1. Příklad na asymptotickou složitost rekurzivního algoritmu [4b]

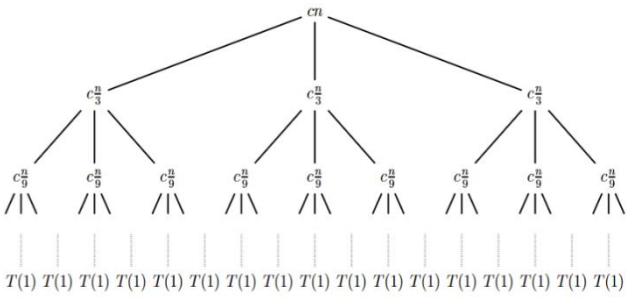
Analýzou stromů rekurze (recursion/tree method) odhadněte co nejtěsnější horní mez asymptotické složitosti (velké $\mathcal{O}$) pro:

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

Rekurzivní strom nakreslete, vyznačte jeho hloubku a složitost v jednotlivých uzlech stromu. Vypočtěte 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 [4b]

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

> *Na začátku funkce neobsahuje podposloupnost* $\{a_0, \ldots, a_{i-1}\}$ *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 pravděpodobnost [3b]

Jaká je střední hodnota návratové hodnoty randomizovaného algoritmu reprezentovaného metodou **`f`** pro dané **`n`** (celé kladné číslo) a **`t`** z intervalu [0,1]?
Funkce `nextDouble` vrací rovnoměrně rozdělená náhodná čísla z intervalu [0, 1).

```
public static double f(int n, double t) {
      Random r = new Random();
      double[] a = new double[n];
      for(int i = 0; i< n; i++) {
          if(r.nextDouble() >= t) {
              a[i] = 1.0;
          } else {
              a[i] = -2.0;
          }
      }
      double s = 0.0;
      for(int i=0; i<n; i++) s+= a[i];
      return s/n;
}
```

Řešení:

Pravděpodobnost, že a[i] bude 1 je 1-t, tj. v poli máme 1-t jedniček. Pravděpodobnost, že a[i] bude -2 je t, v poli máme t hodnot -2. Jejich součet je $1 * (1 - t) + t(-2) = 1 - 3t$.

4. Příklad na asymptotickou složitost [3b]

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 ε = 0,1 > 0, $\log_4 3 < 1$. Dále musí platit podmínka regularity, tj. 3·f(n/4) ≤ c·f(n) pro dostatečně veliká 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. Řazení quicksort [3b]

Je dáno pole celý čísel - { 8, 5, 7, 9, 11, 6, 17, 18 }, které chceme setřídit vzestupně. Uvažujeme algoritmus quicksort z přednášky – klasickou verzi Hoare's partition.

Určete, který z prvků bude prvním pivotem a proveďte první dělení (partition).

Řešení:

Prvním pivotem je číslo 8, po prvním dělení bude posloupnost vypadat následovně { 6, 5, 7, 8, 11, 9, 17, 18 }.

6. Příklad na úpravu haldy [3b]

Je dáno pole *A* = [100, 47, 39, 15, 13, 36, 25, 14, 9]. Pole reprezentuje maximální haldu. 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. $\log_2 n$.