dvou cest $P_1 = \langle v_1 v_2 \rangle$ a $P_2 = \langle v_3 v_4 \rangle$ budeme využívat operátor +, tedy $P_1 + P_2 = \langle v_1 v_2 v_3 v_4 \rangle$.

Algorithm 5: Herbert Weinbalttův algoritmus

Input: $G := (V, E), n := |V|$
Output: L_{cycles}

```
1  Procedure CONCAT(isRecursion, Path)
       // Inicializace lokálních proměnných
2      cycleTails ← EmptyList
3      toAddSave ← EmptyList
4      toAddToControl ← EmptyList
5      added ← EmptyList
6      v ← END(Path)
7      for cycle ∈ L_cycles do
8          tail ← TAIL(v, cycle)
9          if tail = ∅ ∨ tail ∈ L_cycles then
10             continue
11         end
12         cycleTails.append(tail)
13         if ∃v_k ∈ tail : v_k ∈ Path then
14             continue
15         end
16         cycleEnd ← END(cycle)
17         if S_v[cycleEnd] = 2 then
18             toAddToControl.extend(CONCAT(True, Path + tail))
19             continue
20         else
21             newCycle ←⟨cycleEnd⟩ + TAIL(cycleEnd, TT)+
                                Path + TAIL(END(Path), cycle)
22             if isRecursion then
23                 toAddToControl.append(newCycle)
24             else
25                 toAddSave.append(newCycle)
26             end
27         end
28     end
29     if isResursion then
30         return toAddToControl
31     else
32         L_cycles.extend(toAddSave)
33         added.extedn(toAddSave)
34         for cycle ∈ toAddToControl do
35             if cycle ∉ added then
36                 L_cycles.append(cycle)
37                 added.append(cycle)
38             end
39         end
40     end
41 end
```

```
42 Procedure EXAMINE(v)
43     if S_v[v] = 0 then
44         S_v[v] ← 1
45         TT.append(v)
46     else if S_v[v] = 1 then
47         L_cycles.append(⟨v⟩ + TAIL(v, TT) + ⟨v⟩)
48     else
49         CONCAT(False, [v])
50 end
51 Procedure EXTEND
52     while TT ≠ ∅ do
53         u ← END(TT)
54         possible_v ← {v ∈ V | (u, v) ∈ E ∧ S_e[u][v] = 0}
55         if possible_v = ∅ then
56             S_v[u] ← 2
57             TT.removeLast()
58         else
59             v ← PickOne(possible_v)
60             S_e[u][v] ← 2
61             EXAMINE(v)
62         end
63     end
64 end
   // Inicializace globálních proměnných
65 TT ← EmptyList
66 S_e[0 . . . n − 1][0 . . . n − 1] ← 0                    // nulová matice n × n
67 S_v[0 . . . n − 1] ← 0
68 for v ∈ V do
69     if S_v[v] = 0 then
70         S_v[v] ← 1
71         TT.append(v)
72         EXTEND()
73     end
74 end
75 return L_cycles
```

Před spuštěním vlastního prohledávání grafu jsou všechny uzly, které mají hodnotu d_+ nebo d_- rovnou nule a jejich hrany zanedbány (odstraněny) algoritmem 3. Cílem tohoto kroku je eliminovat uzly, které nemohou tvořit cykly a tím snížit velikost grafu nad kterým bude prohledávání prováděno.

Algoritmus vybere jeden uzel grafu (*počáteční uzel cesty*) a začne prozkoumávat všechny cesty vycházející z tohoto uzlu. Pokud se během vytváření cesty některý uzel navštíví vícekrát, je tato podcesta označena za cyklus a konstrukce cyklu se navrátí k předchozímu uzlu, pro který existují doposud neprozkoumaní následníci. Pokud již žádný takový uzel

neexistuje, zvolí algoritmus nový, doposud nezvolený, *počáteční uzel cesty*. V případě, že takový uzel neexistuje, algoritmus skončí.

V průběhu výpočtu je cesta vedoucí z *počátečního uzlu cesty* reprezentovaná seznamem TT ("trial thread"). Při návratu je odstraněn poslední uzel (nejvzdálenější od *počátečního uzlu*) cesty TT.

Když algoritmus dospěje k uzlu v, který se již byl v minulosti vyhodnocen, pak před zpětným navrácením zkontroluje, zda některá z doposud vyhodnocovaných hran netvoří nový cyklus. Pokud se uzel v nachází v TT, pak existuje právě jeden cyklus, který je tvořen sjednocením v s *TAIL* TT s respektem z uzlu v. Pokud se uzel v již nenachází v TT, pak může cyklus existovat pouze pokud byly již nějaké cykly obsahující v objeveny. V takovém případě je spuštěna rekurzivní procedura $CONCAT$, která se snaží nalézt cestu, která začíná na uzlu v a končí na některém uzlu u, který je stále na TT. Z každé takto nalezené cesty je vytvořen cyklus sjednocením této cesty s *TAIL* TT s respektem k u. (Nechť $C_1 = \langle v_1 v_2 v_1 \rangle$ a $C_2 = \langle v_2 v_3 v_2 \rangle$ jsou dva již objevené cykly, $TT = \langle v_1 \rangle$ a algoritmus objeví již jednou zpracovaný uzel v_3. A takovém případě je konkatenací podcesty $\langle v_1 \rangle$ z TT, podcesty $\langle v_3 v_2 \rangle$ z C_2 a podcesty $\langle v_1 \rangle$ z C_1 vytvořen nový cyklus $\langle v_1 v_3 v_2 v_1 \rangle$.)

4.2 Časová složitost

Teorém 4.2.1 *Časová složitost Herbert Weinblattova algoritmu pro výčet všech cyklů v orientovaném grafu je* $\mathcal{O}((|V| + |E|) * (c + 1))$*, kde* c *je počet cyklů v grafu.*

Důkaz. TODO. □

4.3 Prostorová složitost

Teorém 4.3.1 *Prostorová složitost Herbert Weinblattova algoritmu pro výčet všech cyklů v orientovaném grafu je* $\mathcal{O}(c * |V| + |V|^2)$*, kde* c *je počet cyklů v grafu.*

Důkaz. Algoritmus využívá tři globální pomocné struktury. Prostorová složitost TT je $\mathcal{O}(|V|)$ protože maximální délka cyklu je shora omezena na $|V| + 1$. Prostorová složitost matice S_e je $\mathcal{O}(|V|^2)$. A prostorová složitost pole S_v je $\mathcal{O}(|V|)$. Prostorovou složitost lokálních pomocných proměnných *added*, $cycleTails$, $toAddSave$ a $toAddToControl$ můžeme zanedbat, protože jejich data jsou obsažena v listu všech detekovaných cyklů L_{cycles}. Prostorová složitost listu L_{cycles} je $\mathcal{O}(c * |V|)$, protože obsahuje c cyklů[1], kde délka každého cyklu může být až $|V| + 1 \simeq |V|$.

Pokud by prohledávaný graf neobsahoval žádný cyklus, potom budou vždy inicializovány globální pomocné struktury s prostorovou složitostí $\mathcal{O}(|V|^2)$.

Bylo dokázáno, že prostorová složitost Herbert Weinblattova algoritmu pro výčet všech cyklů v orientovaném grafu je $\mathcal{O}(c * |V| + |V|^2)$, kde c je počet cyklů v grafu. □

[1] V případě úplného grafu je počet cyklů rovem mohutnosti symetrické grupy $|S_{|V|}| = |V|!$.

Kapitola 5

Hongbo Liu a Jiaxin Wang

V roce 2006 spolu Hongo Liu a Jiaxin Wang publikovali algoritmus [5] pro výčet všech cyklů v grafu. Tento algoritmus pracuje v exponenciální časové složitosti oproti algoritmu Jonsona [3], Tiernana [9] nebo zmíněného Weinblattova algoritmu [10]. V porovnáni s nimi je ale jednodušší pro porozumění, čímž jsou minimalizovány chyby v implementaci způsobené chybnou interpretací. Na druhou stranu Liuův a Wangův přístup je neefektivní pro velké grafy. Nicméně existují případy, pro které je jejichž řešení efektivnější.

Algoritmus využívá frontu zkoumaných cest $P_0 \dots P_n$ pro $n \in \mathbb{N}_0$, kde pro délky cest ve frontě platí $|P_n| \leq |P_0| + 1$ a zároveň $|P_{i-1}| \leq |P_i|$. Díky tomu jej lze snadno využít pro výčet všech cyklů délek maximálně $k \in \mathbb{N}_0$ be z nutnosti nalezení všech cyklů.

5.1 Popis algoritmu

Algoritmus využívá pomocné funkce *END*, která již byla definována dříve 4.1.1 a *HEAD*.

Definice 5.1.1 ***HEAD*** *je unární funkce, které pro cestu* $\langle v_0 v_1 \dots v_n \rangle$*, vrací první uzel cesty* v_0*.*

Pro sjednocení dvou cest $P_1 = \langle v_1 v_2 \rangle$ a $P_2 = \langle v_3 v_4 \rangle$ budeme využívat operátor $+$, tedy $P_1 + P_2 = \langle v_1 v_2 v_3 v_4 \rangle$.

Aby se zabránilo duplicitní detekci cyklů, které byly generovány z jiných počátečních uzlů, ale jinak jsou si zcela ekvivalentní, je každému uzlu v přiřazena různá hodnota $ord(v) \in \mathbb{N}_0$, kde pro $u, v \in V$ platí $ord(u) = ord(v) \iff u = v$. Při prohledávání cesty P v grafu pak může být uzel v připojen na konec cesty P pouze pokud $ord(v) > ord(END(P))$.

Před spuštěním prohledávání grafu jsou všechny uzly, které mají hodnotu d_+ nebo d_- rovnou nule a jejich hrany zanedbány (odstraněny) algoritmem 3. Cílem tohoto kroku je eliminovat uzly, které nemohou tvořit cykly a tím snížit velikost grafu nad kterým bude prohledávání prováděno.

Při inicializaci proměnných (1–4) jsou všechny cesty délek 0 vloženy do fronty cest Q. V hlavním cyklu (5–19) se algoritmus snaží vytvořit cykly délky k pro cesty délek $k-1$. Cyklus délky k je tvořen sjednocením cesty P délky $k-1$ a hrany $(END(P), HEAD(P))$. Dále z cesty P na základě přechodů vedoucích z $END(P)$ generuje nové cesty $P + \langle v \rangle$ pro $v \in \{v \in Adj[END(P)] \,|\, v \notin P \wedge ord(v) > ord(HEAD(P))\}$ a vkládá je do fronty cest Q.

Pokud je fronta cest Q prázdná (nebylo již možné vygenerovat další cesty), je algoritmus ukončen a všechny objevené cykly jsou vráceny v seznamu L_{cycles}.

Algorithm 6: Liuův a Wangův algoritmus

Input: $G := (V, E)$
Output: L_{cycles}

```
   // Inicializace
 1 Q ← EmptyQueue
 2 for v ∈ V do
 3 |   ENQUEUE(Q, ⟨v⟩)
 4 end
 5 while Q ≠ ∅ do
 6 |   P ← DEQUEUE(Q)
 7 |   head ← HEAD(P)
 8 |   end ← END(P)
 9 |   for v ∈ Adj[end] do
10 |   |   if v = head then
11 |   |   |   L_cycles.append(P + ⟨v⟩)
12 |   |   else if ord(v) > ord(head) then
13 |   |   |   if v ∉ P then
14 |   |   |   |   ENQUEUE(Q, P + ⟨v⟩)
15 |   |   |   end
16 |   |   end
17 |   end
18 end
19 return L_cycles
```

5.2 Časová složitost

Teorém 5.2.1 *Časová složitost algoritmu, který publikovali Hongo Liu a Jiaxin Wang je* $\mathcal{O}(2^{|V|})$.

Důkaz. Inicializace a naplnění fronty (1–4) Q má časovou složitost $\mathcal{O}(|V|)$. V hlavním cyklu (5–18) jsou postupně vygenerovány všechny cesty P, pro které platí $\forall v \in P : ord(v) \geq ord(HEAD(P))$. Takových to cest je až $\sum_{k=1}^{n:=|V|} \binom{n}{k} = 2^n - 1$. Celková časová složitost algoritmu je tedy $\mathcal{O}(2^{|V|})$. □

5.3 Prostorová složitost

Teorém 5.3.1 *Prostorová složitost algoritmu, který publikovali Hongo Liu a Jiaxin Wang je* $\mathcal{O}(2^{|V|})$.

Důkaz. Prostorová složitost výstupního seznamu L_{cycles} je $\mathcal{O}(c * |V|)$, protože délka cyklu může být až $|V| + 1 \simeq 1$.

Do fronty Q se postupně ukládají všechny cesty v grafu, kde pro každý uzel v cesty P musí platit $ord(v) \geq HEAD(P)$. Algoritmem ude celkem zpracováno až $\sum_{k=1}^{n:=|V|} \binom{n}{k}$.

Hodnota této funkce dosahuje maxima pro $k = \lceil n/2 \rceil =: k_{0.5}$. Tedy prostorová složitost fronty Q je $\mathcal{O}(\binom{n}{k_{0.5}}) \in \mathcal{O}(2^{|V|})$.

Celková prostorová složitost algoritmu je $\mathcal{O}(2^{|V|})$. □

Kapitola 6

Návrh programu

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget

felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Kapitola 7

Použití programu

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget

felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Kapitola 8

Experimenty

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget

felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Kapitola 9

Závěr

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Literatura

[1] Allen, F. E. Program optimization. *Research Report RC-1959*. IBM Watson Research Center, Yorktown Heights, N.Y. april 1966.

[2] Deo, N. *Graph Theory with Applications to Engineering and Computer Science*. Dover Publications, 2017. ISBN 9780486820811. Dostupné z: https://books.google.cz/books?id=DSBMDgAAQBAJ.

[3] Johnson, D. B. Finding All the Elementary Circuits of a Directed Graph. *SIAM Journal on Computing*. 1975, sv. 4, č. 1, s. 77–84. DOI: 10.1137/0204007. Dostupné z: https://doi.org/10.1137/0204007.

[4] Křivka, Z. a Masopust, T. *Grafové algoritmy*. VUT Brno, Fakulta informačních technologií, 2018. Dostupné z: http://www.fit.vutbr.cz/study/courses/GAL/public/gal-slides.pdf.

[5] Liu, H. a Wang, J. A new way to enumerate cycles in graph. In: *Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06)*. 2006, s. 57–57. DOI: 10.1109/AICT-ICIW.2006.22.

[6] Médard, M. a Lumetta, S. Network Reliability and Fault Tolerance. In:. Duben 2003. DOI: 10.1002/0471219282.eot281. ISBN 9780471219286.

[7] Rozenfeld, H. D., Kirk, J. E., Bollt, E. M. a Avraham, D. ben. Statistics of cycles: how loopy is your network? *Journal of Physics A: Mathematical and General*. IOP Publishing. may 2005, sv. 38, č. 21, s. 4589–4595. DOI: 10.1088/0305-4470/38/21/005. Dostupné z: https://doi.org/10.1088%2F0305-4470%2F38%2F21%2F005.

[8] Rushdi, A. a Alsogati, A. Matrix Analysis of Synchronous Boolean Networks. *International Journal of Mathematical, Engineering and Management Sciences*. Duben 2021, sv. 6, s. 598–610. DOI: 10.33889/IJMEMS.2021.6.2.036.

[9] Tiernan, J. C. An Efficient Search Algorithm to Find the Elementary Circuits of a Graph. *Commun. ACM*. New York, NY, USA: Association for Computing Machinery. dec 1970, sv. 13, č. 12, s. 722–726. DOI: 10.1145/362814.362819. ISSN 0001-0782. Dostupné z: https://doi.org/10.1145/362814.362819.

[10] Weinblatt, H. A New Search Algorithm for Finding the Simple Cycles of a Finite Directed Graph. *J. ACM*. New York, NY, USA: Association for Computing Machinery. jan 1972, sv. 19, č. 1, s. 43–56. DOI: 10.1145/321679.321684. ISSN 0004-5411. Dostupné z: https://doi.org/10.1145/321679.321684.