Algoritmi probabilişti

R. Trîmbiţaş
UBB

10 decembrie 2015

Rezumat

Se prezintă algoritmi Monte Carlo şi Las Vegas.

1 Introducere

Dacă într-un algoritm permitem o acţiune aleatoare cum ar fi aruncarea unei monede, adică generarea unui număr aleator, lărgim clasa de probleme rezolvabile şi totodată, în anumite cazuri, putem să accelerăm algoritmii. Un algoritm care are cel puţin un pas ce constă din generarea unui număr aleator se va numi **algoritm probabilist**. Algoritmii probabilişti sunt de multe ori mai simpli, mai rapizi şi mai uşor de analizat decât cei determinişti (pe care totuşi nu îi înlocuiesc).

Aleatorismul este o resursă utilă în proiectarea algoritmilor. El permite:

- conceperea unor algoritmi care să asigure, în ,,majoritatea cazurilor", aceeaşi performanţă pentru cazul cel mai nefavorabil ca un algoritm determinist complicat;
- să se realizeze lucruri care nu ştim cum s-ar putea face determinist, cum ar fi testul de primalitate, determinarea cu cost liniar a arborelui de acoperire minimal;
- să se realizeze lucruri care s-au demonstrat a fi imposibile în mod determinist.

Vom considera două clase de importante de algoritmi probabilişti: algoritmi Las Vegas în timp polinomial şi algoritmi Monte Carlo în timp polinomial.

Algoritmii Las Vegas în timp polinomial au următoarele caracteristici:

- depind de o sursă de numere aleatoare veritabile;
- returnează întotdeauna rezultatul corect;
- timpul de execuţie poate fi prost, dar timpul mediu de execuţie este bun.

Algoritmii Monte Carlo în timp polinomial au caracteristicile:

- depind de o sursă de numere aleatoare veritabile;
- pot să dea uneori rezultate eronate, dar cu o probabilitate mică;
- probabilitatea depinde numai de numerele aleatoare utilizate nu şi de intrare;
- timpul de execuţie este un polinom fixat.

2 Algoritmi Las Vegas

Ca prim exemplu de algoritm Las Vegas s-ar putea da algoritmul *Quicksort* cu pivotul selectat aleator. Nu insistăm asupra acestui algoritm, deoarece este cunoscut. Pentru detalii a se consulta [CLR].

Vom mai da un exemplu de algoritm Las Vegas pentru:

PROBLEMA ELEMENTULUI MAJORITAR. Se dă un tablou $A[1..n]$ de întregi şi se ştie că unul dintre elemente apare de cel puţin $n/2$ ori (element majoritar). Să se determine elementul.

Avem următorul algoritm:

```
FIND-MAJORITY(A,n)
repetă nedefinit
    alege aleator un indice i în domeniul 1..n;
    determină numărul j de apariţii ale lui A[i] în a;
    dacă (j ≥ n/2) atunci returnează j.
```

Deoarece există un element majoritar, algoritmul va găsi unul cu o probabilitate $\geq \frac{1}{2}$. Numărul mediu de încercări necesare pentru a-l găsi este

$$\sum_{i=1}^{\infty} \frac{i}{2^i} = 2.$$

Fiecare încercare necesită un timp liniar, deci numărul mediu de operaţii (după toate generările de numere aleatoare) este $O(n)$. În concluzie algoritmul are un timp de execuţie liniar.

Observaţia 1 *1. Algoritmul nu este robust. Dacă nu există element majoritar, algoritmul intră într-un ciclu infinit. Putem rezolva această problemă oprindu-ne după un număr fixat de încercări. Dar în acest mod putem da un răspuns eronat (totuşi cu o probabilitate mică) şi algoritmul se transformă într-un algoritm de tip Monte Carlo.*

2. În practică calitatea generatorului de numere aleatoare poate influenţa crucial rezultatul.

3 Algoritmi Monte Carlo

Vom considera problema: dându-se trei matrice pătratice de ordinul n, A, B şi C să se verifice dacă $C = AB$.

Algoritmul clasic de înmulţire rezolvă problema în timp $O(n^3)$. Algoritmul lui Strassen (vezi [CLR]) rezolvă problema în timp $O(n^{\log_2 7})$. Cel mai bun algoritm determinist cunoscut are un timp de execuţie $O(n^{2.376})$ ([CLR]).

Să analizăm următorul algoritm de tip Monte Carlo datorat lui Freivald.

`repetă de` k `ori`
    `generează un vector aleator` $x \in \{-1, 1\}^n$;
    `dacă` $A(Bx) \neq Cx$ `atunci returnează ''diferit'';`
`returnează ''egal''.`

Teorema 2 *Probabilitatea de eroare p a algoritmului de mai sus verifică $p \leq 2^{-k}$.*

Demonstraţie. Dacă $ABx \neq Cx$, atunci există indicii i, j astfel încât

$$c_{i,j} \neq \sum_{l=1}^{n} a_{il} b_{lj}.$$

Atunci $ABx \neq Cx$ fie pentru $x_j = 1$ fie pentru $x_j = -1$. Astfel, putem spune la o încercare, cu o probabilitate $\leq 1/2$, dacă $AB \neq C$. Deoarece încercările sunt independente, $p \leq 2^{-k}$. ■

Timpul de execuţie este $O(n^2)$. Problema existenţei unui algoritm determinist pentru rezolvarea acestei probleme în timp pătratic este deschisă (ea ar putea fi mai simplă decât problema înmulţirii în timp pătratic).

Următorul exemplu se referă la verificarea unor identităţi.

Fie $f(x_1, \ldots, x_n)$ un polinom cu coeficienţi raţionali de n variabile de grad cel mult k în fiecare dintre variabile. Vrem să decidem dacă $f \equiv 0$. Ideea de bază este să înlocuim variabilele cu numere aleatoare şi să calculăm valoarea polinomului. Dacă aceasta nu este zero polinomul nu poate fi identic nul. Dacă pentru un număr de încercări suficient de mare, se obţine de fiecare dată valoarea zero, probabilitatea ca polinomul să nu fie identic nul este mică. Vom alege pentru variabile valori întregi din intervalul $[0, N-1]$, independente şi uniform distribuite. Are loc următorul rezultat:

Lema 3 (Schwarz) *Dacă f nu este identic nul şi valorile ξ_i sunt independente şi uniform distribuite în intervalul $[0, N-1]$, atunci*

$$P\left(f\left(\xi_1, \ldots, \xi_n\right) = 0\right) \leq \frac{kn}{N}.$$

Demonstraţie. Se face prin inducţie după n. Lema este adevărată pentru $n = 1$, deoarece un polinom într-o variabilă de grad k poate avea cel mult k rădăcini. Fie $n > 1$ şi să ordonăm f după puterile lui x_1:

$$f = f_0 + f_1 x_1 + f_2 x_1^2 + \cdots + f_t x_1^t,$$

unde $f_0, \ldots, f_t$ sunt polinoame în variabilele $x_2, \ldots, x_n$, termenul f_t nu este identic 0 şi $t \leq k$. Aplicând formula probabilităţii totale avem

$$\begin{aligned}
&P\left(f(\xi_1, \ldots, \xi_n) = 0\right) \leq \\
&P\left(f(\xi_1, \ldots, \xi_n) = 0 | f_t(\xi_2, \ldots, \xi_n) = 0\right) P\left(f_t(\xi_2, \ldots, \xi_n) = 0\right) + \\
&+ P\left(f(\xi_1, \ldots, \xi_n) = 0 | f_t(\xi_2, \ldots, \xi_n) \neq 0\right) P\left(f_t(\xi_2, \ldots, \xi_n) \neq 0\right) \\
\leq &P\left(f_t(\xi_2, \ldots, \xi_n) = 0\right) + P\left(f(\xi_1, \ldots, \xi_n) = 0 | f_t(\xi_2, \ldots, \xi_n) \neq 0\right).
\end{aligned}$$

Primul termen poate fi estimat folosind ipoteza inducţiei, iar al doilea este cel mult k/N (căci ξ_1 este independentă de variabilele $\xi_2, \ldots, \xi_n$ şi de aceea dacă ultimele sunt fixate astfel ca $f_t \neq 0$ şi f ca polinom în x_1 nu este identic nul, atunci probabilitatea ca ξ_1 să fie rădăcină este cel mult k/N). Deci

$$P\left(f(\xi_1, \ldots, \xi_n) = 0\right) \leq \frac{k(n-1)}{N} + \frac{k}{N} \leq \frac{kn}{N}.$$

■

Aceasta ne conduce la următorul algoritm: calculăm $f(\xi_1, \ldots, \xi_n)$ pentru valorile întregi ξ_i care sunt numere (pseudo) aleatoare independente distribuite uniform discret în intervalul $[0, 2kn]$. Dacă obţinem o valoare diferită de 0 ne oprim : f nu este identic nul. Dacă obţinem valoarea 0 repetăm calculul. Dacă obţinem valoarea 0 de, să zicem 100 de ori, ne oprim şi decidem că f este identic nul.

Observaţia 4 *Dacă numărul de repetări de repetări este l, probabilitatea ca algoritmul să decidă eronat că $f \equiv 0$ este $< 2^{-l}$, deoarece probabilitatea de a greşi la o încercare este $\leq 1/2$, iar încercările sunt independente.*

Problem 5 *Să se verifice identităţile:*

$$(a+b+c)^3 - a^3 - b^3 - c^3 = 3(a+b)(a+c)(b+c),$$

Liouville

$$\sum_{1 \leq i < j \leq 4} \left[(x_i + x_j)^4 + (x_i - x_j)^4\right] = 6\left(x_1^2 + x_2^2 + x_3^2 + x_4^2\right)^2,$$

Viète

$$x^3 + y^3 = \left(\frac{x^4 + 2xy^3}{x^3 - y^3}\right)^3 + \left(\frac{y^4 + 2x^3y}{y^3 - x^3}\right)^3.$$

Bibliografie

[1] Radu Trîmbiţaş – *Metode statistice*, Presa Universitară Clujeană, 2000.

[2] Gaston Gonnet – Determining Equivalence of Expressions in Random Polynomial Time, *Proceedings of the 16th ACM Symposium on the Theory of Computing*, Washington DC, April 1984, pp. 334-341.

[CLR] C. Cormen, T. Leiserson, R. Rivest – *Algorithms*, MIT Press, 1994.