Algoritmi Avansați 2023

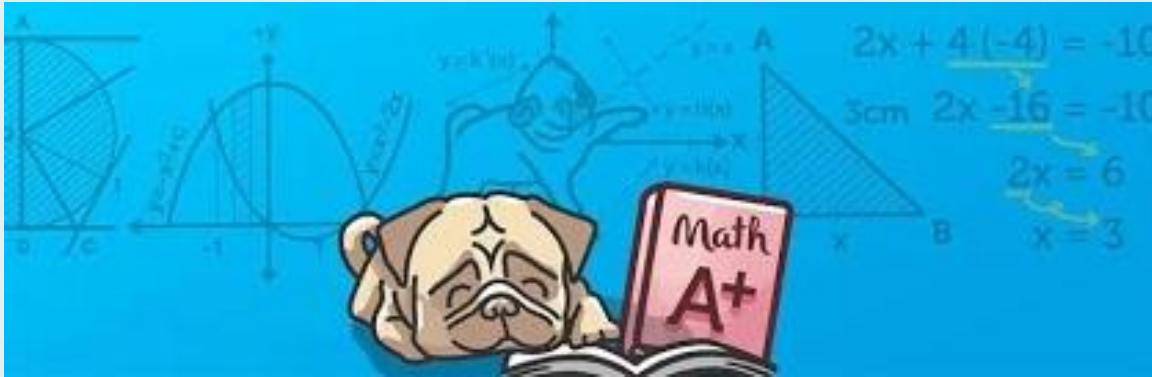
c-12

Randomized Algorithms

Lect. Dr. Ștefan Popescu

Email: stefan.popescu@fmi.unibuc.ro

Grup Teams:



Cuprins

Descriere

Probleme:

- Check Matrix multiplication
- Quicksort

Algoritmi probabilisti

- Ce sunt?

Algoritmi probabilisti

- Ce sunt algoritmii probabilisti?

  Orice algoritm care generează aleator un element $r\in\{1,2,...,R\}$ si efectuează decizii în funcție de valoarea acestuia

Algoritmi probabilisti

- Ce sunt algoritmii probabilisti?

  Orice algoritm care generează aleator un element $r \in \{1,2,...,R\}$ si efectuează decizii în funcție de valoarea acestuia

Un astfel de algoritm poate rula un număr diferit de pași și poate oferi output-uri diferite pe aceeași intrare. Astfel devine relevant sa avem mai multe iterații ale algoritmului pe un același input!

Algoritmi probabilisti

Algoritmii probabilisti pot fi impartiti in 2 (sau 3) clase:

Algoritmi probabilisti

Algoritmii probabilisti pot fi impartiti in 2 (sau 3) clase:

- Algoritmi Monte Carlo:
  - rulează în timp polinomial (rapid) și oferă un răspuns ”probabil” corect

Algoritmi probabilisti

Algoritmii probabilisti pot fi impartiti in 2 (sau 3) clase:

- Algoritmi Monte Carlo:
  - rulează în timp polinomial (rapid) și oferă un răspuns ”probabil” corect
- Algoritmi Las Vegas:
  - oferă mereu răspunsul corect în timp ”probabil” rapid

Algoritmi probabilisti

Algoritmii probabilisti pot fi impartiti in 2 (sau 3) clase:

- Algoritmi Monte Carlo:
  - rulează în timp polinomial (rapid) și oferă un răspuns ”probabil” corect
- Algoritmi Las Vegas:
  - oferă mereu răspunsul corect în timp ”probabil” rapid
- Algoritmi *Atlantic City*:
  - rulează în timp ”probabil” rapid și oferă un rezultat ”probabil” corect.

Algoritmi Monte Carlo

Exemplu de problemă

Algoritmi Monte Carlo

Matrix Multiplication:

Algoritmi Monte Carlo

Matrix Multiplication:

Fie A,B - două matrici pătratice de dimensiune *nxn*. Dorim să efectuăm calculul AxB.

Algoritmi Monte Carlo

Matrix Multiplication:

Fie A,B - două matrici pătratice de dimensiune *nxn*. Dorim să efectuăm calculul AxB.

Alternative:

Algoritmi Monte Carlo

Matrix Multiplication:

Fie A,B - două matrici pătratice de dimensiune *nxn*. Dorim să efectuăm calculul AxB.

Alternative:

- Implementare naivă. Complexitate ?

Algoritmi Monte Carlo

Matrix Multiplication:

Fie A,B - două matrici pătratice de dimensiune *nxn*. Dorim să efectuăm calculul AxB.

Alternative:

- Implementare naivă. Complexitate: $O(n^3)$

Algoritmi Monte Carlo

Matrix Multiplication:

Fie A,B - două matrici pătratice de dimensiune *nxn*. Dorim să efectuăm calculul AxB.

Alternative:

- Implementare naivă. Complexitate: $O(n^3)$
- Strassen (1969). $O(n^{\log 7})=O(n^{2,81})$

Algoritmi Monte Carlo

Matrix Multiplication:

Fie A,B - două matrici pătratice de dimensiune *nxn*. Dorim să efectuăm calculul AxB.

Alternative:

- Implementare naivă. Complexitate: $O(n^3)$
- Strassen (1969). $O(n^{\log 7})=O(n^{2,81})$
- Coppersmith-Winograd (1990). $O(n^{2,376})$

Algoritmi Monte Carlo

Matrix Multiplication Check

Fie A,B,C - trei matrici pătratice de dimensiune *nxn*. Dorim să verificăm dacă AxB=C

Algoritmi Monte Carlo

Matrix Multiplication Check

Fie A,B,C - trei matrici pătratice de dimensiune *nxn*. Dorim să verificăm dacă AxB=C

Se poate mai bine decât ”calea directă”?

Algoritmi Monte Carlo

Matrix Multiplication Check

Fie A,B,C - trei matrici pătratice de dimensiune *nxn*. Dorim să verificăm dacă AxB=C

Se poate mai bine decât ”calea directă”?

DA!

Monte Carlo: Frievald's Algorithm

Algoritm probabilist cu următoarele proprietăți:

Fie A, B, C - matricile din problemă.

- Dacă AxB=C, atunci algoritmul va returna întotdeauna *"DA"*
- Dacă AxB≠C, atunci algoritmul va returna *"NU"* cu o probabilitate >=1/2

Monte Carlo: Frievald’s Algorithm

Problemă: A,B,C - 3 matrici pătrate de dimensiune *nxn*; Trebuie să verificăm dacă AxB=C.

Monte Carlo: Frievald's Algorithm

Problemă: A,B,C - 3 matrici pătrate de dimensiune *nxn*; Trebuie să verificăm dacă AxB=C.

Soluție:

1. Generam un vector binar *r* de lungime *n* cu $Pr[r_i=1]=½$.
2. Dacă Ax(Br)=Cr, return "DA"
3. Altfel return "NU"

Monte Carlo: Frievald's Algorithm

Soluție:

1. Generam un vector binar *r* de lungime *n* cu $Pr[r_i=1]=½$.
2. Dacă $Ax(Br)=Cr$, return "DA"
3. Altfel return "NU"

Complexitate?

Monte Carlo: Frievald's Algorithm

Complexitate: $O(n^2)$

Soluție:

1. Generam un vector binar *r* de lungime *n* cu $Pr[r_i=1]=½$.
2. Dacă $Ax(Br)=Cr$, return "DA"
3. Altfel return "NU"

Monte Carlo: Frievald's Algorithm

Complexitate: $O(n^2)$

Soluție:

1. Generam un vector binar *r* de lungime *n* cu $Pr[r_i=1]=½$.
2. Dacă Ax(Br)=Cr, return "DA"
3. Altfel return "NU"

Observație: Dacă $AxB \neq C$, atunci $Pr[Ax(Br) \neq Cr] \geq 1/2$

Justificare Pt simplitate vom presupune ca matricile sunt binare (doar elemente de 0 si 1)

Algoritmi Las Vegas

Quicksort. (C.A.R. Hoare, Moscova, 1959)

Algoritmi Las Vegas

<u>Quicksort.</u> (C.A.R. Hoare, Moscova, 1959)

Algoritm Bazat pe strategia Divide-et-Impera

Primește ca input un șir A de elemente comparabile, returnează șirul A sortat.

Sortează oarecum asemănător ca sortarea prin inserție: la fiecare pas se fixează un element pe poziția sa.

Algoritmi Las Vegas

Quicksort. (C.A.R. Hoare, Moscova, 1959)

Pașii:

- Divide: se alege un element x din șirul A pe post de pivot. Se partiționează A în L (elementele <x), G (elementele >x) și E (elementele =x).
- Conquer: aplicăm recursiv sortarea pe șirurile L, respectiv G
- Combinare: ...

Algoritmi Las Vegas

<u>Basic Quicksort.</u>

1. Alegem pivotul x ca fiind fie A[1], fie A[n]
2. În mod repetat eliminăm fiecare element y din A
    a. inserăm y fie în L, G, sau E, în funcție de relația față de x

Fiecare inserție și ștergere durează O(1)

Partiționarea durează O(n)

Algoritmi Las Vegas

Basic Quicksort.

1. Alegem pivotul x ca fiind fie A[1], fie A[n]
2. În mod repetat eliminăm fiecare element y din A
    a. inserăm y fie în L, G, sau E, în funcție de relația față de x

Fiecare inserție și ștergere durează O(1)

Partiționarea durează O(n)

detalii în CLRS pag 171; Analiza algortimului:**Justificare** -$O(n^2)$

Algoritmi Las Vegas

Quicksort.

Q: Cum ne asigurăm ca găsim un pivot bun?

Algoritmi Las Vegas

<u>Quicksort.</u>

Q: Cum să asigurăm ca găsim un pivot bun?

A: Găsirea medianei!

Q: Timp?

Algoritmi Las Vegas

Quicksort.

Q: Cum să asigurăm ca găsim un pivot bun?

A: Găsirea medianei!

Q: Timp?

A: Găsirea medianei se face în timp asimptotic liniar!

Algoritmi Las Vegas

<u>Quicksort: Median selected as Pivot</u>

- Ne asigură faptul că L și G sunt mereu echilibrate ca mărime
- Analiză complexitate: -prima Θ(n) este din cauza selectiei medianei, iar a doua pentru pasul de partiție.
- Avem: $T(n) = 2T\left(\frac{n}{2}\right) + \theta(n) + \theta(n)$

$$T(n) = \theta\left(n \cdot \log_2 n\right)$$

Algoritmi Las Vegas

<u>**Quicksort: Median selected as Pivot**</u>

- Ne asigură faptul că L și G sunt mereu echilibrate ca mărime
- Analiză complexitate: -prima Θ(n) este din cauza selectiei medianei, iar a doua pentru pasul de partiție.
- Avem: $T(n) = 2T\left(\frac{n}{2}\right) + \theta(n) + \theta(n)$

$$T(n) = \theta\left(n \cdot \log_2 n\right)$$

In practică acest algoritm **<u>perfomează mai prost</u>** decât varianta Basic.

Algoritmi Las Vegas

Randomized Quicksort:

- la fiecare pas al recursiei, pivotul este ales aleator.
- Este echivalent cu varianta Basic.
- Detalii în CLRS pag 181-184

Algoritmi Las Vegas

<u>**Paranoid Quicksort:**</u>

1. Repetă:
    a. Alegem un pivot x aleator din A
    b. Partitionam A în L, G, E, în funcție de x
2. Până când partițiile rezultate sunt de forma:
    a. $|L| \leq 3/4|A|$ și $|G| \leq 3/4|A|$
3. Apelăm recursiv algoritmul pe L și G

Analiza algortimului: [Justificare]

Course notes in English