

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	$\theta(n \log_2 n)$	$\theta(n^2)$	$\theta(n \log_2 n)$	$O(n^2)$	in-place ($\theta(1)$)
Sortare prin interclasare	$\theta(n \log_2 n)$	$\theta(n \log_2 n)$	$\theta(n \log_2 n)$	$\theta(n \log_2 n)$	out-of-place ($\theta(n)$)

1. Să se demonstreze că:

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

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

2. Construiți un algoritm cu timpul $\theta(n \log_2 n)$.

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

```
găsit ← fals
pentru  $i \leftarrow 1, n$  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:

```
 $k \leftarrow 0$ 
pentru  $i \leftarrow 1, n$  execută
    pentru  $j \leftarrow 1, x_i$  execută
         $k \leftarrow k + x_j$ 
    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 $\theta(n \log_2 n)$.

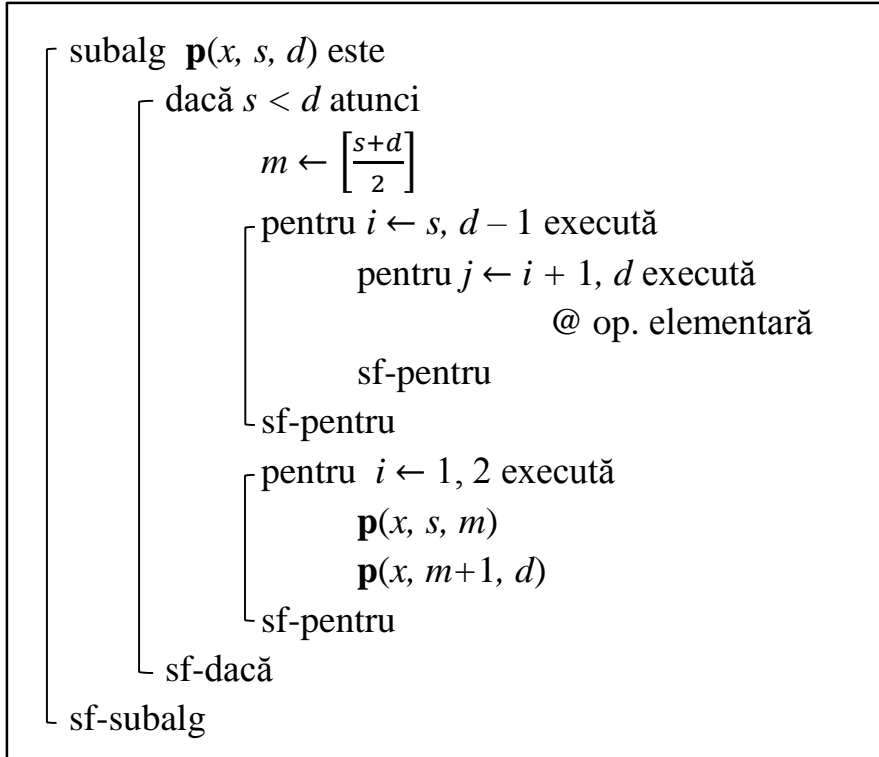
6. Să se determine complexitatea în timp a următorului algoritm:

```
┌ pentru  $i \leftarrow 1, n$  execută  
│   @ op. elementară  
└ sf-pentru  
   $i \leftarrow 1$   
   $k \leftarrow \text{adev}$   
  ┌ cât-timp ( $i \leq n - 1$  și  $k = \text{adev}$ ) execută  
  │    $j \leftarrow i$   
  │    $k_1 \leftarrow \text{adev}$   
  │   ┌ cât-timp ( $j \leq n$  și  $k_1 = \text{adev}$ ) execută  
  │   │   @ op. elementară  
  │   │    $j \leftarrow j + 1$   
  │   │   ...  
  │   └ sf-cât-timp  
  │    $i \leftarrow i + 1$   
  │   @ op. elementară  
  │   ...  
  └ sf-cât-timp
```

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

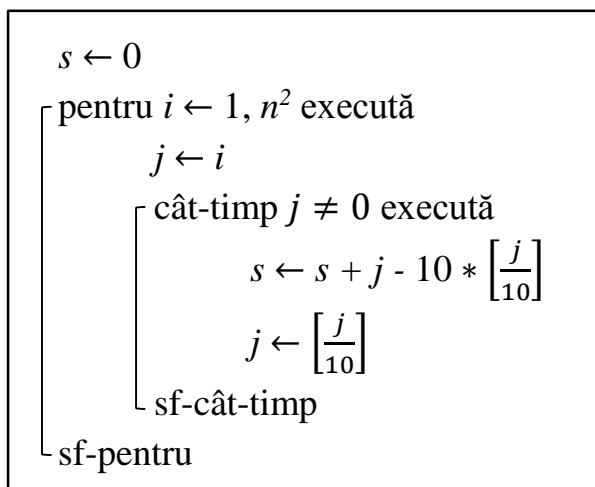
```
 $s \leftarrow 0$   
┌ pentru  $i \leftarrow 1, n^2$  execută  
│    $j \leftarrow i$   
│   ┌ cât-timp  $j \neq 0$  execută  
│   │    $s \leftarrow s + j$   
│   │    $j \leftarrow j - 1$   
│   └ sf-cât-timp  
└ sf-pentru
```

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

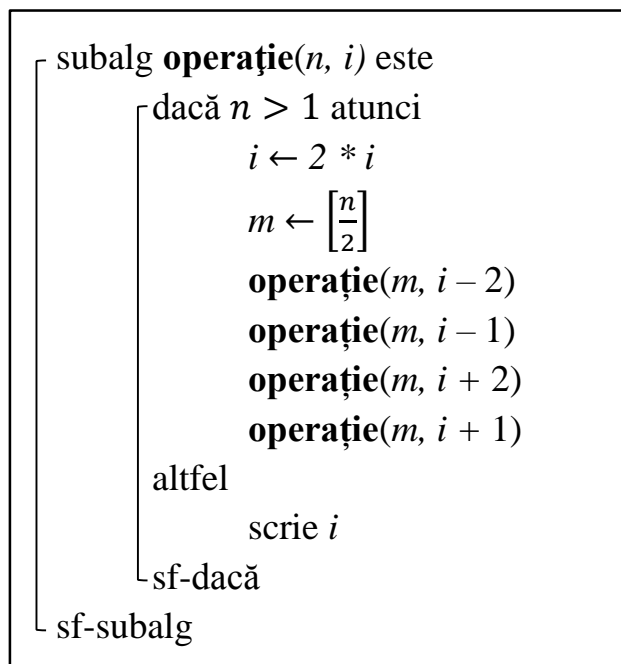


Apel: **p**($x, 1, n$)

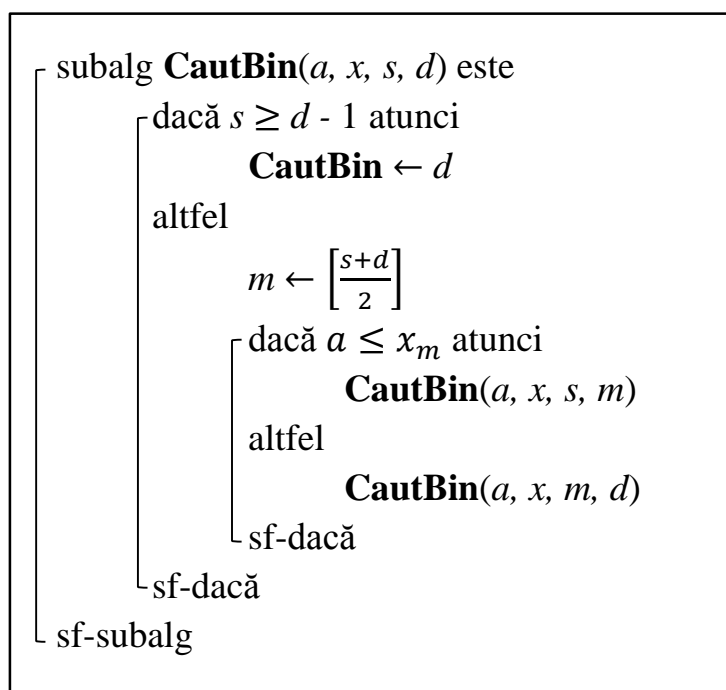
9. Să se determine complexitatea în timp a următorului subalgoritm:



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



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



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.