Complexități

Algoritm	Complexitate timp				Complexitate spațiu
	CF	CD	CM	CT	
Căutare secvențială	$\theta(n)$	$\theta(n)$	$\theta(n)$	$\theta(n)$	$\theta(1)$
Căutare binară (în șir ordonat)	$\theta(1)$	$\theta(log_2 n)$	$\theta(log_2 n)$	$O(log_2 n)$	$\theta(1)$
Sortare prin selecție	$\theta(n^2)$	$\theta(n^2)$	$\theta(n^2)$	$\theta(n^2)$	in-place ($\theta(1)$)
Sortare prin inserție	$\theta(n)$	$\theta(n^2)$	$\theta(n^2)$	$O(n^2)$	in-place ($\theta(1)$)
Sortare prin metoda bulelor	$\theta(n)$	$\theta(n^2)$	$\theta(n^2)$	$O(n^2)$	in-place ($\theta(1)$)
Sortare rapidă	$\theta(nlog_2 n)$	$\theta(n^2)$	$\theta(nlog_2 n)$	$O(n^2)$	in-place ($\theta(1)$)
Sortare prin interclasare	$\theta(nlog_2 n)$	$\theta(nlog_2 n)$	$\theta(nlog_2 n)$	$\theta(nlog_2 n)$	out-of-place ($\theta(n)$)

1. Să se demonstreze că:

$$O(f) + O(g) = O(\max\{f, g\})$$

Definiție: $T(n) \in O(f(n)) <=> \exists\, n_0, c > 0\ a.î.\ 0 \leq T(n) \leq cf(n), \forall\, n \geq n_0.$

2. Construiți un algoritm cu timpul $\theta(nlog_2 n)$.

3. Să se determine complexitatea în timp a următorilor algoritmi:

```
găsit ← fals
pentru i ← 1, n execută
    dacă x_i = a atunci
        găsit ← adevărat
    sf-dacă
sf-pentru
```

```
găsit ← fals
cât-timp găsit = fals execută
    dacă x_i = a atunci
        găsit ← adev
    sf-dacă
sf-cât-timp
```

4. Fie *x* un tablou de întregi nenegativi. Dându-se urmatorul algoritm, să se calculeze complexitatea acestuia în timp:

```
k ← 0
pentru i ← 1, n execută
    pentru j ← 1, x_i execută
        k ← k + x_j
    sf-pentru
sf-pentru
```

5. Se consideră problema de a determina dacă un șir arbitrar de numere conține cel puțin 2 termeni egali. Arătați că acest fapt poate fi realizat în $\theta(nlog_2 n)$.

6. Să se determine complexitatea în timp a următorului algoritm:

```
pentru i ← 1, n execută
        @ op. elementară
sf-pentru
i ← 1
k ← adev
cât-timp ( i ≤ n – 1 și k = adev) execută
        j ← i
        k_1 ← adev
        cât-timp ( j ≤ n  și k_1 = adev) execută
                @ op. elementară
                j ← j + 1
                ...
        sf-cât-timp
        i ← i + 1
        @ op. elementară
        …
sf-cât-timp
```

7. Să se determine complexitatea în timp a următorului algoritm:

```
s ← 0
pentru i ← 1, n^2 execută
        j ← i
        cât-timp j ≠ 0 execută
                s ← s + j
                j ← j – 1
        sf-cât-timp
sf-pentru
```

8. Să se determine complexitatea în timp a următorului subalgoritm:

```
subalg  p(x, s, d) este
    dacă s < d atunci
            m ← [(s+d)/2]
        pentru i ← s, d – 1 execută
                pentru j ← i + 1, d execută
                        @ op. elementară
                sf-pentru
        sf-pentru
        pentru  i ← 1, 2 execută
                p(x, s, m)
                p(x, m+1, d)
        sf-pentru
    sf-dacă
sf-subalg
```

Apel: **p**(*x*, 1, *n*)

9. Să se determine complexitatea în timp a următorului subalgoritm:

```
s ← 0
pentru i ← 1, n^2 execută
        j ← i
    cât-timp j ≠ 0 execută
            s ← s + j - 10 * [j/10]
            j ← [j/10]
    sf-cât-timp
sf-pentru
```

10. Să se determine complexitatea în timp a următorului subalgoritm:

```
subalg operaţie(n, i) este
    dacă n > 1 atunci
        i ← 2 * i
        m ← ⌈n/2⌉
        operație(m, i – 2)
        operație(m, i – 1)
        operație(m, i + 2)
        operație(m, i + 1)
    altfel
        scrie i
    sf-dacă
sf-subalg
```

11. Să se determine complexitatea în timp a algoritmului recursiv de căutare binară:

```
subalg CautBin(a, x, s, d) este
    dacă s ≥ d - 1 atunci
        CautBin ← d
    altfel
        m ← ⌈(s+d)/2⌉
        dacă a ≤ x_m atunci
            CautBin(a, x, s, m)
        altfel
            CautBin(a, x, m, d)
        sf-dacă
    sf-dacă
sf-subalg
```

Apel: **CautBin**(a, x, 1, n)

12. Să se scrie un algoritm recursiv pentru a verifica dacă șirul $x_1, x_2, \dots, x_n$ este ordonat și să se calculeze complexitatea în timp a acestui algoritm.

13. Să se scrie un algoritm recursiv pentru a verifica dacă șirul $x_1, x_2, \dots, x_n$ conține cel puțin un element par și să se calculeze complexitatea în timp a acestui algoritm.