Cvičení 12

Příklad 1: Pro stroje RAM zkonstruované v příkladě 2 ze cvičení 10 určete co nejpřesněji jejich časovou a paměťovou složitost:

- Nejprve vždy specifikujte, co přesně považujete v daném případě za velikost vstupu.
- Poté určete časovou a paměťovou složitost při použití jednotkové míry. Určete přesné počty provedených instrukcí a použitých paměťových buněk, ne jen asymptotické odhady.
- Alespoň přibližně odhadněte složitost při použití logaritmické míry. Bude se nějak lišit vaše definice velikosti vstupu při použití jednotkové a logaritmické míry?

Řešení:
Zadání (e):

```
loop1:  READ
        JZERO  loop2
        STORE  *2
        LOAD   1
        ADD    =1
        STORE  1
        JUMP   loop1
loop2:  LOAD   1
        JZERO  end
        SUB    =1
        STORE  1
        LOAD   *2
        WRITE
        JUMP   loop2
  end:  HALT
```

Pro vstup $a_1, a_2, \ldots, a_n, 0$ se provede celkem $7n + 2 + 7n + 2 + 1 = 14n + 5$ instrukcí. Použije se n buněk na adresách $2, 3, \ldots, n + 1$ pro uložení čísel $a_1, a_2, \ldots, a_n$ a dále pak buňky 0 a 1. Dohromady se tedy během výpočtu použije $n + 2$ buněk.

Příklad 2: Určete co nejpřesněji časové složitosti následujících fragmentů kódu. Výsledky vyjádřete v asymptotické notaci pomocí Θ.

Poznámka: Jako velikost vstupu uvažujte hodnotu n. Můžete předpokládat, že hodnoty všech proměnných jsou již načteny v paměti.

a)
for $i := 1$ **to** $n * n$ **do**
    for $j := 1$ **to** i **do**
        $A[i, j] := A[i, j] + b[j]$
    end for

end for

Řešení: Vnější cyklus se provede n^2-krát. Počet iterací vnitřního cyklu však závisí na proměnné i vnějšího cyklu. Celkový počet průchodů vnitřním cyklem je

$$\sum_{i=1}^{n^2} i = 1 + 2 + 3 + \cdots + (n^2 - 1) + n^2 = (1 + n^2)\frac{n^2}{2} = \frac{n^4}{2} + \frac{n^2}{2} = \Theta(n^4)$$

Časová složitost tohoto fragmentu kódu je tedy $\Theta(n^4)$.

b)
$x := 0$
for $i := 1$ **to** n **do**
  $j := i * i$
  while $j > 0$ **do**
    if $d[j] < R[i, j]$ **then**
      $R[i, j] := x - 1$
      $x := d[j]$
    end if
    $j := j - 1$
  end while
end for

Řešení:

$$\sum_{i=1}^{n} i^2 = \Theta(n^3)$$

Podrobnější zdůvodnění:

$$\sum_{i=1}^{n} i^2 \leq \sum_{i=1}^{n} n^2 = n \cdot n^2 = n^3$$

$$\sum_{i=1}^{n} i^2 \geq \sum_{i=\lceil \frac{n}{2} \rceil}^{n} i^2 \geq \sum_{i=\lceil \frac{n}{2} \rceil}^{n} \left(\frac{n}{2}\right)^2 \geq \frac{n}{2} \cdot \left(\frac{n}{2}\right)^2 = \frac{1}{8}n^3$$

c)
$i := 1$
while $i < n$ **do**
  $q[i] := q[i] + i$
  $i := i + i$
end while

Řešení: $\Theta(\log n)$

d)
$i := 1;\ \ j := 1$
while $i < n$ **do**
  $E[i, j] := E[i, j] \bmod s[i]$
  $i := i + j$
  $j := j + 1$

end while

Řešení: Potřebujeme zjistit pročet průchodu cyklem. Nutně to bude takové největší číslo k, pro které platí $1 + 2 + \cdots + k \leq n$. Protože

$$1 + 2 + \cdots + k = (1 + k) \cdot \frac{k}{2} \approx \frac{1}{2}k^2$$

Z rovnosti $\frac{1}{2}k^2 \approx n$ dostáváme, že hodnota k je přibližně rovna $\sqrt{2n}$, tj. že počet průchodů cyklem (a celková složitost) je $\Theta(\sqrt{n})$.

Poznámka: Alternativně můžeme najít největší k takové, že $(1 + k) \cdot \frac{k}{2} \geq n$, jako jedno z řešení kvadratické rovnice

$$\frac{1}{2}k^2 + \frac{1}{2}k - n = 0$$

Řešení této kvadratické rovnice vypadá takto

$$k = \frac{-\frac{1}{2} \pm \sqrt{\left(\frac{1}{2}\right)^2 + 4 \cdot \frac{1}{2} \cdot n}}{2 \cdot \frac{1}{2}} = -\frac{1}{2} \pm \sqrt{\frac{1}{4} + 2n}$$

Vzhledem k tomu, že hledáme maximální hodnotu k, která je nezáporná a která je celým číslem, řešením bude hodnota $-\frac{1}{2} + \sqrt{\frac{1}{4} + 2n}$ zaokrouhlená na nejbližší menší celé číslo, tj. přibližně $\sqrt{2n}$.

e)
```
s := 1
while s ≤ n do
    i := 0
    while i < n do
        a[i] := a[b[i]] * s
        i := i + s
    end while
    s := s * 2
end while
```

Řešení: Při průchodech vnějším cyklem nabývá s postupně hodnot $1, 2, 4, 8 \ldots$ Počet průchodů vnějším cyklem je tedy zhruba $\lg n$ (pozn.: $\lg n = \log_2 n$). Při každém průchodu vnějším cyklem se vnitřní cyklus provede zhruba n/s krát. Celkový počet průchodů vnitřním cyklem je tedy zhruba

$$\frac{n}{1} + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \cdots + \frac{n}{n} \approx 1 + 2 + 4 + 8 + \cdots + 2^{\lg n} = \sum_{j=1}^{\lg n} 2^j =$$

$$= \frac{2^{\lg n + 1} - 1}{2 - 1} = 2 \cdot 2^{\lg n} - 1 = 2n - 1 = \Theta(n)$$

To nám dává horní odhad časové složitosti $O(n)$. Na druhou stranu, při prvním průchodu vnějším cyklem se vnitřní cyklus provede n krát, což nám dává spodní odhad. Časová složitost tohoto fragmentu kódu je tedy $\Theta(n)$.

f)
```
s := 1
while s ≤ n do
    i := 0
    while i < n do
        a[i] := a[i] + s
        i := i + s
    end while
    s := s + 1
end while
```

Řešení: Proměnná s nabývá při průchodech vnějším cyklem postupně hodnot $1, 2, \ldots, n$, přičemž podobně jako v předchozím případě se při každém průchodu vnějším cyklem vnitřní cyklus provede zhruba n/s krát. Celková počet průchodů vnitřním cyklem (a celková složitost fragmentu kódu) je tedy

$$\sum_{k=1}^{n} \frac{n}{k} = n \cdot \sum_{k=1}^{n} \frac{1}{k} = n \cdot \Theta(\log n) = \Theta(n \log n)$$

Poznámka: Využije se vztah pro součet harmonické řady

$$\ln(n+1) \leq \sum_{k=1}^{n} \frac{1}{k} \leq \ln n + 1\,.$$

Příklad 3: Popište pseudokódem libovolný algoritmus pro řešení následujícího problému a co nejpřesněji odhadněte jeho časovou a paměťovou složitost:

VSTUP: Sekvence přirozených čísel $a_1, a_2, \ldots, a_n$.

VÝSTUP: Čísla $a_1, a_2, \ldots, a_n$ setříděná od nejmenšího po největší.

Poznámka: V algoritmu se nemusíte zabývat načítáním vstupu a výpisem výstupu.

Příklad 4: Popište pseudokódem libovolný algoritmus pro řešení následujícího problému a co nejpřesněji odhadněte jeho časovou a paměťovou složitost:

VSTUP: Matice A, B, jejichž prvky jsou celá čísla.

VÝSTUP: Matice $A \cdot B$.

Poznámka: V algoritmu se nemusíte zabývat načítáním vstupu a výpisem výstupu.

Příklad 5: Jednoduchá implementace Euklidova algoritmu pro výpočet největšího společného dělitele dvou přirozených čísel a, b počítá výsledek následovně:

```
EUCLID(a, b):
    if b = 0 then
```

```
        return a
    else if a > b then
        return EUCLID(b, a − b)
    else
        return EUCLID(a, b − a)
    end if
```

Jaká je časová složitost tohoto algoritmu?

Poznámka: Jako velikost vstupu uvažujte celkový počet bitů čísel a, b. Předpokládejte, že aritmetické operace trvají jednotkový čas.

Řešení: S každým rekurzivním voláním se jedno z čísel a, b zmenší aspoň o 1. Nemůže tedy nastat více než $2 \cdot 2^k = 2^{k+1}$ rekurzivních volání. Každé volání trvá konstantní čas. Na druhou stranu si lze snadno představit zadání, při kterém bude 2^k iterací skutečně nutných: $a = 1$, $b = 2^k$. Celkem tedy je složitost algoritmu $\Theta(2^k)$.

Příklad 6: Jaká je časová složitost následující efektivnější implementace Euklidova algoritmu?

```
EUCLID(a, b):
    while b ≠ 0 do
        c := a mod b
        a := b
        b := c
    end while
    return a
```

Poznámka: Stejně jako v předchozím příkladě uvažujte jako velikost vstupu celkový počet bitů čísel a, b a předpokládejte, že aritmetické operace trvají jednotkový čas.