

Algoritmi i strukture podataka 2019./2020.

Složenost - zadatci za vježbu

1. Kolika je vremenska složenost sljedećeg odsječka:

```
int main() {  
    cout << "Dobar dan!";  
    return 0;  
}
```

$O(1) = \Omega(1) = \Theta(1)$

2. Kolika je vremenska složenost sljedećih odsječaka:

a) <pre>int main() { for (int i = 0; i < 8; i++) cout << "Dobar dan!"; return 0; }</pre> <p>$\Theta(8) = \Theta(1)$</p>	b) <pre>int main() { for (int i = 0; i < n; i++) cout << "Dobar dan!"; return 0; }</pre> <p>$\Theta(n)$</p>
--	--

3. Koje su od navedenih tvrdnji istinite:

- a) $T(n) = n^3 + n + \log(n) \in \Theta(n^3)$ ✓
- b) $T(n) = n^3 + n \notin O(n^3)$
- c) $T(n) = n^3 + n^2 + n \in \Omega(n)$
- d) $T(n) = n^3 + n \in o(n^3)$
- e) $T(n) = n^3 + n \in \omega(n^3)$
- f) $T(n) = 2^e \in \Theta(1)$
- g) $T(n) = 2^n \in O(2^n + n^4)$
- h) $T(n) = (n + 1)^3 \in \Theta(5n^3 + n)$

4. Kolika je vremenska složenost sljedećeg odsječka u ovisnosti o n :

```
int brojac = 0;  
for (int i = 0; i < n; i++)  
    for (int j = 0; j < i; j++)  
        ++ brojac;
```

5. Kolika je vremenska složenost sljedećeg odsječka u ovisnosti o k :

```
int brojac = 0;  
for (int i = k; i > 0; i /= 2)  
    for (int j = 0; j < i; j++)  
        ++ brojac;
```

6. Zadano je polje A od n članova tipa `int`. Odredite vremensku složenost sljedećih postupaka:

- a) ispis svih n članova polja
- b) zbroj svih n članova polja
- c) dohvaćanje člana $A[1]$

d) dohvat najvećeg člana u polju

7. Odredite vrijeme izvođenja u O , Ω i, ako je moguće, Θ notaciji za funkciju f1:

```
void f1(int n) {  
    int i;  
    for (i = n; i > 0; i /= 2) {  
        printf("%d\n", i % 2);  
    }  
}
```

8. Zadano je polje A od n elemenata tipa int. Odredite vrijeme izvođenja u O , Ω i, ako je moguće, Θ notaciji za sljedeći programski odsječak (**MI 2015./2016.**):

```
if (n <= 10) {  
    g(n); // obavlja se u  $\Theta(n)$  vremenu  
}  
else {  
    for (i = n - 1; i >= 0; i--) {  
        if (A[i] > A[i - 1]) {  
            g(i);  
        }  
    }  
}
```

Naputak: Članovi polja mogu, ali i ne moraju, biti sortirani (uzlazno ili silazno).

9. Odredite vrijeme izvođenja sljedećih programskih odsječaka pomoću O notacije i varijable n (**LJR 2018./2019.**).

a)

```
int fun(int k) {  
    int cost;  
    for (int i = 0; i < k; ++i) cost = cost + (i * k);  
    return cost;  
}  
  
...  
// poziv funkcije fun  
answ = fun(n);
```

b)

```
int sum;  
for (int i = 0; i < n; ++i) {  
    if (n < 1000)  
        sum++;  
    else  
        sum += fun(n);  
}
```

c)

```
for (int i = 0; i < n + 100; ++i) {  
    for (int j = 0; j < i * n; ++j) {  
        sum = sum + j;  
    }  
    for (int k = 0; k < n + n + n; ++k) {  
        c[k] = c[k] + sum;  
    }  
}
```

10. Zadano je polje B cijelih brojeva velikih između 1 i n . Odredite vrijeme izvođenja u O , Ω i Θ notaciji za funkcije f_1 i f_2 (MI 2018./2019.).

a) // $O(fPom(n)) = \Omega(fPom(n)) = \Theta(fPom(n)) = 1$

```
int f1(int n) {
    int s = 0;
    for (int i = n; i >= 1; i /= 2) {
        for (int j = 1; j <= i; j++) {
            s = s + fPom(i + j);
        }
    }
    return s;
}
```

b) // $O(gPom(n)) = \Omega(gPom(n)) = \Theta(gPom(n)) = n$

```
int f2(int n, int *B) {
    int i, j, s;
    s = 0;
    for (i = n / 2; i <= n; i++)
        if (B[i] > n / 2)
            for (j = 1; j < n; j += n/2) {
                s += B[i] * gPom(n - j);
            }
        else {
            s += B[i] * gPom(n - i) * gPom(i);
        }
    return s;
}
```

11. Odredite, gdje je moguće, vrijeme izvođenja u O , Ω i Θ notaciji u ovisnosti o m . Ako se vrijeme izvođenja ne može odrediti, navedite tako u rješenju. Pretpostaviti da je $m > 2$ (IR 2017./2018.). Polje c je polje cijelih brojeva čiji elementi mogu imati vrijednost 0 ili 1.

```
s=0;
for (i=0; i<m; i++){
    if (c[i]==0){
        for (j=0; j<m; j++)
            s += A[i*m+j]*B[j];
    }
    else {
        for (j=0; j<m; j++)
            for (k=j+1; k<m; k++)
                s += A[k*m+i]*A[j*m+i];
    }
}
```

12. Odredite, gdje je moguće, vrijeme izvođenja u O , Ω i Θ notaciji. Ako se vrijeme izvođenja ne može odrediti, navedite tako u rješenju (**LJR 2015./2016.**).

a)

/* A je polje $n \times n$ cijelih brojeva, sve varijable su tipa int,
funkcija f ima asimptotsku složenost $\Theta(n)$ */

```
tSum = 0;
for (i = 0; i < n; i++)
    for (j = 1; j <= n; j *= 2)
        tsum += A[n*i + j - 1] * f(n, i, j);
for (i = 0; i < n; i++) {
    sSum = A[i*n + 0] + A[i*n + 1] + A[i*n + 2];
    if (sSum > 0) {
        for (k = 0; k < n; k++)
            sSum += A[n*i + k] * f(n, i, k + 1);
        tSum += sSum
    }
}
```

13. Zadano je polje cijelih brojeva A koje ima n elemenata (n je paran broj), za koje vrijedi

$$A[0] < A[2] < A[4] < \dots < A[n-2] < A[1] < A[3] < \dots < A[n-1]$$

Napišite funkciju `traziBroj` koja vraća 1, ako je x element polja A, odnosno 0 ako nije. Funkcija `traziBroj` treba imati složenost $O(\log_2 n)$ (**JR 2018./2019.**)

Napomena: nije dozvoljeno korištenje pomoćnih polja i struktura, kao ni promjena vrijednosti elemenata polja.

Rješenja:

1. $\Theta(1)$
 2. a) $\Theta(1)$ b) $\Theta(n)$
 3. Istinite tvrdnje: a), c), f), g), h)
 4. Za $i = 0$, unutarnja petlja se obavi 0 puta; za $i = 1$, unutarnja petlja se obavi jedanput; ..., za $i = n-1$, unutarnja petlja se obavi $(n-1)$ puta.
Ukupno: $0 + 1 + 2 + \dots + (n-1) = n \cdot (n-1)/2$, tj. vremenska složenost je $\Theta(n^2)$
 5. Za $i = k$, unutarnja petlja se obavi k puta;
za $i = k/2$, unutarnja petlja se obavi $k/2$ puta;
za $i = k/4$, unutarnja petlja se obavi $k/4$ puta;
...
za $i = 1$, unutarnja petlja se obavi jedanput.
Ukupno: $1 + \dots + k/4 + k/2 + k \approx 2k$, tj. vremenska složenost je $\Theta(k)$
 6. a) $\Theta(n)$ b) $\Theta(n)$ c) $\Theta(1)$
d) ako je polje sortirano uzlazno/silazno: $\Theta(1)$; ako nije sortirano: $\Theta(n)$
 7. $O(\log n)$, $\Omega(\log n)$, $\Theta(\log n)$
 8. $O(n^2)$, $\Omega(n)$ (najbolji slučaj: silazno sortirano polje; najlošiji slučaj: uzlazno sortirano polje)
 9. a) $O(n)$ b) $O(n^2)$ c) $O(n^3)$
 10. a) $O(n)$, $\Omega(n)$, $\Theta(n)$
- Broj izvođenja petlje po j je manji ili jednak od:
- $$n + n/2 + n/4 + \dots + 1 = n (1 + 1/2 + (1/2)^2 + \dots + (1/2)^{\log_2 n}) \leq n n (1 + 1/2 + (1/2)^2 + \dots) = n \cdot (1 / (1 - 1/2)) = 2n$$
- Zadano je $\Theta(f_{\text{pom}}(n)) = \Theta(1)$, pa zaključujemo da je $O(f_1(n)) = O(n)$, $\Omega(f_1(n)) = \Omega(n)$ i $\Theta(f_1(n)) = \Theta(n)$.
- b) $O(n^2)$, $\Omega(n^2)$, $\Theta(n^2)$
- Petlja po i se izvodi $n/2$ puta, a unutar te petlje se poziva funkcija g_{Pom} .
Ako je $B[i] > n/2$, onda se funkcija g_{Pom} poziva dva puta, pa je složenost tog dijela $\Theta(n)$.
Ako je $B[i] \leq n/2$ tada se opet funkcija g_{Pom} poziva dva puta pa je složenost i tog dijela $\Theta(n)$. Kada sve spojimo ($n/2$ ponavljanja koda čija je složenost $\Theta(n)$) dobijemo da je:
- $$O(f_2(n)) = O(n^2), \Omega(f_2(n)) = \Omega(n^2) \text{ i } \Theta(f_2(n)) = \Theta(n^2).$$
11. Najlošiji slučaj je kada su svi indikatori $c[i]$ jednaki 1 i složenost iznosi $O(m^3)$. Najbolji slučaj je kada su svi indikatori $c[i]$ jednaki 0 i složenost je $\Omega(m^2)$. Θ nije moguće odrediti.

12. $O(n^3)$, $\Omega(n^2 \log(n))$

13.

```
int traziBroj(int x, int A[], int n) {
    int manji, sredina;
    int pIndex, zIndex;
    pIndex = 0;
    zIndex = n/2;
    // indikator tražim li među elementima
    // polja s parnim indeksima ili
    // elementima polja s neparnim indeksima
    if (x < A[1]) manji = 1; else manji = 0;

    while (pIndex < zIndex) {
        sredina = (zIndex + pIndex) / 2;
        if (manji) {
            // traženje među elementima s parnim indeksima
            if (A[2*sredina] == x) return 1;
            if (A[2*sredina] < x) {
                pIndex = sredina + 1;
            }
            else {
                zIndex = sredina ;
            }
        }
        else {
            // traženje među elementima s neparnim indeksima
            if (A[2 * sredina + 1] == x) return 1;
            if (A[2 * sredina + 1] < x) {
                pIndex = sredina + 1;
            }
            else {
                zIndex = sredina ;
            }
        }
    }
    return 0;
}
```