Vysoké učení technické v Brně
Fakulta informačních technologií

TIN

Úloha 3

18. prosince 2023 Adam Zvara (xzvara01)

1. Majme problém Lenkinho zasadacieho poriadku na svadbe definovaný následovne:
$LP = \{\langle H\rangle\#\langle M\rangle\#\langle N\rangle\#\langle S\rangle\#\langle v\rangle \mid H$ je množina hostí, M a N sú ireflexívne binárne relácie *musí sedieť s* resp. *nechce sedieť s*, S je počet stolov a V je počet miest u stolu$\}$. Dokážte, že takto zadefinovaný problém je NP-úplný.

$LP \in NP$: majme nedeterministický viacpáskový TS T, ktorý rozhoduje LP v polynomiálnom čase pracujúci následovne. T overí validitu svojho vstupu, čo realizuje v čase $\mathcal{O}(n^2)$. T nedeterministicky uhádne rozdelenie hostí ku stolom a zapíše ich na jednotlivé pásky. Každá páska predstavuje jeden stôl a hostia na páske spolu sedia pri danom stole. Ďalej TS T overí, že pre dané rozdelenie sú splnené podmienky zo zadania:

1. hostia, ktorí pri sebe musia sedieť pri sebe sedia
2. hostia, ktorí pri sebe nechcú sedieť pri sebe nesedia
3. počet hostí pri každom stole nepresahuje V

Podmienka 3 je splniteľná v čase $\mathcal{O}(n)$, keďže vyžaduje lineárny prechod cez všetky pásky. Podmienka 1 je splniteľná v čase $\mathcal{O}(n)$, keďže vyžaduje lineárny prechod cez množinu M a kontrolu pásiek v lineárnom čase. Nakoniec posledná podmienka je taktiež skontrolovaná v čase $\mathcal{O}(n^2)$ a teda všetky kontroly je T schopný vykonať v polynomiálnom čase (existuje polynomiálny verifikátor). Ak sú priradenia hostí ku stolom platné, T akceptuje jeho vstup, inak zamieta.

LP je NP-ťažký: dokážeme zostrojením polynomiálnej redukcie σ, ktorá ukáže, že
$GRAPH - COLORING \leq_p LP$, pričom $GRAPH - COLORING$ je problém obarvenia grafu, ktorý je definovaný ako:

$GRAPH-COLORING = \{\langle G\rangle\#\langle k\rangle \mid$ G je kód neorientovaného grafu a k je kód počtu bariev, ktoré pridelíme vrcholom grafu tak, aby žiadne 2 vrcholy spojené hranou nemali rovnakú barvu$\}$.

Je známe, že problém $GRAPH - COLORING$ je NP-úplný a teda danou redukciou dokážeme, že problém LP je NP-ťažký, čo v spojení s tým, že $LP \in NP$ dokazuje, že LP je NP-úplný. Zostrojíme teda TS M_σ, ktorý bude realizovať redukciu:

$$\sigma(\langle G\rangle\#\langle k\rangle) = \langle H\rangle\#\langle M\rangle\#\langle N\rangle\#\langle S\rangle\#\langle v\rangle$$

pracujúcu následujúcim spôsobom. Ak problém $GRAPH - COLORING$ nie je valídne zakódovaný problém, TS M_σ vráti následujúce kódy: $H = \{A, B\}, M = N = \{\}, S = v = 1$, kde A, B sú kódy dvoch hostí (je zrejmé, že neexistuje usadenie 2 hostí pri jednom stole s kapacitou jedna). V opačnom prípade vracia M_σ následujúce kódy:

$$\begin{aligned} H &= V \\ M &= \{\} \\ N &= \{(i,j) \mid i,j \in V \wedge i \neq j \wedge \{i,j\} \in E\} \\ S &= k \\ v &= |V| \end{aligned}$$

Neformálny popis redukcie: redukcia mapuje vrcholy grafu G na množinu hostí H, teda každý hosť predstavuje jeden vrchol v grafe G. Hrany grafu G sa namapujú na množinu N, teda ak sú v grafe spojené 2 vrcholy jednou hranou, tak daní hostia pri sebe nechcú sedieť. Počet barev k sa namapuje na počet stolov S. Tieto kroky samotné by nám stačili na redukovanie problému $GRAPH - COLORING$ na LP, ale musíme ešte vyriešiť množinu M a veľkosť stolu v. Ideálne to môžeme vyriešiť tak, aby nám tieto časti nekomplikovali život a nijako nevplývali na zvyšok problému. Môžeme teda zafixovať množinu M na prázdnu množinu, ktorá nebude mať vplyv na usadenie hostí pri stoloch a v zvoliť natoľko veľké, aby nám taktiež nijako neovplyvňovalo riešenie, teda zvolili sme $v = |V|$ čo znamená, že pri každom stole môžu sedieť všetci hostia.

Je zrejmé, že pri (takto naformulovanej redukcii) ak existuje riešenie problému $GRAPH - COLORING$, tak bude existovať riešenie problému LP a naopak, teda
$\langle G\rangle\#\langle k\rangle \in GRAPH - COLORING \iff \sigma(\langle G\rangle\#\langle k\rangle) \in LP$.

V poslednom kroku musíme ukázať, že TS M_σ pracuje v polynomiálnom čase. Kontrola správnosti formátu problému $GRAPH - COLORING$ prebieha v čase $\mathcal{O}(n)$. Na vytvorenie množiny hostí H, počtu stolov S a veľkosti jedného stolu v je potrebný lineárny prechod cez vstup, čo odpovedá $\mathcal{O}(n)$. Vytvorenie relácie N vyžaduje lineárny prechod cez množinu hrán grafu G ($\mathcal{O}(n)$), pričom pre každý vrchol kontrolujeme, či sa

skutočne nachádza v množine V ($2 \cdot \mathcal{O}(n)$), teda zložitosť tejto operácie je $\mathcal{O}(n^2)$. Vytvorenie relácie M prebieha v konštantnom čase $\mathcal{O}(1)$. TS M_σ teda pracuje v čase $\mathcal{O}(n^2)$.

Ukázali sme, že $LP \in NP$ a zároveň že $GRAPH - COLORING \leq_p LP$, z čoho vyplýva, že problém LP je NP-úplný.

* * *

2. Implementácia operácie `vlož_a_uřež(k)` realizovanú nad obojstranne viazaným zoznamom tak, aby sekvencia n operácií bežala v čase $\mathcal{O}(n)$.

Naivná implementácia tejto operácie pozostávajúca z lineárneho prechod zoznamom od počiatočného prvku až po prvok, pred ktorý máme vložiť nový prvok k je nevyhovujúca. Pre sekvenciu N operácií `vlož_a_uřež(k)`, pričom každá operácia vloží prvok o 1 väčší ako predchádzajúca sa dostaneme na časovú zložitosť $\mathcal{O}(n^2)$.

Môžeme však využiť invariant, ktorý vyplýva z funkcie `vlož_a_uřež` – zoznam je po každom vykonaní tejto operácie zoradený (pretože dochádza ku vymazaniu položiek > k). Z tejto vlastnosti algoritmu vyplýva to, že môžeme zoznam prechádzať od posledného prvku, následne nájdeme nový index, kde vložíme prvok k a popri tom, ako budeme prechádzať zoznam môžeme vypisovať a zároveň mazať prvky > k (v zadaní nie je podmienka, že prvky musia byť vypísané v poradí, v ktorom sa v zozname nachádzajú).

Algorithm 1: vloz_a_urez(DLL, k) // DLL = Doubly Linked List

```
while (last = DLL.tail()) ≠ NULL do
    if last ≥ k then
        print(last)
        DLL.pop()
    end
    else
        break
    end
end
DLL.insert_end(k)
```

Intuitívne si môžeme predstaviť, že "drahá" operácia bude skutočne prebiehať iba po určitej sekvencii kratších operácií (operácia vloženia malého prvku do zoznamu s veľkým množstvom prvkov bude drahá, ale takýchto operácií nemôže nastať n, pretože pri vložení veľmi malého prvku sa vymaže veľká časť zoznamu).

Amortizovanú zložitosť sekvencie n operácií `vlož_a_uřež` môžeme analyzovať pomocou metódy účtov:

operácia	cena	kredity
tail	1	1
compare (≥)	1	1
print	1	0
pop	1	0
insert_end	1	3

Vysvetlenie ohodnotenia jednotlivých operácií: do tabuľky si zapíšeme všetky čiastkové operácie nachádzajúce sa v metóde `vlož_a_uřež`. Keďže je cena týchto operácií konštantná, priradíme im ceny rovné 1.

Operácia `insert_end`: táto operácia si v kreditoch zaplatí samu za seba. Zároveň táto operácia však vkladá do zoznamu nový prvok, ktorý môže byť niekedy v ďalšom volaní tejto funkcie vypísaný (+1) a zároveň s tým aj vymazaný (+1). Kredity pre túto operáciu sú teda 3.

Operácie `print` a `pop`: tieto operácie pracujú len s tými prvkami, ktoré boli vložené do zoznamu operáciou `insert_end` a zároveň ešte neboli vymazané. Ich práca však už bola predplatená a z toho dôvodu je počet pridelených kreditov 0.

Operácie `tail` a `compare`: tieto operácie jednoducho predplácajú svoje potenciálne volanie do budúcna nad prvkami, preto je im priradený 1 kredit.

Stav účtu si môžeme vypísať pre malú sekvenciu operácií `vlož_a_uřež`:

obsah zoznamu	**operácia**	**sub-operácia**	**stav účtu**
		—	***0***
[]	`vloz_a_urez(1)`	DLL.tail	***0***
		DLL.insert_end	***2***
		DLL.tail	***2***
[1]	`vloz_a_urez(5)`	compare ($5 \geq 1$)	***2***
		DLL.insert_end	***4***
		DLL.tail	***4***
[1, 5]	`vloz_a_urez(10)`	compare ($10 \geq 5$)	***4***
		DLL.insert_end	***6***
		DLL.tail	***6***
		compare ($3 \geq 10$)	***6***
		print	***5***
		pop	***4***
[1, 5, 10]	`vloz_a_urez(3)`	compare ($3 \geq 5$)	***4***
		print	***3***
		pop	***2***
		compare ($3 \geq 1$)	***2***
		DLL.insert_end	***4***

Ako môžeme vidieť, platí invariant, že počet kreditov na účte je rovný dvojnásobku počtu prvkov v danom zozname, z čoho vyplýva, že sú splnené podmienky pre metódu účtov a sekvencia n operácií tejto metódy má zložitosť $\leq 5n (\in \mathcal{O}(n))$ – cena jedného volania metódy v sekvencii n volaní je $\mathcal{O}(1)$.