Teorie algoritmů — 7. týden

Marie Demlová
http://math.fel.cvut.cz/en/people/demlova

2. 4. 2024

Realistický model počítače

Model počítače

1. **Paměť** se skládá z potenciálně nekonečného počtu *slov*, každé má *adresu*.
2. **Program** je uložen v některých slovech paměti. Každé slovo — jednoduchá instrukce. Připouštíme nepřímé adresovaní.
3. Každá instrukce obsahuje omezený počet slov a mění obsah nejvýše jedné adresy.

Příklady instrukcí: copy, add (hodnotu jedné buňky k hodnotě jiné buňky), jump a pod.

Vztah mezi Turingovými stroji a počítači

Věta 1. Ke každému Turingovu stroji M existuje program P takový, že oba mají stejné chování. Navíc, jestliže M potřeboval n kroků, P má časovou složitost $\mathcal{O}(n^2)$.

Věta 2. Pro každý program P existuje Turingův stroj M s pěti páskami takový, že P i M mají stejné chování.

Vztah mezi Turingovými stroji a počítači

Věta 3. Jestliže program P splňuje následující podmínky:

- program obsahuje pouze instrukce, které zvětšují délku binárně zapsaného čísla maximálně o jednu;
- program obsahuje pouze instrukce, které Turingův stroj s více páskami provede na slovech délky k v $\mathcal{O}(k^2)$ krocích,

pak Turingův stroj z věty 2 simuluje n kroků programu P pomocí $\mathcal{O}(n^3)$ svých kroků.

Důsledek. Je dán program P, který splňuje podmínky z předchozí věty. Pak existuje Turingův stroj s jednou páskou, který má stejné chování jako P a n kroků programu P simuluje pomocí $\mathcal{O}(n^6)$ svých kroků.

Rozhodovací úlohy a jazyky

Rozhodovací úlohy

Rozhodovací úloha/problém je taková úloha, jejímž „řešením“ je buď odpověď „ANO“ nebo odpověď „NE“.

Příklady

- SAT — splňování Booleovských formulí (v CNF).
- Existence Hamiltonovské cesty/cyklu/kružnice.
- Je možné obarvit graf 3 barvami?

Rozhodovací úlohy a jazyky

Rozhodovací verze optimalizačních problémů.

Pro každý optimalizační problém vytvoříme jeho rozhodovací verzi např. takto:

- Je dán souvislý neorientovaný graf G spolu s délkami hran $a\colon E \to \mathbb{Z}$. Je dáno přirozené číslo K. Existuje kostra G délky nejvýše K?
- Je dána instance TSP a přirozené číslo K. Existuje trasa délky nejvýše K?

Rozhodovací úlohy a jazyky

Věta.

Kdyby existoval polynomiální algoritmus $\mathcal{A}$, který řeší rozhodovací verzi TSP, pak také existuje polynomiální algoritmus, který řeší optimalizační verzi TSP.

Rozhodovací úlohy a jazyky

Rozhodovací úlohy a jazyky.

Je dán rozhodovací úloha $\mathcal{U}$. Všechny instance této úlohy rozdělíme do dvou skupin — ANO instance a NE instance.

Instance úlohy je možné zakódovat jako slovo nad vhodnou abecedou Σ. Jazyk

$$L_{\mathcal{U}} = \{w \mid w \text{ je kód ANO instance}\}$$

je jazyk úlohy $\mathcal{U}$.

Slova nad Σ, které neodpovídají žádné instanci úlohy, považujeme za NE instance.

Třídy složitosti

Třída $\mathcal{P}$.

Rozhodovací úloha $\mathcal{U}$ patří do třídy $\mathcal{P}$ právě tehdy, když existuje deterministický Turingův stroj M, který přijímá jazyk $L_{\mathcal{U}}$ a pracuje s polynomiální časovou složitostí.

Příklady.

- Rozhodovací verze problému minimální kostry.
- Rozhodovací verze úlohy nalezení nejkratších cest v ohodnoceném grafu bez cyklů záporné délky.
- Rozhodovací verze nalezení maximálního toku v síti.
- Rozhodovací verze úlohy nalezení minimálního řezu.

Třídy složitosti

Třída $\mathcal{NP}$.

Rozhodovací úloha $\mathcal{U}$ patří do třídy $\mathcal{NP}$ právě tehdy, když existuje nedeterministický Turingův stroj M, který přijímá jazyk $L_{\mathcal{U}}$ a pracuje s polynomiální časovou složitostí.

Příklady.

- SAT problém.
- Existence hamiltonovské cesty/kružnice, hamiltonovského cyklu.
- Rozhodovací verze problému batohu.
- Rozhodovací verze problému klik.

Třídy složitosti

Nedeterministický „algoritmus“.

Nedeterministický algoritmus se skládá ze dvou fází

- v první fázi se vygeneruje řetězec s;
- na základě vstupní instance a na základě s (deterministický) algoritmus dá odpověď buď „ano“ nebo „ne“.

Nedeterministický algoritmus řeší úlohu, jestliže

1) pro každou ANO instanci existuje řetězec s, pro který druhá fáze odpoví ano;
2) pro žádnou NE instanci neexistuje řetězec s, pro který druhá fáze odpoví ano.

Nedeterministcký algoritmus pracuje v čase $\mathcal{O}(T(n))$, jestliže každý průchod oběma fázemi 1 a 2 pro instanci velikosti n potřebuje $\mathcal{O}(T(n))$ kroků.

Třída $\mathcal{NPC}$

Redukce a polynomiální redukce úloh.

Jsou dány dvě rozhodovací úlohy $\mathcal{U}$ a $\mathcal{V}$. Řekneme, že úloha $\mathcal{U}$ se redukuje na úlohu $\mathcal{V}$, jestliže existuje algoritmus (program pro počítač, Turingův stroj) M, který pro každou instanci I úlohy $\mathcal{U}$ zkonstruuje instanci I' úlohy $\mathcal{V}$ a to tak, že

I je ANO-instance $\mathcal{U}$ právě tehdy, když I' je ANO-instance $\mathcal{V}$.

Značíme to

$$\mathcal{U} \lhd \mathcal{V}.$$

Jestliže M pracuje s polynomiální časovou složitostí, $\mathcal{U}$ se polynomiálně redukuje na $\mathcal{V}$, a píšeme

$$\mathcal{U} \lhd_p \mathcal{V}.$$

Třída $\mathcal{NPC}$

Tvrzení. Jsou dány tři rozhodovací úlohy $\mathcal{U}$, $\mathcal{V}$ a $\mathcal{W}$. Jestliže platí

$$\mathcal{U} \lhd_p \mathcal{V} \quad \text{a} \quad \mathcal{V} \lhd_p \mathcal{W},$$

pak

$$\mathcal{U} \lhd_p \mathcal{W}.$$

Třída $\mathcal{NPC}$

$\mathcal{NP}$ úplné úlohy.

Rozhodovací úloha $\mathcal{U}$ je $\mathcal{NP}$ úplná jestliže

1. $\mathcal{U}$ patří do třídy $\mathcal{NP}$;
2. každá $\mathcal{NP}$ úloha se polynomiálně redukuje na $\mathcal{U}$.

$\mathcal{NPC}$ je třída všech $\mathcal{NP}$ úplných úloh.

Tvrzení.

Jsou dány dvě $\mathcal{NP}$ úlohy $\mathcal{U}$ a $\mathcal{V}$, pro které platí $\mathcal{U} \lhd_p \mathcal{V}$. Pak

1. jestliže $\mathcal{V}$ je v $\mathcal{P}$, pak také $\mathcal{U}$ je v $\mathcal{P}$;
2. jestliže $\mathcal{U}$ je $\mathcal{NP}$ úplná úloha, pak také $\mathcal{V}$ je $\mathcal{NP}$ úplná úloha.

Třída $\mathcal{NPC}$

Věta.

$$\mathcal{P} \subseteq \mathcal{NP}.$$

Zda $\mathcal{P} = \mathcal{NP}$ otevřená otázka.

Tvrzení.

Kdyby některá $\mathcal{NP}$ úplná úloha patřila do třídy $\mathcal{P}$ (tj. byla by polynomiálně řešitelná), pak $\mathcal{P} = \mathcal{NP}$.