# Analiza Algoritmilor- Etapa3
# Range Minimum Query

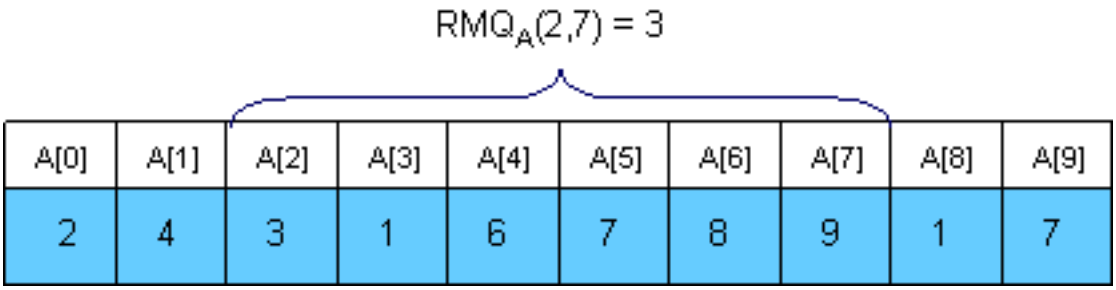
Ciorceanu Andrei Razvan - 324CA
Universitatea Politehnica din Bucuresti
Facultatea de Automatica si Calculatoare

15 noiembrie 2020

1. **Introducere**

   (a) *Descrierea problemei rezolvate*

   Problema rezolvata pe parcursul acestei teme va fi cea a gasirii elementului minim dintr-un interval.In domeniul de Computer Science, RMG(Range Minimum Query) rezolva problema gasirii unei valori minime dintr un sub-tablou al unui tablou de elemente comparabile.




Ilustrarea modului de functionare a unui algoritm de tip RMG

   (b) *Exemple de aplicatii practice pentru problema aleasa*

   Gasirea pozitiei elementului minim intr-un subtablou A[i..j] al unui tablou A de marime n este o operatie fundamentala in multe aplicatii cum ar fi traversarea unui arbore succint sau listarea tuturor elementelor distincte intr-un interval al unui tablou.Un alt exemplu din realitate ar fi determinarea celor mai mici timpi pentru stabilirea pole-position intr-o cursa de Formula 1.

(c) *Specificarea solutiilor alese*
In cazul in care sub-tabloul nu se modifica(nu sunt sterse sau adaugate elemente pe parcursul algoritmului),iar setul de date este cunoscut de dinaintea inceperii rezolvarii problema poate fi rezolvata cu o solutia aparent banala.Adaugarea sub-tabloului intr-o structura de date asigura rezolvarea rapida,iar solutia consta in parcurgerea fiecarui sub-tablou A si stocarea minimului intr-un alt tablou B,astfel incat B[i, j] = min(A[i...j]);

Segment Tree este o structura de date de tip heap care raspunde efectiv interogarilor de tip RMG si care permite in acelasi timp o flexibilitate,putand modifica tabloul.Segment Tree trebuie sa afle suma elementelor intre indicii l si r (ri=la[i] ) si de asemenea sa permita modificari ale elementelor din tablou (a[i]=x) fiind capabil sa proceseze fiecare interogare cu o complexitate de O(logn). Un principal avantaj al acestui algoritm este complexitatea buna,iar un dezavantaj ar fi complexitatea in ceea ce priveste implementarea si codul.

O alta metoda este Tournament Method in care impartim tabloul in 2 parti si comparam minimul si maximul din cele 2 parti pentru a afla minimul si maximul intregului tablou. Aceasta metoda se bazeaza pe principiul Divide et Impera iar un principal avantaj este faptul ca este usor de implementat avand in vedere partea de cod,un dezavantaj ar fi complexitatea destul de mare.

(d) *Specificarea criteriilor de evaluare alese pentru validarea solutiilor*

Fiecare algoritm de gasire a minimului dintr-un interval va fi evaluat in functie de comportamentul sau pe mai multe tipuri de seturi de date (numere intregi atat pozitive cat si negative),pentru a evidentia avantajele si dezavantajele in ceea ce priveste timpul de executie si precizia cu care vor afisa output-ul.In fisierul de input am generat teste cu numere alese la intamplare,ordonate crescator sau descrescator,teste in care un numar se repeta de mai multe ori,etc..

2. **Prezentarea solutiilor**

(a) *Descrierea modului in care functioneza algoritmii alesi, analiza complexitati solutiilor si prezentarea principalelor avantaje si dezavantaje pentru solutiile luate in considerare*

*Solutia banala*
In acest caz sub-tabloul nu se modifica(nu sunt sterse sau adaugate elemente pe parcursul algoritmului),iar setul de date este cunoscut de dinaintea inceperii rezolvarii problemei.Adaugarea sub-tabloului intr-o structura de date asigura rezolvarea rapida,iar solutia consta in parcurgerea fiecarui sub-tablou A si stocarea minimului intr-un alt tablou B,astfel incat B[i, j] = min(A[i...j]); RNG poate fi rezolvata foarte usor cu o simpla cautare in B,dar aceasta metoda nu este buna din punct de vedere al complexitatii,existand ($n^2$) interogari pentru un tabou de n elemente.

*Segment Tree*
Este o structura de data care raspunde efectiv interogarilor de tip RMG si care permite in acelasi timp o flexibilitate,putand modifica tabloul. Un exemplu ar fi: avem tabloul a[0...n1],Segment Tree trebuie sa afle suma elementelor intre indicii l si r (ri=la[i]) si de asemenea sa permita modificari ale elementelor din tablou (a[i]=x) fiind capabil sa proceseze fiecare interogare cu o complexitate de O(logn).Calculam si stocam suma elementelor din tot tabloul a[0...n1], apoi impartim acest tablou in 2 sub-tablouri de dimensiune egala a[0...n/2] si a[n/2+1...n1] pentru care calculam suma si o stocam.Vom repeta procedeul pentru cele 2 sub-intervale pana cand vom avea doar intervale cu un singur element.Putem organiza acest procedeu subforma unui arbore binar cu segmentul initial a[0...n1] ca radacina tocmai de aceea complexitatea acestui algoritm o sa fie O(logn). (1+2+4++2log2n=2log2n+1¡4n.) Un principal avantaj al acestui algoritm este complexitatea buna,iar un dezavantaj ar fi complexitatea in ceea ce priveste implementarea si codul.

*Tournament Method*
Impartim tabloul in 2 parti si comparam minimul si maximul din cele 2 parti pentru a afla minimul si maximul intregului tablou.Complexitatea algoritmului este O(n).Algoritmul se aseamana metodei Divide-et-Impera prin impartirea la fiecare pas in alte sub-intervale pentru care se pune aceeasi problema.Numarul de comparatii in cel mai bun caz este 3n/2-2 si in cel mai rau caz este 3n/2-1. Un principal avantaj este faptul ca este usor de implementat avand in vedere partea de cod,un dezavantaj ar fi complexitatea destul de mare.

3. **Evaluare**

   (a) *Descrierea modalitatii de construire a setului de teste folosite pentru validare*

Fiecare algoritm din cei prezentati mai sus va fi evaluat si folosit in functie de mai multe criterii de evaluare si teste care contin seturi de date diferite(alese random cat si personalizate).

-*Seturi generale de date*: numere alese random pentru a testa functionalitatea generala a algoritmilor.

-*Cazuri nefavorabile*: pentru a testa complexitatea si functionarea daca datele sunt nefavorabile(de exemplu numerele din tablou sunt puse intr o ordine crescatoare/descrescatoare).

-*Cazuri favorabile*: pentru a testa eficienta algoritmilor(de exemplu daca in tablou sunt putine numere).

-*Cazuri particulare*: aceste seturi vor fi implementate de noi si nu generate random ca la cele general(de exemplu daca in tablou sunt mai multe dubluri sau tabloul contine acelasi numar repetat de mai multe ori).

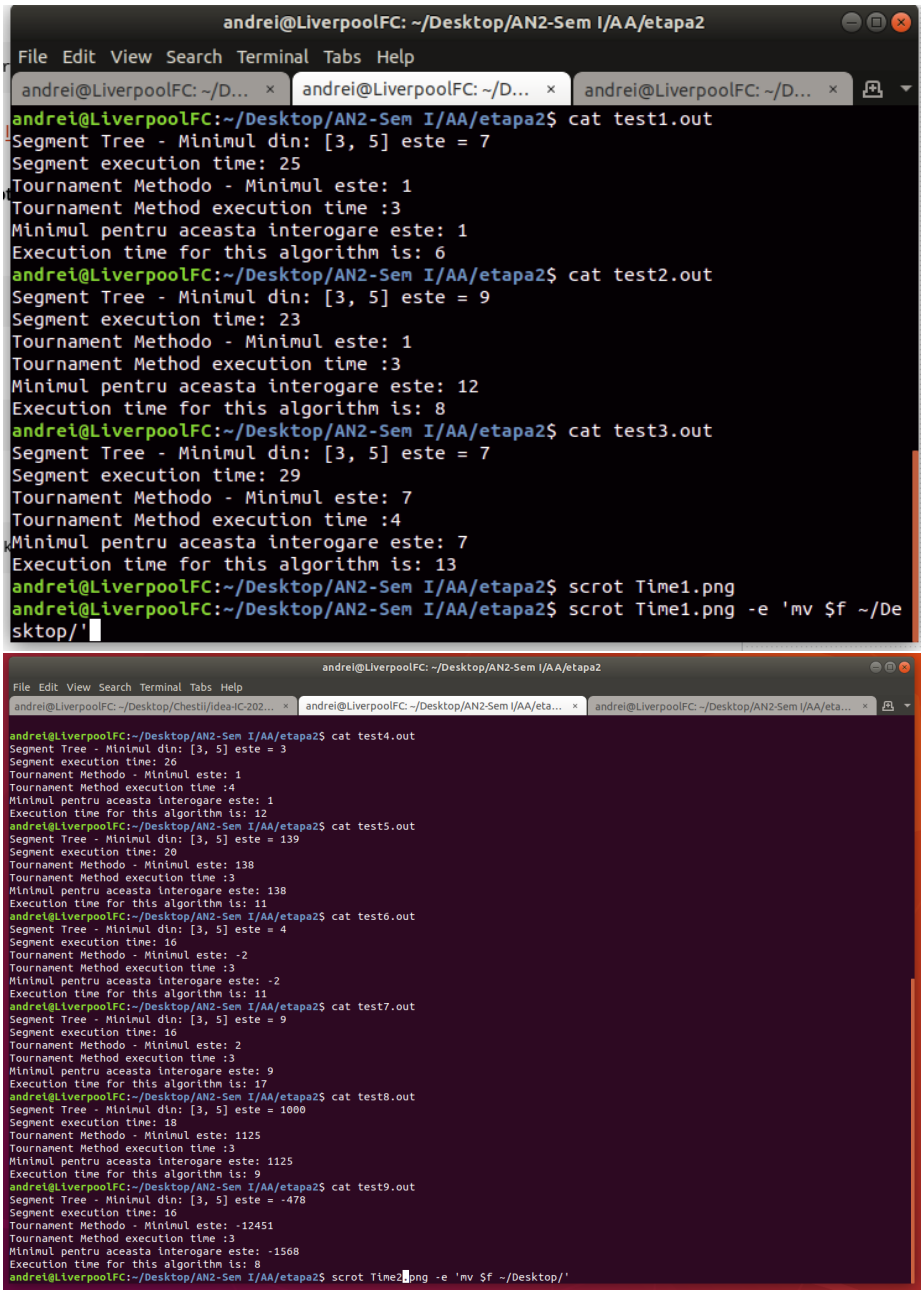
(b) *Mentionati specificatiile sistemului de calcul pe care ati rulat testele*

Testele au fost rulate pe un sistem de calcul care are urmatoarele specificatii:

Operating System:Linux, Ubuntu-18.04.1
Processor: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
Bios Memory: 10240 kB
System Memory:8068464 kB
CPU MHz:800.366

(c) *Ilustrarea folosind poze, a rezultatelor evaluarii solutiilor pe setul de teste*

**Execution Time**
Vom prezenta pozele cu timpii de executie a fiecarui test pe fieacre din cei 3 algoritmi si vom observa diferentele




(d) *Prezentarea valorilor obtinute pe teste*
Asa cum putea observa din cele 9 teste rulate,Tournament Method are un timp de executie de aprox. 3-4 milisecunde,valoarea timpului

de executie pentru algoritmul banal este de circa 7-9 milisecunde pe test, iar Segment Tree are cel mai mare timp de executie cu valori intre 16-26 milisecunde pe test. Chiar daca Segment Tree este mai util din punct de vedere al memoriei,acesta folosind un heap acesta are un timp de executie relativ mare. Ca urmare, pentru testele cu un tablou relativ mic Tournament Method este cel mai eficient algoritm din punct de vedere al timpului de executie.

4. **Concluzii**

In concluzie,dupa datele precizate mai sus cred ca Tournament Method este cel mai bun algortitm din punct de vedere al timpului de executie si al implementarii care este una relativ usoara,chair daca Segment Tree este mai util din punct de vedere al memoriei folisite acest lucru fiind vizibil pentru seturi cu foarte multe numere generate.

5. **Referinte relevante**


https://www.geeksforgeeks.org/maximum-and-minimum-in-an-array/
https://cp-algorithms.com/data$_s tructures/segment_t ree.html$
$https://en.wikipedia.org/wiki/Range_m inimum_q uery$
$https://www.quora.com/How-does-the-tournament-method-for-finding-the-maximum-and-minimum-element-in-an-array-consist-of-3n-2-2-comparisons$
$https://en.wikipedia.org/wiki/Rangeminimumquery$