Karel Horák, Petr Ryšavý

30. března 2016

Katedra počítačů, FEL, ČVUT

Asymptotická složitost

Příklad 1

Pole a obsahuje N prvků. Určete asymptotickou složitost daného kódu v závislosti na N.

```
for( i = 0; i < N; i++ )
    for( j = i; j < N; j++ )
        for( k = j; k < N; k++ )
            print( a[i]*a[j] - a[k] );
```

Příklad 2

Pole a obsahuje N prvků. Určete asymptotickou složitost daného kódu v závislosti na N.

```
for( i = 0; i < N; i++ )
    a[i] = 1;
for( i = 1; i < N; i++ )
    while( a[i] <= 2*a[i-1] ) {
        print( a[i] );
        a[i]++;
    }
```

Příklad 3

Pole a obsahuje N prvků. Určete asymptotickou složitost daného kódu v závislosti na N.

```
for( i = 0; i < N; i++ )
    a[i] = N;
for( i = 0; i < N; i++ )
    while( a[i] > 0 ) {
        print( a[i] );
        a[i] = a[i]/2;
    }
```

Příklad 4

Pole a obsahuje N prvků. Určete asymptotickou složitost daného kódu v závislosti na N.

```
for( i = 0; i < N; i++ )
    a[i] = i;
for( i = 0; i < N; i++ )
    while( a[i] > 0 ) {
        print( a[i] );
        a[i] = a[i]/2;
    }
```

Příklad 5

Spočtěte jednoduché Θ odhady pro funkci $f(n)$, kde $f(n)$ je následující:

- 10^{80} (počet atomů ve vesmíru)
- $\log_{\ln 5}\left(\log^{\log 100} n\right)$
- $(20n)^7$
- $5^{\log 3} n^3 + 10^{80} n^2 + \log(3) n^{3.1} + 6006$
- $\log \binom{n}{\frac{n}{2}}$ (nápověda: použijte Stirlingův aproximační vzorec $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$).

[http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/recitation-videos/MIT6_006F11_rec01.pdf]

Příklad 6

Nechť $f(n)$ a $g(n)$ jsou asymptoticky kladné funkce. Dokažte nebo vyvraťte následující tvrzení:

$f(n) \in \mathcal{O}(g(n))$ implikuje, že $2^{f(n)} \in \mathcal{O}(2^{g(n)})$

$f(n) \in \mathcal{O}((f(n))^2)$.

[https://fmse.info.uaic.ro/getteachingfile/53/Probleme_Complexitati_PA_sol.pdf/, strana 6]

Příklad 7

Pole `A` celých čísel je uspořádáno v neklesajícím pořadí. Kromě toho je dáno číslo `b`. Máme najít minimální hodnotu výrazů

```
A[i]+A[j] - b,
abs( A[i]*A[j] - b ),
```

kde `i`, `j` probíhají přes všechny indexy pole `A`. Číslo `b` a všechny hodnoty v poli mohou být záporné i nezáporné. Navrhněte efektivní algoritmus a určete jeho asymptotickou složitost.

Rekurze

Příklad 8

Určete výstup následujícího programu.

```
public class Recursive {
    private int counter = 0;

    public int doStuff(int x, int y)     {
        counter++;
        if (x < 1 || y < 1)
            return 1;
        return x + y + doStuff(x - 1, y - 2);
    }

    public static void main(String[] args) {
        Recursive x = new Recursive();
        System.out.println("Answer = " + x.doStuff(3, 3) +
            " Counter = " + x.counter);
        System.out.println("Answer = " + x.doStuff(6, 6) +
            " Counter = " + x.counter);
    }
}
```

[http://www.cis.gvsu.edu/~kurmasz/Teaching/OldCourses/CS163/CS163_W08/WebPage/Homework/Recursion.pdf]

Příklad 9

Sestrojte rekurzivní algoritmus, který spočte

$$a^b,$$

kde a a b jsou přirozená čísla. Dodržte následující signaturu metody

```
public interface Power {
        public BigInteger power(BigInteger a,
                                BigInteger b);
}
```

Pokud Váš algoritmus potřebuje lineární počet násobení vzhledem k b, sestrojte algoritmus, který potřebuje pouze $\mathcal{O}(\log b)$ násobení.

Příklad 10

Napište rekurzivní algoritmus, který porovná dva řetězce: `int compareTo(String s1, String s2)`. Pamatujte, že `compareTo` vrací:

- záporné číslo, pokud `s1 < s2` v lexikografickém uspořádání,
- 0, pokud `s1 == s2`,
- kladné číslo, pokud `s1 > s2` v lexikografickém uspořádání.

Modifikujte předchozí kód tak, aby opakovaně nekopíroval části řetězců. [`http://www.cis.gvsu.edu/~kurmasz/Teaching/OldCourses/CS163/CS163_W08/WebPage/Homework/Recursion.pdf`]

Příklad 11

Pochopte jak funguje rekurzivní algoritmus pro řešení problému Hanojských věží. Můžete použít `https://cw.fel.cvut.cz/wiki/_media/courses/a4b33alg/alg02b.pdf`.

Složitost rekurzivních algoritmů a mistrovská věta

Příklad 12

Složitost rekurzivního algoritmu je dána vztahem

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{6}\right) + T\left(\frac{n}{3}\right) + n.$$

Určete jeho asymptotickou složitost.

[`https://courses.engr.illinois.edu/cs173/sp2011/Homework/hw9-solutions.pdf`]

Příklad 13

Složitost rekurzivního algoritmu je dána vztahem

$$A(n) = 2A\left(\frac{n}{4}\right) + \sqrt{n}.$$

Určete jeho asymptotickou složitost.

Určete $A(1024)$ a $A(4096)$.

[`https://courses.engr.illinois.edu/cs173/sp2011/Homework/hw9-solutions.pdf`]

Příklad 14

Složitost rekurzivního algoritmu je dána rekurencí:

$$T(n) = 3T(\lfloor \tfrac{n}{2} \rfloor) + n$$
$$T(n) = T(\tfrac{n}{2}) + n^2$$
$$T(n) = 4T(\tfrac{n}{2} + 2) + n$$

Použijte metodu stromu rekurze k nalezení vhodného horního odhadu funkce $T(n)$. Ověřte výsledek substituční metodou a použitím Mistrovské věty.

Příklad 15

Ackermanova funkce $A(n, m)$ je definována níže. Doplňte do tabulky hodnoty $A(n, m)$.

$$A(n,m) = \begin{cases} m+1 & \text{pro } n = 0, \\ A(n-1,1) & \text{pro } n > 0, m = 0, \\ A(n-1, A(n, m-1))) & \text{pro } n > 0, m > 0. \end{cases}$$

$A(n,m)$	$m=0$	$m=1$	$m=2$	$m=3$	$m=4$
$n=0$					
$n=1$					
$n=2$					
$n=3$					
$n=4$					

Stromy, BVS a prohledávání stavového prostoru

Příklad 16

Napište rekurzivní verzi operace `TreeMinimum`, která vrátí referenci na uzel s nejmenší hodnotou v BVS.

Můžete využít kód, který jste napsali na minulých cvičeních.

Příklad 17

Napište funkci, jejímž vstupem bude ukazatel (reference) na uzel X v BVS a výstupem ukazatel (reference) na uzel s nejbližší vyšší hodnotou ve stromu.

Příklad 18

Navrhněte rekurzívní funkci, která určí, zda má dvojice binárních stromů stejnou strukturu. Dva binární stromy považujeme za strukturně stejné, pokud se dají nakreslit tak, že po položení na sebe pozorovateli splývají, bez ohledu na to, jaká obsahují data. Kořeny musí být položeny na sobě.

Příklad 19

Máme projít pravidelným binárním stromem a navštívit všech jeho N uzlů. Jediné dvě možnosti pohybu v každém uzlu jsou buď posun do některého bezprostředního potomka nebo skok zpět do kořene stromu. Každý posun nebo skok trvá jednu mikrosekundu. Určete, za jak dlouho lze úkol splnit, pokud

strom má minimální možnou hloubku,

strom má maximální možnou hloubku.

Příklad 20

Přelévání vody mezi nádobami.

Máme 3 nádoby o různých objemech. Žádná nádoba na sobě nemá stupnici a nádoby mají dokonce tak roztodivné tvary, že nám znemožňují odhadování množství vody v nich. Stále ale můžeme naměřit i jiné množství tekutiny. Můžeme přelévat obsah jedné nádoby do druhé tak dlouho, dokud nepřelijeme všechno nebo dokud se druhá nádoba nezaplní. Jaký je nejmenší počet přelití, abychom vyřešili následující úlohu? Ve všech variantách je na počátku největší nádoba plná.

Máme 3 nádoby o objemech $8\,\mathrm{l}$, $5\,\mathrm{l}$ a $3\,\mathrm{l}$. Přeléváním se chceme dostat do stavu, kdy je počáteční množství rozděleno na dva stejné díly.