Algoritmi EXPONENȚIALI

Mihaela Oltean

Cât de rapid este un algoritm exponențial? Aceasta este întrebarea la care dorim să răspundem în cadrul acestui articol. Suntem convinși că vă imaginați care este răspunsul... Totuși, vă invităm să cercetați problema dintr-un punct de vedere ușor inedit.

În cartea *Computers and Intractability - A guide to the theory of NP-Completeness* [3] autorii ne prezintă tabelul 1, care conține timpul teoretic de execuție al unor algoritmi polinomiali și exponențiali.

Timpii de execuție prezentați în tabelul 1 au fost determinați pentru un calculator ipotetic care efectuează 10^6 operații pe secundă.

Acest mod teoretic de calcul al timpului de execuție diferă față de timpul real de execuție al unui program având aceeași complexitate.

Diferențele sunt datorate în primul rând faptului că diverse operații elementare necesită timpi de execuție diferiți.

De exemplu, se știe că o comparație este mai costisitoare decât o adunare. Un alt exemplu îl constituie operatorul la nivel de biți (and) care este implementat astfel încât să se execute în paralel pentru toți biții implicați. Astfel operația de conjuncție binară între două variabile reprezentate pe 32 de biți se execută într-un pas și nu în 32 de pași.

Alți factori care influențează timpul de execuție al programelor sunt legați de viteza (frecvența) diferitelor componente ale calculatoarelor (procesor, memorie *RAM*, disc hard).

În acest articol este realizată o analiză a timpului real de execuție al programelor de calculator. Pentru această analiză am folosit câțiva algoritmi exponențiali, des întâlniți în literatura de specialitate, pe care i-am rulat pe calculatoare având diverse configurații.

Alegerea algoritmilor exponențiali în detrimentul celor polinomiali s-a făcut din două motive:

- timpul de execuție al unor algoritmi polinomiali este foarte mic chiar și atunci când cantitatea de date de intrare este mare;
- la concursurile de informatică multe dintre problemele propuse se pot rezolva rapid folosind metoda *backtracking* [2, 3] în detrimentul unor metode mai eficiente (de exemplu programarea dinamică [1, 2]); unii elevi consideră că în acest fel au rezolvat problema fără să realizeze că de fapt timpul de execuție al programului lor este inacceptabil pentru anumite teste; în alte situații elevii se străduiesc (și adesea fără succes) să găsească metode mai inteligente, necrezând că un *backtracking* optimizat s-ar încadra în limita de timp;
- este tot mai des întâlnită concepția (greșită de altfel): *Calculatorul meu are un Pentium 4 Extreme Edition la 3,74 GHz. Pot să rulez orice program instantaneu!*; este adevărat că viteza calculatoarea a crescut foarte mult in ultimii 10 ani, dar foarte probabil unii algoritmi nu vor rula instantaneu nici pe calculatoarele anului 3000!

G. Moore

	10	20	30	40	50	60
n	0.00001 s	0.00002 s	0.00003 s	0.00004 s	0.00005 s	0.00006 s
n^2	0.0001 s	0.0004 s	0.0009 s	0.0016 s	0.0025 s	0.0036 s
n^3	0.001 s	0.008 s	0.027 s	0.064 s	0.125 s	0.216 s
n^5	0.1 s	3.2 s	24.3 s	1.7 min	5.2 min	13.0 min
2^n	0.001 s	1.0 s	17.9 min	12.7 zile	35.7 ani	366 secole
3^n	0.059 s	58 min	6.5 ani	3855 ani	$2 \cdot 10^8$ sec	$1.3 \cdot 10^{13}$ sec

Tabelul 1

Alegerea problemelor s-a făcut astfel încât mulțimea datelor de intrare să fie mică: pentru problemele selectate doar un singur număr este citit de la intrare.

În acest fel rezultatele prezentate în acest articol pot fi reproduse cu ușurință. Problemele analizate în acest articol sunt prezentate în continuare.

Problema damelor (P_1)

Să se amplaseze *n* dame pe o tablă de șah cu *n* linii și *n* coloane, astfel încât oricare două dame să nu se atace reciproc. [5]

Săritura calului (P_2)

Să se determine un drum al calului pe o tablă de șah cu *n* linii și *n* coloane, astfel încât fiecare căsuță să fie vizitată o singură dată. [5]

Partițiile unui număr întreg (P_3)

Se consideră un număr întreg *n*. Să se determine toate posibilitățile distinc-

n	C_1	C_2	C_3	C_1	C_2	C_3
	o soluție			toate soluțiile		
10	< 0,01 s	< 0,01 s	< 0,01 s	< 0,01 s	< 0,01 s	< 0,01 s
12	< 0,01 s	< 0,01 s	< 0,01 s	1,30 s	1,10 s	1,09 s
14	< 0,01 s	< 0,01 s	< 0,01 s	50,14 s	47,10 s	42,04 s
15	< 0,01 s	< 0,01 s	< 0,01 s	6,20 min	5,34 min	4,49 min
24	1,24 s	1,42 s	1,40 s	netestat	netestat	netestat
25	0,70 s	0,14 s	0,14 s	netestat	netestat	netestat
26	2,25 s	1,80 s	1,75 s	netestat	netestat	netestat
27	2,76 s	2,10 s	1,80 s	netestat	netestat	netestat
28	20,00 s	15,42 s	12,11 s	netestat	netestat	netestat

Tabelul 2: Problema P_1

te de a-l descompune pe *n* în sumă de numere mai mici sau egale cu el. Două descompuneri care diferă doar prin ordinea termenilor ***nu*** se consideră distincte. [4, 5]

Soluțiile acestor probleme se bazează pe metoda *backtracking* [2]. Descrierile soluțiilor pentru aceste probleme pot fi găsite în majoritatea cărților care tratează această metodă.

Nu vom insista asupra soluțiilor deoarece scopul nostru nu este de a prezenta și exemplifica metoda *backtracking*, ci de a analiza timpul de execuție al diferiților algoritmi exponențiali.

Deoarece unele probleme au mai multe soluții, am analizat, în cazurile în care a fost posibil, timpul necesar obținerii unei soluții și timpul necesar obținerii tuturor soluțiilor.

De exemplu, în cazul problemei P_3 am analizat doar timpul necesar generării tuturor soluțiilor, deoarece obținerea unei singure soluții nu necesită timp de lucru exponențial.

Pentru aceste probleme am analizat timpul de execuție rulând programele respective pe mai multe sisteme având configurații diferite.

Caracteristicile sistemelor implicate în această comparație sunt următoarele:

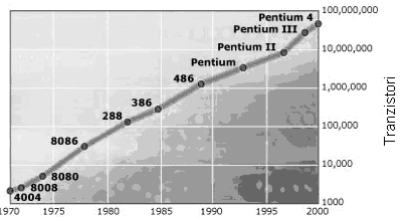
- *Intel Celeron*, 850 MHz (C_1);
- *Intel Pentium 4*, 1,4 GHz (C_2);
- *Intel Pentium 4*, 2,26 GHz (C_3).

n	C_1	C_2	C_3
	o soluție		
7	0.05 s	0.05 s	0.05 s
8	8 s	6.5 s	5,30 s
9	43 s	36 s	30.11 s

Tabelul 3: Problema P_2

Timpii necesari rulării celor trei probleme sunt prezentați în tabelele 2, 3 și 4. Aceste tabele ne permit să tragem următoarele concluzii:

- diferența dintre timpii de execuție este neglijabilă în cazul în care numărul datelor este mic (vezi problema P_1 pentru valori cel mult egale cu 15);
- diferența (în număr de secunde) dintre rulările pe diverse configurații crește foarte mult pe măsură ce dimensiunea spațiului de căutare crește.



Numărul tranzistorilor din procesoarele Intel

În loc de concluzii...

Adevăr: *Viteza calculatoarelor a crescut foarte mult în ultimul deceniu. Gordon Moore* [6, 7] a făcut în anul 1965 o faimoasă observație referitoare la numărul de tranzistori într-un circuit integrat. El a prezis că densitatea tranzistoarelor se va dubla la fiecare 18 luni. Graficul de pe această pagină ilustrează veridicitatea acestei observații (până acum).

Mit: *Un calculator Pentium 4 la 3,74 GHz rulează orice program instantaneu*. Nu este adevărat! Majoritatea programelor care există pe piață și care sunt rulate pe calculatoarele personale conțin doar algoritmi polinomiali.

Mit: *În lume există rețele formate din mii de calculatoare și super-calculatoare care pot executa orice program instantaneu.* Nu este adevărat! Dacă un program se execută pe un procesor într-o oră, înseamnă că pe un calculator cu 1000 procesoare se va executa în cel puțin 3,6 secunde.

Mit: *În 100 ani vom avea calculatoare care rulează orice program instantaneu.* Nu știm! Tehnologia curentă nu permite această viteză oricât de mult s-ar micșora un tranzistor. Trebuie dezvoltată o altă tehnologie, total diferită de cea curentă, care să permită execuția instantanee a unei instrucțiuni. Din păcate o astfel de tehnologie nu există și nici nu avem nici cea mai mică idee despre ea. Nici nu putem prezice când și dacă va fi inventată.

Bibliografie

[1] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1957

[2] T. Cormen, C. Leiserson, R. Rivest, *Introducere în algoritmi*, Editura Computer Libris Agora, Cluj-Napoca, 2000.

[3] D. Johnson, M. Garey, *Computers and Intractability - A guide to the theory of NP-Completeness*, Freeman&Co, San Francisco, 1979.

[4] I. Tomescu, *Introducere în combinatorică*, Editura Tehnică, București, 1982.

[5] Colecția *GInfo*, Agora Media, Târgu Mureș.

[6] G. Moore, *No Exponential is Forever - but We Can Delay 'Forever*, International Solid State Circuits Conference, February 10, 2003.

[7] G. Moore, *Cramming More Components Onto Integrated Circuits*, Electronics, April, 1965

n	C_1	C_2	C_3
	toate soluțiile		
70	1,23 s	1,04 s	1,02 s
75	2,50 s	2,14 s	2,00 s
80	4,90 s	4,40 s	3,15 s
85	10,0 s	7,45 s	5,59 s
90	19,21 s	16,03 s	12,09 s

Tabelul 4: Problema P_3