REKURZE, MISTROVSKÁ VĚTA

Karel Horák, Petr Ryšavý

2. března 2016

Katedra počítačů, FEL, ČVUT

Příklad 1

Uvažme následující algoritmus.

```
// array contains N>0 random integers
public int[] dummySort(int[] array) {
    final int[] array2 = new int[array.length];
    for(int i = 0; i < array2.length; i++) {
        final int minIndex = findMinimum(array);
        array2[i] = array[minIndex];
        array[minIndex] = Integer.MAX_VALUE;
    }
    return array2;
}

public int findMinimum(int[] array) {
    int minIndex = 0;
    for(int i = 1; i < array.length; i++) {
        if(array[i] < array[minIndex]) minIndex = i;
    }
    return minIndex;
}
```

Určete asymptotickou složitost algoritmu.

Rekurze

Situace kdy algoritmus volá sám sebe za účelem vyřešení problému.

Problém se rozloží na jednodušší podproblémy, které lze řešit stejným algoritmem.

Problém rozkládáme dokud nezískáme dostatečně malý podproblém, který umíme snadno vyřešit.

Mnoho rekurzivních algoritmů je postaveno na myšlence „rozděl a panuj“ (anglicky *divide and conquer*).

dále viz. Rekurze

Příklad 2

Uvažme následující algoritmus. Ten počítá čísla Fibonacciho posloupnosti

$$F(0) = 0, \qquad F(1) = 1, \qquad F(n) = F(n-1) + F(n-2).$$

```
public int fibonacci(int n) {
    if(n == 0) { return 0; }
    if(n == 1) { return 1; }
    return fibonacci(n-1) + fibonacci(n-2);
}
```

Nakreslete strom volání pro $F(4)$ a $F(5)$. Určete složitost výpočtu v jednotlivých uzlech. Kolikrát je funkce `fibonacci` volána?

Příklad 3

Pomocí rekurzivní funkce vypište pro zadané kladné číslo N posloupnost čísel

$$1 \quad 2 \quad \cdots N-1 \quad N \quad N \quad N-1 \quad \cdots \quad 2 \quad 1.$$

Pokud se budete nudit, zkuste napsat rekurzivní funkci, která spočte kombinační číslo $\binom{n}{k}$ za využití identity

$$\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}.$$

Příklad 4

Uvažme následující algoritmus pro hledání prvku q v seřazeném poli délky n.

1. Pokud je pole prázdné, vrať „prvek nenalzen“.
2. Porovnej q s prostředním prvkem m.
   - 2.1 Je-li $q = m$, pak vrať index prostředního prvku.
   - 2.2 Je-li $q < m$, hledej v levé polovině pole.
   - 2.3 Je-li $q > m$, hledej v pravé polovině pole.

Nakreslete strom rekurzivního volání pro pole $[1, 3, 8, 15, 42, 95, 107]$ a dotaz $q = 42$. Určete asymptotickou složitost algoritmu.

Příklad 5

Uvažme následující algoritmus pro řazení pole délky n.

1. Seřaď levou polovinu pole.
2. Seřaď pravou polovinu pole.
3. Zkombinuj výsledky do seřazeného pole.

Jaká je celková asymptotická složitost, víte-li, že třetí krok lze provést v $c \cdot n$ operacích (pro nějakou kladnou konstantu c). Nakreslete strom rekurzivních volání.

Master theorem

Nechť $a \geq 1$ a $b > 1$ jsou konstanty, nechť $f(n)$ je funkce a nechť $T(n)$ je definováno pro nezáporná celá čísla rekurencí

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n).$$

Pokud $f(n) \in \mathcal{O}(n^{\log_b a - \varepsilon})$ pro nějakou konstantu $\varepsilon > 0$, potom

$$T(n) \in \Theta(n^{\log_b a}).$$

Pokud $f(n) \in \Theta(n^{\log_b a})$, potom

$$T(n) \in \Theta(n^{\log_b a} \log n).$$

Pokud $f(n) \in \Omega(n^{\log_b a + \varepsilon})$ pro nějakou konstantu $\varepsilon > 0$ a pokud $af(\frac{n}{b}) \leq cf(n)$ pro nějakou konstantu $c < 1$ a všechna dostatečně velká n, potom

$$T(n) \in \Theta(f(n)).$$

Příklad 6

Uvažme následující algoritmus pro řazení pole délky n.

1. Seřaď levou polovinu pole.
2. Seřaď pravou polovinu pole.
3. Zkombinuj výsledky do seřazeného pole.

Za užití Mistrovské věty určete jeho asymptotickou složitost, víte-li, že třetí krok lze provést v $c \cdot n$ operacích (pro nějakou kladnou konstantu c).

Příklad 7

Nechť $\mathbf{X}$ a $\mathbf{Y}$ jsou matice z $\mathbb{R}^{n\times n}$, kde n je přirozené číslo. Uvažme násobení matic $\mathbf{Z} = \mathbf{XY}$. Zde platí

$$z_{ij} = \sum_{k=1}^{n} x_{ik} y_{kj}. \tag{1}$$

Jak dlouho trvá spočíst součin dvou matic pomocí této definice.

Příklad 8

Nyní uvažme násobení matic pomocí rekurze. Pro jednoduchost předpokládejme, že n je mocninou dvou. Pak můžeme matice násobit po blocích. Představíme, si že matice $\mathbf{X}$ a $\mathbf{Y}$ rozdělíme na 4 stejně velké podmatice

$$\mathbf{X} = \left(\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{D} \end{array}\right) \quad \mathbf{Y} = \left(\begin{array}{c|c} \mathbf{E} & \mathbf{F} \\ \hline \mathbf{G} & \mathbf{H} \end{array}\right).$$

Pak můžeme využít vztahu

$$\mathbf{X} \cdot \mathbf{Y} = \left(\begin{array}{c|c} \mathbf{AE} + \mathbf{BG} & \mathbf{AF} + \mathbf{BH} \\ \hline \mathbf{CE} + \mathbf{DG} & \mathbf{CF} + \mathbf{DH} \end{array}\right)$$

pro výpočet součinu $\mathbf{XY}$. Jaká je složitost takovéhoto rekurzivního algoritmu.

Příklad 9

V předchozím případě jsme potřebovali 8 rekurzivních volání funkce násobení matic. Německý matematik Volker Strassen[1] si v roce 1969 všiml, že pokud vhodně násobí součty různých podmatic, pak je schopný ušetřit jedno rekurzivní volání. Označíme-li

$$\mathbf{P}_1 = \mathbf{A}(\mathbf{F} - \mathbf{H}) \qquad \mathbf{P}_2 = (\mathbf{A} + \mathbf{B})\mathbf{H} \qquad \mathbf{P}_3 = (\mathbf{C} + \mathbf{D})\mathbf{E}$$
$$\mathbf{P}_4 = \mathbf{D}(\mathbf{G} - \mathbf{E}) \qquad \mathbf{P}_5 = (\mathbf{A} + \mathbf{D})(\mathbf{E} + \mathbf{H}) \qquad \mathbf{P}_6 = (\mathbf{B} - \mathbf{D})(\mathbf{G} + \mathbf{H})$$
$$\mathbf{P}_7 = (\mathbf{A} - \mathbf{C})(\mathbf{E} + \mathbf{F}),$$

pak platí

$$\mathbf{X} \cdot \mathbf{Y} = \begin{pmatrix} \mathbf{AE} + \mathbf{BG} & \mathbf{AF} + \mathbf{BH} \\ \mathbf{CE} + \mathbf{DG} & \mathbf{CF} + \mathbf{DH} \end{pmatrix} = \begin{pmatrix} \mathbf{P}_5 + \mathbf{P}_4 - \mathbf{P}_2 + \mathbf{P}_6 & \mathbf{P}_1 + \mathbf{P}_2 \\ \mathbf{P}_3 + \mathbf{P}_4 & \mathbf{P}_1 + \mathbf{P}_5 - \mathbf{P}_3 - \mathbf{P}_7 \end{pmatrix}.$$

Jaká je asymptotická složitost násobení matic pomocí Strassenova algoritmu?

[1] Strassen, Volker, *Gaussian Elimination is not Optimal*, Numer. Math. 13, p. 354-356, 1969

Příklad 10

Uvažme následující algoritmus pro řazení pole délky n.

1. Seřaď levou polovinu pole.
2. Seřaď pravou polovinu pole.
3. Zkombinuj výsledky do seřazeného pole.

Lojza Patlal nedával pozor na přednášce z algoritmizace a třetí krok naimplementoval se složitostí $c \cdot n^2$ operací (pro nějakou kladnou konstantu c). Za užití Mistrovské věty určete asymptotickou složitost algoritmu, který vytvořil.