

1. Příklad na asymptotickou složitost rekurzivního algoritmu

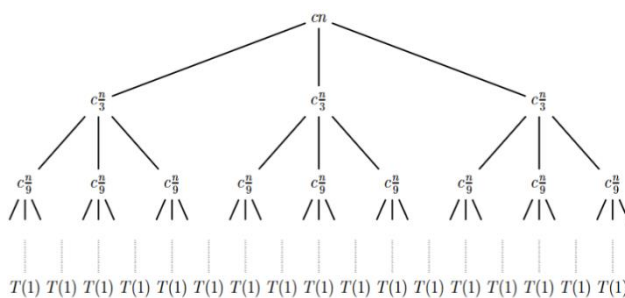
Analýzou stromů rekurze (recursion/tree method) odhadněte co nejtěsnější horní mez asymptotické složitosti (velké \mathcal{O}) pro:

$$T(n) = \begin{cases} 3T\left(\frac{n}{3}\right) + n & n > 1 \\ 1 & n = 1 \end{cases}$$

Rekurzivní strom nakreslete, vyznačte jeho hloubku a složitost v jednotlivých uzlech stromu. Vypočtete celkovou složitost stromu.

Uvažujte pouze velikosti instancí, které jsou mocninou 3 ($n = 3^k$).

Řešení:



Výška stromu - $\log_3 n$

Počet triviálních případů - $3^{\log_3 n} = n^{\log_3 3} = n^1 = n$

Složitost na jednotlivých úrovních - $c \frac{n}{3^i} \cdot 3^i$

Celková složitost – součet složitostí v uzlech na vrchních $\log_3 n - 1$ úrovních plus složitost triviálních případů:

$$\sum_{i=0}^{\log_3 n - 1} c \frac{n}{3^i} \cdot 3^i + n = cn \cdot \log_3 n + n = \mathcal{O}(n \log_3 n) = \mathcal{O}(n \log_2 n)$$

2. Příklad na čtení kódu

Pro funkci $f(a, i)$, jejíž vstupem je jednorozměrné pole a a index i , definujeme následující invariant:

Na začátku funkce neobsahuje podposloupnost $\{a_0, \dots, a_{i-1}\}$ prvek sedm.

Dokažte platnost daného invariantu při volání funkce $f(a, 0)$. Platí daný invariant i při volání $f(a, 1)$? Určete, co vrátí volání $f(a, 0)$.

```
int f(int[] a, int i) {
    if (i == a.length - 1) {
        if (a[i] == 7) return i;
        else return -1;
    }
    if (a[i] == 7) return i;
    else return f(a, i + 1);
}
```

Řešení:

Pro $i=0$ je podposloupnost prázdná a invariant platí.

Při běhu programu: předpokládáme, že podposloupnost $\{a_0, \dots, a_{i-2}\}$ neobsahuje 7.

Ukážeme, že pak také podposloupnost $\{a_0, \dots, a_{i-1}\}$ neobsahuje na začátku funkce sedmičku. Podle předpokladu by pouze $a_{i-1} = 7$. Prvek a_{i-1} je zpracován v předchozí iteraci a pokud by se rovnal 7, výpočet by skončil a funkce by vrátila hodnotu $i - 1$. Prvek a_{i-1} není 7, proto došlo k rekurzivnímu volání (poslední řádek programu). Podposloupnost $\{a_0, \dots, a_{i-1}\}$ proto neobsahuje na začátku funkce sedmičku.

Pro volání $f(a, 1)$ invariant neplatí, například pro pole $[7, 1, 2, 3]$ – kontrola přítomnosti sedmičky probíhá až od prvku s indexem 1.

Volání $f(a, 0)$ vrátí pozici nejlevější sedmičky. Pokud číslo sedm v poli není, vrátí -1.

3. Příklad na rekurzi

Vypočtete, kolik celkem času (v milisekundách) zabere jedno zavolání funkce `rekur(n)`; za předpokladu, že provedení příkazu `xyz()`; trvá vždy jednu milisekundu a že dobu trvání všech ostatních akcí zanedbáme.

```
void rekur(int x) {
    if (x < 1) return;
    rekur(x-1);
    xyz();
    rekur(x-1);
}
```

Úlohu řešte obecně pro vstupní hodnotu n . Následně vyčíslíte čas volání funkce `rekur(4)`;

Řešení:

Rekurentní vztah platný pro daný výpočet je $T(n) = 2T(n-1) + 1$. Výraz postupně upravíme podle principů matematické indukce $T(n) = 2T(n-1) + 1 = 4T(n-2) + 2 + 1 = 8T(n-3) + 4 + 2 + 1 = \dots = 2^{n-1}T(1) + 2^{n-2} + 2^{n-3} + \dots + 1 = 2^n - 1$.

Pro $n = 4$ dostáváme: $2^4 - 1 = 15$.

4. Příklad na asymptotickou složitost

Odhadněte asymptotickou složitost funkce $T(n)$:

$$T(n) = \begin{cases} 1 & n = 1 \\ 3T\left(\frac{n}{4}\right) + n \log_2 n & n > 1 \end{cases}$$

Použijte mistrovskou metodu (Master teorém). Pečlivě zkontrolujte všechny předpoklady věty a určete hodnoty jednotlivých konstant!

Řešení:

Abychom mohli použít mistrovskou metodu musíme porovnat $n^{\log_4 3} = n^{0,79}$ a $n \log_2 n$ ($a=3$, $b=4$). Zřejmě $f(n) = n \log_2 n = \Omega(n^{\log_4 3 + \varepsilon})$, kde $\varepsilon = 0,1 > 0$, $\log_4 3 < 1$. Dále musí platit podmínka regularity, tj. $3 \cdot f(n/4) \leq c \cdot f(n)$ pro dostatečně velká n .

$$3f\left(\frac{n}{4}\right) \leq cf(n), 3 \frac{n}{4} \log_2 \frac{n}{4} \leq cn \log_2 n, \frac{3}{4}n(\log_2 n - 2) \leq cn \log_2 n, \frac{3}{4} < c < 1$$

Platí tedy třetí možnost teorému a $T(n) = \theta(n \log_2 n)$.

5. Příklad na úpravu haldy

Je dáno pole $A = [100, 47, 39, 15, 13, 36, 25, 14, 9]$. Pole reprezentuje maximální haldou. Při řazení byl z haldy odstraněn kořenový prvek 100, který byl nahrazen číslem 8. Upravte pole tak, aby zůstalo maximální haldou.

Odhadněte shora počet výměn, které jsou nutné pro úpravu vlastností haldy.

Řešení: Postupné úpravy haldy:

$A = [8, 47, 39, 15, 13, 36, 25, 14, 9]$

$A = [47, 8, 39, 15, 13, 36, 25, 14, 9]$

$A = [47, 15, 39, 8, 13, 36, 25, 14, 9]$

$A = [47, 15, 39, 14, 13, 36, 25, 8, 9]$

Počet výměn je shora ohraničen hloubkou stromu, tj. $\log_2 n$.