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?

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
```

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
```

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

d)
```
i := 1;  j := 1
while i < n do
```

```
        E[i, j] := E[i, j] mod s[i]
        i := i + j
        j := j + 1
    end while
```

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
```

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
```

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.

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.