Elementul minim dintr-un interval (RMQ)

Careja Alexandru-Cristian

Facultatea de Automatica si Calculatoare, Universitatea Politehnica Bucuresti

Splaiul Independentei 290, 060029 Bucuresti, Romania  
careja.alexandru@gmail.com

**Rezumat.** Fiind dat un vector cu N elemente intregi, RMQ cere elementul minim dintr-un subvector al vectorului initial. In acest studiu voi compara principalii algoritmi de rezolvare ai problemei gasirii elementului minim dintr-un interval. Voi explica algoritmii in cauza si voi da exemple de cateva interogari si rezultatul care ar trebui returnat. Dupa aceasta etapa, voi trece la implementarea efectiva a solutiilor (in Python) si verificarea acestora cu un set de teste care va pune la incercare eficienta si corectitudinea algoritmilor. Voi analiza complexitatea solutiilor si voi prezenta principalele avantaje si dezavantaje ale fiecareria. In urma analizei, voi incheia cu felul in care as aborda problema in practica si in ce situatii as opta pentru una din solutiile alese.

**Cuvinte cheie:** RMQ, LCA, interogare, interval, vector, matrice, arbore, complexitate, eficienta, asimptota, optim.

1. Introducere
   1. Cerinta

Dat fiind un vector A cu N elemente de tip intreg, trebuie raspuns eficient la intrebarea: “Care este elementul minim in intervalul care incepe la pozitia x si se termina la pozitia y?”. Se considera ca se da vectorul si apoi se fac M interogari fara a se modifica intre timp vectorul.

### Observatie: Voi considera pozitia 0 ca fiind pozitia primului element din vector

**Tabel 1.** Exemple de interogari RMQ

|  |  |  |
| --- | --- | --- |
| Vector input | X, Y | Elementul minim |
| 10, 2, 3, 49, 33, 2, 5, 78, 12, 11, 90 | **0,10** | **2** |
| 10, 2, 3, 49, 33, 2, 5, 78, 12, 11, 90 | **7, 10** | **11** |
| 1, 2, 3, 4, 5, 6, 54, 213, -4, 40, 9, 9 | **3, 11** | **-4** |
| 1001, 2002, 900, 1200, 6000, 555 | **0, 1** | **1001** |
| 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 ,13 ,14 | **3,3** | **4** |

* 1. Aplicatii practice

Range Minimum Query este un algoritm foarte des folosit pentru rezolvarea problemei LCA (Lowest Common Ancestor – gasirea celei mai mic stramos intr-un arbore, RMQ si LCA fiind doua probleme echivalente), dar nu numai. RMQ mai este folosit si in preprocesarea stringurilor si la suffix arrays, o noua structura de date care suporta cautari de stringuri aproape la fel de rapide ca suffix trees, dar care foloseste mai putina memorie si necesita mai putin efort de programare.

* 1. Solutii de rezolvare
* Sparse table
* Cartesian Tree & algoritmul Farach-Colton and Bender
* Segment Tree
  1. Evaluarea solutiilor

In vederea intocmirii unui set de teste, voi genera un set de date de intrare, cu diverse valori pentru N si M. Pentru fiecare fisier de intrare voi genera random N numere intregi si M indici x si y, cu conditia ca perechile de indici sa fie diferite pentru fiecare test.

Validarea corectitudinii o voi face prin compararea rezultatului generat de fiecare solutie analizata cu rezultatul obtinut prin metoda banala(cea care raspunde in O(n) la fiecare interogare), folosind o functie de comparare a fisierelor.

Eficienta solutiilor va fi evaluata pe de o parte, pe foaie prin calcularea complexitatii algoritmului, cat si pe calculator prin masurarea timpului de executie pe cateva fisiere de intrare mari, si prin compararea cu timpul algoritmului banal.

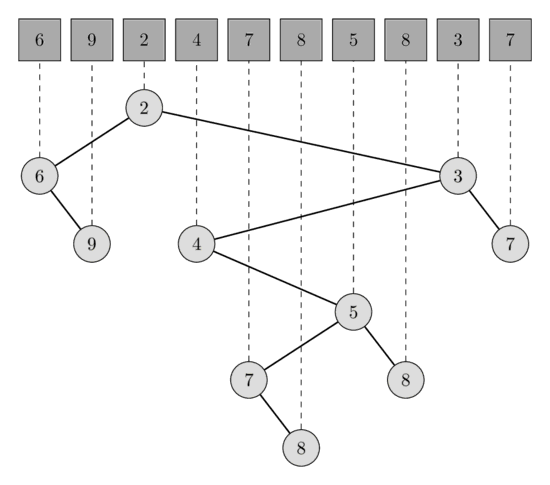
1. Prezentarea Solutiilor
   1. Sparse Table

Este foarte bine cunoscut faptul ca orice numar nenegativ poate fi scris intr-un mod unic ca o suma de puteri descrescatoare ale lui doi (de exemplu 13 = 1101 in baza 2, adica 13 = 8 + 4 + 1). Pentru un numar n, exista maximum logaritm in baza 2 din n termeni in suma. De asemenea, orice vector poate fi reprezentat in mod unic ca o reuniune de vectori de lungimi care sunt puteri descrescatoare ale lui doi (de exemplu [2: 23] = [2: 17] U [18: 21] U [22: 23], unde vectorul complet are lungimea de 22 si vectorii in care l-am descompus au lungimile 16, 4, 2). La uniunea vectorilor sunt maxim logaritm in baza 2 din n vectori. Principala idee din spatele metodei Sparse Table este sa precomputeze toate raspunsurile de RMQ posibile pentru fiecare vector de lungime egala cu o putere de-a lui doi. Raspunsul la o interogare diferita de cele precomputate pana acum se precomputeaza prin impartirea vectorului dat in vectori de lungime egala cu puteri ale lui doi, si uitandu-ne la raspunsurile deja precomputate si combinandu-le, ajungand la un raspuns final. In rezolvarea problemei RMQ prin metoda Sparse Table pornim de la ipoteza ca vectorul nu se va modifica de-a lungul seriei de interogari.

* 1. Cartesian Tree & algoritmul Farach-Colton & Bender

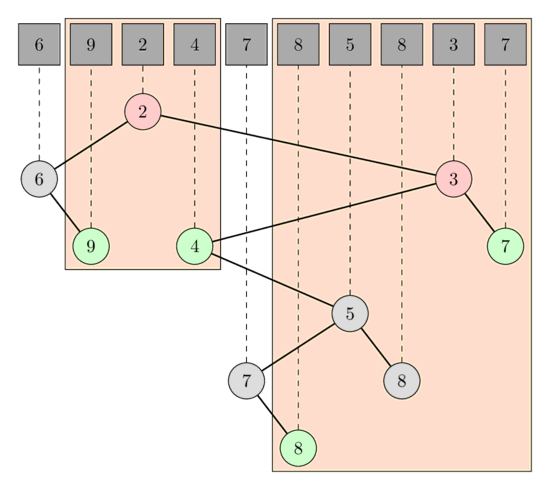
Pornim de la construirea unul arbore Cartezian din vectorul dat. Un arbore Cartezian este un arbore binar cu proprietatea de min-Heap (valoarea nodului parinte este mai mica decat decat valorile nodurilor copii ai sai) astfel incat parcurgerea inordine a arborelui sa viziteze nodurile in ordinea in care se gasesc in vectorul dat.

**Imagine 1.** Vector de 10 elemente si arborele Cartezian corespunzator



Valoarea minima din intervalul [l, r] este echivalenta cu cel mai mic stramos comun al nodului corespondent elementului de la indexul l si nodului corespondent elementului de la indexul r. In urmatoare imagine se poate vedea rezolvarea interogarii LCA (cel mai mic stramos comun) pentru inputul [1, 3] si [5, 9].

**Imagine 2.** Ilustrare a celui mai mic stramos comun



Construirea arborelui Cartezian se face adaugand elemente succesiv, avand in vedere ca dupa fiecare element adaugat sa pastram proprietatea de arbore Cartezian. Adaugarea un element A[i] poate modifica doar nodurile cele mai din dreapta incepand de la radacina si luand copilul din dreapta in mod repetat. Subarborele nodului cu cea mai mica valoare, dar mai mare sau egala decat elementul curent A[i] devine subarborele stang al nodului curent si arborele cu radacina A[i] va deveni noul subarbore drept al nodului cu cea mai mare valoare mai mica decat A[i].  
 Algoritmul Farach-Colton & Bender face turul Euler pe arborele Cartezian si il memoreaza intr-un vector E. Imparte vectorul E in blocuri de 0.5log in baza 2 din N, unde N este lungimea lui E. Pentru fiecare bloc, calculeaza elementul minim si il memoreaza intr-un vector M si construieste un Sparse Table pentru M.

Daca se da o interogare l, r si l si r sunt in blocuri diferite, atunci raspunsul va fi compus din sufixul blocului lui l, incepand de la el, prefixul blocului lui r, terminandu-se la r, si minimul dintre blocurile dintre l si r. Valorile din vectorul E difera cu o unitate una fata de cea de langa, astfel ca daca scadem din fiecare bloc valoarea primului element, vom avea o secventa formata numai din 1 si -1. Iar pentru ca blocurile sunt de o dimensiune atat de mica, numarul de secvente posibile este unul redus.

Referinte

1. Autori: Hao Yuan, Mikhail J. Atallah : Data Structures for Range Minimum Queries in Multidimensional (2014)
2. Forumul [www.geeksforgeeks.org](http://www.geeksforgeeks.org) , ultima accesare 13/11/2019
3. <https://cp-algorithms.com/sequences/rmq.html> , ultima accesare 13/11/2019
4. Blogul TopCoder [www.topcoder.com](http://www.topcoder.com) , ultima accesare 13/11/2019