
Algoritmi și structuri de date (I). Seminar 5: Stabilirea ordinului de complexitate pornind de la numărul de operații efectuate. Proprietăți și calcule cu ordine de complexitate. Analiza complexității în cazul mediu.

Problema 1 Să se determine ordinul de creștere al valorii variabilei x (în funcție de n) după execuția algoritmilor **alg1**, **alg2**, **alg3**, **alg4**, **alg5** și **alg6**.

alg1(integer n)	alg2(integer n)
1: $x \leftarrow 0$	1: $x \leftarrow 0$
2: for $i \leftarrow 1, n$ do	2: $i \leftarrow n$
3: for $j \leftarrow 1, i$ do	3: while $i \geq 1$ do
4: for $k \leftarrow 1, j$ do	4: $x \leftarrow x + 1$
5: $x \leftarrow x + 1$	5: $i \leftarrow i \text{DIV} 2$
6: end for	6: end while
7: end for	7: return x
8: end for	
9: return x	

alg3(integer n)	alg4(integer n)
1: $x \leftarrow 0$	1: $x \leftarrow 0$
2: $i \leftarrow n$	2: $i \leftarrow n$
3: while $i \geq 1$ do	3: while $i \geq 1$ do
4: for $j \leftarrow 1, n$ do	4: for $j \leftarrow 1, i$ do
5: $x \leftarrow x + 1$	5: $x \leftarrow x + 1$
6: end for	6: end for
7: $i \leftarrow i \text{DIV} 2$	7: $i \leftarrow i \text{DIV} 2$
8: end while	8: end while
9: return x	9: return x

Indicație. **alg1**:

$$x = \sum_{i=1}^n \sum_{j=1}^i \sum_{k=1}^j 1 = \sum_{i=1}^n \sum_{j=1}^i j = \sum_{i=1}^n \frac{i(i+1)}{2} = \frac{n(n+1)(n+2)}{6} \in \Theta(n^3)$$

alg2: x va conține numărul de cifre ale reprezentării în baza 2 a lui n , adică $\lfloor \lg_2(n) \rfloor + 1 \in \Theta(\lg n)$.

alg3: Spre deosebire de exemplul anterior, pentru fiecare valoare a lui i variabila x este mărită cu n , deci valoarea finală va fi $n(\lfloor \lg_2(n) \rfloor + 1) \in \Theta(n \lg n)$.

alg4: Este similar algoritmului **alg3** însă pe linia 4 limita superioară a ciclului **for** este i în loc de n . Variabila x va conține suma $n + \lfloor \frac{n}{2} \rfloor + \dots + \lfloor \frac{n}{2^k} \rfloor$ cu k având proprietatea că $2^k \leq n < 2^{k+1}$. Ordinul de mărime al lui x se poate stabili pornind de la observația că $2^{k+1} - 1 \leq x < 2(2^{k+1} - 1)$. Deci $x \in \Theta(2^k) = \Theta(n)$.

alg5(integer n)	alg6(integer n)
1: $x \leftarrow 0$	1: $x \leftarrow 0$
2: $i \leftarrow 1$	2: $i \leftarrow 2$
3: while $i < n$ do	3: while $i < n$ do
4: $x \leftarrow x + 1$	4: $x \leftarrow x + 1$
5: $i \leftarrow 2 * i$	5: $i \leftarrow i * i$
6: end while	6: end while
7: return x	7: return x

alg5: Variabila x va conține cel mai mare număr natural k cu proprietatea că $2^k \leq n$, deci $x = \lfloor \lg_2 n \rfloor + 1 \in \Theta(\lg n)$.

alg6: Variabila x va conține cel mai mare număr natural k cu proprietatea că $2^{2^k} \leq n$ deci $x = \lfloor \lg_2 \lg_2 n \rfloor + 1 \in \Theta(\lg_2 \lg_2 n)$.

Problema 2 Să se determine termenul dominant și ordinul de creștere pentru expresiile:

- (a) $2\lg n + 4n + 3n\lg n$
- (b) $2 + 4 + 6 + \dots + 2n$
- (c) $\frac{(n+1)(n+3)}{n+2}$
- (d) $2 + 4 + 8 + \dots + 2^n$

Indicație. (a) Termenul dominant în $2\lg n + 4n + 3n\lg n$ este evident $3n\lg n$ iar ordinul de creștere este $n\lg n$.
 (b) Termenul dominant al sumei $2 + 4 + 6 + \dots + 2n$ nu este $2n$ ci n^2 întrucât $2 + 4 + 6 + \dots + 2n = n(n+1)$. Deci ordinul de creștere este chiar n^2 .
 (c) Termenul dominant este $n^2/(n+2)$ iar ordinul de creștere este n .
 (d) Întrucât $2 + 4 + 8 + \dots + 2^n = 2(1 + 2 + 4 + \dots + 2^{n-1}) = 2(2^n - 1)$, termenul dominant este $2 \cdot 2^n$ iar ordinul de creștere este 2^n .

Problema 3 Să se arate că:

- (a) $n! \in \mathcal{O}(n^n)$, $n! \in \Omega(2^n)$
- (b) $\lg n! \in \Theta(n\lg n)$
- (c) $2^n \in \mathcal{O}(n!)$
- (d) $\sum_{i=1}^n i\lg i \in \mathcal{O}(n^2\lg n)$
- (e) $\lg(n^k + c) \in \Theta(\lg n)$, $k > 0, c > 0$.
- (f) $\sum_{i=1}^n i^k \in \Theta(n^{k+1})$

Indicație. (a) Întrucât $n! \leq n^n$ pentru orice n natural, rezultă imediat că $n! \in \mathcal{O}(n^n)$. Pe de altă parte, $n! \geq 2^{n-1}$ pentru orice n , deci $n! \in \Omega(2^n)$.
 (b) Se pornește de la aproximația Stirling $n! \simeq \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$ care este adevărată pentru valori mari ale lui n . Mai exact, există $c_1, c_2 \in \mathbb{R}_+$ și $n_0 \in \mathbb{N}$ astfel încât

$$\sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \frac{c_1}{n}\right) \leq n! \leq \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \frac{c_2}{n}\right)$$

pentru orice $n \geq n_0$.

Prin logaritmare se obține:

$$\ln(2\pi n)/2 + \ln(1 + c_1/n) + n \ln n - n \leq \ln n! \leq \ln(2\pi n)/2 + \ln(1 + c_2/n) + n \ln n - n$$

deci $\ln n! \in \Theta(n \ln n)$. Facând abstracție de baza logaritmului se obține $\lg n! \in \Theta(n\lg n)$

- (c) Cum $2^n < n!$ pentru orice $n \geq 4$ rezultă că $2^n \in \mathcal{O}(n!)$.
- (d) $\sum_{i=1}^n i\lg i \leq \lg n \sum_{i=1}^n i \leq n^2\lg n$, deci $\sum_{i=1}^n i\lg i \in \mathcal{O}(n^2\lg n)$.
- (e) Pentru valori suficient de mari ale lui n are loc $\lg n^k \leq \lg(n^k + c) \leq \lg(2n^k)$, adică $k\lg n \leq \lg(n^k + c) \leq k\lg n + \lg 2$, deci $\lg(n^k + c) \in \Theta(\lg n)$.
- (f) Întrucât fiecare termen al sumei este mai mic sau cel mult egal cu n^k rezultă imediat că $\sum_{i=1}^k < n \cdot n^k = n^{k+1}$ deci suma este de ordin $\mathcal{O}(n^{k+1})$. Pentru a justifica faptul că suma este din $\Theta(n^{k+1})$ se poate folosi formula lui Faulhaber din care rezultă că suma este un polinom de grad $k+1$ de argument n (pentru o demonstrație a acestui fapt poate fi consultată lucrarea <https://www.emis.de/journals/AMEN/2018/AMEN-170803.pdf>).

Problema 4 Care dintre următoarele afirmații sunt adevărate ? Demonstrați.

- (a) $\sum_{i=1}^n i^2 \in \Theta(n^2)$, $\sum_{i=1}^n i^2 \in \mathcal{O}(n^2)$, $\sum_{i=1}^n i^2 \in \Omega(n^2)$
 (b) $cf(n) \in \Theta(f(n))$, $f(cn) \in \Theta(f(n))$
 (c) $2^{n+1} \in \mathcal{O}(2^n)$, $2^{2n} \in \mathcal{O}(2^n)$?

Indicație. (a) Întrucât $\sum_{i=1}^n i^2 = n(n+1)(2n+1)/6$ rezultă că $\sum_{i=1}^n i^2 \in \Theta(n^3)$, deci $\sum_{i=1}^n i^2 \in \Omega(n^2)$ însă celelalte două afirmații sunt false.

(b) Întrucât $c_1 f(n) \leq cf(n) \leq c_2 f(n)$ pentru constante c_1 și c_2 satisfăcând $0 < c_1 \leq c \leq c_2$ rezultă că $cf(n) \in \Theta(f(n))$. În schimb $f(cn) \in \Theta(f(n))$ nu este adevărată pentru orice funcție f și orice constantă c . De exemplu $f(n) = \exp(n)$ și $c > 1$ nu satisfac această proprietate întrucât nu există c' astfel încât $\exp(cn) \leq c' \exp(n)$ pentru valori mari ale lui n .

(c) Întrucât $2^{n+1} = 2 \cdot 2^n$ rezultă că $2^{n+1} \in \Theta(2^n)$ deci implicit și $2^{n+1} \in \mathcal{O}(2^n)$. Pe de altă parte, $\lim_{n \rightarrow \infty} 2^{2n}/2^n = \infty$ prin urmare $2^{2n} \notin \mathcal{O}(2^n)$ dar $2^{2n} \in \Omega(2^n)$.

Problema 5 Propuneți un algoritm pentru determinarea celor mai mici două valori dintr-un tablou. Determinați numărul de comparații efectuate asupra elementelor tabloului și stabiliți ordinul de complexitate al algoritmului.

Indicație. O variantă de algoritm este descrisă în `valori_minime`.

```

valori_minime(integer a[1..n])
1: if  $a[1] < a[2]$  then
2:    $min1 \leftarrow a[1]$ ;  $min2 \leftarrow a[2]$ ;
3: else
4:    $min1 \leftarrow a[2]$ ;  $min2 \leftarrow a[1]$ ;
5: end if
6: for  $i \leftarrow 3, n$  do
7:   if  $a[i] < min1$  then
8:      $min2 \leftarrow min1$ ;  $min1 \leftarrow a[i]$ 
9:   else if  $a[i] < min2$  then
10:     $min2 \leftarrow a[i]$ 
11:   end if
12: end for
13: return  $min1, min2$ 

```

În cazul cel mai nefavorabil numărul de comparații efectuate asupra elementelor tabloului este $T(n) = 1 + 2(n-2) = 2n-3$ deci algoritmul aparține lui $\mathcal{O}(n)$.

Problema 6 Propuneți un algoritm de complexitate liniară pentru a determina tabloul frecvențelor cumulate pornind de la tabloul frecvențelor simple. Pentru un tablou (f_1, \dots, f_n) de frecvențe, tabloul frecvențelor cumulate (fc_1, \dots, fc_n) se caracterizează prin $fc_i = \sum_{j=1}^i f_j$.

Rezolvare. Este ușor de văzut că un algoritm care calculează pentru fiecare i valoarea sumei $fc_i = \sum_{j=1}^i f_j$ necesită efectuarea a $\sum_{i=2}^n \sum_{j=1}^i 1 = \sum_{i=2}^n i = n(n+1)/2 - 1 \in \Theta(n^2)$. Observând că $fc_i = fc_{i-1} + f_i$ pentru $i = \overline{2, n}$ și $fc_1 = f_1$ rezultă că frecvențele cumulate pot fi calculate prin algoritmul `frecvente_cumulate` descris în continuare.

```

frecvente_cumulate(integer f[1..n])
integer i, fc[1..n]
1:  $fc[1] \leftarrow f[1]$ 
2: for  $i \leftarrow 2, n$  do
3:    $fc[i] = fc[i-1] + f[i]$ 
4: end for
5: return  $fc[1..n]$ 

```

Problema 7 Să se stabilească ordinul de complexitate pentru următorii algoritmi ce prelucreză date de volum n .

```

(a)  

---


      for  $i \leftarrow 1, n$  do
        ... ( $\Theta(1)$ )
        for  $j \leftarrow 1, 2i$  do
          ... ( $\Theta(1)$ )
           $k \leftarrow j$  ( $\Theta(1)$ )
          while  $k \geq 0$  do
            ... ( $\Theta(1)$ )
             $k \leftarrow k - 1$ 
          endwhile
        endfor
      endfor


---


(b)  

---


       $h \leftarrow 1$ 
      while  $h \leq n$  do
        ... ( $\Theta(1)$ )
         $h \leftarrow 2 * h$ 
      endwhile


---


(c)  

---


      calcul ( $x[0..n-1]$ )
       $k \leftarrow 0$ 
      for  $i \leftarrow 0, n-1$  do
        for  $j \leftarrow 0, n-1$  do
           $y[k] \leftarrow x[i] * x[j]$ 
           $k \leftarrow k + 1$ 
        endfor
      endfor return  $y[1..k]$ 


---



```

Rezolvare: (a):

$$T(n) = \sum_{i=1}^n \sum_{j=1}^{2i} \sum_{k=0}^j 1 = \sum_{i=1}^n \sum_{j=1}^{2i} (j+1) = \sum_{i=1}^n 2i^2 + 3i = \frac{n(n+1)(2n+1)}{3} + \frac{3n(n+1)}{2} \in \Theta(n^3)$$

(b): $T(n) \in \Theta(\lg n)$

(c): $T(n) \in \Theta(n^2)$

Problema 8 Propuneți un algoritm de complexitate $\Theta(n^2)$ și unul de complexitate $\Theta(n)$ pentru evaluarea unui polinom de grad n .

Rezolvare:

<code>polinom1(real $a[0..n]$, x)</code>	<code>polinom2(real $a[0..n]$, x)</code>
1: $y \leftarrow 0$	1: $y \leftarrow 0$
2: for $i \leftarrow 0, n$ do	2: $k \leftarrow n$
3: $k \leftarrow 1$	3: while $k \geq 0$ do
4: for $j \leftarrow 1, i$ do	4: $y \leftarrow y * x + a[k]$
5: $k \leftarrow k * x$	5: $k \leftarrow k - 1$
6: end for	6: end while
7: $y \leftarrow y + a[i] * k$	7: return y
8: end for	
9: return y	

În ambele cazuri dimensiunea problemei este n , iar operația dominantă este înmulțirea. Pentru algoritmul `polinom1` numărul de înmulțiri efectuate este $\sum_{i=0}^n \sum_{j=1}^i 1 = \sum_{i=0}^n i = n(n+1)/2 \in \Theta(n^2)$. În cazul algoritmului `polinom2` numărul de înmulțiri efectuate este $n+1$ deci ordinul de complexitate este $\Theta(n)$.

Probleme suplimentare/teme

1. Să se determine ordinul de mărime al variabilei x după execuția următoarelor prelucrări (și implicit ordinul de complexitate a algoritmului) :

```
1:  $x \leftarrow 0$ 
2:  $j \leftarrow n$ 
3: while  $j \geq 1$  do
4:   for  $i \leftarrow 1, j$  do
5:      $x \leftarrow x + 1$ 
6:   endfor
7:    $j \leftarrow j \text{DIV} 3$ 
8: endwhile
```

2. Se consideră un text constituit din N cuvinte, lungimea maximă a unui cuvânt fiind M . Asupra textului se aplică următoarea prelucrare:

Pas 1. Se ordonează crescător literele din fiecare cuvânt

Pas 2. Se ordonează lexicografic cuvintele din text

De exemplu, textul "exemplu de transformare a textului" va suferi următoarele transformări:

Pas 1. "eelmpux ed aaefmnorrst a eilttuux"

Pas 2. "a aaefmnorrst ed eelmpux eilttuux"

Stabiliți ordinul de complexitate al algoritmului în ipoteza că: (a) se utilizează un algoritm de sortare de complexitate pătratică, $O(n^2)$; (b) se utilizează un algoritm de sortare de complexitate $O(n \lg n)$.

3. Se consideră următorul algoritm:

```
1: alg(int  $n$ )
2:  $d \leftarrow 2$ 
3: while  $d * d < n$  do
4:   if  $n \text{ MOD } d == 0$  then
5:     return False
6:   end if
7:    $d \leftarrow d + 1$ 
8: end while
9: return True
```

Care este ordinul de complexitate al algoritmului?

4. Se consideră următorul algoritm:

```
1: alg(int  $n$ , int  $k$ )
2:  $p \leftarrow 1$ 
3: while  $k > 0$  do
4:   if  $k \text{ MOD } 2 == 1$  then
5:      $p \leftarrow p * n$ ;  $k \leftarrow k - 1$ 
6:   else
7:      $n \leftarrow n * n$ ;  $k \leftarrow k \text{ DIV } 2$ 
8:   end if
9: end while
10: return  $p$ 
```

Care este cel mai favorabil caz? Care este cel mai defavorabil caz? Care este ordinul de complexitate a algoritmului?

5. Se consideră următorul algoritm:

```

1: alg(int  $n$ )
2:  $i \leftarrow n/2$ ;  $nr \leftarrow 0$ 
3: while  $i \leq n$  do
4:    $j \leftarrow n$ 
5:   while  $j \geq 1$  do
6:      $nr \leftarrow nr + 1$ 
7:      $j \leftarrow j/2$ 
8:   end while
9:    $i \leftarrow i + 1$ 
10: end while
11: return  $nr$ 

```

Ce returnează algoritmul? Care este ordinul de complexitate?