Laborator 6 - Complexitatea algoritmilor-0

Complexitatea algoritmilor

6.1 Timpul de execuţie şi ordinul de creştere al unui algoritm

Pentru a estima eficienţa de timp a unui algoritm se poate analiza numărul de operaţii care vor efectuate, relativ la dimensiunea datelor de intrare.

Considerând că instrucţiunile sunt executate secvenţial, fără calcul paralel, şi că operaţiile elementare (adunare, scădere, înmulţire, atribuire, evaluarea unei expresii logice etc.) au costul o unitate de timp, putem aproxima timpul de execuţie al unui algoritm.

Exercitiul 6.1. Pentru un n dat, se calculează suma S = 1 * 2 + 2 * 3 + ... + n * (n + 1) într-un mod iterativ.

Op.	Algoritm	Cost	Nr. rep.
1	$S \leftarrow 0$		
2	**for** $i \leftarrow 1, n$ **do**		
3	$S \leftarrow S + i * (i + 1)$		
	end for		

Operaţiile 1 şi 3 sunt elementare şi au costul 1. În particular, putem considera că operaţia 3 este formată dintr-o adunare, o înmulţire, o adunare şi o atribuire, caz în care costul poate fi evaluat la 4 unităţi de timp. Vom vedea, însă că valorile constante nu participă semnicativ la estimarea complexităţii de timp.

Operaţia 2 are costul 2 * (n + 1), deoarece se constituie din atribuirea succesivă a valorilor 1, 2, . . . , n + 1, variabilei i şi verificarea condiţiei i <= n pentru fiecare din aceste valori.

Atunci, timpul de execuţie se calculează în funcţie de n astfel:

$$T(n) = 1*1 + 2*(n+1)*1 + 1*n = 3*n +3$$

Considerând valori mari ale datelor de intrare, ordinul de creştere al timpului de execuţie al unui algoritm este dat de termenul dominant al acestuia. În cazul exemplului de mai sus, termenul dominant este n.

Pentru a compara diferiţi algoritmi din punct de vedere al complexităţii de timp se folosesc următoarele instrumente:

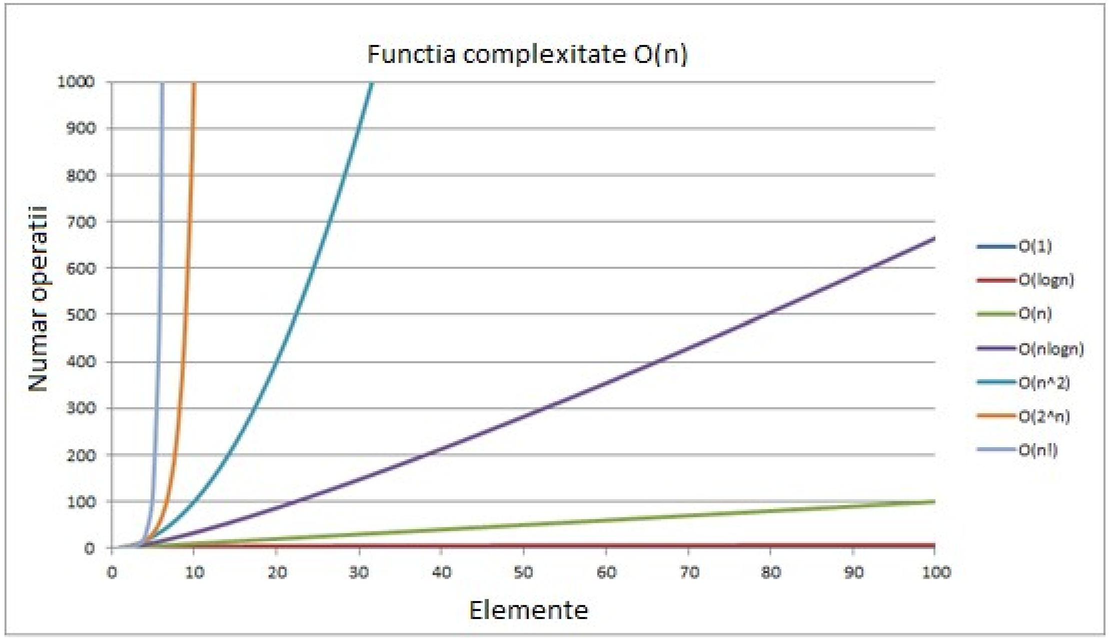
$$\Theta(\mathrm{g(n)}) = \{f(n) \mid \exists\, c_1, c_2 \in \mathbb{R}^+ \text{ şi } n_0 \in \mathbb{N} \text{ a.î.}$$
$$0 \le c_1 * g(n) \le f(n) \le c_2 * g(n), \forall\, n \ge n_0\}$$
$$\mathcal{O}(\mathrm{g(n)}) = \{f(n) \mid \exists\, c \in \mathbb{R}^+ \text{ şi } n_0 \in \mathbb{N} \text{ a.î.}$$
$$0 \le f(n) \le c * g(n), \forall\, n \ge n_0\}$$

În exemplul de mai sus avem că T(n) ∊ Θ(n) şi spunem că algoritmul are complexitate liniară. Evident, dacp T(n) ∊ Θ(n), atunci T(n) ∊ O(n).

Dacă expresia timpului de execuţie depinde de datele de intrare, aşa cum vom vedea mai jos, în câteva exemple, se pot explicita, în funcţie de numărul de operaţii executate, cazurile cel mai favorabil, mediu sau cel mai nefavorabil. Cel mai util pentru a stabili o limită de timp a unui algoritm este cazul cel mai nefavorabil. Acest caz descrie numărul maxim de operaţii care s-ar putea efectua la o rulare a algoritmului.

Se poate scrie următoarea ierarhie:

$$\mathcal{O}(1) \in \mathcal{O}(\log\log \mathrm{n}) \in \mathcal{O}(\log \mathrm{n}) \in \mathcal{O}(\mathrm{n}) \in \mathcal{O}(\mathrm{n}\log \mathrm{n}) \in \mathcal{O}(\mathrm{n}^2) \in \mathcal{O}(\mathrm{a}^\mathrm{n}).$$



6.2 Aplicaţii

Să se calculeze timpul de execuţie pentru următorii algoritmi:

Exercitiul 6.2. Produsul a două matrici cu m linii şi n coloane, respectiv n linii şi p coloane. Matricea C va fi matricea rezultată în urma înmulţirii A cu B.

Op.	Algoritm	Cost	Nr. rep.
1	**for** $i \leftarrow 0, m-1$ **do**		
2	**for** $j \leftarrow 0, p-1$ **do**		
3	$c[i][j] \leftarrow 0$		
4	**for** $k \leftarrow 0, n-1$ **do**		
5	$c[i][j] \leftarrow c[i][j] + a[i][k] * b[k][j]$		
	end for		
	end for		
	end for		

Obţinem:

T(m; n; p) = ?, atunci termenul dominant este ?.

Exercitiul 6.3. Calcularea valorii maxime dintr-un şir de n numere.

Op.	Algoritm	Cost	Nr. rep.
1	$max \leftarrow v[0]$		
2	**for** $i \leftarrow 1, n-1$ **do**		
3	**if** $max < v[i]$ **then**		
4	$max \leftarrow v[i]$		
	end if		
	end for		

Obţinem:

T(n) = ?.

Cazul cel mai favorabil: ?.

Cazul cel mai nefavorabil: ?

În ambele cazuri termenul dominant este ?.

Exercitiul 6.4. Căutarea secvenţială a unui element x într-un şir de n numere.

Op.	Algoritm	Cost	Nr. rep.
1	$gasit \leftarrow 0$		
2	$i \leftarrow 0$		
3	**while** $i < n$ AND $gasit = 0$ **do**		
4	**if** $x = v[i]$ **then**		
5	$gasit \leftarrow 1$		
	else		
6	$i \leftarrow i + 1$		
	end if		
	end while		

Obţinem T(n) = ?

Cazul cel mai favorabil: ?

Cazul cel mai nefavorabil: ?

Exercitiul 6.5. Căutarea binară a unui element x într-un şir ordonat crescător de n numere.

Op.	Algoritm	Cost	Nr. rep.
1	$gasit \leftarrow 0$		
2	$p \leftarrow 0$		
3	$u \leftarrow n - 1$		
4	**while** $p <= u$ AND $gasit = 0$ **do**		
5	$m \leftarrow (p + u)/2$		
6	**if** $x = v[m]$ **then**		
7	$gasit \leftarrow 1$		
	else		
8	**if** $x > v[m]$ **then**		
9	$p \leftarrow m + 1$		
	else		
10	$u \leftarrow m - 1$		
	end if		
	end if		
	end while		

Obţinem: T(n) = ?

Cazul cel mai favorabil: ?

Cazul cel mai nefavorabil: ?

Last modified: Wednesday, 18 January 2023, 12:02 PM

PREVIOUS ACTIVITY

NEXT ACTIVITY