Curs Algoritmi Fundamentali (AF)
Analiza eficienţei algoritmilor. Complexităţi

Lect. dr. Alexandra Băicoianu

Universitatea *Transilvania* din Braşov
Facultatea de Matematică şi Informatică

2020

Agenda

Eficienţa unui algoritm şi rolul ei practic

Rolul ei practic?
Vom compara numai algoritmii despre care ştim că sunt corecţi.
Putem compara doi algoritmi în raport cu **cantitatea de memorie necesară** şi **viteza de lucru**, deci timpul necesar rezolvării problemei.
Timpul necesar execuţiei depinde de numărul operaţiilor ce trebuie executate, iar numărul operaţiilor efectuate depinde de datele de intrare, care se schimbă de la o execuţie la alta.
Scopul principal al analizei eficienţei algoritmilor este acela de a determina modul în care timpul de execuţie creşte odată cu creşterea dimensiunii problemei.

Au fost propuse câteva probleme:

Se consideră un tablou cu n elemente având valori din $1, ..., n$. Tabloul poate avea toate elementele distincte sau poate exista o pereche de elemente cu aceeaşi valoare (o singură astfel de pereche). Să se verifice dacă elementele tabloului sunt toate distincte sau dacă există o pereche de elemente distincte.
Cât este complexitatea algoritmului propus?

Analiza unui algoritm

În general, analiza unui algoritm se desfăşoară în două etape:

- analiza apriorică: constă în aprecierea din punct de vedere temporal a operaţiilor care se utilizează şi a costului lor relativ. Aceasta conduce la stabilirea expresiei unei funcţii care mărgineşte timpul de execuţie al algoritmului;

Analiza unui algoritm

În general, analiza unui algoritm se desfăşoară în două etape:

- analiza apriorică: constă în aprecierea din punct de vedere temporal a operaţiilor care se utilizează şi a costului lor relativ. Aceasta conduce la stabilirea expresiei unei funcţii care mărgineşte timpul de execuţie al algoritmului;
- testul ulterior: constă în stabilirea unui număr suficient de seturi de date iniţiale care să acopere practic toate posibilităţile de comportament ale algoritmului. Această etapă se finalizează prin culegerea unor date în baza cărora pot fi elaborate o serie de statistici referitoare la consumul de timp specific execuţiei algoritmului în cauză (profilul algoritmului).

Analiza eficienţei unui algoritm

Analiza eficienţei algoritmilor înseamnă estimarea volumului de resurse de calcul (spaţiu de memorie, timp de execuţie) necesare execuţiei algoritmilor. Analiza eficienţei este utilă pentru a compara algoritmii între ei şi pentru a obţine informaţii privind resursele de calcul necesare pentru execuţia lor.

Avantajul oferit de analiza teoretica este acela că ea permite studiul eficienţei algoritmului pentru cazuri de orice mărime.

Cum se poate măsura eficienţa algoritmilor?

Pentru estimarea timpului de execuţie este necesar să se stabilească un **model de calcul** şi o **unitate de măsură** a timpului de execuţie (timpul necesar execuţiei unei prelucrări elementare - prelucrări de bază: asignare, operaţii aritmetice, comparaţii, operaţii logice).

Timpul de execuţie al algoritmului: $T(n)$ = numărul de operaţii elementare.

Estimarea timpului de execuţie are ca scop determinarea dependenţei dintre numărul de operaţii elementare executate şi dimensiunea problemei executate.

Cum se poate măsura eficienţa algoritmilor? (Exemplu)

Să se analizeze numărul de operaţii pentru **calculul sumei primelor** n **numere**.

- Scriere de pseudocod pentru rezolvare.

Cum se poate măsura eficienţa algoritmilor? (Exemplu)

Să se analizeze numărul de operaţii pentru **calculul sumei primelor** n **numere**.

- Scriere de pseudocod pentru rezolvare.
- Să se calculeze numărul de operaţii pentru $n = 5$. Câte operaţii avem pentru $n = 10$? Cât este $T(n)$?

Cum se poate măsura eficienţa algoritmilor? (Exemplu)

Să se analizeze numărul de operaţii pentru **calculul sumei primelor n numere**.

- Scriere de pseudocod pentru rezolvare.
- Să se calculeze numărul de operaţii pentru $n = 5$. Câte operaţii avem pentru $n = 10$? Cât este $T(n)$?
- Putem şi altă rezolvare? Cât este $T(n)$?

Cum se poate măsura eficienţa algoritmilor? (Alt exemplu)

Fie $T(n) = n^4 + 2 * n^2 + 10 * n + 500$.

n	n^4	T(n)
1	1	513
3	81	629
5	625	1225
7	2401	3069
10	10000	10800
100	100000000	100021500

Cu cât creşte valoarea lui n, cu atât devine mai mică diferenţa dintre $T(n)$ şi valoarea lui n^4. Astfel, pentru valori mai mari ale lui n putem spune că $T(n) \approx n^4$.
Ordinul de creştere expune modalitatea în care creşte termenul dominant al timpului de execuţie în raport cu dimensiunea problemei.

Analiza în cel mai favorabil şi cel mai defavorabil caz

Analiza în cazul cel mai favorabil furnizează o margine inferioară pentru timpul de execuţie şi permite identificarea algoritmilor ineficienţi (dacă un algoritm are un cost ridicat chiar şi în cel mai favorabil caz atunci el nu reprezintă o soluţie acceptabilă).

Analiza în cazul cel mai defavorabil furnizează cel mai mare timp de execuţie în raport cu toate datele de intrare de dimensiune n (reprezintă o margine superioară a timpului de execuţie). Este important de precizat faptul că că marginea superioară a timpului de execuţie este mai importantă decât marginea inferioară.

Analiza în cazul mediu

Cazul cel mai favorabil sau cazul cel mai defavorabil sunt cazuri particulare/excepţii.

Scopul analizei **cazului mediu** este să furnizeze informaţii privind comportarea algoritmului în cazul unor date de intrare arbitrare. Analiza în cazul mediu se bazează pe cunoaşterea distribuţiei de probabilitate a datelor de intrare. Timpul mediu de execuţie este valoarea medie (în sens statistic) a timpilor de execuţie corespunzători diferitelor instanţe ale datelor de intrare.

Care sunt etapele analizei eficienţei?

- Identificarea dimensiunii problemei;

Care sunt etapele analizei eficienţei?

- Identificarea dimensiunii problemei;
- Stabilirea operaţiei dominante (cea mai costisitoare operaţie din algoritm);

Care sunt etapele analizei eficienţei?

- Identificarea dimensiunii problemei;
- Stabilirea operaţiei dominante (cea mai costisitoare operaţie din algoritm);
- Determinarea numărului de execuţii ale operaţiei dominante;

Care sunt etapele analizei eficienţei?

- Identificarea dimensiunii problemei;
- Stabilirea operaţiei dominante (cea mai costisitoare operaţie din algoritm);
- Determinarea numărului de execuţii ale operaţiei dominante;
- Dacă numărul de execuţii ale operaţiei dominante depinde de proprietăţile datelor de intrare, atunci se analizează cel mai favorabil caz, cel mai defavorabil caz şi cazul mediu;

Care sunt etapele analizei eficienţei?

- Identificarea dimensiunii problemei;
- Stabilirea operaţiei dominante (cea mai costisitoare operaţie din algoritm);
- Determinarea numărului de execuţii ale operaţiei dominante;
- Dacă numărul de execuţii ale operaţiei dominante depinde de proprietăţile datelor de intrare, atunci se analizează cel mai favorabil caz, cel mai defavorabil caz şi cazul mediu;
- Se stabileşte ordinul (clasa) de complexitate.

Analiza asimptotică: O (Omicron), Ω (Omega), Θ (Theta)

Analiza timpilor de execuţie pentru dimensiuni mici ale problemei nu permite diferenţierea algoritmilor eficienţi de cei ineficienţi. Diferenţele dintre ordinele de creştere devin relevante pe parcursul creşterii dimensiunii problemei.

Analiza asimptotică are ca obiectiv studiul proprietăţilor timpului de execuţie atunci când dimensiunea problemei tinde către infinit. Există trei notaţii folosite pentru diverse clase de eficienţă: O (Omicron), Ω (Omega) şi Θ (Theta).

Notaţia O (Omicron)

Desemnează marginea asimptotică superioară a unei funcţii.
Fie $f, g : \mathbb{N} \to \mathbb{R}_+$ două funcţii care depind de dimensiunea problemei.
Spunem că funcţiile sunt în relaţia:

$$f(n) = O(g(n))$$

dacă şi numai dacă

$$\exists n_0 \in \mathbb{N}, \exists c > 0 \mid 0 <= f(n) <= c * g(n), \forall n >= n_0$$

Pentru valori mari ale lui n, $f(n)$ este mărginită superior de $g(n)$ multiplicată cu o constantă pozitivă:

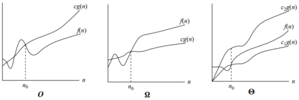

Figure: Ordine de creştere asimptotică (Corman et. al: Introduction to algorithms, MIT Press, 2009)

Notaţia O (Omicron)

$f(n) = n^4 + 2n^2 + 10n + 500$, $g(n) = n^5$ $f(n) = O(g(n))$ dacă găsim constantele pozitive c şi n_0, astfel încât $0 <= n^4 + 2n^2 + 10n + 500 <= c * n^5, \forall n >= n_0$. Alegem $c = 2$.

Table: Exemplu notaţia O

n	$f(n)$	$2 * g(n)$
1	513	2
3	629	486
4	828	2048
5	1225	6250
7	3069	33614

Pentru orice $n_0 >= 4$, vom avea
$0 <= n^4 + 2n^2 + 10n + 500 <= 2 * n^5$, deci avem $f(n) = O(n^5)$.

Notaţia *O* (Omicron) - Observaţii şi alte exemple

O este folosită pentru a descrie o margine superioară, ea este utilizată pentru a mărginii cazul cel mai nefavorabil de execuţie al unui algoritm.

Notaţia pune în evidenţă o margine superioară a complexităţii algoritmului în aceeaşi măsură pentru orice intrare.

Pentru exemplul cu **suma primelor** n **elemente** varianta 1 şi varianta 2: există mai multe valori pentru $g(n)$?

Alte exemple:

$$3n^2 - 100n + 6 = O(n^2), 3n^2 > 3n^2 - 100n + 6$$

$$3n^2 - 100n + 6 = O(n^3), 0.01n^3 > 3n^2 - 100n + 6$$

Notaţia O (Omicron) - Proprietăţi

1. $f(n) \in O(f(n))$ (reflexivitate)

Notaţia O (Omicron) - Proprietăţi

1. $f(n) \in O(f(n))$ (reflexivitate)
2. $f(n) \in O(g(n)), g(n) \in O(h(n)) \Rightarrow f(n) \in O(h(n))$ (tranzitivitate)

Notaţia O (Omicron) - Proprietăţi

1. $f(n) \in O(f(n))$ (reflexivitate)
2. $f(n) \in O(g(n)), g(n) \in O(h(n)) \Rightarrow f(n) \in O(h(n))$ (tranzitivitate)
3. Dacă $T(n) = a_d * n^d + a_{d-1} * n^{d-1} + ... + a_1 * n + a_0$ atunci $T(n) \in O(n^k)$ pentru orice $k >= d$. (funcţii polinomiale)

Notaţia O (Omicron) - Proprietăţi

1. $f(n) \in O(f(n))$ (reflexivitate)
2. $f(n) \in O(g(n)), g(n) \in O(h(n)) \Rightarrow f(n) \in O(h(n))$ (tranzitivitate)
3. Dacă $T(n) = a_d * n^d + a_{d-1} * n^{d-1} + ... + a_1 * n + a_0$ atunci $T(n) \in O(n^k)$ pentru orice $k >= d$. (funcţii polinomiale)
4. Dacă pentru cazul cel mai nefavorabil obţinem $T(n) <= g(n)$, atunci $T(n) \in O(g(n))$.