Sada: 3 Příklad: 3 IV003 Algoritmy a datové struktury II

Jméno: Jiří Novotný UČO: 409963

Jméno: Henrich Lauko UČO: 410438

a) Algoritmus pre odstránenie vrcholu z grafu tak, aby sa nezmenili vzdialenosti ostatných vrcholov pracuje na princípe prelinkovania vchádzajucich hrán do a z vrcholu v.

Algoritmus 1: REMOVEVERTEX(G, v)

vstup: graf $G = (V, E)$ obsahujúci odoberaný vrchol v
výstup: graf G', ktorý neobsahuje vrchol v, nezmenenými najkračšími vzdialenosťami

$E' \leftarrow \emptyset$
$V' \leftarrow V \setminus \{v\}$
foreach $i \in V'$ **do**
    foreach $j \in V'$ **do**
        $value \leftarrow min\{E(i,j); E(i,v) + E(v,j)\}$// *neexistujúca hrana má hodnotu* ∞
        if $value \neq \infty$ **then**
            E' addEdge $(i, j, value)$// *prida hranu s hodnotou value*
        fi
    end foreach
end foreach
return $G' = (E', V')$

Časová zložitosť Časová zložitosť algoritmu je $\mathcal{O}(|V'^2|)$, keďže v algoritme iterujeme dvoma zanorenými cyklami cez množinu vrcholov V', v tele cyklu sa indexy nemenia a operácie vnútri cyklov su konštantné, teda časová zložitosť cyklov je $\mathcal{O}(|V'|) \cdot \mathcal{O}(|V'|) = \mathcal{O}(|V'^2|)$. Pri inicializácii kopírujem množinu V do V' čo je vykonané v lineárnom čase, číže to časovú zložitosť nezhorší.

Korektnosť Odobratie vrchulo v mohlo narúšiť jedine dĺžky najkratších ciest na ktorých sa vrchol v nachádzal. Čo môžeme reprezentovať tak, že existuje nejaká cesta medzi (a, b) v pôvodnom grafe G, môžu nastať dva prípady:

1. v sa na najkratšéj ceste nachádza v tom prípade existuje taká dvojica vrcholov (i, j), kde i je predchodca v na najktratšej ceste (a, b) a j je jeho následnik v tom prípade do grafu G' pridáme hranu $(i, j) = (i, v) + (v, j)$, teda najkratšia cesta sa nezmení

2. v sa na najkratšej ceste nenachádza teda hrany (i, v) a (v, j) nepoškodia najkratšie cesty a rátame pôvodnú hranu medzi (i, j)

Keďže algoritmus pri prechode cyklov prechádza konečne veľkú množinu vrcholov je aj konečný, teda je aj korektný.

b) Algoritmus pre hľadanie najkratších ciest z a do vrcholu v využije znalosti dĺžok najkračších ciest v grafe G', ktoré skombinuje s vchádzajucimi a vychádzajucimi hranami z vrcholu v.

Jméno: Jiří Novotný UČO: 409963

Jméno: Henrich Lauko UČO: 410438

Algoritmus 2: ComputeDistance($G, dist, v$)

vstup: vrchol $v \in G$, kde $G = (V, E)$ je graf s nenapočitanými najkratšími cestami v matici $dist$, najkratšie cesty z a do v sú iniciálne ∞
výstup: dopočítaná matica $dist$
1 **foreach** $i \in V$ **do**
2   **foreach** $j \in V$ **do**
3     $dist[i, v] \leftarrow min\{dist[i, j] + E(j, v); E(i, v)\}$// *vstupne hrany*
4     $dist[v, i] \leftarrow min\{dist[j, i] + E(v, j); E(v, i)\}$// *vystupne hrany*
5   **end foreach**
6 **end foreach**
7 **return** $dist$

Časová zložitosť Časová zložitosť algoritmu je $\mathcal{O}(|V^2|)$, keďže v algoritme iterujeme dvoma zanorenými cyklami cez množinu vrcholov V, v tele cyklu sa indexy nemenia a operácie vnútri cyklov su konštantné, teda časová zložitosť cyklov je $\mathcal{O}(|V'|) \cdot \mathcal{O}(|V'|) = \mathcal{O}(|V'^2|)$, čo je aj zložitosť celého algoritmu.

Korektnosť Z konečnosti forcyklov vyplíva, že je algoritmus konečný. Úprava matice $dist$ je správna, keďže pre dopočítanie vzdialenosti z a do vrcholu v nám stačí prejsť všetkých následnikov a predchodcov. Najkratšie vzdialenosti opäť môžeme definovať ako cestu dĺžky $dist[a, b]$, môžeme predpokladať ak existuje nejaká cesta z následnikov v do b tak najkratšia cesta bude tá, kde súčet cesty z následnika a do následnika je najkratší(riadok 4. v algoritme). Obdobne pre najkratšie cesty do vrcholu v. Teda je algoritmus korektný.

c) Algoritmus spočítania všetkých vzdialeností si môžeme predstaviť tak, že najprv vytvoríme prázdny graf do ktorého vždy pridáme jeden vrchol z pôvodneho grafu, následne budeme aplikovať algoritmus ComputeDistance zo zadania b).

Algoritmus 3: ComputeAllDistances(G)

vstup: graf $G = (E, V)$, ktorému napočítame všetky vzdialenosti
výstup: spočítané vzdialenosti $dist$
$G' \leftarrow \emptyset$// $G' = (V', E')$
$dist \leftarrow$ matica veľkosti $|V| \times |V|$ inicializovaná na ∞
foreach $v \in V$ **do**
  pridaj vrchol v do V'
  foreach $e \in E$ kde existuje hrana medzi v a vrcholmi V' **do**
    pridaj hranu e do E'
  end foreach
  $dist \leftarrow$ ComputeDistance($G', dist, v$)
end foreach
return dist

Jméno: Jiří Novotný UČO: 409963

Jméno: Henrich Lauko UČO: 410438

Časová zložitosť Je v $\mathcal{O}(|V^3|)$, keďže časová zložitosť prvého forcyklu je $\mathcal{O}(|V|)$ (prechádza všetky vrcholy), vnorené operácie operácie majú zložitosť $\mathcal{O}(|V|^2)$ - ComputeDistance bolo odvodené v podpríklade b) a prejdenie všetkých hrán je v $\mathcal{O}(|V^2|)$. Teda celková zložitosť je $\mathcal{O}(|V|) \cdot \mathcal{O}(|V^2|) = \mathcal{O}(|V^3|)$.

Korektnosť Aby bol algoritmus korektný musí byť konečný a parciálne korektný. Konečnosť je zjavná keďže focyklami prechádzame konečné množiny vrcholov a hran. Konečnosť ComputeDistance bola dokázaná v predchádzajúcom príklade. Keďže predpokládame, že ComputeDistance je korektný postupným pridávaním vrcholov môžeme tvrdiť, že aj ComputeAllDistances je korektný, keďže predpoklad algoritu ComputeDistance je že má napočítané všetky najkratšie vzdialenosti okrem vzdialeností pridávaneho vrcholu. čo v prvok kroku platí vzdialeností 0 vrcholov sú spočítané. Následne pridávame vždy len jeden vrchol, číže vždy máme len jeden vrchol s nenapočitanými vzdialenosťami. Teda je algoritmus korektný.

d) Algoritmus pracuje na princípe, že pri každom pridaní vrcholu skontroluje či sa zmenila vzdialenosť do seba sama.

Algoritmus 4: ComputeAllDistancesWithCycles(G)

vstup: graf $G = (E, V)$, ktorému napočítame všetky vzdialenosti
výstup: spočítané vzdialenosti *dist*

```
dist ← ComputeAllDistances(G)
for v ∈ V do
    dist ← ComputeDistance(G, dist, v)
    if dist[v, v] < 0 then
        for i ∈ V do
            dist[i, v] ← −∞// neexistuje najkratšia cesta
            dist[v, i] ← −∞// neexistuje najkratšia cesta
        od
    fi
od
return dist
```

Časová zložitosť Zo zadania c) vieme, že algoritmus ComputeAllDistances má časovú zložitosť $\mathcal{O}(|V^3|)$. V následnej verifikácii zápornych cyklov prechádzame $\mathcal{O}(|V|) \cdot \mathcal{O}(|V^2|)$ kde $\mathcal{O}(|V^2|)$ je zložitosť ComputeDistance. Teda celková časová zložitosť je $\mathcal{O}(|V^3|)$.

Korektnosť Algoritmus je z dokázania predchádzajúcich algoritmov a konečnosti forcyklov konečný. Algoritmus korektne nájde všetky vzdialenosti v v grafe pomocou ComputeAllDistances. Následne ak existuje v grafe záporný cyklus musí existovať kratšia cesta z vrcholu do seba sam, v ComputeAllDistances sa s takouto situáciou nepredpokladá preto neni nutné sa vracať a prepočítavať vzdialenosti, v našom algoritme pre detekciu použijeme znova spočítanie vzdialeností pre daný vrchol a pokial existuje záporný cyklus nájde algoritmus zápornu cestu zo v do v v tom prípade definujeme neexistujúcu najkratšiu cestu ako $-\infty$ cez všetky cesty vedúce cez tento vrchol. Teda je algoritmus korektný.