Kapitola 3

Státnice - Odhady složitosti

3.1 Dolní odhady složitosti problémů

Definice (Složitost problému)

Složitost problému je složitost asymptoticky nejlepšího možného algoritmu, který řeší daný problém (ne nejlepšího známého).

Každý konkrétní algoritmus dává horní odhad složitosti. Dolní odhady (až na triviální – velikost vstupu, výstupu) jsou složitější.

Věta (Dolní odhad složitosti mediánu)

Pro výběr k-tého z n prvků je třeba alespoň $n-1$ porovnání, tj. problém je $\Omega(n)$.

Důkaz

Intuitivně potřebuju medián, i kdyby mi spadnul z nebe, porovnat se všemi ostatními prvky, abych vůbec zjistil, jestli to je medián ...

Věta (Dolní odhad složitosti třídění)

Pro každý třídící algoritmus, založený na porovnávání prvků, existuje vstupní posloupnost, pro kterou provede $\Omega(n \log n)$ porovnání.

Důkaz

Nakreslím si rozhodovací strom jako model algoritmu – všechny vnitřní uzly odpovídají nějakému porovnání, které algoritmus provedl, jejich synové jsou operace, které nasledovaly po různých výsledcích toho porovnání (BÚNO jsou-li prvky různé, bude strom binární). Listy odpovídají setříděným posloupnostem. Aby byl algoritmus korektní, musí mít strom listy se všemi $n!$ možnými pořadími prvků.

Pro plýtvající algoritmus mohou existovat listy, neodpovídající žádné permutaci, tj. porovnává stejnou dvojici prvků dvakrát (a jedna z možností už nemůže nastat). Pro rovnoměrné rozdělení je očekávaný čas průměrná délka cesty od kořene k listům, nejhorší čas je výška stromu.

Označím výšku jako h, pak počet listů je $\leq 2^h$ a tedy $n! \leq 2^h$, tj. $h \geq \log n!$, pro dolní odhad $\Omega(n \log n)$ stačí odhad faktoriálu $n! < n^{\frac{n}{2}}$, případně můžu použít Stirlingův vzorec $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$ a dostávám $\Theta(n \log n)$.

3.2 Amortizovaná složitost

Definice (Amortizovaná složitost)

Typicky se používá pro počítání časové složitosti operací nad datovými strukturami, počítá průměrný čas na 1 operaci při provedení posloupnosti operací. Dává realističtější odhad složitosti posloupnosti všech operací, než měření všech nejhorším případem.

Známe 3 metody amortizované analýzy:

- agregační
- účetní
- potenciálová

Problémy:

- Inkrementace binárního čítače – do binárního čítače délky k postupně přičteme n-krát jedničku. Počet bitových operací na 1 přičtení je v nejhorším případě $O(\log n)$, ale amortizovaně dojdeme k $O(1)$.
- Vkládání do dynamického pole – začnu s prázdným polem délky 1 a postupně vkládám n prvků. Pokud je stávající pole plné, alokujeme dvojnásobné a kopírujeme prvky. Počet kopírování prvků na jedno vložení je až $O(n)$, ale amortizovaně opět $O(1)$.

Algoritmus (Agregační metoda)

Spočítáme nejhorší čas pro celou posloupnost n operací – $T(n)$, amortizovaný čas na 1 operaci je pak $\frac{T(n)}{n}$.

- **Binární sčítání**: v průběhu n přičtení se i-tý bit překlopí $\lfloor \frac{n}{2^i} \rfloor$-krát, takže celková cena překlopení je $\leq n \cdot \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n$, tj. amortizovaně na jedno přičtení $\frac{2n}{n} = \Theta(1)$
- **Vkládání**: cena i-tého vložení do pole je $c_i = \begin{cases} i \text{ pokud } \exists k : i - 1 = 2^k \\ 1 \text{ jinak} \end{cases}$. Celkem dostávám $T(n) = \sum_{i=1}^{n} c_i = n + \sum_{j=0}^{\lfloor \log n \rfloor} 2^j \leq n + 2n = 3n$, takže na jedno vložení vyjde $\frac{3n}{n} = \Theta(1)$.

Algoritmus (Účetní metoda)

Od každé operace vyberu urč. pevný "obnos", kterým onu operaci "zaplatím". Pokud něco zbyde, dám to na účet, pokud bude oprace naopak dražší než onen obnos, z účtu vybírám. Zůstatek na účtu musí být stále nezáporný – pokud uspěji, obnos je amortizovaná cena 1 operace.

- **Binární sčítání**: Při každém přičtení je právě jeden bit překlopen z 0 na 1. Proto každému bitu zavedeme účet a za přičtení budeme vybírat 2 jednotky. Jedna je použita na překlopení daného bitu z 0 na 1 a druhá uložena na právě jeho účet; překlápění z 1 na 0 jsou hrazeny z účtů (protože každý bit, který má nastavenu 1 má na účtu právě 1 jednotku, projde to). Amortizovaná cena tedy vyjde $2 = \Theta(1)$.
- **Vkládání**: Od každého vložení vyberu 3 jednotky – na vlastní vložení, na překopírování právě vloženého prvku při příštím vložení a na příští překopírování odpovídajícího prvku v levé polovině pole (na pozici $n - \frac{s}{2}$), který obnos ze svého vložení vyčerpal. Po expanzi je celkem na všech účtech 0, jindy víc, tj. amortizovaná cena operace je $3 = \Theta(1)$.

Algoritmus (Potenciálová metoda)

Je to podobné bankovní, roli účtu hraje nějaká funkce w, která popisuje vhodnost jednotlivých konfigurací $D_0, D_1, \ldots$. Potřebuji potom, aby $w(D_i) \geq 0 \ \forall i$. Amortizovaná složitost i -té operace o je potom:

$$am(o_i) = T(o_i) + w(D_{i+1}) - w(D_i)$$

Složitost nejhoršího případu celé posloupnosti operací může být mnohem "rychlejší" než posloupnost nejhorších případů jednotlivých operací:

$$\sum_{i=1}^{n} T(o_i) \leq \sum_{i=1}^{n} am(o_i) + w(D_0)$$