Teoretická informatika

2022/2023

Domácí úloha 3

Pavel Šesták (xsesta07)

Brno, 18. prosince 2022

Obsah

Seznam obrázků

Seznam tabulek

Teoretická informatika (TIN) – 2022/2023
Úkol 3
(max. zisk 5 bodů – 10 bodů níže odpovídá 1 bodu v hodnocení předmětu)

1. Dokažte redukcí, že následující jazyk L není ani částečně rozhodnutelný:

   $$L = \{\langle M_1 \rangle \sharp \langle M_2 \rangle \mid M_1 \text{ a } M_2 \text{ jsou Turingovy stroje takové, že } L(M_1) \leq L(M_2)\}$$

   (Připomínáme, že $L(M_1) \leq L(M_2)$ znamená, že $L(M_1)$ se redukuje na $L(M_2)$.)

   15 bodů

2. Pro danou konečnou podmnožinu M přirozených čísel a číslo k je úkolem rozhodnout, zda existuje rozklad R množiny M, který má maximálně k tříd, a kde každé dva různé prvky i, j stejné třídy r jsou nesoudělné. Dokažte, že se jedná o NP-úplný problém. Čísla jsou kódována binárně.[1]

   Inspirujte se zde: `https://en.wikipedia.org/wiki/List_of_NP-complete_problems`

   15 bodů

3. Uvažujte *množinový term* t definovaný gramatikou

   $$t ::= \emptyset \mid (t \cup t) \mid \mathsf{add\_max}(t) \mid \mathsf{remove\_min}(t)$$

   Hodnotu $h(t)$ termu t (slova z jazyka gramatiky) definujeme induktivně jako množinu přirozených čísel:

   $$\begin{array}{rlrl} h(\emptyset) = \emptyset & & h(\mathsf{add\_max}(t)) = h(t) \cup \{\max(h(t)) + 1\} & \text{(vložení maxima plus 1)} \\ h(t \cup t') = h(t) \cup h(t') & & h(\mathsf{remove\_min}(t)) = h(t) \setminus \{\min(h(t))\} & \text{(odebrání minima)} \end{array}$$

   ($\max(\emptyset)$ definujeme jako 0 a $\min(\emptyset)$ jako ∞)

   (a) Navrhněte pseudokód algoritmu, který vyhodnotí daný term t, tj., vrátí $h(t)$.
   Algoritmus bude term vyhodnocovat rekurzivně od listů ke kořeni.
   Reprezentujte hodnoty termů, množiny čísel, jako dvousměrně vázané seznamy čísel. Vysvětlete, jak implementovat operace tak, aby pro konečné množiny S, S' mělo vyhodnocení operací $\mathsf{add\_max}(S)$ a $\mathsf{remove\_min}(S)$ konstantní časovou složitost, a aby vyhodnocení sjednocení S a S' mělo složitost lineární k maximu množiny s menším maximem. (Máte k dispozici operace získání odkazu na první/poslední/další prvek dvousměrného seznamu, vložení prvku před/za odkazovanou pozici, které mají konstantní časovou složitost. Nepopisujte jejich implementaci na úrovni manipulace s ukazateli.)

   (b) Ukažte, že Váš algoritmus má lineární časovou složitost (běží v čase $O(|t|)$[2]).
   Čas výpočtu $h(t)$ bude součtem času potřebného k vyhodnocení všech výskytů operací v t.
   Budete potřebovat argumentovat, že čas vyhodnocení každého sjednocení $h(t \cup t')$ na základě $h(t)$ a $h(t')$ se amortizuje časem potřebným pro výpočet $h(t)$ a $h(t')$.[3]

   Zvolte vhodnou úroveň abstrakce algoritmů a důkazu složitosti. Vágní nebo zbytečně technický popis nebude akceptován.

   20 bodů

[1] Můžete předpokládat, že hodnota n-tého nejmenšího prvočísla je polynomiální k n.
Můžete také použít následující dodatečné předpoklady: (1) výpočet prvních n prvočísel je polynomiální k n, (2) faktorizace binárně zapsaného čísla je v NP. Předpoklady (1) a (2) byste ale měli být schopni zdůvodnit. Pokud je nepoužijete, tj., nebudete je pouze předpokládat ale zdůvodníte je (nebo příklad správně vyřešíte bez nich), bude to oceněno odpuštěním až cca 10 bodů v případě chyb jinde.

[2] $|t|$ je jednoduše délka slova t.

[3] Uvědomte si, že naivním součtem složitostí nejhoršího případů jednotlivých operací bychom dostali příliš velkou, kvadratickou, složitost vyhodnocení $h(t)$: čas potřebný k vyhodnocení sjednocení je lineární k velikosti operandů, velikost operandu je ohraničena počtem výskytů $\mathsf{add\_max}$ v t, vyhodnocení jednoho sjednocení je tedy v $O(|t|)$. Sjednocení se vyskytuje maximálně $|t|$-krát, dohromady je tedy čas všech sjednocení v $O(|t|)^2$.

1 Důkaz redukcí, že jazyk L není ani částečně rozhodnutelný

1.1 Použité jazyky v rámci redukční funkce

co-HP = {<M>#<w>| TS M na w nezastaví}
HP = {<M>#<w>| TS M na w zastaví}

1.2 Redukce na co-HP

Uvažmě redukci $co - HP \leq L$
Požadovaná redukce je funkce $\sigma : \{0, 1, \#\}^* \rightarrow \{0, 1, \#\}^*$

Algorithm 1: Algoritmus popisující chování redukční funkce σ, která je implementována úplným Turingovým strojem M_x

Input : <M>#<w>
Output: $<M_1>\#<M_2>$

M_x sestrojí kód TS M_2, pro který platí L(M_2) = HP
M_x provede syntaktickou kontrolu vstupu
if *se nejedná o korektní instanci problému pro co-HP* **then**
    M_x sestrojí kód TS M_1, pro který platí L(M_1) = co-HP
else
    M_x sestrojí kód TS M_1, který pracuje následovně:
    M_1 spustí simulaci M na vstupu w (Kód M a w jsou přímo uloženy v kódu TS M_1)
    if *Simulace M na vstupu w skončí* **then**
        M_1 dále pracuje tak, že L(M_1) = co-HP
    else
        M_1 cyklí při simulaci M na vstupu w a tedy L(M_1) = $\emptyset$
    end
end
return $<M_1>\#<M_2>$

1.3 Ověření totálnosti redukční funkce a zachovává ní příslušnosti do jazyka

Je zřejmé, že mohou nastat tři situace:
1. Vstup <M>#<w> není korektní instance co-HP problému:

$$< M > \# < w > \notin co - HP \Leftrightarrow co - HP \not\leq HP$$

2. TS M na w nezastaví:

$$< M > \# < w > \in co - HP \Leftrightarrow \emptyset \leq HP$$

3. TS M na w zastaví:

$$< M > \# < w > \notin co - HP \Leftrightarrow co - HP \not\leq HP$$

Funkce sigma je tedy totální a zachovává příslušnost.

2 Důkaz NP-úplnosti

Problém popsaný v bodu 2 zadaní označíme jako L_P. NP-úplný problém je právě když náleží do NP a je NP-těžký.

2.1 Důkaz, že $L_P \in NP$

Sestrojíme 4 páskový NTS M takový, že L(M) = L_P a M pracuje v polynomiálním čase. Turingův stroj M implementuje Eukleidův algoritmus. První páska obsahuje instanci problému. Druhá a třetí páska obsahuje nedeterministicky zvolená čísla. Čtvrtá páska obsahuje zbytek po dělení.
1. M nedeterministicky zvolí dva různé prvky i, j z jedné třídy.
2. M překopíruje i na druhou pásku (u)
3. M překopíruje j na třetí pásku (w)
4. M vydělí druhou pásku třetí a zbytek po dělení uloží na čtvrtou pásku (r)
5. M překopíruje obsah třetí pásky na druhou pásku
6. M překopíruje obsah čtvrté pásky na třetí pásku
7. Pokud třetí páska není prázdná pokračuj krokem 4.
8. Pokud obsah druhé pásky obsahuje jeden symbol, který není blank tak M akceptuje.
9. Jinak M zamítá.

2.1.1 Analýza časové složitosti NTS M

Je zřejmé, že M pracuje v polynomiálním čase vzhledem k délce operandu dělení.
M zvolí dva prvky což je v O(1)
Výpočet zbytku po dělení lze provést v O(n) a je zde provedeno log(n) krát, takže O(n*log(n))[2].
M přepíše druhou a třetí pásku log(n) krát, tedy 2 * O(log(n)*n)
M rozhodne výsledek v O(1)

Algoritmus tedy pracuje v O(n*log(n)) na stroji s RAM. Turingův stroj zvládne simulovat stroj s RAM v čase O(t^2(n) * log(n))
Ve výsledku tedy náš stroj bude pracovat v třídě asymptotické složitosti $O((n * log(n))^2 * log(n))$ což je polynomiální.

2.2 Důkaz, že L_P je NP-těžký

Pro důkaz NP-těžkosti L_P použijeme redukci z problému K-obarvitelnosti, který je NP-úplný. Ukážeme tedy, že k-obarvitelnost grafu se redukuje na náš problém, čímž ukážeme NP-těžkost.

K-obarvitelnost = {<G>#<k>| G je obarvitelný k barvami a žádní dva sousedi nemají stejnou barvu}

$L_{SP} \leq L_P$

Redukce má tvar σ(<G>#<k>) → <M>#<k> a implementuje jí čtyř páskový úplný Turingův stroj M_x

Algorithm 2: Algoritmus popisující chování redukční funkce σ, která je implementována úplným Turingovým strojem M_x

Input : <G>#<k>
Output: <M>#<k>

M_x zkontroluje, že <G>#<k> je korektní instance K-obarvitelnosti
if *<G>#<k> není korektní instance K-obarvitelnosti* **then**
    M_x zkonstruuje kód množiny <M>, kde M je tří prvková množina {2,4,6}
    M_x zkonstruuje kód čísla <k>, kde k = 2
    return <M>#<k>
end
M_x zkonstruuje |E| prvočísel, která si uloží na druhou pásku a jednotlivá prvočísla oddělí oddělovačem #. M_x přiřadí každé hraně e ∈ E právě jedno prvočíslo, přičemž toto mapování si uloží na třetí pásku
M_x začne sestrojovat výstupní množinu M, pro kterou platí: |M| = |V|. Každé číslo množiny bude odpovídat jednomu vrcholu, kde hodnota vrcholu je rovna součinu ohodnocení hran prvočísli, které do daného vrcholu vedou
return <M>#<k>

2.2.1 Analýza časové složitosti redukční funkce

Syntaktická kontrola vstupu O(n)
Nesprávná instance O(1)
Konstrukce k prvočísel je polynomiální
Tvorba výstupní množiny O(n^2)
Redukce tedy pracuje v polynomiálním čase.

2.2.2 Analýza časové složitosti vypočtení prvních k prvočísel

Ověření zdali je číslo prvočíslem jsme schopni v polynomiálním čase ověřit například pomocí AKS testu prvočíselnosti[1], který pracuje v $\tilde{O}(log(n)^{12})$, $\tilde{O}$ je asymptotická časová složitost, která ignoruje logaritmické prvky. Tento algoritmus je tedy polynomiální. Dále jsme schopni generovat čísla dokud nedostanem k prvočísel a pro každé ověřit pomocí AKS prvočíselnost. Tedy ten algoritmus se pustí nkrát, ale časová složitost bude stále polynomiální.

3 Pseudokód návrhu algoritmu pro vyhodnocení termu

Algorithm 3: Algoritmus popisující metody pro vyhodnocení termu

```
procedure ADD_MAX(t : DoubleLinkedList)
newMax = t.First()
if newMax == ∅ then
    newMax = 0
end
newMax = newMax + 1
t.InsertFirst(newMax)
return t
end procedure

procedure REMOVE_MIN(t : DoubleLinkedList)
t.RemoveLast() //RemoveLast has no effect on empty list
end procedure

procedure UNION(t : DoubleLinkedList, u : DoubleLinkedList)
result:DoubleLinkedList
//Iterate over lists until one of them is empty
while (tItem = t.First()) != NULL ∧ (uItem = u.First()) != NULL do
    if tItem.isEqual(uItem) then
        result.InsertLast(tItem)
        t.RemoveFirst()
        u.RemoveFirst()
    end
    if tItem > uItem then
        result.InsertLast(tItem)
        t.RemoveFirst()
    end
    if tItem < uItem then
        result.InsertLast(uItem)
        u.RemoveFirst()
    end
end
//Iterate over unempty list
while (tItem = t.First()) != NULL do
    result.InsertLast(tItem) t.RemoveFirst()
end
while (uItem = u.First()) != NULL do
    result.InsertLast(uItem) u.RemoveFirst()
end
return result
end procedure
```

Reference

[1] AKS primality test.
URL https://en.wikipedia.org/wiki/AKS_primality_test

[2] Gollova, A.: Counting modulo n and its time complexity. Dostupné z https://math.fel.cvut.cz/en/people/gollova/emkr/e-mcr2a.pdf.