Algoritmi probabiliști

Monte Carlo & Las Vegas

Slide 26.

Observatie 1: Daca $A x B = C$, atunci algoritmul returnează mereu “DA”

$A(Br)=(AB)r=Cr$ - deci algoritmul va returna “DA”

Afirmatie 2: **Daca $AB \neq C$, atunci $Prob[ABr \neq Cr] \geq 1/2$**

Slide 26.
Observatie 1: Daca AxB=C, atunci algoritmul returnează mereu “DA”
A(Br)=(AB)r=Cr - deci algoritmul va returna “DA”

Afirmatie 2: **Daca AB≠C, atunci Prob[ABr≠Cr]≥1/2**

fie D=AB-C.

Slide 26.

Observatie 1: Daca $A \times B = C$, atunci algoritmul returnează mereu “DA”

$A(Br)=(AB)r=Cr$ - deci algoritmul va returna “DA”

Afirmatie 2: **Daca $AB \neq C$, atunci $Prob[ABr \neq Cr] \geq 1/2$**

fie $D=AB-C$.

Ipoteza de lucru spune ca $AB \neq C$, deci $D \neq O_{n,n}$. Deci cu siguranta exista vectori r astfel incat $Dr \neq O$.

Slide 26.
Observatie 1: Daca AxB=C, atunci algoritmul returnează mereu “DA”
A(Br)=(AB)r=Cr - deci algoritmul va returna “DA”

Afirmatie 2: **Daca AB≠C, atunci Prob[ABr≠Cr]≥1/2**

fie D=AB-C.

Ipoteza de lucru spune ca AB≠C, deci D≠On,n. Deci cu siguranta exista vectori r astfel incat Dr≠O.

Scopul este sa aratam ca exista o multitudine de astfel de valori pentru r.

Slide 26.
Observatie 1: Daca AxB=C, atunci algoritmul returnează mereu “DA”
A(Br)=(AB)r=Cr - deci algoritmul va returna “DA”

Afirmatie 2: **Daca AB≠C, atunci Prob[ABr≠Cr]≥1/2**

fie D=AB-C.

Ipoteza de lucru spune ca AB≠C, deci D≠On,n. Deci cu siguranta exista vectori r astfel incat Dr≠O.

Scopul este sa aratam ca exista o multitudine de astfel de valori pentru r.
Mai exact, vom arata ca Prob[Dr ≠ O]≥1/2

Mai exact, vom arata ca Prob[Dr ≠ O]≥1/2

Vom arata ca pentru fiecare r, cu proprietatea ca Dr=O, exista un r' "croit" pentru r astfel incat Dr' ≠ O.

Mai exact, vom arata ca Prob[Dr ≠ O]≥1/2

Vom arata ca pentru fiecare r, cu proprietatea ca Dr=O, exista un r' "croit" pentru r astfel incat Dr' ≠ O.

Daca D≠O, exista i,j astfel încât di,j≠0.

Mai exact, vom arata ca Prob[Dr ≠ O]≥1/2

Vom arata ca pentru fiecare r, cu proprietatea ca Dr=O, exista un r' "croit" pentru r astfel incat Dr' ≠ O.

Daca D≠O, exista i,j astfel încât di,j≠0.

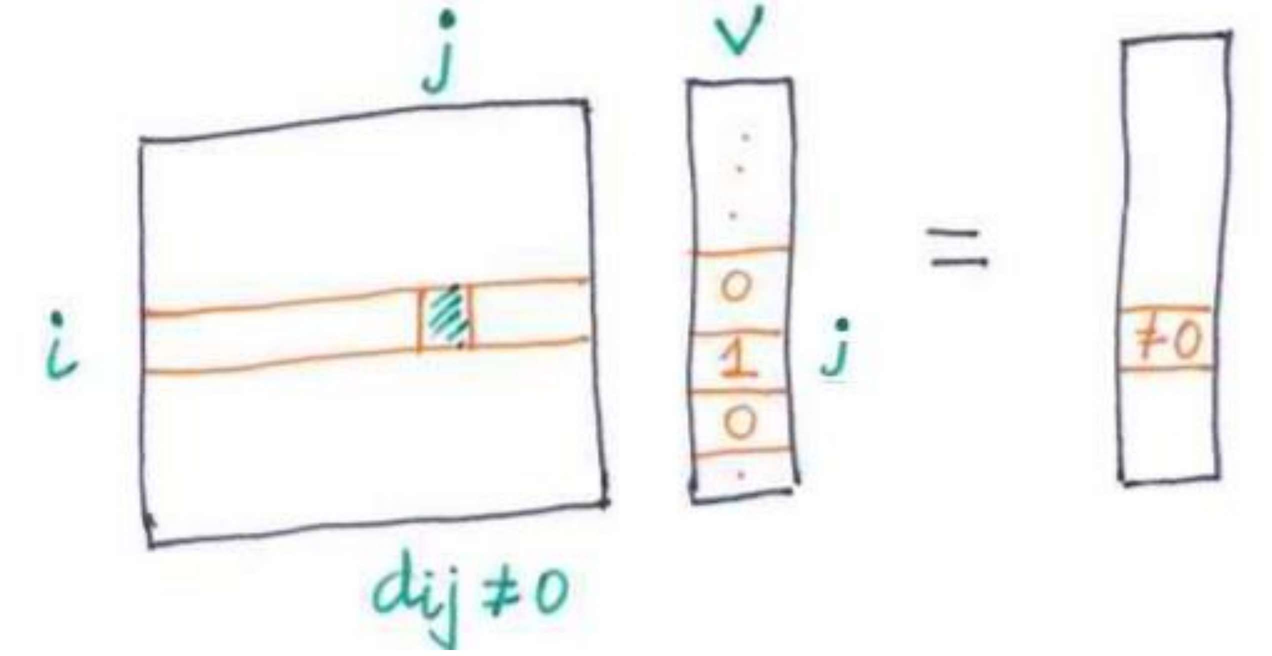
Alegem v - un vector de lungime n cu toate elementele 0, mai puțin elementul de pe poziția j.

Mai exact, vom arata ca Prob[Dr ≠ O]≥1/2

Vom arata ca pentru fiecare r, cu proprietatea ca Dr=O, exista un r' "croit" pentru r astfel incat Dr' ≠ O.

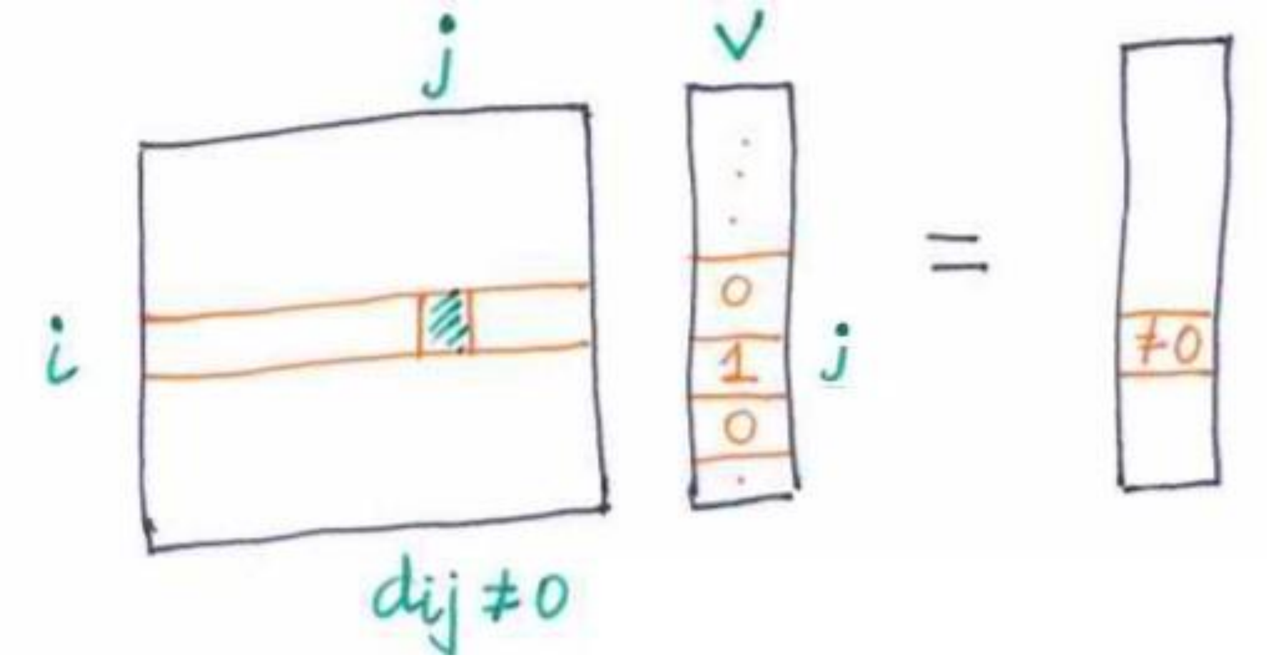
Daca D≠O, exista i,j astfel încât di,j≠0.

Alegem v - un vector de lungime n cu toate elementele 0, mai puțin elementul de pe poziția j.



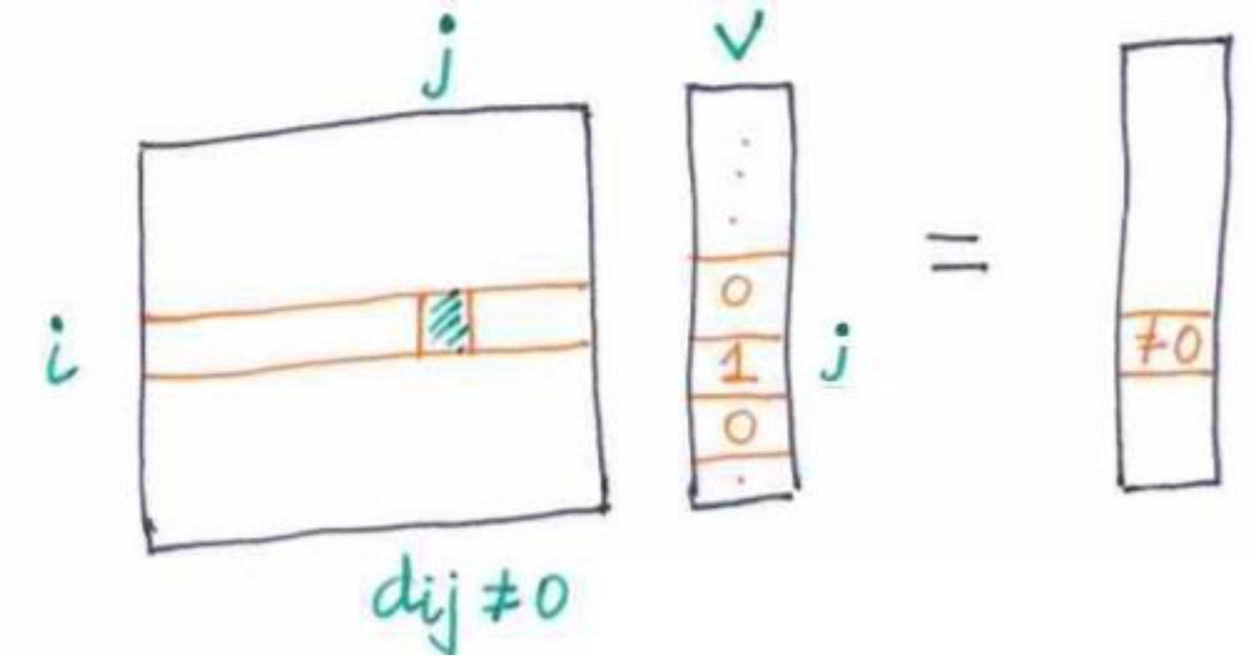
Mai exact, vom arata ca Prob[Dr ≠ O]≥1/2

Fie r un vector generat aleator cu Dr=O.



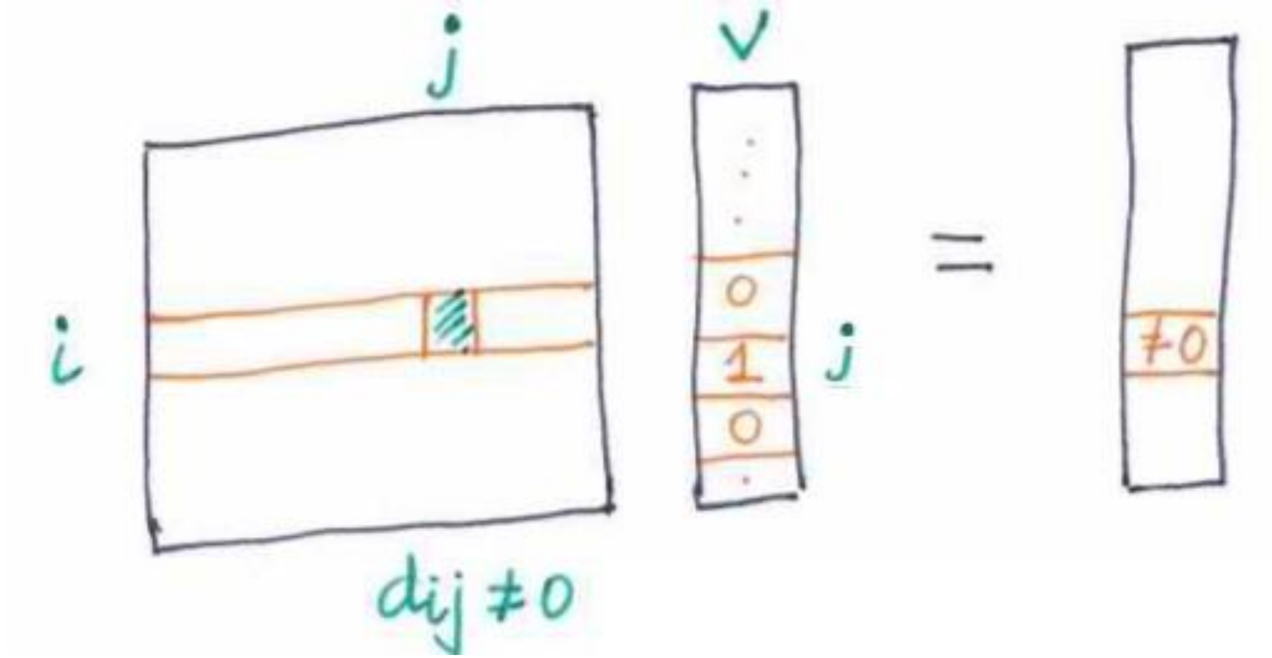
Mai exact, vom arata ca Prob[Dr ≠ O]≥1/2

Fie r un vector generat aleator cu Dr=O.
Pentru fiecare vector astfel generat, putem calcula un vector r’=r+v.



Mai exact, vom arata ca Prob[Dr ≠ O]≥1/2

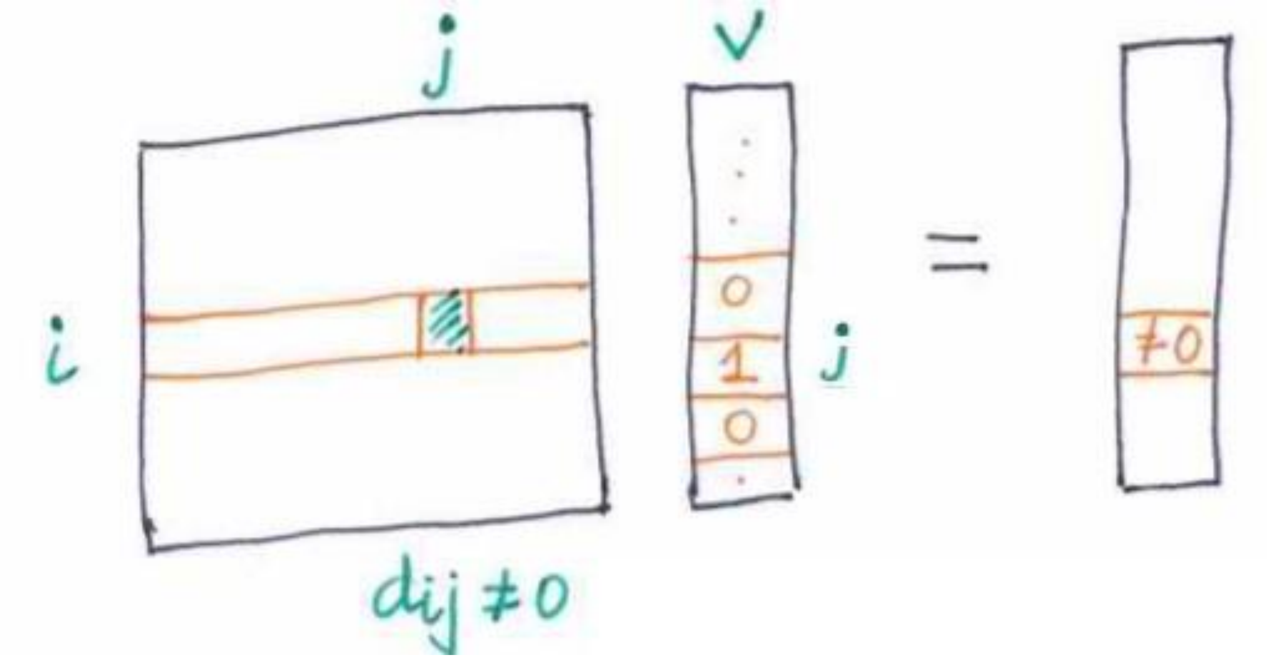
Fie r un vector generat aleator cu Dr=O.
Pentru fiecare vector astfel generat, putem calcula un vector r’=r+v.
Deoarece v este 0 peste tot mai puțin pe poziția j, r’ va fi identic cu r peste tot, mai puțin pe poziția j, unde va fi $r'_j = (r_j + v_j)$ mod 2.



Mai exact, vom arata ca Prob[Dr ≠ O]≥1/2

Fie r un vector generat aleator cu Dr=O.
Pentru fiecare vector astfel generat, putem calcula un vector r’=r+v.
Deoarece v este 0 peste tot mai puțin pe poziția j, r’ va fi identic cu r peste tot, mai puțin pe poziția j, unde va fi $r'_j = (r_j + v_j)$ mod 2.

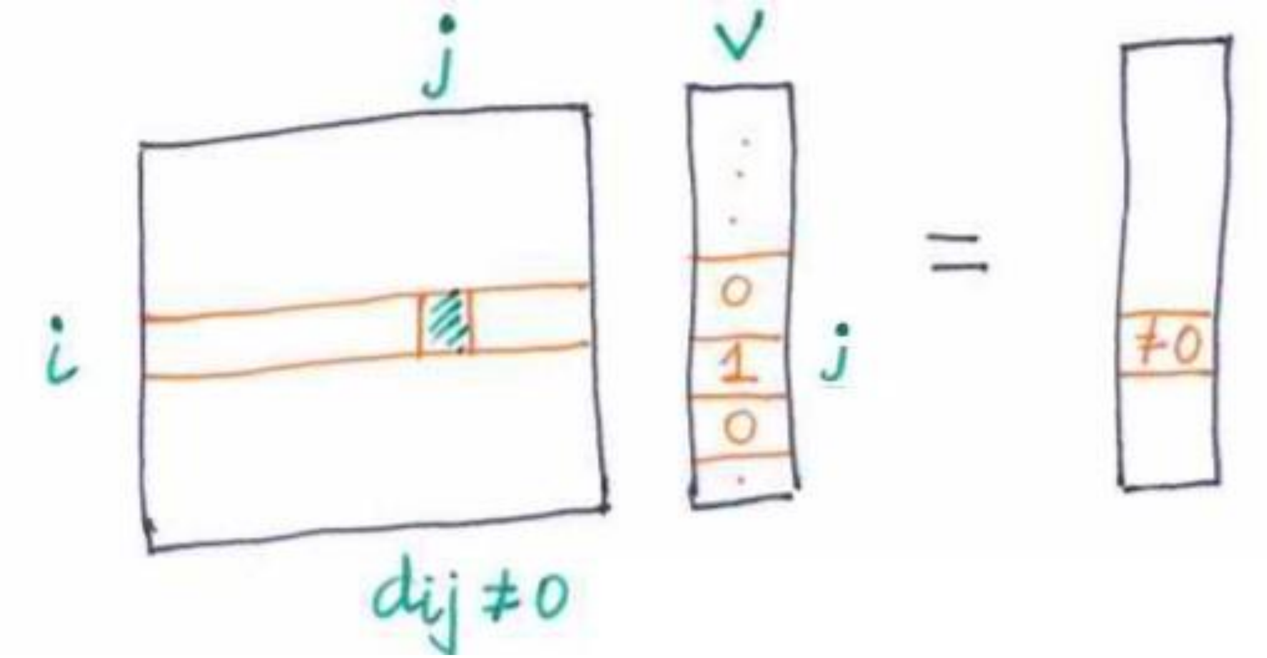
Pe cale de consecință avem Dr’=D(r+v)≠O.



Mai exact, vom arata ca Prob[Dr ≠ O]≥1/2

Fie r un vector generat aleator cu Dr=O.
Pentru fiecare vector astfel generat, putem calcula un vector r'=r+v.
Deoarece v este 0 peste tot mai puțin pe poziția j, r' va fi identic cu r peste tot, mai puțin pe poziția j, unde va fi $r'_j = (r_j + v_j)$ mod 2.

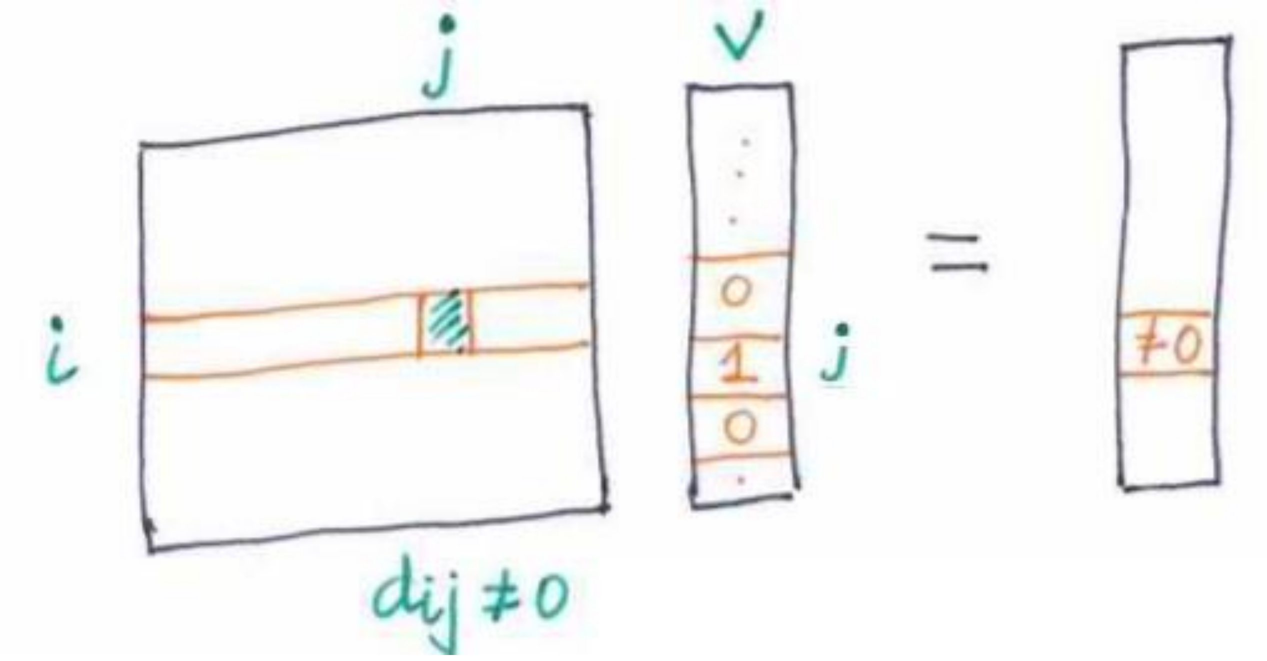
Pe cale de consecință avem Dr'=D(r+v)≠O.
Am aratat ca:
pentru fiecare r cu proprietatea ca ne furnizeaza un false positive (Dr=O), noi putem sa construim macar un r', unic pentru r, astfel incat Dr'≠O.



Mai exact, vom arata ca Prob[Dr ≠ O]≥1/2

Fie r un vector generat aleator cu Dr=O.
Pentru fiecare vector astfel generat, putem calcula un vector r'=r+v.
Deoarece v este 0 peste tot mai puțin pe poziția j, r' va fi identic cu r peste tot, mai puțin pe poziția j, unde va fi $r'_j = (r_j + v_j)$ mod 2.

Pe cale de consecință avem
Dr'=D(r+v)≠O.
Am aratat ca:
pentru fiecare r cu proprietatea ca ne furnizeaza un false positive (Dr=O), noi putem sa construim macar un r', unic pentru r, astfel incat Dr'≠O.
Deci Prob[Dr=O, iar D≠O]<½
q.e.d



The Solovay-Strassen Test (slides 27-28)

Defintitii:

- Simbolul Legendre – pentru orice număr prim *p,* avem:
  - $\left[\frac{b}{p}\right] = \begin{cases} 0 \mid cmmdc(b,p) \neq 1 \\ 1 \mid \exists\ x\ a.\hat{\imath}.\ x^2 \equiv b \bmod p \\ -1 \mid alfel \end{cases}$

The Solovay-Strassen Test

Defintitii:

- Simbolul Legendre – pentru orice număr prim *p,* avem:
  - $\left[\frac{b}{p}\right] = \begin{cases} 0 \mid cmmdc(b,p) \neq 1 \\ 1 \mid \exists\ x\ a.\hat{\imath}.\ x^2 \equiv b \bmod p \\ -1 \mid alfel \end{cases}$

Ex: $\left[\frac{10}{7}\right] = -1$

$$cmmdc(7{,}10) = 1; 10\ nu\ este\ reziduu\ quadratic\ pt\ 7$$

Ex: $\left[\frac{5}{11}\right] = 1$

$$cmmdc(5{,}11) = 1; 4^2 \equiv 5\ mod\ 11$$

The Solovay-Strassen Test

Defintitii:

- Simbolul Legendre – pentru orice număr prim *p,* avem:
  - $\left[\frac{b}{p}\right] = \begin{cases} 0 \mid cmmdc(b,p) \neq 1 \\ 1 \mid \exists\ x\ a.\hat{\imath}.\ x^2 \equiv b\ mod\ p \\ -1 \mid alfel \end{cases}$
- Simbolul Jacobi – generalizare pentru un *n* impar, nu neapărat prim
- Fie descompunerea lui *n* în factori primi: $n = \prod p_i^{\alpha_i}$ atunci avem:
- $\left(\frac{k}{n}\right) = \prod \left[\frac{k}{p_i}\right]^{\alpha_i}$

The Solovay-Strassen Test

- Simbolul Jacobi – generalizare pentru un *n* impar, nu neapărat prim
- Fie descompunerea lui *n* în factori primi: $n = \prod p_i^{\alpha_i}$ atunci avem:
- $\left(\frac{k}{n}\right) = \prod \left[\frac{k}{p_i}\right]^{\alpha_i}$

- Ex: $\left(\frac{15}{21}\right): 21 = 3 * 7; deci \left(\frac{15}{21}\right)$ = [$\frac{15}{3}$] $*$[$\frac{15}{7}$]=0;
- Ex: $\left(\frac{10}{21}\right): 21 = 3 * 7; deci \left(\frac{10}{21}\right)$ = [$\frac{10}{3}$] $*$[$\frac{10}{7}$]=1$*$−1=−1;

The Solovay-Strassen Test

- Simbolul Jacobi – generalizare pentru un *n* impar, nu neapărat prim
- Fie descompunerea lui *n* în factori primi: $n = \prod p_i^{\alpha_i}$ atunci avem:
- $\left(\frac{k}{n}\right) = \prod \left[\frac{k}{p_i}\right]^{\alpha_i}$

- Dacă *n* este prim, atunci simbolul Jacobi este egal cu simbolul Legendre.

The Solovay-Strassen Test

- Simbolul Jacobi – generalizare pentru un *n* impar, nu neapărat prim
- Fie descompunerea lui *n* în factori primi: $n = \prod p_i^{\alpha_i}$ atunci avem:
- $\left(\frac{k}{n}\right) = \prod \left[\frac{k}{p_i}\right]^{\alpha_i}$
- Dacă *n* este prim, atunci simbolul Jacobi este egal cu simbolul Legendre.

n \ k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1																
3	0	1	−1														
5	0	1	−1	−1	1												
7	0	1	1	−1	1	−1	−1										
9	0	1	1	0	1	1	0	1	1								
11	0	1	−1	1	1	1	−1	−1	−1	1	−1						
13	0	1	−1	1	1	−1	−1	−1	−1	1	1	−1	1				
15	0	1	1	0	1	0	0	−1	1	0	0	−1	0	−1	−1		
17	0	1	1	−1	1	−1	−1	−1	1	1	−1	−1	−1	1	−1	1	1

The Solovay-Strassen Test

Simbolul Jacobi;

- Proprietăți:
- $\left(\frac{2}{n}\right) = \begin{cases} 1 \mid n \equiv \pm 1 \ mod\ 8 \\ -1 \mid n \equiv \ \pm 3 \ mod \\ 0 \mid altfel \end{cases}$

The Solovay-Strassen Test

Simbolul Jacobi;

- Proprietăți:
- $\left(\frac{2}{n}\right) = \begin{cases} 1 \mid n \equiv \pm 1 \; mod \; 8 \\ -1 \mid n \equiv \; \pm 3 \; mod \\ 0 \mid altfel \end{cases}$

- $\left(\frac{k}{n}\right) = \left(\frac{n}{k}\right) * (-1)^{\left(\frac{(k-1)(n-1)}{4}\right)}$ pentru *k* și *n* impare

The Solovay-Strassen Test

- Criteriul lui Euler – pentru orice număr prim impar *p* și un număr întreg impar *b*, avem

$$\left(\frac{b}{p}\right) \equiv b^{\frac{p-1}{2}} \, mod \, p$$

The Solovay-Strassen Test

- Criteriul lui Euler – pentru orice număr prim impar *p* și un număr întreg impar *b*, avem

$$\left(\frac{b}{p}\right) \equiv b^{\frac{p-1}{2}} \, mod \, p$$

- Dacă *p* **nu este prim** iar *b* **verifică criteriul lui Euler** atunci ***b* este numit Euler Liar pentru *p***
- Dacă *p* nu este prim, va exista cel puțin un element *b* din mulțimea {1, 2, ... p-1} care să nu verifice criteriul lui Euler
- Ba mai mult, numărul “mincinoșilor” va fi <50%

The Solovay-Strassen Test

Solovay–Strassen (n)
$alegem\ \ 'a' aleator\ din\ \{1,2,\ldots,n-1\}$
$if\ cmmdc(a,n) > 1\text{:}\ return$ "nu e prim"
$if\ \left(\frac{a}{n}\right) \neq a^{\frac{n-1}{2}}\ :\ return$ "nu e prim"
$return\ "prim"$

The Solovay-Strassen Test

Solovay–Strassen (n)
$alegem\ 'a'aleator\ din\ \{1,2,\dots,n-1\}$
$if\ cmmdc(a,n) > 1: return$ "nu e prim"
$if\ \left(\frac{a}{n}\right) \neq a^{\frac{n-1}{2}}: return$ "nu e prim"
$return\ "prim"$

50% șanse de ”false positive” – *a* să fie Eulerian Liar

Complexitate: $\boldsymbol{O(log^3 n)}$ per iterație

Slide 32 relatia de recurenta worst case pt basic quicksort:

$$T(n) = T(0) + T(n-1) + \theta(n)$$

$$= \theta(1) + T(n-1) + \theta(n)$$

$$= \theta\left(n^2\right)$$

“Paranoid Quicksort” slide 40:

Se tot alege un pivot pana cand se nimerește să alegem un pivot “bun”
Ce este un pivot “bun”?

“Paranoid Quicksort” slide 40:

Se tot alege un pivot pana cand se nimerește să alegem un pivot “bun”
Ce este un pivot “bun”?

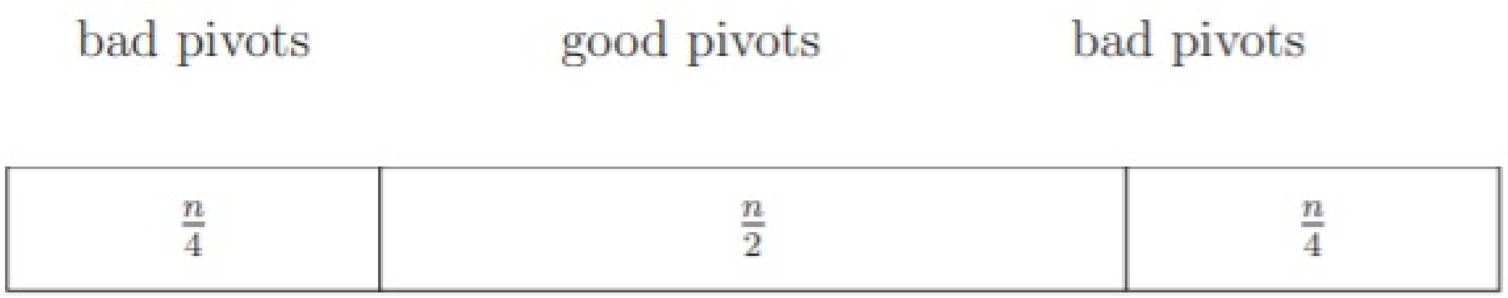
Acela pentru care partițiile L si G nu depășesc (¾)*n.

“Paranoid Quicksort” slide 40:

Se tot alege un pivot pana cand se nimerește să alegem un pivot “bun”
Ce este un pivot “bun”?

Acela pentru care partițiile L si G nu depășesc (¾)*n.

Un pivot slab este acela pentru care ori L, ori G depășește ca dimensiune valoarea ¾ *n



“Paranoid Quicksort” slide 40:

Alegând aleator un pivot, ce probabilitate este ca acel pivot ales sa fie “bun”?

“Paranoid Quicksort” slide 40:

Alegând aleator un pivot, ce probabilitate este ca acel pivot ales sa fie “bun”?

½

“Paranoid Quicksort” slide 40:

Alegând aleator un pivot, ce probabilitate este ca acel pivot ales sa fie “bun”?

½

Daca la fiecare pas al quicksort trebuie sa repet alegerea unui pivot pana când nimeresc unul bun, in medie cate selecții trebuie făcute?

“Paranoid Quicksort” slide 40:

Alegând aleator un pivot, ce probabilitate este ca acel pivot ales sa fie “bun”?

½

Daca la fiecare pas al quicksort trebuie sa repet alegerea unui pivot pana când nimeresc unul bun, in medie cate selecții trebuie făcute?

2 selecții

“Paranoid Quicksort” slide 40:

Alegând aleator un pivot, ce probabilitate este ca acel pivot ales sa fie “bun”?

½

Daca la fiecare pas al quicksort trebuie sa repet alegerea unui pivot pana când nimeresc unul bun, in medie cate selecții trebuie făcute?

2 selecții

Fie T(n) un upper bound pentru numărul de pasi necesari in paranoid quicksort.
T(n) este compus din:

“Paranoid Quicksort” slide 40:

Alegând aleator un pivot, ce probabilitate este ca acel pivot ales sa fie “bun”?

½

Daca la fiecare pas al quicksort trebuie sa repet alegerea unui pivot pana când nimeresc unul bun, in medie cate selecții trebuie făcute?

2 selecții

Fie T(n) un upper bound pentru numărul de pasi necesari in paranoid quicksort.
T(n) este compus din:

- numărul de pași necesar pentru a sorta partitia L

“Paranoid Quicksort” slide 40:

Alegând aleator un pivot, ce probabilitate este ca acel pivot ales sa fie “bun”?

½

Daca la fiecare pas al quicksort trebuie sa repet alegerea unui pivot pana când nimeresc unul bun, in medie cate selecții trebuie făcute?

2 selecții

Fie T(n) un upper bound pentru numărul de pasi necesari in paranoid quicksort.
T(n) este compus din:

- numărul de pași necesar pentru a sorta partitia L
- numărul de pași necesar pentru a sorta partitia G

“Paranoid Quicksort” slide 40:

Alegând aleator un pivot, ce probabilitate este ca acel pivot ales sa fie “bun”?

½

Daca la fiecare pas al quicksort trebuie sa repet alegerea unui pivot pana când nimeresc unul bun, in medie cate selecții trebuie făcute?

2 selecții

Fie T(n) un upper bound pentru numărul de pasi necesari in paranoid quicksort.
T(n) este compus din:

- numărul de pași necesar pentru a sorta partitia L
- numărul de pași necesar pentru a sorta partitia G
- numărul de iterații necesar pentru alegerea pivotului si partiționarea finala după un pivot “bun” (nr de iterații)*c*n

“Paranoid Quicksort” slide 40:

Fie T(n) un upper bound pentru numărul de pasi necesari in paranoid quicksort.
T(n) este compus din:

- numărul de pași necesar pentru a sorta partitia L
- numărul de pași necesar pentru a sorta partitia G
- numărul de iterații necesar pentru alegerea pivotului si partiționarea finala după un pivot “bun” (nr de iterații)*c*n

$$T(n) \le \max_{\frac{n}{4} \le i \le \frac{3}{4}n} (T(i) + T(n-i)) + (nt\ de\ iteratii\ pt\ alegerea\ pivotului) \cdot cn$$

$$nr\ iteratii\ = 2$$

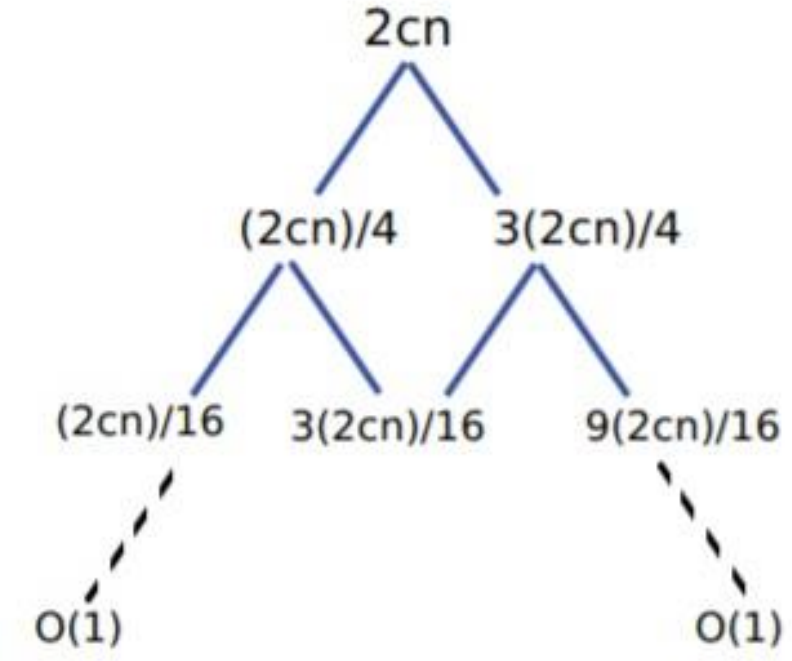
$$T(n) \le T\left(\frac{n}{4}\right) + T\left(\frac{3}{4}n\right) + 2cn$$

“Paranoid Quicksort” slide 40:

$$T(n) \le \max_{\frac{n}{4} \le i \le \frac{3}{4}n} (T(i) + T(n-i)) + (nt\ de\ iteratii\ pt\ alegerea\ pivotului) \cdot cn$$

$$nr\ iteratii = 2$$

$$T(n) \le T\left(\frac{n}{4}\right) + T\left(\frac{3}{4}n\right) + 2cn$$



Înălțimea arborelui de derivare nu poate fi mai mult decât $log_{4/3}(2cn)$.

$$T(n) = \theta(n \log n)$$