Elementul minim dintr-un interval

CUPRINS

1.Introducere

Problema minimului dintr-un interval (Range minimum query), se refera la gasirea unei valori(int, float, string etc.) Minimul dintr-un interval al unui set de date / multime (un vector cu sau fara dublicate), in domeniu stiintific definitia se traduce in, cautarea intr-un sub-array dintr-un array de elemente comparabile, elementul minim al acestuia.

In general o problema de interogare a minimului pune la dispozitie 3 parametrii: vectorul, numarul de elemente si numarul de interogari, deci programul rezultat trebuie sa se bazeze pe aceste resurse. Din toate aceste lucruri putem deduce ca in orice caz de RMQ avem pus la dispozitie

2.La ce este folositor un algoritm RMQ?

La o prima privire acest subiect poate fi considerat "inutil", insa, in practica, daca ne documentam aflam ca reprezinta un utilitar care rezolva foarte multe probleme si multe aplicatii folosesc un algoritm RMQ, daca stam si ne gandim la ce putem sa aplicam acest utilitar raspunsul este ca il putem utiliza oriunde, noi lucram cu seturi de date si vrem sa construim o aplicatie care este foarte rapida si foarte "benefica", acum daca avem un set mare de date si vrem sa facem interogari pentru a aflat niste valori minime/maxime sau niste valori dupa anumite criterii incepem sa vedem utilitatea acestui "sablon" (deoarece RMQ este doar un tip de determinare),.

3.Avantaje

1. *Raspunde rapid la probleme de interogare.*
2. *Spatiul de alocare este egal cu spatiul vectorului (sau chiar vectorul).*

4.Dezavantaje

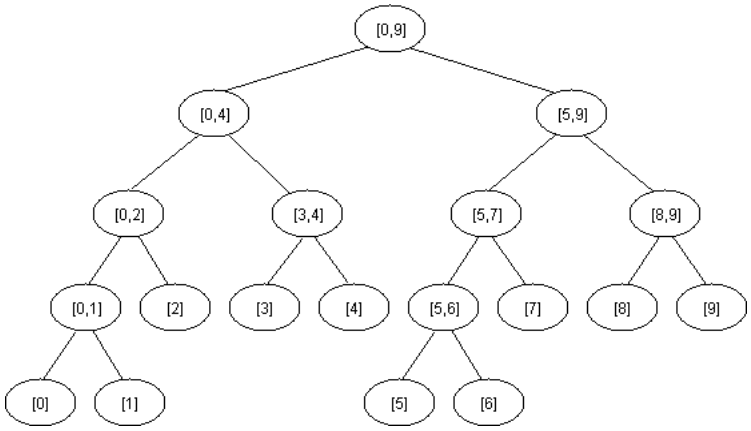
1. *Timp mare de preprocesare, cel putin O(n)*
2. *Daca se schimba valori din vectorul atunci trebuie sa facem de la capat preprocesarea (in majoritatea algoritmilor).*

5.Utilizari pentru RMQ

1. *Computing the lowest common ancestor in a tree.*
2. *Computing the longest common prefix in a string.*

In fiecare tabel se vor genera: timpulMediu de executie de preprocesare si de generare a minimului din intervalul [0-n] pentru un input intre [10-1000] cu scopul de a observa cresterea timpului in raport cu inputul acestuia.

6.SegmentTree



6.1.Descriere

1. *Segment tree reprezinta o structura de date de tip arbore folosit pentru stocarea de informatii ale unui interval sau segment de date, aceasta structura de date are doi pasi, un pas de preprocesare in care se prelucreaza informatiile (se construieste "arborele") si o parte de interogare care reprezinta generarea rezultatului pe un interval de interogare.*
2. *Un arbore segment este mai special deoarce raspunde foarte rapid pentru determinarea minimului pe un anumuit interval deoarce in preprocesare se aseaza la inceputul arborelui*

elementul care reprezinta minimul pe intervalul [0 – N], copilul din stanga va reprezenta valoarea minimala pentru segmentul [0 si (0-N/2)] adica [0 – N/2] iar cel din dreapta [N/2 – N].

3. *Avantajul pe care il ofera acest tip de asezare il constituie complexitatea pe care o are raspunsul la interogare, are o complexitatea O(log N), deoarece se parcurge arborele si se cauta intervalul pe care il dorim, cum se parcurge arborele, complexitatea este data de inaltimea acestuia iar inaltimea arborelui este mereu log N deci obtinem un raspuns foarte rapid.*

6.2.Preprocesare

1. *Preprocesre acestui algoritm se desfasoara in 2 etape, etapa in care se initializeaza vectorul pe care l-am folosit ca si suport cat si apelarea propriu zisa a metodei de preprocesare. Un utilizator are acces doar la metoda de initializare care se ocupa de intializarea structurii de date, aceasta metoda primeste un singur parametru, vectorul cu elemente pe care il punem la dispozitie. Aceasta etapa are o complexitate de O(n) deoarce se parcurge recursiv fiecare element din vector si se construieste vectorul de preprocesari.*

2. *Etapa de constructie buildHeap primeste ca parametrii vectorul de elemente, pozitia de start a preprocesarii, pozitia de final a preprocesarii si nodul in care ne aflam, la inceput va verifica daca pozitiile date ca parametrii sunt egale (daca s-a parcurs tot vectorul), in caz affirmativ – vectorul de preprocesari va pune nodul in care ne aflam pe pozitia de start sau final pentru ca oricum sunt egale; in caz contrar - se va calcula punctul mediu dintre cele doua puncte start si final si se va apela recursiv aceasi functie pentru copilul din dreapta al arborelui apoi pentru copilul din stanga al arborerului, care vor intoarce o valoarea dupa preprcesarea elementelor de sub nodul aferent, iar in vectorul de preprcesari in pozitia nodului curent se va pune valoarea cea mai mica dintre copilul stang si cel drept care reprezinta elementul cu valoarea cea mai mica pe intervalul respective.*

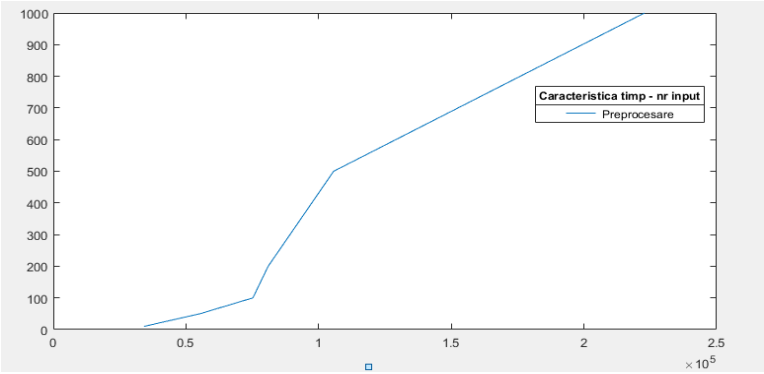
*6.3.*Interogarea

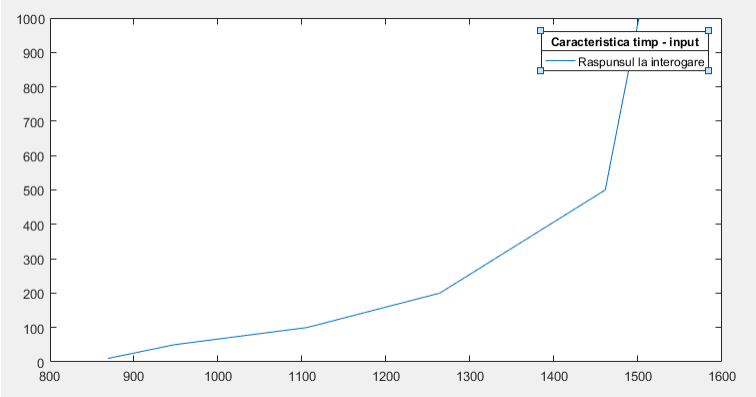
- *Interogarea, ca si preprocesarea se desfasoara in doua etape, prima etapa este reprezentata de o metoda la care are acces persoana care vrea sa afle minimul pe un interval, getRMQ care primeste ca prametru intervalul pe care vrem sa calculam minimul, un punct de start si unul de stop, algoritmul va verfica daca intervalul este unul valid, daca nu este un interval valid atunci va intoarce 0 si un mesaj de eroare. In cazul in care este un interval valid se va apela metoda*

buildRmq, scopul metodei este de a parcurge arborele si de a cauta intervalul pe care noi l-am cerut. Fiind o metoda recursiva care are o complexitate de O(log N), deoarece se parcurge inaltimea arborelui (aceasta inaltime nu poate sa depaseasca log N).

6.4.Testarea algoritmului

- In interiorul arhivei in fisierul SegmentaionTreeTimpMediu.txt se gasesc teste pentru determinarea timpului mediu pentru un vector de dimensiunea 10, 50, 100, 200, 500, 1000 la o singura interogare pe tot intervalul, cu urmatoarele date am construit doua grafice, unul cu timpul de preprocesare si altul cu timpul de interogare

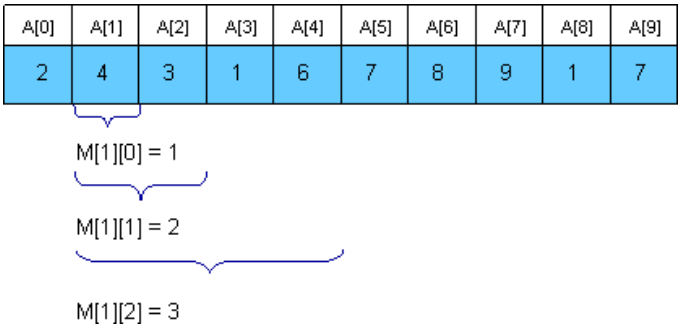




Size:	10	50	100	200	500	1000
Preprocesare	34212	55308	75220	80987	105649	222815
Raspuns	869	948	1106	1264	1461	1501

In urma testelor se oberva ca timpul de executie este proportional cu N * log N la constructie si log N la preprocesare. Amplificarea de timp este data din cauza folosirii listelor.

7.SparseTable



7.1.Descriere

1. *Sparse Table este un algoritm de tip RMQ pentru un vector de marimea* 2^n.
2. *Se foloseste o matrice ajutatoare de NxlogN/ NxN.*
3. *Elementul de pe pozitia (i,j) reprezinta minimul pentru intervalul* $(i, 2^j)$.
4. *Algortimul de desfasoara in 2 etape, o etapa de preprocesare si o alta etapa de interogare.*
5. *Beneficiile pe care le are acest algoritm o are partea de interogare care are un raspuns constant in O(1). In implementarea pe care am abordat-o nu are nu timp constant, deoarce am folosit liste, iar adaugarea si eliminarea elementelor din lista*

nu are un timp constant; facand abstractie de acest lucru se observa o usoara modificare a timpului de executie din cauza parcurgerii listei, dar in acelasi mod se oberva ca timpul de executie nu difera foarte mult de la o interogare la alta si se poate deduce ca timpul executarii unei interogari, fara lista, este unul constant.

7.2.Preprocesare

1. *La primul pas se initializeaza matricea, iar mai apoi se scrie in prima celula a fiecarui rand numarul randului - care reprezinta valoarea pentru intervalele de dimensiune 1.*

2. *In pasul urmator primul for reprezinta preprocesarea datelor de la intervale mici spre intervale mari pentru o rapiditate in determinarea solutiei, iar al doilea for este necesar pentru determinarea minimului fiecarui interval cu dimensiunea 2^k, unde k reprezinta numarul de iteratie din for, in interiorul acestuia se verifica daca elementul din vectorul de pe poztia elementului din matricea i j este mai mic fata de fata de ultimul element din sirul 2^k, in caz afirmativ elementul din matricea((ij)) va deveni elementul din vector, in caz contrar celalalt element se va pune in matricea (i,j), in final elementul din matricea(i,j) va contine mimimul pentru intervalul (i, 2^j)) al vectorului.*

3. *Complexitatea preprocesarii la prima vedere este O(N^2)), deoarece se fac doua bucle, dar analizand interiorul buclelor obsevam ca outputul se va scrie si parcurge in forma triunghiulara in interiorul matricei fapt ce reduce complexitatea de la O(N^2)) la O(N*logN).*

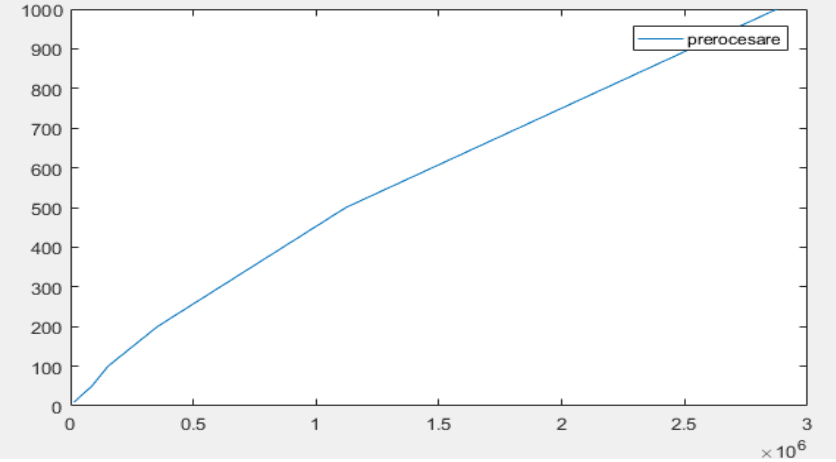
*7.3.*Interogarea

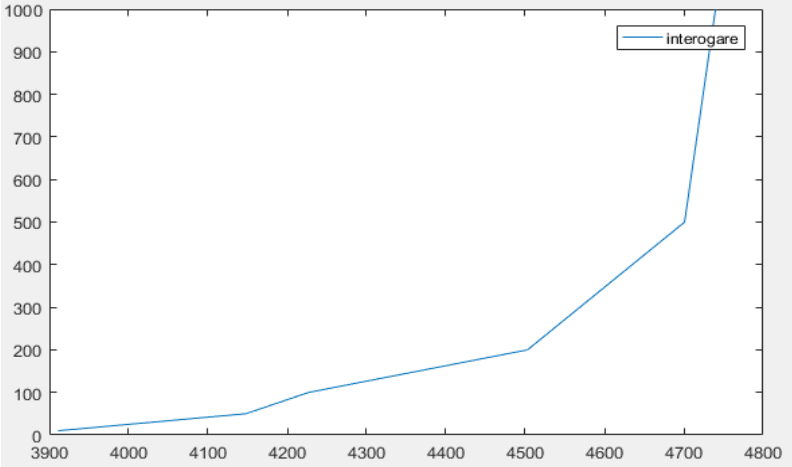
1. *Avantajul acestei metode o constituie timpul de raspuns, O(1), cel mai rapid raspuns pe care il putem obtine, din cauza preprocesarii, am asezat elementele din marice intr-un mod convenabil noua care permite interogarea foarte rapida pentru un interval bine determinat.*

2. *In interiorul metodei se calculeaza log2(SfarsitulIntervalului – InceputulIntervalului + 1)) pentru a determina locatia minimului in interiorul matricii, apoi se verifica daca elementul din matricea(StartRange,log2) este mai mic decat capatul tabelei; daca da atunci se va intorce primul element, in caz contrar, al doilea element. Metoda are o complexitate constanta in O(1), deoarce in interiorul ei nu se parcurg bucle si nici nu se desfasoara alte preprcesarii.*

7.4.Testarea algoritmului

- *In interiorul arhivei in fisierul SparseTableTimpMediu.txt se gasesc teste pentru determinarea timpului mediu pentru un vector de dimensiunea 10, 50, 100, 200, 500, 1000 la o singura interogare pe tot intervalul, cu urmatoarele date am construit doua graffice, unul cu timpul de preprocesare si altul cu timpul de interogare.*

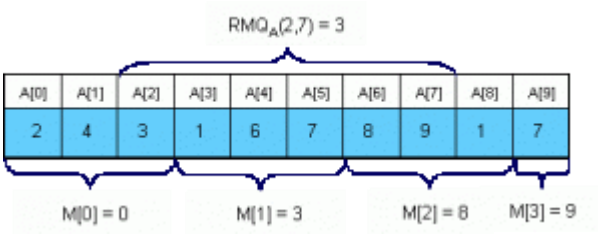




Size:	10	50	100	200	500	1000
Preprocesare	15328	86953	152731	353936	1120950	2875185
Raspuns	3911	4148	4227	4503	4701	4740

In urma testelor se oberva ca timpul de executie este proportional cu N * log N la constructie si $constant$ la preprocesare. Amplificarea de timp este data din cauza folosirii listelor.

8.Split and Query



8.1.Descriere

1. *Acest algoritm presupune impartirea vectorului cu elemente in $\sqrt{N}$ parti de dimensiunea $\sqrt{N}$, apoi stocarea indicelui la care se afla valoarea minima din intervalul respectiv intr-un alt vector, tot acest proces reprezinta faza de preprocesare, apoi pentru determinarea minimului vom parcurge acel vector pe care l-am construit si vom determina minimul pe intervalul pe care il dorim in functie de indicii din vectorul rezultat.*

2. *Beneficiul acestui algoritm este ca etapa de preprocesare dureaza foarte putin (se parcurge vectorul, fapt ce duce la o complexitate de O(N)), iar in etapa de determinare a minimului se parcurge vecotrul pe care l-am construit, iar acel vector are dimensiunea $\sqrt{N}$, fapt ce duce la o complexitate de O($\sqrt{N}$)).*

*8.2.*Preprocesare

1. *In etapa de preprocesare se imparte vectorul in in $\sqrt{N}$ parti de dimensiunea $\sqrt{N}$, incepe etapa de determinare a indicelui cu elementul cel mai mic din interiorul partii respective, in implementare acest lucru se realizeaza in for-ul cu numarul doi care va parcurge partea respective, cea in care vrem sa calculam minimul, iar mai apoi se va trece la urmatoarea parte, adica se va incrementa i-ul din primul for pentru a trece la urmatoarea parte, la sfarsitul algoritmului vom avea un vector de diminesiunea $\sqrt{N}$.*
2. *Complexitatea este data de dimensiunea vectorului deoarce se parcurge vectorul avem o complexitate O(N).*

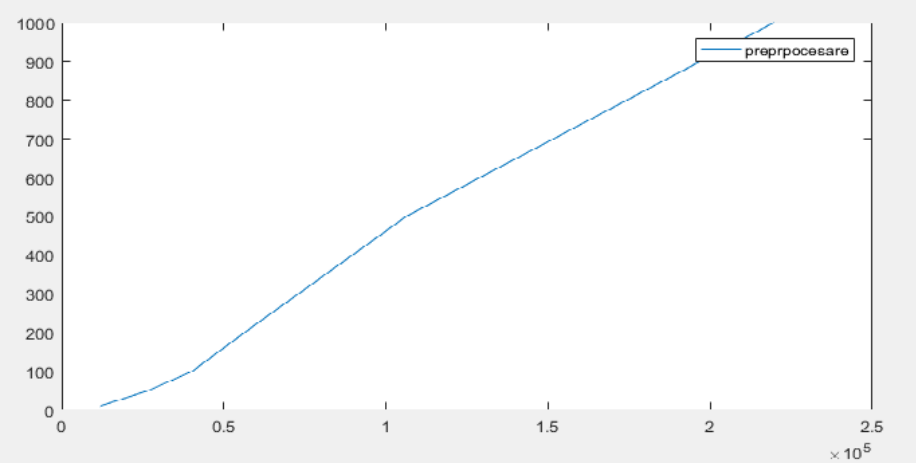
8.3.Interogarea

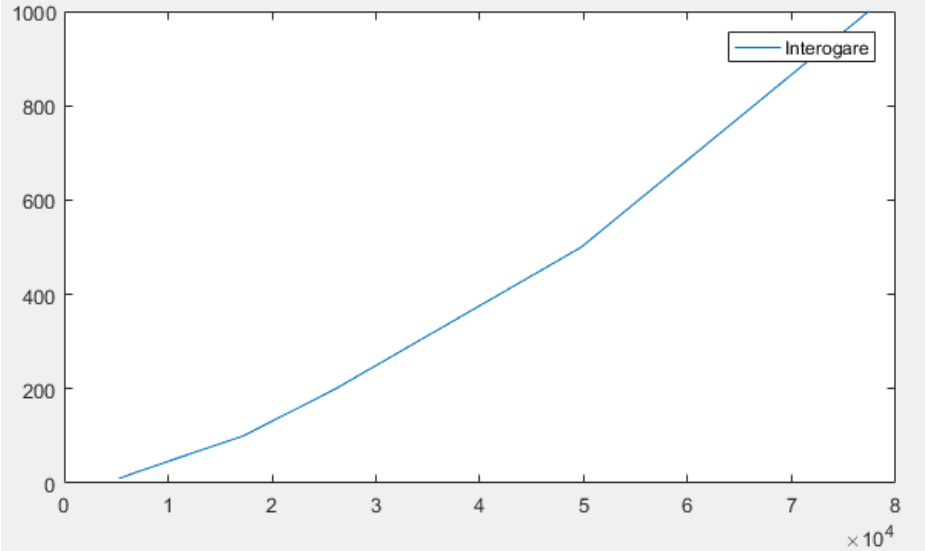
1. *La etapa de interogare se parcurge vectorul pe care l-am preprocesat de dimensiune $\sqrt{N}$, la pimul pas se afla unde este salvat indincele cu valoarea minima prin impartirea valorii de start la $\sqrt{N}$ la fel si punctul de final, apoi se parcurge intervalul intre punctul de start si punctul de start / $\sqrt{N}$ si se determina minimul pe intervalul acela, iar apoi pe cel dintre start / $\sqrt{N}$ si punctul de final / $\sqrt{N}$ si se alege cel mai mic minim dintre cele doua intervale. Acest lucru se face, deoarece intervalul de interogare poate sa cuprinda mai multe sub-intrevale preprocesate, deci se afla cel mai mic minim din intervalele pe care noi le avem deja.*
2. *Complexitatea acestui algoritm este O($\sqrt{N}$), deoarce in interiorul metodei se parcurge o distanta de maxim $\sqrt{N}$ in urma preprocesarii, in felul acesta putem spune ca aveam o complexitate maxima (Worst case) care este data de lungimea vectorului preprocesat.*

8.4.Testarea algoritmului

1. *In interiorul arhivei in fisierul SplitAndQueryTimpMediu.txt se gasesc teste pentru determinarea timpului mediu pentru un vector de dimensiunea 10, 50, 100, 200, 500, 1000 la o singura interogare pe tot intervalul, cu urmatoarele date am construit doua grafice, unul cu timpul de preprocesare si altul cu timpul de interogare.*

2. *DIN CAUZA LISTELOR TIMPUL ESTE DILATAT, DECI REZULTA UN TIMP MAI MARE DE RASPUNS, DAR AM FOLOSIT LISTE IN TOATE IMPLEMENTARILE DECI LA FINAL CIRTERIILE DE DECIZIE VOR FI ACELEASI.*





Size:	10	50	100	200	500	1000
Preprocesare	11891	26627	40256	56375	106311	219694
Raspuns	5253	10508	17224	26113	49738	77392

In urma testelor se oberva ca timpul de executie este proportional cu N la constructie si $\sqrt{N}$ la preprocesare. Aamplificarea de timp este data din cauza folosirii listelor.

*9.*Testarea pentru interogari = lungimea vectorului

- *In fisierul M ≡ N.txt din arhiva se gaseste testarea algoritmiilor la m interogari pentru o dimensiune de 10, 50, 100, 200, 500, 1000 elemente, am realizat cate 100 de teste pentru fiecare dimensiune si am reusit sa determin o valoare medie pentru timpul de executie, in interiorul fisierului se gaseste inputu.*

Dimensiune	10	50	100	200	500	1000
			SegmentTree			
Preprocesare	60031	289581	340149	483210	768002	824495
Interogare	6717	7506	7112	6716	6716	4741
			SparseTable			
Preprocesare	50568	138667	306174	586668	768102	844495
Interogare	3407	3506	3901	3321	3716	3136
			SplitAndQuery			
Preprocesare	20543	40691	68741	66766	102321	111013
Interogare	20543	22642	27654	25679	25037	24247

10.Concluzii

1. ***La capitolul de preprocesare cel mai bun algoritm este Split and Query avand cel mai bun timp de exectuie.***
2. ***La capitolul de interogare cel mai bun algoritm este SparseTable avand cel mai bun raspuns, acesta este aproape de un timp constat.***
3. ***La capitolul de reintorducere a elementelor in vector cel mai bun algoritm este Split and Query, deoarece preprocesarea acestuia este mult mai rapida si copenseaza mai apoi prin raspunsul la interogare.***

*11.*Referinte

[1] - https://www.geeksforgeeks.org/segment-tree-set-1-range-minimum-query/
[2] - https://mayanknatani.wordpress.com/2013/07/15/range-minimum-query/