České vysoké učení technické v Praze
Fakulta elektrotechnická

Tématické okruhy ke státní závěrečné zkoušce pro magisterský studijní program Otevřená Informatika (OI)

http://www.fel.cvut.cz/cz/education/master/topicsOI.html

OI Mgr - Společné

Vygenerováno: 19. června 2014 23:35

Obsah

1 Amortizovaná složitost. Prioritní fronty, haldy (binární, d-regulární, binomiální, Fibonacciho), operace nad nimi a jejich složitost.

2 Neorientované a orientované grafy, jejich reprezentace. Prohledávání grafu (do hloubky a do šířky), topologické uspořádání, souvislost, stromy, minimální kostra.

3 Lexikální analyzátor, syntaktický strom, syntaktický analyzátor shora dolů, LL(1) gramatiky, rozkladové tabulky.

4 Algoritmy vyhledávaní v textu s lineární a sublineární složitostí, (naivní, Boyer-Moore), využití konečných automatů pro přesné a přibližné hledání v textu.

5 Algoritmus, správnost algoritmu, složitost algoritmu, složitost úlohy, třída P, třída NP.

Algoritmus. *Algoritmem* rozumíme dobře definovaný proces, tj. posloupnost výpočetních kroků, který přijímá hodnoty (zadání, vstup) a vytváří hodnoty (řešení, výstup).

Řekneme, že algoritmus $\mathcal{A}$ *řeší* úlohu $\mathcal{U}$, jestliže pro každý vstup (každou instanci problému $\mathcal{U}$) vydá správné řešení.

Správnost algoritmu K ověření správnosti algoritmu je třeba ověřit 2 věci:

1. algoritmus se na každém vstupu zastaví
2. algoritmus po zastavení vydá správný výstup - řešení

Variant. Pro důkaz faktu, že se algoritmus na každém vstupu zastaví, je založen na nalezení tzv. *variantu.* Variant je hodnota udaná přirozeným číslem, která se během práce algoritmu snižuje až nabude nejmenší možnou hodnotu (a tím zaručuje ukončení algoritmu po konečně mnoha krocích).

Invariant. *Invariant*, též *podmíněná správnost algoritmu*, je tvrzení, které:

- platí před vykonáním prvního cyklu algoritmu, nebo po prvním vykonání cyklu
- platí-li před vykonáním cyklu, platí i po jeho vykonání
- při ukončení práce algoritmu zaručuje správnost řešení

Složitost algoritmu

Algoritmy lze rozdělit do několika tříd složitosti na základě času a paměti, jež potřebují ke svému vykonání na různých typech Turingových strojů. [1]

Časovou složitost algoritmu udáváme jako asymptotický odhad $T(n)$ času potřebného pro vyřešení každé instance velikosti n.

Asymptotický růst funkcí

Definujeme několik symbolů (množin).

Symbol $\mathcal{O}$. Je dána nezáporná funkce $g(n)$. Řekneme, že nezáporná funkce $f(n)$ je $\mathcal{O}(g(n))$, jestliže existuje kladná konstanta c a přirozené číslo n_0 tak, že

$$f(n) \leq c \cdot g(n) \text{ pro všechny } n \geq n_0$$

$\mathcal{O}(g(n))$ můžeme též chápat jako třídu všech nezáporných funkcí $f(n)$:

$$\mathcal{O}(g(n)) = \{f(n) \mid \exists c > 0, n_o \text{ tak, že } f(n) \leq c \cdot g(n)\ \forall n \geq n_0\}$$

Další symboly:

- $\Omega(g(n)) = \{f(n) \mid \exists c > 0, n_o \text{ tak, že } f(n) \geq c \cdot g(n)\ \forall n \geq n_0\}$
- $\Theta(g(n)) = \{f(n) \mid \exists c_1, c_2 > 0, n_o \text{ tak, že } c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\ \forall n \geq n_0\}$
- $o(g(n)) = \{f(n) \mid \forall c > 0 \exists n_o \text{ tak, že } 0 \leq f(n) < c \cdot g(n)\ \forall n \geq n_0\}$
- $\omega(g(n)) = \{f(n) \mid \forall c > 0 \exists n_o \text{ tak, že } 0 \leq c \cdot g(n) < f(n)\ \forall n \geq n_0\}$

Tranzitivita $\mathcal{O}, \Omega, \Theta$. Máme dány tři nezáporné funkce $f(n), g(n), h(n)$

- Jestliže $f(n) \in \mathcal{O}(g(n))$ a $g(n) \in \mathcal{O}(h(n))$, pak $f(n) \in \mathcal{O}(h(n))$
- Jestliže $f(n) \in \Omega(g(n))$ a $g(n) \in \Omega(h(n))$, pak $f(n) \in \Omega(h(n))$
- Jestliže $f(n) \in \Theta(g(n))$ a $g(n) \in \Theta(h(n))$, pak $f(n) \in \Theta(h(n))$

Reflexivita $\mathcal{O}, \Omega, \Theta$. Pro všechny nezáporné funkce $f(n)$ platí:

- $f(n) \in \mathcal{O}(f(n))$
- $f(n) \in \Omega(f(n))$
- $f(n) \in \Theta(f(n))$

Master Theorem

Používá se pro určení asymptotického časového odhadu u rekurentních vztahů.
Jsou dána přirozená čísla $a \geq 1, b \geq 1$ a funkce $f(n)$. Předpokládejme, že funkce $T(n)$ je dána na přirozených číslech rekurentním vztahem

$$T(n) = aT\left(\frac{n}{b}\right) + f(n), \text{ kde } \frac{n}{b} \text{ znamená buď } \lfloor \tfrac{n}{b} \rfloor \text{ nebo } \lceil \tfrac{n}{b} \rceil.$$

1. Jesltiže $f(n) \in \mathcal{O}(n^{\log_b a - \epsilon})$ pro nějakou konstantu $\epsilon > 0$, pak $T(n) \in \Theta(n^{\log_b a})$.
2. Jesltiže $f(n) \in \Theta(n^{\log_b a})$, pak $T(n) \in \Theta(n^{\log_b a} \lg n)$.
3. Jesltiže $f(n) \in \Omega(n^{\log_b a + \epsilon})$ pro nějakou konstantu $\epsilon > 0$ a jestliže $af(\frac{n}{b}) \leq cf(n)$ pro nějakou konstantu $c < 1$ pro všechna dostatečně velká n, pak $T(n) \in \Theta(f(n))$.

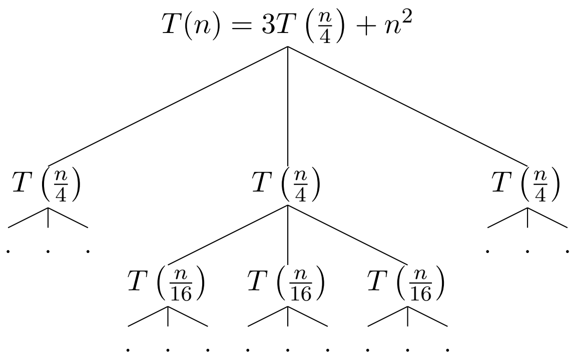
Poznámka. MT nepokrývá všechny případy.

► **Příklad:**

$T(n) = 6T\left(\frac{n}{4}\right) + n^2. \lg(n)$ // 3. případ $n^2 \lg(n) \in \Omega(n^{\log_4 6})$
$6\left(\frac{n}{4}\right)^2 \lg\left(\frac{n}{4}\right) \leq c \cdot n^2 \lg(n)$ // roznásobení
$\frac{6}{16} n^2 \lg\left(\frac{n}{4}\right) \leq c \cdot n^2 \lg(n)$ // $\lg\left(\frac{n}{4}\right)$ si "zvětším"na $\lg(n)$
$\frac{6}{16} n^2 \lg(n) \leq c \cdot n^2 \lg(n)$ // pokrátím (vydělím) $n^2 \lg(n)$
$\frac{6}{16} \leq c$ // $c < 1$, platí
$\rightarrow T(n) = \Theta(n^2 \lg(n))$

Řešení rekurzivních vztahů pomocí rekurzivních stromů

▶ **Příklad:**



Vytvoříme si jednotlivé hladiny stromu, který popisuje rekurzivní výpočet funkce $T(n)$. V nulté hladině máme pouze $T(n)$ a hodnotu n^2, kterou potřebujeme k výpočtu $T(n)$ (známe-li $T\left(\frac{n}{4}\right)$.

V první hladině se nám výpočet $T(n)$ rozpadl na tři výpočty $T(n)$. K tomu potřebujeme hodnotu $3 \cdot (\frac{n}{4})^2 = \frac{3}{16}n^2$.

Při přechodu z hladiny i do hladiny $i+1$ se každý vrchol rozdělí na tři a každý přispěje do celkové hodnoty jednou šestnáctinou předchozího. Proto je součet v hladině i roven $(\frac{3}{16})^i n^2$.

Poslední hladina má vrcholy označené hodnotami $T(1)$ a tím rekurze končí. Počet hladin odpovídá $\log_4 n$. V poslední hladině je $3^{\log_4 n} = n^{\log_4 3}$ hodnot $T(1)$. Proto platí

$$T(n) = \sum_{i=0}^{\log_4 n} \left(\frac{3}{16}\right)^i n^2 + \Theta(n^{\log_4 3})$$

Odtud

$$T(n) < n^2 \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i + \Theta(n^{\log_4 3}) = n^2 \frac{1}{1 - \frac{3}{16}} + \Theta(n^{\log_4 3}) = \frac{16}{13} n^2 + \Theta(n^{\log_4 3})$$

Proto $T(n) \in \Theta(n^2)$

Složitost úlohy

Složitost úlohy je složitost nejlepšího algoritmu řešícího danou úlohu.

Turingův stroj (Turing machine - TM)

Je teoretický model počítače, který se skládá z:

- z **řídící jednotky**, která se může nacházet v jednom z konečně mnoha stavů
- potenciálně **nekonečné pásky** (nekonečné na obě strany) rozdělené na jednotlivé pole
- **čtecí hlavy**, která umožňuje číst obsah polí a přepisovat obsah polí pásky

Je dán sedmicí $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$, kde

- Q je konečná množina stavů
- Σ je konečná množina vstupních symbolů
- Γ je konečná množina páskových symbolů, přitom $\Sigma \subset \Gamma$
- B - je prázdný symbol (*blank*), jedná se o páskový symbol, který není vstupním symbolem (tj. $B \in \Gamma \setminus \Sigma$)
- δ je přechodová funkce, tj. parciální zobrazení z množiny $(Q \setminus F) \times \Gamma$ do množiny $Q \times \Gamma \times L, R$, (zde L znamená pohyb hlavy o jedno pole doleva, R pohyb doprava)
- $q_0 \in Q$ je počáteční stav
- $F \subset Q$ je množina koncových stavů

Nedeterministický TM Je takový Turingův stroj, u kterého připustíme, aby v jedné situaci mohl provézt několik různých kroků.

Přijímaný a rozhodovaný jazyk TM. Vstupní slovo $w \in \Sigma^*$ je *přijato* Turingovým strojem M právě tehdy, když se Turingův stroj na slově w úspěšně zastaví. Množinu slov $w \in \Sigma^*$, které Turingův stroj přijímá, se nazývá *jazyk přijímaný* M a značíme ho $L(M)$.

Turingův stroj *rozhoduje* jazyk L, jestliže tento jazyk přijímá a navíc se na každém vstupu zastaví.

Třída složitosti - $\mathcal{P}$

Třída $\mathcal{P}$ Řekneme, že rozhodovací úloha $\mathcal{U}$ leží ve třídě $\mathcal{P}$, jestliže **existuje deterministický** Turingův stroj, který **rozhodne** jazyk $L_{\mathcal{U}}$ a pracuje v **polynomiálním čase**; tj. funkce $T(n)$ je $\mathcal{O}(p(n))$ pro nějaký polynom $p(n)$.

Příklady $\mathcal{P}$ úloh:

- **Minimální kostra v grafu**. Je dán neorinetovaný graf G s ohodnocením hran c. Je dáno číslo k. Existuje kostra grafu ceny menší nebo rovno k?
- **Nejkratší cesty v acyklickém grafu**. Je dán acyklický graf s ohodnocením hran a. Jsou dány vrcholy r a c. Je dáno číslo k. Existuje orientovaná cesta z vrcholu r do vrcholu c délky menší nebo rovno k?

- **Toky v sítích**. Je dána síť s horním omezením c, dolním omezením l, se zdrojem z a spotřebičem s. Dále je dáno číslo k. Existuje přípustný tok od z do s velikosti alespoň k?
- **Minimální řez**. Je dána síť s horním omezením c, dolním omezením l. Dále je dáno číslo k. Existuje řez, který má kapacitu menší nebo rovnu k?

Třída složitosti - $\mathcal{NP}$

Třída $\mathcal{NP}$ Řekneme, že rozhodovací úloha $\mathcal{U}$ leží ve třídě $\mathcal{NP}$, jestliže **existuje nedeterministický** Turingův stroj, který **rozhodne** jazyk $L_{\mathcal{U}}$ a pracuje v **polynomiálním čase**.

Příklady $\mathcal{NP}$ úloh:

- **Kliky v grafu**. Je dán neorinetovaný graf G a číslo k. Existuje klika v grafu G o alespoň k vrcholech?
- **Nejkratší cesty v obecném grafu**. Je dán orientovaný graf s ohodnocením hran a. Jsou dány vrcholy r a v. Je dáno číslo k. Existuje orientovaná cesta z vrcholu r do vrcholu c délky menší nebo rovno k?
- **k-barevnost**. Je dán neorientovaný graf G. Je graf G k-barevný?
- **Knapsack**. Je dáno n předmětů $1, 2, \ldots, n$. Každý předmět i má cenu c_i a váhu w_i. Dále jsou dána čísla A a B. Je možné vybrat předměty tak, aby celková váha nepřevýšila A a celková cena byla alespoň B?

Otázka obsahuje texty, úryvky a definice z [3].

6 NP-úplné a NP-těžké úlohy, Cookeova věta, heuristiky na řešení NP-těžkých úloh, pravděpodobnostní algoritmy.

Třída složitosti - NPC (NP-complete,NP-úplná)

7 Metoda větví a mezí. Algoritmy pro celočíselné lineární programování. Formulace optimalizačních a rozhodovacích problémů pomocí celočíselného lineárního programování. Toky a řezy. Multi-komoditní toky.

ILP - celočíselné lineární programování

Úloha celočíselného lineárního programování (LP) je zadána maticí $\mathbf{A} \in \mathbf{R}^{m \times n}$ a vektory $b \in \mathbf{R}^m, c \in \mathbf{R}^n$. Cílem je najít takový vektor $x \in \mathbf{Z}^n$, že platí $\mathbf{A} \cdot x \leq b$ a $c^T \cdot x$ je maximální.

Obvykle se celočíselné lineární programování zapisuje ve tvaru:

$$max(c^T \cdot x : \mathbf{A} \cdot x \leq b, x \in Z^n)$$

Pokud bychom takovou úlohu řešili pomocí lineárního programování s tím, že bychom výsledek zaokrouhlili, nejenom že bychom neměli zaručeno že výsledné řešení bude optimální ale ani to, zda bude přípustné. Zatímco úloha LP je řešitelná v polynomiálním čase, úloha ILP je tzv. NP-těžká (NP-hard), neboli není znám algoritmus, který by vyřešil libovolnou instanci této úlohy v polynomiálním čase. Protože prostor řešení ILP není konvexní množina, nelze přímo aplikovat metody konvexní optimalizace.

Formulace optimalizačních a rozhodovacích problémů pomocí ILP.

Algoritmy pro celočíselné lineární programování

1. Výčtové metody (Enumerative Methods)
2. Metoda větví a mezí (Branch and Bound)
3. Metody sečných nadrovin (Cutting Planes Methods)

Výčtové metody (Enumerative Methods)

Výpočet je založen na prohledávání oblasti zahrnující všechna přípustná řešení [1, 2]. Vzhledem k celočíselnému omezení proměnných je počet těchto řešení konečný ale jejich počet je extrémně vysoký. Proto je tato metoda vhodná pouze pro malé problémy s omezeným počtem diskrétních proměnných. Postup je možno zobecnit na úlohu MIP (mixed IP) tak, že ke každé kombinaci diskrétních proměnných je vyřešena úloha LP kde jsou diskrétní proměnné považovány za konstanty. [2]

Metoda větví a mezí (Branch and Bound)

text + obr s prikladem

Metody sečných nadrovin (Cutting Planes Methods)

Další skupinou algoritmů jsou metody sečných nadrovin (cutting plane method), založené podobně jako metoda větví a mezí na opakovaném řešení úlohy LP. Výpočet je prováděn iterativně tak, že v každém kroku je přidána další omezující podmínka zužující oblast přípustných řešení. Každá nová omezující podmínka musí splňovat tyto vlastnosti:

1. Optimální řešení nalezené pomocí LP se stane nepřípustným.
2. Žádné celočíselné řešení přípustné v předchozím kroku se nesmí stát nepřípustným.

Nové omezení splňující tyto vlastnosti je přidáno v každé iteraci. Vzniklý ILP program je vždy znovu řešen jako úloha LP. Proces je opakován, dokud není nalezeno přípustné celočíselné řešení. Konvergence takovéhoto algoritmu potom závisí na způsobu přidávání omezujících podmínek. Mezi nejznámější metody patří Dantzigovi řezy (Dantzig cuts) a Gomoryho řezy (Gomory cuts). [2]

Formulace optimalizačních a rozhodovacích problémů pomocí celočíselného lineárního programování

Toky a řezy

Multi-komoditní toky

8 Nejkratší cesty. Úloha obchodního cestujícího. Heuristiky a aproximační algoritmy. Metoda dynamického programování. Problém batohu. Pseudo-polynomiální algoritmy.

1. SPT - definovat, spousta problemu se na to da prevezt, trojuhelnikova nerovnost
2. TSP - definovat, slozitost, dukaz NP z redukce z Ham Cycle + dukaz neexistence aprox alg, heuristiky 2 aprox atd
3. Knapsack - definovat, dynamicke programovani, 2aprox, pseudo polym alg

9 Rozvrhování na jednom procesoru a na paralelních procesorech. Rozvrhování projektu s časovými omezeními. Programování s omezujícími podmínkami.

1. klasicke rozvhovani, multi, atd.
2. list scheduling
3. vsechny ty algoritmy

Reference

[1] Ing. Pavel Mička. Třídy složitosti a Turingovy stroje. Dostupné z: http://www.algoritmy.net/article/5774/Tridy-slozitosti.

[2] Ing. Přemysl Šůcha, Ph.D. *Celočíselné lineární programování* [online]. 2004. Dostupné z: http://support.dce.felk.cvut.cz/pub/hanzalek/_private/ref/sucha_ilp.pdf.

[3] Prof. RNDr. Marie Demlová, CSc. *Slidy k přednáškám TAL* [online]. 2014. Dostupné z: http://math.feld.cvut.cz/demlova/teaching/tal/predn_tal.html.