Analiza Algoritmilor
Range Minimum Query

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

15 noiembrie 2020

1 Rezumat:

Proiectul se refera la analiza problemei gasirii elemtului minim dintr-un interval de la o pozitie x pana la o pozitie y, problema denumita si "Range Minimum Query" care va fi rezolvata cu ajutorul algoritmilor "Segment Tree" si "Tournament Method". Pe parcursul proiectului, voi redacta codul pentru acesti algoritmi, cât si testele necesare pentru a verifica eficienta acestora, pentru respectarea normelor de complexitate si adaptarea in cazul eventualelor schimbari din partea utilizatorului. La sfarsit voi trage concluziile daca algoritmii sunt eficienti si daca rezolva cat mai rapid si simplu problema propusa.

Keywords: -complexitate -minim -interval -Segment Tree -Tournament Method

2 Introducere:

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. Avand in vedere un tablou A [1 ... n] de n obiecte preluate dintr-un set bine ordonat(cum ar fi numere), interogarea minima a intervalului RMQ A (l, r) = arg min A [k] (cu 1 l k r n) returneaza pozitia elementului minim in sub-tabloul A specificat A [l ... r]. De exemplu: Pentru A = [0,5,2,5,4,3,1,6,3], raspunsul pentru valoarea minima din sub-tabloul A[3 . . . 8] = [2,5,4,3,1,6] este 7, as A[7] = 1.

3 Solutii:

- **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 subintervale pana cand vom avea doar intervale cu un singur element.Putem organiza acest procedeu sub forma 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 subintervale 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.

- **Criterii de evaluare si teste**: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 particulate*: 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).

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