TUGAS AKHIR - IF184802

PENERAPAN STRUKTUR DATA *SEGMENT TREE* PADA RANCANG ALGORITMA: STUDI KASUS *URI ONLINE JUDGE: RANGEL AND THE ARRAY GAME II*

ALDI FEBRIANSYAH
NRP 05111440000015

Dosen Pembimbing 1
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing 2
Rizky Januar Akbar S.Kom., M.Eng.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

Halaman ini sengaja dikosongkan

TUGAS AKHIR - IF184802

PENERAPAN STRUKTUR DATA *SEGMENT TREE* PADA RANCANG ALGORITMA: STUDI KASUS *URI ONLINE JUDGE: RANGEL AND THE ARRAY GAME II*

ALDI FEBRIANSYAH
NRP 05111440000015

Dosen Pembimbing 1
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing 2
Rizky Januar Akbar S.Kom., M.Eng.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

Halaman ini sengaja dikosongkan

UNDERGRADUATE THESES - IF184802

IMPLEMENTATION OF SEGMENT TREE DATA STRUCTURE ON ALGORITHM DESIGN ON: CASE STUDY IN URI ONLINE JUDGE PROBLEM ARRAY GAME II

ALDI FEBRIANSYAH
NRP 05111440000015

Supervisor 1
Rully Soelaiman, S.Kom., M.Kom.

Supervisor 2
Rizky Januar Akbar S.Kom., M.Eng.

INFORMATICS DEPARTMENT
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

Halaman ini sengaja dikosongkan

PENERAPAN STRUKTUR DATA SEGMENT TREE PADA RANCANG ALGORITMA: STUDI KASUS URI ONLINE JUDGE: RANGEL AND THE ARRAY GAME II

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Algoritma Pemrograman
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
Aldi Febriansyah
NRP : 05111440000015

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Rully Soelaiman S.Kom., M.Kom. .....
NIP: 19700213 199402 1 001 (Pembimbing 1)

Rizky Januar Akbar S.Kom., M.Eng. .....
NIP: 19870103 201404 1 001 (Pembimbing 2)

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
DEPARTEMEN TEKNIK INFORMATIKA

SURABAYA
JANUARI 2019

Halaman ini sengaja dikosongkan

PENERAPAN STRUKTUR DATA SEGMENT TREE PADA RANCANG ALGORITMA: STUDI KASUS URI ONLINE JUDGE: RANGEL AND THE ARRAY GAME II

Nama	: ALDI FEBRIANSYAH
NRP	: 05111440000015
Departemen	: Informatika FTIK-ITS
Pembimbing 1	: Rully Soelaiman, S.Kom., M.Kom.
Pembimbing 2	: Rizky Januar Akbar S.Kom., M.Eng.

ABSTRAK

Perkembangan teknologi informasi dalam beberapa dekade terakhir sangat pesat, terutama dalam hal proses komputasi yang memungkinkan pengambilan keputusan dapat dilakukan dengan cepat dan akurat. Salah satu bidang permasalahan yang menarik untuk dieksplorasi dalam bidang komputasi adalah query processing.

Query processing adalah suatu permasalahan untuk memproses beberapa query dari suatu data dengan indeks yang dinamis sesuai dengan kriteria masing-masing query. Tujuan dari query processing adalah menyelesaikan semua query dengan waktu yang efisien dan biaya komputasi yang rendah oleh karena itu diperlukan algoritma yang optimal untuk mencapai hal tersebut.

Topik tugas akhir ini mengangkat permasalahan yang terdapat pada situs penilaian daring URI dengan nomor permasalahan 2849 dan judul rangel and the array game II. Pada permasalahan ini diberikan sebuah array bilangan dengan panjang N dan lima bilangan l, r, k, g, d sebanyak Q

kali. Bilangan l dan r merepresentasikan indeks batas suatu rentang pencarian dalam array. Pada interval indeks l dan r diminta untuk mencari bilangan terkecil ke-K dan frekuensi bilangan tersebut muncul pada rentang. Pada permasalahan ini memiliki tantangan untuk merancang algoritma yang dapat melakukan pemrosesan tiap case secara optimal pada array yang dinamis karena array yang diproses untuk setiap case berbeda sesuai dengan indeks i dan j. Dengan batasan waktu selama 3 detik, nilai N antara 1 hingga 10^5 dan jumlah query sampai 10^5 maka apabila dikerjakan tanpa adanya perencanaan dan perancangan struktur data dan algoritma yang tepat akan menghasilkan status Time Limit Exeeded.

Pada tugas akhir ini diimplementasikan struktur data segment tree. Struktur data segment tree digunakan dengan memanfaatkan sifat dari struktur data segment tree, yaitu node parent akan menyimpan representasi nilai dari children nodenya. Hasil dari Tugas Akhir ini diharapkan dapat menjadi acuan dalam pemilihan, desain, dan implementasi struktur data yang tepat untuk memecahkan permasalahan di atas secara optimal dan diharapkan dapat berkontribusi pada perkembangan ilmu pengetahuan dan teknologi informasi kedepannya.

Kata Kunci: *segment tree, struktur data, range query*

IMPLEMENTATION OF SEGMENT TREE DATA STRUCTURE ON ALGORITHM DESIGN ON: CASE STUDY IN URI ONLINE JUDGE PROBLEM ARRAY GAME II

Name	: ALDI FEBRIANSYAH
NRP	: 05111440000015
Major	: Informatics Department Faculty of IC-ITS
Supervisor 1	: Rully Soelaiman, S.Kom., M.Kom.
Supervisor 2	: Rizky Januar Akbar S.Kom., M.Eng.

ABSTRACT

The development of information technology in recent decades is very rapid, especially in terms of computing processes that enable decision making to be done quickly and accurately. One of the most interesting issues to explore in computing is query processing.

Query processing is a problem for processing multiple queries from a data with a dynamic index according to the criteria of each query. The purpose of query processing is to solve all queries with efficient time and low computational cost and therefore an optimal algorithm is needed to achieve this.

This final project topic will use a problem in URI online judge website which has 2849 as its problem number and "Rangel and Array Game II" as its title. In this problem, an array of numbers with N length and Q times of five numbers l, r, k, g, d is given. The numbers l and r represent the indices of extreme of thee search interval in the array. On each search interval required to find the Kth smallest number and the frequency of said number in the interval. The challenge of this problem is to design an algorithm that optimally process each

query in dynamic array because the said processed array on every case has different i and j extreme. With 3 seconds time limit and the array's length is up to 10^5 and the number of queries is also up to 10^5. Thus, using if the problem is implemented without a well-planned and well-designed algorithm the result certainly is Time Limit Exceeded.

This Final Project will be implemented by using segment tree data structure. Segment tree data structure used by utilizing the segment tree's nature itself, which is every parent node will store representated value of its children. The results of this final project is expected to help determination, design, and implementation process of the correct data structure to optimally solve the problem mentioned above and contribute to development of science and information technology.

Keywords: *segment tree, struktur data, range query*

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT atas rezeki, berkah, kekuatan dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

PENERAPAN STRUKTUR DATA SEGMENT TREE PADA RANCANG ALGORITMA: STUDI KASUS URI ONLINE JUDGE: RANGEL AND THE ARRAY GAME II

Dengan selesainya Tugas Akhir ini diharapkan apa yang telah dikerjakan penulis dapat memberikan kontribusi bagi perkembangan ilmu pengetahuan terutama di bidang teknologi informasi serta bagi diri penulis sendiri selaku peneliti.

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan baik secara langsung maupun tidak langsung selama penulis mengerjakan Tugas Akhir maupun selama menempuh masa studi antara lain:

- Ibu Dyah Esthi Oetami dan bapak Mujayadi selaku orang tua penulis yang selalu memberikan dukungan dalam dari berbagai aspek dan menjadi penyemangat untuk segera mendapatkan gelar sarjana ini.
- Bapak Rully Soelaiman, S.Kom., M.Kom. selaku Dosen Pembimbing I yang telah banyak memberikan ilmu, pandangan, nasihat, motivasi, dan bimbingan selama penulis menempuh masa perkuliahan maupun selama pengerjaan Tugas Akhir ini.

- Bapak Rizky Januar Akbar S.Kom., M.Eng. selaku Dosen Pembimbing II yang telah memberikan bimbingan dan arahan dalam pengerjaan Tugas Akhir ini dan selama masal perkuliahan.
- Teman-teman angkatan 2014 yang menemani penulis selama masa perkuliahan.
- Seluruh dosen, karyawan, dan teknisi yang sudah memberikan ilmu dan mengisi hari penulis selama masa perkuliahan.

Penulis menyadari masih ada kekurangan pada Tugas Akhir ini sehingga Penulis mengharapkan kritik dan saran yang membangun untuk pembelajaran dan perbaikan supaya Tugas Akhir ini menjadi lebih baik. Semoga melalui Tugas Akhir ini penulis dapat memberikan manfaat untuk pembaca.

Surabaya, Januari 2018

DAFTAR ISI

DAFTAR TABEL

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

DAFTAR SUMBER KODE

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Pada bab ini akan dijelaskan latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan Tugas Akhir.

1.1 Latar Belakang

Di dalam ilmu komputer, terdapat permasalahan pada pengaplikasian tipe struktur data yang tepat guna memecahkan permasalahan yang ada. Hal ini dikarenakan banyaknya jenis struktur data dan kegunaannya yang beragam. Peneliti mencoba mencari beberapa permasalahan yang berkaitan dengan pengaplikasian struktur data dalam dunia pemrograman dan menemukan permasalahan di situs pemrograman online *URI Online Judge*. Di dalam situs tersebut terdapat sebuah permasalahan yang dikenal dengan nama *Rangel and the Array Game II* [1].

Pada permasalahan *Rangel and the Array Game II* diberikan input berupa bilangan integer yang disimbolkan N, hal tersebut akan merepresentasikan banyaknya data pada suatu *array A*. Secara singkat, permasalahan ini bercerita tentang dua tokoh bernama Gugu dan Dabriel yang berkompetisi untuk menebak frekuensi munculnya bilangan terkecil ke-K pada suatu rentang dalam *array A*. Permasalahan dalam pencarian data pada umumnya merupakan suatu permasalahan yang dapat diselesaikan cukup dengan struktur data *array*. Namun, tentu saja apabila pencarian data dilakukan dengan menggunakan struktur data *array*, data-data yang ada pada rentang yang diinginkan harus diurutkan terlebih

dahulu. Hal ini akan menyebabkan kompleksitas waktu penyelesaian menjadi O($n$log$n$) untuk setiap proses pencarian data, sehingga dapat menyebabkan *running time* menjadi cukup lama dan tidak dapat memenuhi kriteria penyelesaian permasalahan *Rangel and the Array Game II*, yaitu tiga detik untuk menyelesaikan setiap berkas masukan.

Setelah mencoba menggunakan algoritma *quickselect* dan mendapatkan hasil *time limit exceeded*, oleh karena itu penulis mencoba mengimplementasikan algoritma dan struktur data lain yang memiliki kompleksitas waktu penyelesaian cenderung lebih cepat pada saat proses pencarian data dalam suatu rentang yang diinginkan.

Segment tree merupakan jenis struktur data *tree* yang setiap *node*-nya menyimpan data rentang dari *child node*-nya, yang menyebabkan kompleksitas waktu pada setiap proses pencarian data hanya sebesar O($\log_2 n$). Sehingga dilakukan percobaan untuk menyelesaikan permasalahan menggunakan struktur data *Segment Tree*, dan membandingkan kinerjanya dengan algoritma *quickselect* dalam menyelesaikan problem *Rangel and the Array Game II*.

1.2 Rumusan Masalah

Perumusan masalah yang terdapat pada Tugas Akhir ini adalah:

1. Bagaimana menerapkan konsep dan merancang struktur data *Segment Tree* untuk menyelesaikan permasalahan *Rangel and the Array Game II* pada situs penilaian *URI Online Judge*?
2. Bagaimana perancangan dan pengimplementasian struktur data *Segment Tree* guna menyelesaikan permasalahan

Rangel and the Array Game II pada situs penilaian *URI Online Judge*?

3. Bagaimana hasil analisis dan pengujian terhadap kinerja struktur data Segment Tree dalam menyelesaikan permasalahan *Rangel and the Array Game II* pada situs penilaian *URI Online Judge*?

1.3 Batasan Masalah

Batasan masalah yang terdapat pada Tugas Akhir ini adalah sebagai berikut:

1. Implementasi algoritma menggunakan bahasa pemrograman C++.
2. Terdapat *array A* yang tidak terurut dengan banyak elemennya disimbolkan N, dengan syarat $\{N \mid 1 \leq N \leq 100.000, N \in \mathbf{Z}\}$.
3. Setiap elemen pada *array A* disimbolkan A_i, memiliki syarat $\{A_i \mid A_i \in \mathbf{Z}\}$.
4. Banyaknya kasus uji disimbolkan Q, dengan syarat $\{Q \mid 1 \leq Q \leq 100.000, Q \in \mathbf{Z}\}$.
5. Setiap kasus uji secara berurutan terdiri dari lima bilangan, yaitu batas kiri rentang dalam *array A*, batas kanan rentang dalam *array A*, sebuah bilangan K menyatakan bilangan terkecil ke-K yang dicari dalam rentang, tebakan Gugu, dan tebakan Dabriel.
6. Batas kiri disimbolkan dengan L dan batas kanan rentang disimbolkan dengan R, dengan syarat $\{(L, R) \mid 1 \leq L \leq R \leq N, L \in \mathbf{Z}, R \in \mathbf{Z}\}$.
7. Bilangan terkecil ke-K dalam rentang interval (A_L, A_R) yang disimbolkan dengan X, dan dipastikan X selalu ada pada rentang interval (A_L, A_R).

8. Bilangan tebakan Gugu disimbolkan dengan *G* dan Dabriel disimbolkan dengan *D*, yang memiliki syarat {(*G*, *D*) | $1 \leq G, D \leq 2^{32}-1$}
9. Batas waktu eksekusi sistem dalam untuk setiap berkas masukan harus < 3 detik.

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Melakukan desain konsep dan rancangan struktur data S*egment Tree* untuk menyelesaikan permasalahan *Rangel and the Array Game II* pada situs penilaian *URI Online Judge*.
2. Mengimplementasikan rancangan struktur data *Segment Tree* untuk menyelesaikan permasalahan *Rangel and the Array Game II* pada situs penilaian *URI Online Judge*.
3. Menganalisis hasil kinerja penyelesaian untuk permasalahan *Rangel and the Array Game II*.

1.5 Manfaat

Manfaat dari Tugas Akhir ini adalah sebagai berikut:

1. Mengetahui pemanfaatan penerapan struktur data *Segment Tree* dalam menyelesaikan suatu permasalahan yang ada.

1.6 Metodologi

Metodologi yang digunakan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal tugas akhir
   Tahap pertama dalam proses pengerjaan Tugas Akhir ini adalah menyusun proposal Tugas Akhir. Pada proposal

Tugas Akhir ini akan diajukan sebuah permasalahan struktur data pada di situs *URI Online Judge* terhadap permasalahan *Rangel and the Array Game II* yang akan diselesaikan dengan penggunaan algoritma yang berkaitan dengan permasalahan tersebut. Luaran dari penyusunan tahap ini adalah tersajinya proposal Tugas Akhir

2. Studi literatur
   Pada tahap pengerjaan Tugas Akhir ini, akan dilakukan studi mendalam terkait struktur data di C++, *segment tree*, dan algoritma-algoritma yang terkait dengan permasalahan di situs *URI Online Judge* terhadap permasalahan *Rangel and the Array Game II*. Adapun sumber studi yang dilakukan didapat dari buku-buku, papers, internet, serta materi perkuliahan yang terkait. Luaran dari studi literatur ini adalah tersajinya daftar pustaka dan referensi.
3. Implementasi Algoritma
   Implementasi algoritma adalah tahapan untuk membangun aplikasi yang digunakan untuk menyelesaikan permasalahan di situs *URI Online Judge* terhadap permasalahan *Rangel and the Array Game II*. Luaran dari tahapan ini adalah struktur data dan algoritma yang dibangun menggunakan bahasa pemrograman C++ dan menggunakan IDE Dev-C++ untuk penyelesaian kasus tersebut.
4. Pengujian dan evaluasi
   Tahap pengujian dan evaluasi akan dilakukan dengan menguji program yang telah dibuat di situs *URI Online Judge* terhadap permasalahan *Rangel and the Array Game II* hingga mendapatkan hasil *Accepted* dari *URI Online Judge*. Luaran dari tahapan ini adalah status *Accepted* dari *URI Online Judge*.
5. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan dan buku Tugas Akhir yang berisi dokumentasi hasil pengerjaan Tugas Akhir.

1.7 Sistematika Penulisan

Berikut adalah sistematika penulisan buku Tugas Akhir ini:

1. BAB I: PENDAHULUAN
   Bab ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi dan sistematika penulisan Tugas Akhir.
2. BAB II: DASAR TEORI
   Bab ini berisi dasar teori mengenai permasalahan dan algoritma penyelesaian yang digunakan dalam Tugas Akhir.
3. BAB III: DESAIN
   Bab ini berisi desain algoritma dan struktur data yang digunakan dalam penyelesaian permasalahan.
4. BAB IV: IMPLEMENTASI
   Bab ini berisi implementasi berdasarkan desain algortima yang telah dilakukan pada tahap desain.
5. BAB V: UJI COBA DAN EVALUASI
   Bab ini berisi uji coba dan evaluasi dari hasil implementasi yang telah dilakukan pada tahap implementasi.
6. BAB VI: PENUTUP
   Bab ini berisi kesimpulan yang didapat dari hasil uji coba yang telah dilakukan.

BAB II
DASAR TEORI

Pada bab ini akan dijelaskan mengenai dasar teori yang menjadi dasar pengerjaan Tugas Akhir ini

2.1 Deskripsi Umum Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini adalah permasalahan yang ditemukan pada situs *URI online judge* yang berjudul *Rangel and the Array Game II*. Permasalahan ini bertujuan untuk mencari *Kth smallest element* (elemen terkecil ke-K) dan mencari frekuensi kemunculan bilangan tersebut pada suatu rentang dalam array yang tidak terurut.

Secara singkat, deskripsi pada permasalahan ini menceritakan Gugu dan Dabriel yang berkompetisi untuk menebak frekuensi munculnya elemen terkecil ke-K pada suatu rentang pada array.

Tabel 2.1 Contoh masukan dari daring *URI Online Judge*

10	5								
1	4	5	2	7	4	5	8	10	1
1	10	1	3	1					
1	5	2	1	4					
2	6	3	1	1					
7	7	1	1	10					
3	8	4	10	4					

Masukan yang diberikan pada daring dimulai dengan masukan panjang *array* dan jumlah *query* pada setiap kasus uji, yang dilambangkan dengan *N* dan *Q*. Pada baris berikutnya masukan yang diberikan sejumlah *N* bilangan yang merupakan nilai dari

masing-masing elemen pada *array A*. Pada *Q* baris berikutnya adalah masukan yang melambangkan query yang diinginkan, pada tiap barisnya akan terdapat masukan *L*, *R*, *K*, *G*, dan *D*, yang secara berurutan adalah batas kiri rentang, batas kanan rentang, nilai *K* yang diinginkan, tebakan dari Gugu, dan tebakan dari Dabriel. Contoh masukan yang diberikan pada daring dapat dilihat pada Tabel 2.1.

Setiap *query* yang dimasukan, memiliki luaran berupa *X* dan *Y*, yang secara berurutan merupakan nilai elemen terkecil ke-K dan frekuensi elemen tersebut muncul dalam rentang, serta luaran yang menunjukan pemenang dari kompetisi antara Gugu dan Dabriel, yang dilambangkan dengan huruf "G" bila Gugu adalah pemenang, huruf "D" bila Dabriel adalah pemenang, dan huruf "E" bila tebakan Gugu dan Dabriel seri.

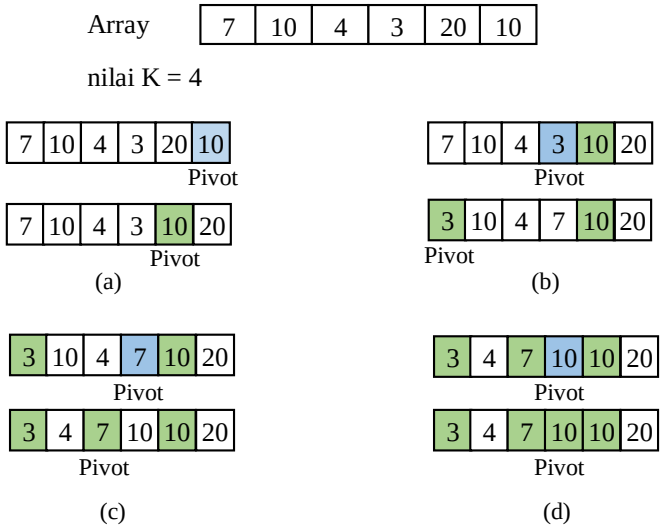
Selain menemukan bilangan yang menjadi elemen terkecil ke-K pada rentang yang ditentukan, penyelesaian yang dibuat untuk permasalahan ini harus menemukan frekuensi elemen yang bernilai sama dengan elemen terkecil ke-K tersebut. Setelah menemukan elemen terkecil ke-K dan frekuensi elemen tersebut, penyelesaian yang dibuat juga menentukan siapa pemenang antara Gugu dan Dabriel yang berkompetisi untuk menebak jumlah elemen yang sama dari bilangan terkecil ke-K tersebut. Contoh luaran pada daring yang sesuai dengan contoh masukan pada Tabel 2.2.

Tabel 2.2 Contoh luaran pada daring

1	2	E
2	1	G
4	2	E
5	1	G
5	2	D

2.2 Algoritma *Quickselect*

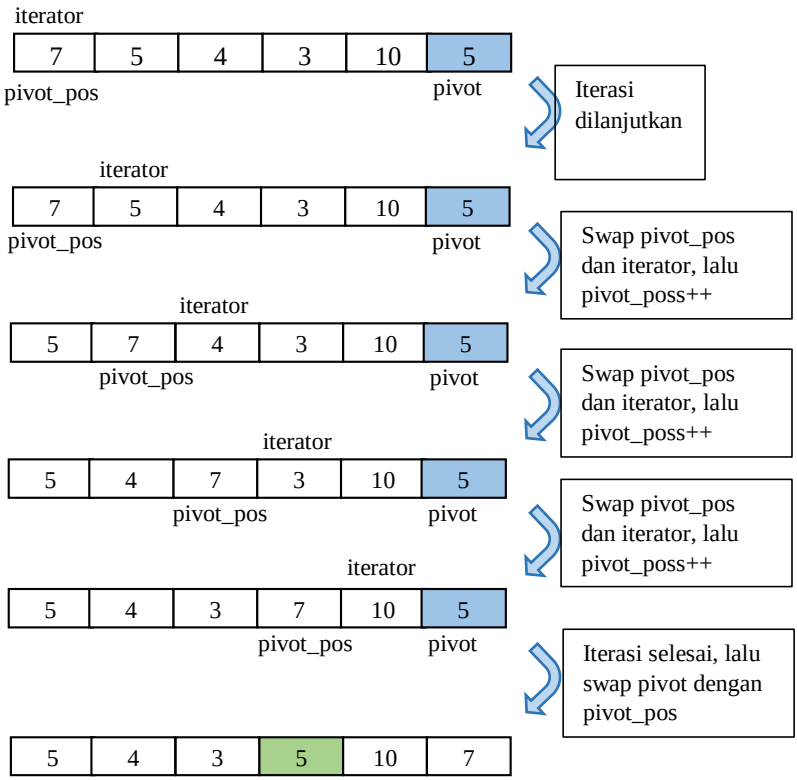
Algoritma *quickselect* merupakan sebuah algoritma pencarian data dalam suatu *array* yang tidak terurut. Cara kerja algoritma *quickselect* bisa dibilang cukup mirip dengan cara kerja algoritma *quicksort*, yakni dengan cara terus memilih elemen sebagai pivot dan menukar data-data yang lain sehingga memiliki urutan yang benar. Hal yang membedakan *quickselect dengan quicksort* adalah algoritma *quickselect* hanya melakukan rekursi pada bagian yang memiliki bilangan terkecil ke-K dan proses rekursi dihentikan bila indeks pivot sama dengan K, sedangkan *quicksort* melakukan proses rekursi pada bagian kiri dan kanan dari indeks dan melakukan proses rekursi hingga seluruh elemen pada *array*, proses ini memiliki rata-rata kompleksitas waktu sebesar O(*n)*, walaupun memiliki kompleksitas waktu pada kemungkinan terburuk O(n^2) [2]. Ilustrasi proses rekursi algoritma *quickselect* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Ilustrasi algoritma *quickselect*

2.2.1 Proses Partisi

Dalam algoritma *quickselect* dilakukan sebuah proses partisi yang berguna untuk menentukan titik pivot dan menempatkannya pada posisinya yang tepat pada setiap rekursi *quickselect*.



Gambar 2.2 Ilustrasi proses partisi pada sebuah *array*

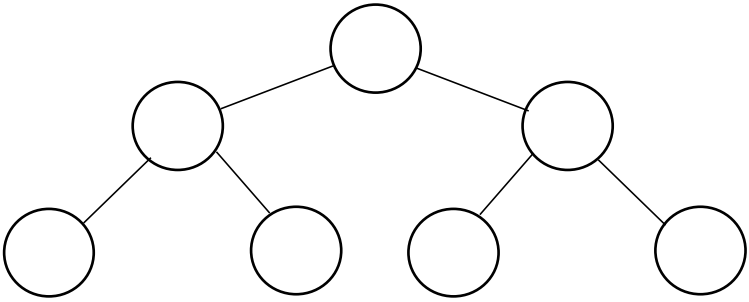
Fungsi partisi bekerja dengan cara meletakan pivot pada indeks paling kanan pada rentang, dan mengasumsikan posisi pivot sebenarnya berada pada elemen paling kiri pada rentang. Fungsi partisi dilanjutkan dengan melakukan iterasi dari batas kiri rentang hingga indeks sebelum pivot. Ilustrasi proses iterarsi dapat dilihat pada Gambar 2.2 dengan posisi pivot sebenarnya disimbolkan dengan *pivot_pos*. Kemudian apabila pada saat iterasi terdapat nilai yang lebih kecil atau sama dengan nilai pada pivot pada indeks

tertentu maka dilakukan penukaran nilai antara pada indeks tersebut dan indeks asumsi posisi pivot, lalu asumsi posisi pivot digeser ke kanan. Ketika proses iterasi selesai dilakukan penukaran nilai pada pivot dan indeks *pivot_pos*.

Ilustrasi pada Gambar 2.2 juga menunjukan bahwa indeks yang benar dalam *array* yang terurut dari pivot adalah pada indeks ke-3 atau merupakan bilangan terkecil ke 4 dalam *array* tersebut.

2.3 *Binary Tree*

Binary Tree merupakan jenis struktur data *Tree*, yang hanya memiliki dua *child* untuk setiap node yang ada, kedua *child* tersebut biasa disebut dengan istilah *left child* dan *right child*. Gambar 2.3 menunjukkan contoh visualisasi dari struktur data *binary tree.*



Gambar 2.3 Visualisasi bentuk struktur data *Binary Tree.*

2.4 Proses *Tree Traversal*

Proses *traversal* adalah sebuah proses yang dilakukan untuk mengunjungi *node-node* yang ada pada struktur data dan melakukan sesuatu terhadap data tersebut [3]. Tidak seperti struktur data linear yang hanya memiliki satu cara pada proses *traversal*, yaitu dengan cara melakukan iterasi pada tiap-tiap node-

nya. Tree memiliki dua jenis cara yang dapat digunakan untuk melakukan proses *traversal.* Proses-proses *traversal* yang dapat dilakukan pada tree adalah *Depth First Traversal* dan *Breadth First Traversal*.

Depth First Traversals adalah proses *traversal* yang memprioritaskan untuk mengunjungi *node-node* terdalam terlebih dahulu. Terdapat tiga jenis proses *Depth First Traversals* yang umum digunakan dalam proses *traverse tree* yaitu *in-Order traversal, pre-Order traversal,* dan *post-Order traversal.*

In-Order Traversal adalah proses *traversal* yang memprioritaskan untuk mengakses data pada *child* paling kiri terlebih dahulu sebelum mengakses *node* itu sendiri dan *child* kanannya.

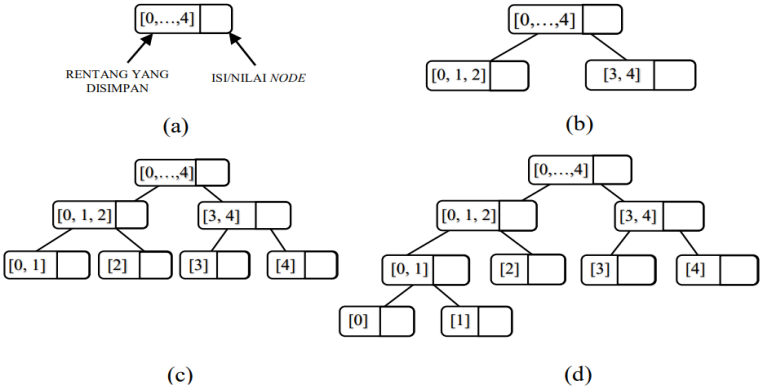
2.5 *Segment Tree*

Dalam ilmu komputer, *segment tree* adalah sebuah struktur data *tree* yang selain menyimpan nilai pada setiap *node*-nya, juga menyimpan informasi tentang interval, atau rentang pada *array*. Dalam pencarian data *segment tree* cukup mencari *node* tertinggi yang dapat mewakili data yang dicari [4]. Umumnya *segment tree* termasuk salah satu struktur data statis, yaitu struktur data yang tidak bisa diubah setelah proses kostruksi strukturnya selesai. Pada umumnya *segment tree* merupakan sebuah *binary tree* yang dimodifikasi agar menyimpan rentang dari *node-node child*-nya.

2.5.1 Konstruksi *Segment Tree*

Dalam membangun sebuah *segment tree* untuk sebuah *array A* dengan panjang elemen sebanyak *n*. Pada umumnya, proses konstruksi sebuah *segment tree* memiliki kompleksitas sebesar O(*n*) [5]. Algoritma yang dapat digunakan untuk mengkonstruksi sebuah *segment tree* adalah sebagai berikut:

1. Buat *node* yang berisikan seluruh elemen *array A* yang akan dijadikan *node root*.
2. Buat 2 child *node* baru yang memiliki *parent node* A[*n*] dan beranggotakan elemen *A*[0,..., *n*/2] dan A[(*n*/2) + 1,..., *n*].
3. Jika elemen di dalam node hanya beranggotakan 1 elemen maka proses dihentikan. Jika tidak maka proses kembali ke langkah kedua.



Gambar 2.4 Ilustrasi proses konstruksi *segment tree*

Ilustrasi pada Gambar 2.4 menunjukan proses konstruksi dari *segment tree* untuk *array A* dengan elemen [0, 1, 2, 3, 4]. Pada Gambar 2.4a menunjukkan *node* yang menyimpan nilai dari seluruh rentang yang ada menjadi *root* dari *segment tree* tersebut, yaitu rentang 0 sampai dengan 4.

Pada Gambar 2.4b menunjukkan proses rekursi untuk membuat dua *child* baru, yakni *child* kiri menyimpan nilai dari rentang 0 hingga 2, dan *child* kanan menyimpan nilai dari rentang 3 dan 4.

Pada Gambar 2.4c menunjukkan lanjutan proses rekursi berikutnya, namun untuk *node* yang menyimpan hanya satu rentang saja, yaitu *node* yang menyimpan rentang 2, 3, dan 4 proses dihentikan, bentuk akhir dari segment tree terlihat pada Gambar 2.4d.

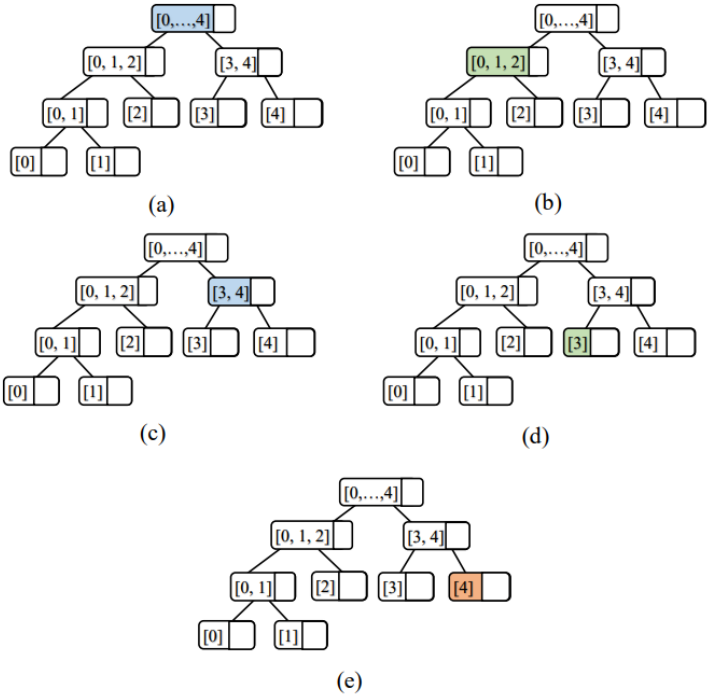
2.5.2 *Query Segment Tree*

Proses *query* pada *segment tree* adalah proses pencarian dan pengambilan nilai yang diinginkan pada *tree*. Umumnya dalam proses *query* pada sebuah *segment tree* dilakukan proses *in-Order traversal* dan memiliki kompleksitas sebesar O(log *n*) karena kedalaman maksimal *tree* adalah log *n*, dengan *n* adalah jumlah data pada *segment tree*. Hal ini karena kemungkinan kasus terburuk pada proses *query* adalah ketika nilai yang diinginkan berada pada *leaf node*, yakni berada pada tingkat terdalam pada *tree* yaitu pada tingkat log *n*[3]. Algoritma yang dapat digunakan dalam proses query pada *segment tree* yang telah dibangun adalah sebagai berikut:

1. Jika rentang yang diinginkan tepat berada atau mencakup keseluruhan pada rentang *node* saat ini, maka proses return nilai dari *node* tersebut.
2. Jika rentang yang diinginkan tidak beririsan sama sekali dengan rentang *node* saat ini maka return 0 atau nilai yang menyatakan rentang yang diinginkan tidak berada pada *node* ini ataupun pada *child node* dari *node* ini.
3. Bila rentang yang diinginkan beririsan dengan sebagian dari rentang *node* saat ini maka dilakukan *query* secara rekursif pada *left child* dan *right child* dari *node* ini.

Proses *query* akan menggunakan *segment tree* yang telah dibangun sebelumnya, yaitu *segment tree* pada Gambar 2.4d dan

dimisalkan rentang yang diinginkan adalah rentang antara 0 sampai dengan 3 diilustrasikan pada Gambar 2.5. Menggunakan algoritma *query* yang telah disebutkan diatas, pada ilustrasi Gambar 2.5 digunakan warna hijau bila memenuhi pernyataan nomor 1, warna merah bila memnuhi pernyataan pada nomor 2, dan warna biru bila memenuhi pernyataan pada nomor 3.



Gambar 2.5 Ilustrasi proses *query* pada *segment tree*

Pada Gambar 2.5a mengilustrasikan proses query dimulai dari *root* segment *tree*, dan proses dilanjutkan secara rekursif pada *left child* dan *right child* dari *root*. Hal ini dilakukan karena rentang

yang diinginkan, yakni 0 hingga 3, beririsan dengan rentang pada node root yang memiliki rentang 0 hingga 4.

Pada Gambar 2.5b mengilustrasikan proses *query* pada *node* yang memiliki rentang 0 hingga 2, maka proses *query* kembali pada *node* sebelumnya dengan me-*return*-kan nilai dari node ini. Hal ini dikarenakan pada *node* ini memiliki rentang 0 hingga 2 yang seluruhnya terdapat pada rentang yang dicari, yakni 0 hingga 3.

Proses *query* dilanjutkan pada *right child* dari *node root*, karena proses *query* pada *left child* telah selesai. Proses ini diilustrasikan pada Gambar 2.5c. Node ini memiliki rentang 3 hingga 4, yang beririsan sebagian dengan rentang yang diinginkan. Oleh karena itu dilakukan proses *query* secara rekursif pada *left child* dan *right child* dari *node* ini.

Pada Gambar 2.5d proses *query* dilakukan pada *node* yang memiliki rentang 3. Karena *node* ini seluruhnya berada pada rentang yang dicari, maka proses *query* kembali pada *node* sebelumnya dengan mengembalikan nilai dari *node* ini.

Pada gambar Gambar 2.5e proses dilanjutkan pada *right child* yang memiliki rentang 4. Karena pada *node* ini memiliki rentang yang sama sekali tidak beririsan dengan rentang yang dicari, maka proses kembali pada node sebelumnya. Karena semua *node* yang perlu dilakukan proses *query* sudah dilewati, maka proses *query* telah selesai.

2.6 Permasalahan *Rangel and the Array Game II*

Permasalahan yang diangkat dalam tugas akhir ini bersumber dari permasalahan *Rangel and the Array Game II* pada situs *URI online judge*. Deskripsi permasalahan ini dapat dilihat pada Gambar 2.6

URI Online Judge | 2849

Rangel and the Array Game II

By Diego Rangel, FACIT Brazil

Timelimit: 3

Always after the programming competitions the participants usually interact. Thinking about it, Rangel is developing an interesting game for the participants to play after a competition. This game will be known as the Array Game.

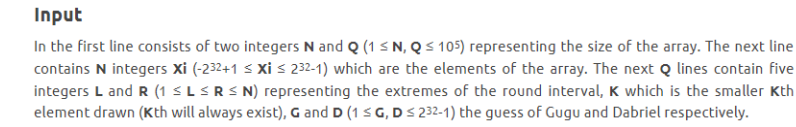
The Array Game works as follows:

- A array with **N** integers is generated randomly and shown for 10 seconds for the challengers.
- Then follows **Q** rounds where players must say how many times the **K**th smaller element appears in a given range.
- Win the round closest to the result.

This year Rangel called his friends Gugu and Dabriel to test the new game and asked you to design the judge to say who Kth is their frequency in the break and who wins the ith round.

Gambar 2.6 Deskripsi soal *Rangel and the Array Game II* pada situs *URI online judge*

Dalam permasalahan tersebut dideskripsikan terdapat tokoh utama yang bernama Rangel telah membuat sebuah permainan yang akan dimainkan setelah kontes programming. Maka rangel memanggil dua tokoh lain yang bernama Gugu dan Dabriel yang akan mencoba dan berkompetisi untuk menebak frekuensi dari bilangan terkecil ke-K dalam suatu rentang pada sebuah *array*.



Input

In the first line consists of two integers **N** and **Q** ($1 \leq N, Q \leq 10^5$) representing the size of the array. The next line contains **N** integers **Xi** ($-2^{32}+1 \leq Xi \leq 2^{32}-1$) which are the elements of the array. The next **Q** lines contain five integers **L** and **R** ($1 \leq L \leq R \leq N$) representing the extremes of the round interval, **K** which is the smaller **K**th element drawn (**K**th will always exist), **G** and **D** ($1 \leq G, D \leq 2^{32}-1$) the guess of Gugu and Dabriel respectively.

Gambar 2.7 Deskripsi masukan pada permasalahan daring *Rangel and The Game II*

Pada deskripsi permasalahan pada daring juga dideskripsikan masukan yang akan diberikan, yang dapat dilihat pada Gambar 2.7. Seperti yang telah dijelaskan pada deskripsi, masukan yang diberikan pada setiap kasus uji dimulai dengan masukan yang

merupakan panjang *array* dan jumlah *query* pada setiap kasus uji, yang dilambangkan dengan *N* dan *Q*. Pada baris berikutnya masukan yang diberikan sejumlah *N* ($1 \leq N \leq 100000$) bilangan yang merupakan nilai dari masing-masing elemen pada *array A*. Pada *Q* ($1 \leq Q \leq 100000$) baris berikutnya adalah masukan yang melambangkan query yang diinginkan, pada tiap barisnya akan terdapat masukan *L*, *R*, *K*, *G*, dan *D*, yang secara berurutan adalah batas kiri dan batas kanan rentang dilambangkan *L* dan *R* ($1 \leq L, R \leq 100000$), nilai *K* yang diinginkan, tebakan dari Gugu yang dilambangkan dengan *G* ($1 \leq G \leq 2^{32}$-1), dan tebakan dari Dabriel yang dilambangkan dengan *D* ($1 \leq D \leq 2^{32}$-1). Contoh masukan yang diberikan pada daring dapat dilihat pada Gambar 2.8.

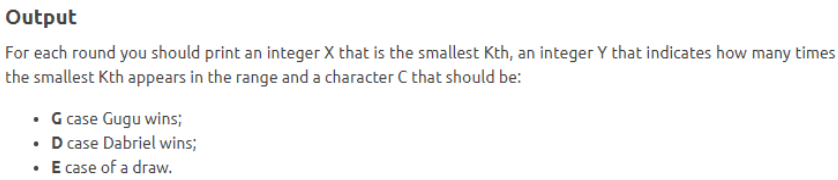
Input Sample

```
10 5
1 4 5 2 7 4 5 8 10 1
1 10 1 3 1
1 5 2 1 4
2 6 3 1 1
7 7 1 0 10
3 8 4 10 4
```

Gambar 2.8 Contoh masukan pada permasalahan daring *Rangel and The Game II*

Setiap *query* yang diterima, memiliki keluaran berupa *X* dan *Y*, yang secara berurutan merupakan nilai elemen terkecil ke-K dan frekuensi elemen tersebut muncul dalam rentang, serta keluaran yang menunjukan pemenang dari kompetisi antara Gugu dan Dabriel, yang dilambangkan dengan huruf "G" bila Gugu adalah pemenang, huruf "D" bila Dabriel adalah pemenang, dan huruf "E" bila tebakan Gugu dan Dabriel seri. Pada situs penilaian daring *URI* juga diberikan deskripsi luaran dari sistem yang diinginkan pada

permasalahan *Rangel and The Game II* yang dapat dilihat pada Gambar 2.9.



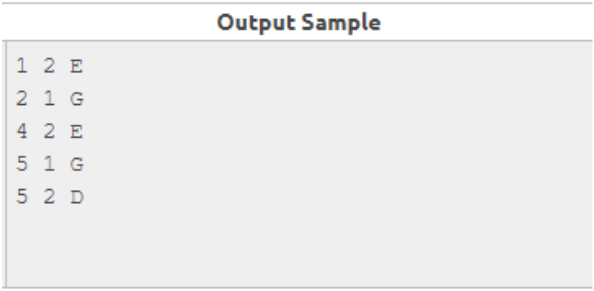
Output

For each round you should print an integer X that is the smallest Kth, an integer Y that indicates how many times the smallest Kth appears in the range and a character C that should be:

- **G** case Gugu wins;
- **D** case Dabriel wins;
- **E** case of a draw.

Gambar 2.9 Deskripsi luaran pada permasalahan daring *Rangel and The Game II*

Selain menemukan bilangan yang menjadi elemen terkecil ke-K pada rentang yang ditentukan, penyelesaian yang dibuat untuk permasalahan ini harus menemukan frekuensi elemen yang bernilai sama dengan elemen terkecil ke-K tersebut. Setelah menemukan elemen terkecil ke-K dan frekuensi elemen tersebut, penyelesaian yang dibuat juga menentukan siapa pemenang antara Gugu dan Dabriel yang berkompetisi untuk menebak jumlah elemen yang sama dari bilangan terkecil ke-K tersebut. Contoh keluaran pada daring yang sesuai dengan contoh masukan pada Gambar 2.10.



Output Sample

```
1 2 E
2 1 G
4 2 E
5 1 G
5 2 D
```

Gambar 2.10 Contoh luaran pada permasalahan daring *Rangel and The Game II*

2.7 Penyelesaian Permasalahan *Rangel and the Array Game II*

Subbab ini menjelaskan pendekatan-pendekatan yang digunakan dalam menyelesaikan permasalahan *Rangel and the Array Game II* dengan penjelasan permasalahan seperti pada subbab 2.6.

2.7.1 Penyelesaian Permasalahan Menggunakan Struktur Data *Array* dan Algoritma *Quickselect*

Pendekatan pertama yang digunakan adalah menggunakan algoritma *quickselect* dan struktur data *array*. Algoritma *quickselect* dipilih karena dapat menemukan bilangan terkecil ke-K tanpa harus mengurutkan seluruh elemen dalam suatu rentang pada *array*.

Proses penyelesaian yang dilakukan diawali dengan menerima semua masukan *N*, *Q*, dan *array A*. Proses pengolahan data *array A* dilakukan setelah menerima *query* yang dimasukan. Setelah menerima masukan *query* akan dilakukan pencarian data pada menggunakan algoritma *quickselect* sesuai dengan penjelasan pada bab sebelumnya. Setelah mendapatkan bilangan terkecil ke-K, proses dilanjutkan dengan mencari frekuensi bilangan terkecil ke-K tersebut pada *array A* dengan cara melakukan iterasi sepanjang rentang yang sesuai dengan masukan *query* yang diberikan. Setelah mendapatkan bilangan terkecil ke-K dan frekuensinya, akan dilakukan pengecekan dengan tebakan dari Gugu dan Dabriel pada masukan *query* tersebut.

Proses penyelesaian diakhiri dengan menampilkan luaran yang didapat dari proses penyelesaian, yaitu bilangan terkecil ke-K, frekuensi bilangan tersebut, dan sebuah huruf yang mewakili pemenang dari *query* ini.

2.7.2 Penyelesaian Permasalahan Menggunakan Struktur Data *Segment Tree*

Penyelesaian permasalahan menggunakan struktur data *segment tree* dipilih karena struktur data ini memiliki keunggulan dalam penyimpanan dan pengambilan data pada suatu rentang tertentu. Karena struktur data *segment tree* menyimpan rentang yang dimiliki *children node*-nya, yang menyebabkan proses pengambilan data pada struktur data *segment tree* menjadi lebih efisien.

Proses penyelesaian yang dilakukan diawali dengan menerima semua masukan *N* dan *Q*. Lalu sebanyak *N* kali masukan akan disimpan *array A* dan pada sebuah *ordered set*. *Set* digunakan dengan tujuan pada proses penyelesaian dapat dilakukan dengan lebih efisien karena sistem memiliki data dari *array A* yang *unique* dan terurut yang disimpan dalam *set* tersebut. Setelah semua data pada *array A* telah diterima, dilakukan konstruksi *segment tree* sesuai dengan data masukan pada *array* A.

Lalu proses penerimaan masukan *query* dilakukan sebanyak *Q* kali. Untuk masing-masing *query*, dilakukan pengambilan bilangan terkecil ke-K dan frekuensinya dengan mengakses *segment tree* yang telah dibuat sebelumnya. Setelah mendapatkan bilangan terkecil ke-K dan frekuensinya, akan dilakukan pengecekan dengan tebakan dari Gugu dan Dabriel pada masukan *query* tersebut.

Proses penyelesaian diakhiri dengan menampilkan luaran yang didapat dari *segment tree*, yaitu bilangan terkecil ke-K dan frekuensi bilangan tersebut, serta sebuah huruf yang mewakili pemenang dari *query* ini.

BAB III
DESAIN

Pada bab ini akan dibahas tentang desain dan algoritma untuk menyelesaikan permasalahan pada Tugas Akhir ini.

3.1 Desain Penyelesaian Permasalahan *Rangel and the Array Game II* Menggunakan Algoritma *Quickselect* dan Struktur Data *Array*

Pada subbab ini akan dijelaskan desain penyelesaian permasalahan *Rangel and the Array Game II* dengan pendekatan algoritma *Quicselect* dan struktur data *Segment Tree*.

3.1.1 Definisi Umum Sistem

Sistem akan menerima masukan berupa sebuah bilangan bulat *N* dan *Q*, yang secara berurutan mewakili banyaknya elemen pada *array A* dan banyaknya jumlah *query* yang akan dilakukan. Kemudian sistem akan menerima masukan bilangan bulat *Xi* sebanyak *N* kali yang mewakili elemen dari *array A*.

Lalu sebanyak *Q* kali sistem akan menerima 5 bilangan bulat *L*, *R*, dan *K* yang secara berurutan mewakili batas kiri pada *array*, batas kanan pada *array*, nilai terkecil ke-*K* pada batas tersebut, serta *G* dan *D* yang merupakan tebakan dari Gugu dan Dabriel. Kemudian sistem memanggil fungsi solve untuk mendapatkan hasil yang diinginkan. Hasil yang didapat dari fungsi solve adalah sebuah *pair* yang berisi *X* yang merupakan bilangan terkecil ke-*K* pada rentang dengan batas *L* dan *R*, dan *Y* yang merupakanfrekuensi munculnya *X* pada rentang dengan batas *L* dan *R*. Kemudian sistem akan memberikan keluaran berupa tiga bilangan bulat, yaitu X, Y, dan sebuah huruf yang mewakili inisial

pemenang antara Gugu, yakni huruf “G”, dan Dabriel, yakni huruf “D”, atau huruf “E” bila tebakan Gugu dan Dabriel memiliki hasil seri. Adapun *pseudocode* dari fungsi main dapat dilihat pada Gambar 3.1.

```
main()

1   input N
2   input Q
3   for i = 0 to N
4         input A[i]
5   while Q – –
6         input L
7         input R
8         input K
9         input G
10        input D
11        copy A to B
12        P = solve(L, R, K, B)
13        G = abs( P.second – G )
14        D = abs( P.second – D )
15        if G < D
16              output P.first, P.second, “G”
17        else if D  < G
18              output P.first, P.second, “D”
19        else
20              output P.first, P.second, “E”
```

Gambar 3.1 *Pseudocode* fungsi main

3.1.2 Desain Algoritma

Sistem terdiri dari empat fungsi utama yang dibutuhkan sistem agar implementasi kode dapat dilakukan dengan ringkas dan

efektif. Keempat fungsi tersebut yaitu, fungsi *partition* yang diperlukan dalam penerapan algoritma *quickselect*, fungsi *getKth* yang berfungsi untuk melakukan *quickselect* pada rentang dalam *array* dan mendapatkan bilangan terkecil ke-K, fungsi *occurrence* yang berfungsi untuk melakukan iterasi pada rentang dalam *array* dan mendapatkan frekuensi bilangan yang diinginkan, fungsi *solve* yang berfungsi untuk mengolah data masukan query dan memanggil fungsi *getKth* dan fungsi *occurence* sesuai dengan data tersebut. Pada subbab ini akan dijelaskan desain algoritma keempat fungsi tersebut.

3.1.2.1 Desain Fungsi *partition*

Dalam algoritma *quickselect* terdapat proses partisi yang berguna untuk menentukan pivot dan menemukan posisi indeks dari pivot bila *array* tersebut terurut. Fungsi *partition* dan *swap* digunakan untuk menjalankan proses tersebut. *Pseudocode* dari fungsi partition dan swap dapat dilihat pada Gambar 3.2.

```
Partition (arr, l, r, k)
1  i = l
2  for j = l  to r – 1
3        if arr[ j ] == arr[ r ]
4             swap( arr[ i ], arr[ j ] )
5             i++
6  swap( arr[ i ], arr[ r ] )
7  return i
```

Gambar 3.2 Pseudocode fungsi partition

3.1.2.2 Desain Fungsi *getKth*

Fungsi *getKth* digunakan untuk mencari bilangan terkecil ke-K pada suatu *array* menggunakan algoritma *quickselect.* Pseudocode fungsi ini dapat dilihat pada Gambar 3.3.

```
getKth(arr, l, r, k)

1  if k > 0 and k ≤ r - l + 1
2        index = partition(arr, l, r)
3  if index – l == k – 1
4        return arr[ index ]
5  if index – l > k -1
6        return getKth( arr, l, index – 1, k )
7  return getKth( arr, index + 1, r – 1, k – index + l - 1 )
```

Gambar 3.3 Pseudocode fungsi getKth

Fungsi *getKth* bekerja dengan cara memanggil fungsi *partition* untuk mendapatkan indeks pivot apabila nilai $k > 0$ dan nilai $k \leq$ panjang rentang antara *l* dan *r*.

3.1.2.3 Desain Fungsi *occurence*

Fungsi *occurrence* digunakan untuk mencari frekuensi munculnya suatu bilangan dalam suatu rentang. Fungsi occurrence bekerja dengan cara melakukan iterasi sepanjang rentang yang diberikan dan menghitung jumlah elemen yang sama pada rentang tersebut. Pseudocode dari fungsi occurrence dapat dilihat pada *Gambar 3.4*.

```
occurence(arr, l, r, x)

1  res = 0
2  for i = l  to r
3       if arr[ i ] == x
4       res++
5  return res
```

Gambar 3.4 Pseudocode fungsi occurence

3.1.2.4 Desain Fungsi *solve*

Fungsi *solve* merupakan fungsi utama yang digunakan untuk menyelesaikan permasalahan *Rangel and the Array Game II*. Fungsi *solve* mulanya akan mencari nilai terkecil ke-*k* dengan cara memanggil fungsi *getKth* dan *occurrence*, dan mengembalikan hasil dari kedua fungsi tersebut dalam bentuk pair.

```
solve(arr, l, r, k)

6  kth = getKth( arr, l, r, k )
7  occ = occurrence( arr, l,  r, kth )
8  return pair( kth, occ )
```

Gambar 3.5 Pseudocode fungsi solve

3.2 Desain Penyelesaian Permasalahan *Rangel and the Array Game II* Menggunakan Struktur Data *Segment Tree*

Pada subbab ini akan dijelaskan desain penyelesaian permasalahan *Rangel and the Array Game II* dengan pendekatan menggunakan struktur data *Segment Tree*.

3.2.1 Definisi Umum Sistem

Sistem akan menerima masukan berupa sebuah bilangan bulat *N* dan *Q*, yang secara berurutan mewakili banyaknya elemen pada *array A* dan banyaknya jumlah *query* yang akan dilakukan. Kemudian sistem akan menerima masukan bilangan bulat *Xi* sebanyak *N* kali yang mewakili elemen dari *array A* dan juga akan disimpan pada sebuah *set* bernama *mSet*. Kemudian sistem akan membuat sebuah *vector* bernama *indices* yang merupakan salinan dari *set mSet*, dan memanggil fungsi *build* yang bertujuan untuk mengkontruksi *segment tree* dari elemen-elemen pada *array A*.

Lalu sebanyak *Q* kali sistem akan menerima 5 bilangan bulat *L*, *R*, dan *K* yang secara berurutan mewakili batas kiri pada *array*, batas kanan pada *array*, nilai terkecil ke-*K* pada batas tersebut, serta *G* dan *D* yang merupakan tebakan dari Gugu dan Dabriel. Kemudian sistem memanggil fungsi solve untuk mendapatkan hasil yang diinginkan. Hasil yang didapat dari fungsi solve adalah sebuah *pair* yang berisi *X* yang merupakan bilangan terkecil ke-*K* pada rentang dengan batas *L* dan *R*, dan *Y* yang merupakanfrekuensi munculnya *X* pada rentang dengan batas *L* dan *R*. Kemudian sistem akan memberikan keluaran berupa tiga bilangan bulat, yaitu X, Y, dan sebuah huruf yang mewakili inisial pemenang antara Gugu, yakni huruf “G”, dan Dabriel, yakni huruf “D”, atau huruf “E” bila tebakan Gugu dan Dabriel memiliki hasil seri. Adapun *pseudocode* dari fungsi main dapat dilihat pada Gambar 3.6.

```
main()
1  input N
2  input Q
3  for i = 0 to N
4       input A[i]
5       mSet.insert A[i]
6  copy mSet to indices
7  build( 1, 0, N – 1 )
8  while Q – –
9       input L
10      input R
11      input K
12      input G
13      input D
14      P = solve(K, L, R)
15      G = abs( P.second – G )
16      D = abs( P.second – D )
17      if G < D
18           output P.first, P.second, “G”
19      else if D  < G
20           output P.first, P.second, “D”
21      else
22           output P.first, P.second, “E”
```

Gambar 3.6 *Pseudocode* fungsi main

3.2.2 Desain Algoritma

Sistem terdiri dari tiga fungsi utama yang dibutuhkan sistem agar implementasi kode dapat dilakukan dengan ringkas dan efektif. Ketiga fungsi tersebut yaitu, fungsi utama, yaitu fungsi *build* yang berfungsi untuk mengkonstruksi *segment tree*, fungsi *query* yang digunakan untuk proses pengambilan data pada *segment tree*, dan fungsi *solve* yang berfungsi untuk mengolah data masukan *query* dan memanggil fungsi *query* sesuai dengan data

tersebut. Pada subbab ini akan dijelaskan desain algoritma ketiga fungsi tersebut.

3.2.2.1 Desain Fungsi *Build*

Fungsi build digunakan untuk mengkonstruksi *segment tree* dari masukan yang telah diterima pada fungsi main. Pseudocode fungsi ini ditunjukkan oleh Gambar 3.7. Variabel *node* merepresentasikan nilai *node* saat ini, *l* merepresentasikan batas kiri dari *node* saat ini, dan *r* merepresentasikan batas kanan dari *node* saat ini.

build(*node*, *l*, *r*)

```
1  if l == r
2      tree[node].push_back( A[l] )
3      return
4  cl[node] = 2 * node
5  cr[node] = 2 * node +1
6  mid[node] = ( l + r ) / 2
7  build( cl[node], l, mid[node] )
8  build( cr[node], mid[node] + 1, r )
9  tree[node] = merge( tree[ cl[ node ] ], tree[ cr[ node ] ] )
```

Gambar 3.7 Pseudocode fungsi build

Fungsi build akan terus membagi dua rentang antara *l* dan *r* secara rekursif hingga tersisa satu elemen saja dalam rentang, atau dengan kata lain ketika variabel *l* bernilai sama dengan variabel *r*. Fungsi build menggunakan fungsi *merge* yang berguna untuk menggabungkan data pada *child* kiri, yaitu *tree*[*cl*], dan *child* kanan, yaitu *tree*[*cr*], dari *tree* pada *node* tersebut, serta menambahkannya pada *tree*[*node*].

3.2.2.2 Desain Fungsi *Query*

Fungsi *query* digunakan untuk mencari frekuensi dari sebuah nilai variabel *x* pada suatu rentang antara variabel *start* dan variabel *end*. Variabel *l* dan *r* adalah variabel yang mewakili batas rentang keseluruhan ketika fungsi *query* pertama kali dipanggil dari fungsi *solve*.

query(*node, start, end, l, r, x, k*)

```
1  if end < l or start > r or start > end
2      return 0
3  if l <= start or r >= end
4      return upper_bound( tree[node], x )
5  left_res = query( cl[ node ], start, mid[node], l, r, x, k )
6  right_res = query( cr[ node ], mid[ node ]+1, end, l, r, x, k )
7  return left_res + right_res
```

Gambar 3.8 *Psudocode* fungsi *query*

Fungsi *query* akan mengembalikan nilai 0 apabila rentang *start* dan *end* tidak beririsan sama sekali dengan rentang antara *l* dan *r*, baris kedua pada Gambar 3.8. Apabila rentang *start* dan *end* tepat beririsan dengan rentang antara *l* dan *r* maka fungsi *query* akan mengembalikan indeks dari *x* yang ada pada *vector tree*[*node*]. Apabila rentang *start* dan *end* beririsan sebagian dengan rentang antara *l* dan *r* maka fungsi *query* secara rekursif akan memanggil *query* untuk *child* kiri dan *child* kanan dari tree.

3.2.2.3 Desain Fungsi *Solve*

Fungsi *solve* merupakan fungsi utama yang digunakan untuk menyelesaikan permasalahan *Rangel and the Array Game II*. Fungsi *solve* akan mencari nilai terkecil ke-*k* dengan berkali-kali memanggil fungsi *query* yang pada awalnya dipanggil dengan argumen *x* sama dengan nilai dari titik tengah variabel *indices*. Setiap iterasi yang dilakukan nilai titik tengah yang digunakan sebagai argumen pada fungsi *query* akan disesuaikan dengan hasil dari fungsi *query sebelumnya*. Ketika variabel *lo* bernilai lebih besar daripada variabel *hi* maka perulangan dihentikan. Selanjutnya dilakukan pemanggilan fungsi *query* yang bertujuan mencari frekuensi dari variabel *ans* dalam rentang antara *l* dan *r* dengan cara mengurangkan hasil fungsi *query* yang memiliki argumen *x* bernilai *ans* dengan hasil fungsi *query* yang memiliki argumen *x* bernilai *ans*–1. Dapat dilihat pada *Gambar 3.9*

```
solve(k, l, r)
1   hi = indices.size() – 1
2   lo = 0
3   while lo <= hi
4       mid = ( lo + hi ) / 2
5       res = query( 1, 0, N-1, l, r, indices[ mid ], k )
6       if res >= k
7           ans = indices[ mid ]
8           hi = mid – 1;
9       else
10          lo = mid + 1;
11  occ_till_ans = query( 1, 0, N-1, l, r, ans, k )
12  occ_bfr_ans = query( 1, 0, N-1, l, r, ans – 1, k )
13  return pair( ans,  occ_till_ans – occ_bfr_ans )
```

Gambar 3.9 *Pseudocode* fungsi *solve*

BAB IV
IMPLEMENTASI

Pada bab ini dijelaskan mengenai implementasi dari desain dan algoritma penyelesaian permasalahan *Rangel and the Array Game II*.

4.1 Lingkungan Implementasi

Lingkungan implementasi dalam pembuatan Tugas Akhir ini meliputi perangkat keras dan perangkat lunak yang digunakan untuk penyelesaian permasalahan Rangel and the Array Game II adalah sebagai berikut:

1. Perangkat Keras:
    - *Processor Intel Core* i5-6200U 4 *Cores* @ 2.3 GHz *up to* 2.4 GHz.
    - Memori 8 GB.
2. Perangkat Lunak:
    - Sistem Operasi: Windows 10 Education 64 bit.
    - IDE: Orwell Bloodshed Dev-C++ 5.41.
    - Compiler: g++ (TDM-GCC 4.9.2 64-bit).
    - Bahasa Pemrogramman: C++

4.2 Rancangan Data

Pada subbab ini dijelaskan mengenai desain data masukan yang diperlukan untuk melakukan proses algoritma, dan data keluaran yang dihasilkan oleh program.

4.2.1 Rancangan Data Masukan

Data masukan adalah data yang akan diproses oleh program sebagai masukan dalam penggunaan algoritma dan struktur data yang telah dirancang dalam Tugas Akhir ini.

Data masukan berupa berkas teks yang berisi data dengan format yang telah ditentukan pada deskripsi permasalahan *Rangel and the Array Game II.* Pada masing-masing berkas data masukan, baris pertama berupa dua bilangan bulat yang secara berurutan merepresentasikan jumlah data yang ada dan jumlah kasus uji yang ada pada berkas tersebut. Baris kedua berisi bilangan bulat sebanyak bilangan pertama pada masukan baris pertama. Lalu baris-baris berikutnya yang sebanyak bilangan kedua pada baris pertama terdapat lima bilangan bulat yang secara berurutan merepresentasikan batas kiri rentang, batas kanan rentang, urutan terkecil dari nilai yang dicari pada rentang, dan dua tebakan dari tokoh pada permasalahan *Rangel and the Array Game II.*

4.2.2 Rancangan Data Keluaran

Data keluaran yang dihasilkan sistem berupa dua bilangan bulat dan sebuah huruf, yang secara berurutan merepresentasikan bilangan yang dicari, frekuensi kemunculan bilangan yang dicari, dan huruf yang mewakili pemenang pada permasalahan *Rangel and the Array Game II.*

4.3 Implementasi Penyelesaian Permasalahan *Rangel and the Array Game II* Menggunakan Algoritma *Quickselect* dan Struktur Data *Array*

Pada subbab ini akan dijelaskan mengenai implementasi dari penyelesaian permasalahan Rangel and the Array Game II sesuai dengan desain pada subbab 3.1.

4.3.1 Penggunaan *Library*

Pada Subbab ini akan dibahas penggunaan *library* yang digunakan dalam sistem. Pada Kode sumber Kode Sumber 4.1 terdapat sebuah *library*, yaitu *library* bits/stdc++.h yang digunakan.

```
1  #include <bits/stdc++.h>
2  using namespace std;
```

Kode Sumber 4.1 Potongan kode sumber library yang digunakan

4.3.2 Implementasi Fungsi *main*

Fungsi main yang diimplementasikan sesuai dengan *pseudocode* pada bab sebelumnya. Implementasi fungsi main dapat dilihat pada Kode Sumber 4.2. Pada fungsi main dilakukan inisialisasi variabel-variabel yang digunakan pada fungsi ini. Fungsi main juga menggunakan fungsi *fastscan* yang berguna untuk menerima masukan.

```
1   int main()
2   {
3       long n, q, l, r, k, g, d;
4       fastscan(n);
5       fastscan(q);
6       long a[n], b[n];
7       for(long i=0; i<n; i++){
8           fastscan(a[i]);
9       }
10      for(long i=0; i<q; i++){
11          fastscan(l);
12          fastscan(r);
13          fastscan(k);
14          fastscan(g);
15          fastscan(d);
16          for(long j=l-1; j<r; j++){
17              b[j] = a[j];
18          }
19          pair<long, long> p = solve(l-1, r-1, k, b);
20          long x = abs(p.second - g);
21          long y = abs(p.second - d);
22          printf("%d %d %c\n", p.first, p.second, (x == y) ?
            'E' : ((x < y) ? 'G' : 'D'));
23      }
24      return 0;
25  }
```

Kode Sumber 4.2 Potongan kode sumber fungsi main

Fungsi *fastscan* yang dapat dilihat pada Kode Sumber 4.3, menggunakan *getchar* karena proses penerimaan masukan lebih cepat daripada penggunaan fungsi *scanf* dan *standard input cin*.

```
1   inline void fastscan(int &number)
2   {
3       bool negative = false;
4       register int c;
5       number = 0;
6       c = getchar();
7       if (c=='-')
8       {
9           negative = true;
10          c = getchar();
11      }
12      for (; (c>47 && c<58); c=getchar())
13          number = (number<<1)+(number<<3) + c - '0';
14      if (negative)
15          number *= -1;
16  }
```

Kode Sumber 4.3 Potongan kode sumber fungsi *fastscan*

4.3.3 Implementasi Fungsi *partition*

Fungsi *build* diimplementasikan sesuai dengan *pseudocode* yang ada pada bab sebelumnya. Hasil implementasi fungsi *build* dapat dilihat pada Kode Sumber 4.4.

```
1    long partition(long arr[], long l, long r)
2    {
3        long x = arr[r];
4        long i = l;
5
6        for (long j = l; j <= r - 1; j++) {
7
8            if (arr[j] <= x) {
9
10               swap(&arr[i], &arr[j]);
11               i++;
12               //printVec(arr, l, r);
13           }
14       }
15       swap(&arr[i], &arr[r]);
16       return i;
17   }
```

Kode Sumber 4.4 Potongan kode sumber fungsi *partition*

4.3.4 Implementasi Fungsi *getKth*

Fungsi *build* diimplementasikan sesuai dengan *pseudocode* yang ada pada bab sebelumnya. Hasil implementasi fungsi *build* dapat dilihat pada Kode Sumber 4.5.

```
1   long getKth(long arr[], long l, long r, long k)
2   {
3
4       if (k > 0 && k <= r - l + 1) {
5
6           long index = partition(arr, l, r);
7
8           if (index - l == k - 1)
9               return arr[index];
10
11          if (index - l > k - 1)
12              //printVec(arr, l, r);
13              return getKth(arr, l, index - 1, k);
14
15
16          //printVec(arr, l, r);
17          return getKth(arr, index + 1, r,
18                              k - index + l - 1);
19
20      }
21
22      return INT_MAX;
23  }
```

Kode Sumber 4.5 Potongan kode sumber fungsi getKth

4.3.5 Implementasi Fungsi *occurrence*

Fungsi *build* diimplementasikan sesuai dengan *pseudocode* yang ada pada bab sebelumnya. Hasil implementasi fungsi *build* dapat dilihat pada Kode Sumber 4.6.

```
1    long occurence(long arr[], long kth, long l, long r){
2        long occur = 0;
3        for(long i=l; i<=r; i++){
4            if(arr[i] == kth){
5                occur++;
6            }
7        }
8        return occur;
9    }
```

Kode Sumber 4.6 Potongan kode sumber fungsi occurence

4.3.6 Implementasi Fungsi *solve*

Fungsi *build* diimplementasikan sesuai dengan *pseudocode* yang ada pada bab sebelumnya.

4.4 Implementasi Penyelesaian Permasalahan *Rangel and the Array Game II* Menggunakan Struktur Data *Segment Tree*

Pada subbab ini akan dijelaskan mengenai implementasi dari penyelesaian permasalahan *Rangel and the Array Game II* sesuai dengan desain pada subbab 3.2.

4.4.1 Penggunaan *Library* dan Variabel Global

Pada Subbab ini akan dibahas penggunaan *library*, *template*, konstanta dan variabel global yang digunakan dalam sistem. Pada Kode Sumber 4.7 terdapat sebuah *library*, yaitu *library* bits/stdc++.h yang digunakan. Didefinisikan MAXNODE bernilai 300005 merupakan jumlah maksimal *node* yang dapat disimpan. Pada Tabel 4.1 dan 4.2 dijelaskan mengenai variabel-variabel global yang digunakan dalam implementasi program.

```
1  #include <bits/stdc++.h>
2  #define MAXNODE 300005
3  #define ll long long
4  #define abs_value(x) ((x < 0) ? -x : x)
5  using namespace std;
```

Kode Sumber 4.7 Potongan kode sumber *library* dan makro yang digunakan

Tabel 4.1 Daftar variabel global bagian 1

No	Nama Variabel	Tipe	Penjelasan
1	A	int []	Digunakan untuk menyimpan seluruh masukan
2	N	int	Digunakan untuk menyimpan banyaknya elemen pada *array A*
3	tree	vector<int> []	Digunakan untuk menyimpan *node-node* dari *segment tree*
4	cl	int []	Digunakan untuk menyimpan indeks *child* kiri dari *segment tree*

Tabel 4.2 Daftar variabel global bagian 2

5	cr	int []	Digunakan untuk menyimpan indeks *child* kanan dari *segment tree*
6	middle	int []	Digunakan untuk menyimpan indeks tengah pada *vector* dalam *node segment tree*
7	mSet	set<int>	Digunakan untuk menyimpan array *A*[] yang *unique* dan terurut
8	indices	vector<int>	Merupakan *vector* salinan dari mSet

4.4.2 Implementasi Fungsi *Main*

Fungsi main yang diimplementasikan sesuai dengan *pseudocode* pada bab sebelumnya. Implementasi fungsi main dapat dilihat pada Kode Sumber 4.8. Pada fungsi main dilakukan inisialisasi variabel-variabel yang digunakan pada fungsi ini. Fungsi main juga menggunakan fungsi *fastscan* yang berguna untuk menerima masukan.

```
1   int main()
2   {
3       int Q, l, r, k, g, d;
4       ll x, y;
5       pair<int, int> p;
6       fastscan(N);
7       fastscan(Q);
8       for(int i = 0; i < N; i++)
9       {
10                  fastscan(A[i]);
11                  mSet.insert(A[i]);
12      }
13      indices.assign(mSet.begin(), mSet.end());
14      build(1, 0, N-1);
15      while(Q--)
16      {
17                  fastscan(l);
18                  fastscan(r);
19                  fastscan(k);
20                  fastscan(g);
21                  fastscan(d);
22                  l--, r--;
23                  p = solve(k, l, r);
24                  x = abs(p.second-(ll)g);
25                  y = abs(p.second-(ll)d);
26                  printf("%d %d %c\n", p.first, p.second, (x == y)?
                    'E' : ((x < y) ? 'G' : 'D'));
27      }
28      return 0;
29  }
```

Kode Sumber 4.8 Potongan kode sumber fungsi main

Fungsi *fastscan* yang dapat dilihat pada Kode Sumber 4.9, menggunakan *getchar* karena proses penerimaan masukan lebih cepat daripada penggunaan fungsi *scanf* dan *standard input cin*.

```
1   inline void fastscan(int &number)
2   {
3       bool negative = false;
4       register int c;
5
6       number = 0;
7       c = getchar();
8       if (c=='-')
9       {
10          negative = true;
11          c = getchar();
12      }
13      for (; (c>47 && c<58); c=getchar())
14          number = (number<<1)+(number<<3) + c - '0';
15      if (negative)
16          number *= -1;
17  }
```

Kode Sumber 4.9 ptongan kode sumber fungsi *fastscan*

4.3.3 Implementasi Fungsi *Build*

Fungsi *build* diimplementasikan sesuai dengan *pseudocode* yang ada pada bab sebelumnya. Hasil implementasi fungsi *build* dapat dilihat pada Kode Sumber 4.10.

```
1   inline void build(int node, int l, int r)
2   {
3           if(l == r)
4           {
5                   tree[node].push_back(A[l]);
6                   return;
7           }
8           cl[node] = 2*node;
9           cr[node] = 2*node+1;
10          middle[node] = ( l +  r )>>1;
11          build(cl[node], l, middle[node]);
12          build(cr[node], middle[node]+1, r);
13          merge(tree[ cl[node] ].begin(), tree[ cl[node] ].end(),
                    tree[ cr[node] ].begin(), tree[ cr[node] ].end(),
                    back_inserter( tree[node] ));
14  }
```

Kode Sumber 4.10 Potongan kode sumber fungsi build

4.4.4 Implementasi Fungsi *Query*

Fungsi *query* diimplementasikan sesuai dengan *pseudocode* yang ada pada bab sebelumnya. Hasil implementasi fungsi *query* dapat dilihat pada Kode Sumber 4.12. Fungsi *query* digunakan sebagai fungsi yang digunakan untuk mencari bilangan terkecil ke-*K* dan juga frekuensi bilangan itu sendiri.

Setelah dilakukan optimasi lebih lanjut fungsi *query* pada penggunaan dalam mencari bilangan terkecil ke-*K* dapat dioptimasi menjadi fungsi *query2,* yang dapat dilihat pada Kode Sumber 4.11. Optimasi yang dilakukan adalah dengan melakukan pencarian bilangan terkecil ke-*K* dilakukan dengan melakukan rekursif ke *child* kiri terlebih dahulu.

```
1  inline int query(int node, int start, int end, int l, int r, int x, int K)
2  {
3          if(end < l || start > r || start > end)
4                  return 0;
5          if(l <= start && r >= end)
6          {
7                  return upper_bound(tree[node].begin(),
                       tree[node].end(), x) - tree[node].begin();
8          }
9          return query(fl[node], start, meio[node], l, r, x) +
                   query(fr[node], meio[node]+1, end, l, r, x);
10 }
```

Kode Sumber 4.12 Potongan kode sumber fungsi query

```
11 inline int query2(int node, int start, int end, int l, int r, int x, int K)
12 {
13         if(end < l || start > r || start > end)
14                 return 0;
15         if(l <= start && r >= end)
16         {
17                 return upper_bound(tree[node].begin(),
                       tree[node].end(), x) - tree[node].begin();
18         }
19         int y = query2(fl[node], start, meio[node], l, r, x, K);
20         if(y >= K)
21                 return y;
22         else
23                 return y + query2(fr[node], meio[node]+1, end, l, r,
                                      x, K);
24 }
```

Kode Sumber 4.11 Potongan kode sumber fungsi query2

4.4.6 Implementasi Fungsi *Solve*

Fungsi *solve* diimplementasikan sesuai dengan *pseudocode* yang ada pada bab sebelumnya. Hasil implementasi fungsi *solve* dapat dilihat pada Kode Sumber 4.13.

```
1.  inline pair<int, int> solve(int K, int L, int R)
2.  {
3.      int lo = 0, hi = indices.size()-1, mid, ans;
4.      while(lo <= hi)
5.      {
6.          mid = (lo+hi)>>1;
7.          if(query(1, 0, N-1, L, R, indices[mid], K) >= K)
8.          {
9.              ans = indices[mid];
10.             hi = mid-1;
11.         }
12.         else
13.             lo = mid+1;
14.     }
15.     return make_pair(ans, query(1, 0, N-1, L, R, ans)
                              – query(1, 0, N-1, L, R, ans-1));
16. }
```

Kode Sumber 4.13 Potongan kode sumber fungsi *solve*

4.5 Data Generator

Data generator digunakan untuk membuat data masukan yang ditujukan untuk proses uji coba. Implementasi data generator yang digunakan dapat dilihat pada Kode Sumber 4.14, Kode *Sumber 4.15*, Kode Sumber *4.16* .

Data generator menggunakan sebuah struct bernama *query* yang berisi lima bilangan dan melambangkan masukan *query* yaitu (*l, r, k, g, d*).

```
#include <sstream>
#include <vector>
#include <time.h>
#include <fstream>
#include <string>
#include <cstdlib>
#include <windows.h>
using namespace std;

struct query
{
   long long L,R,K,G,D;
};

int main()
{
      int T = 20, maxN = 50000;
      string nama_file = "/berkas_masukan_";
      long long N,Q, L, R, K, G, D, temp;
      vector<query> dataset;
      query data;

      while(T--){
            std::stringstream nama;
            nama << maxN << nama_file << T << ".in";
            ofstream file(nama.str());
            srand(time(NULL));

            N = rand()%(maxN+T)+1;
            Q = rand()%(maxN+T)+1;

            file << N << " " << Q << endl;

            for (long long i = 0; i < N; ++i)
```

Kode Sumber 4.14 Ptongan kode sumber data *generator bagian* 1

```
36      for (long long i = 0; i < N; ++i)
37              {
38                      file << rand()%(maxN+T);
39                      if(i < N-1 )
40                              file << " ";
41                      else
42                              file << endl;
43              }
44
45              for (long long i = 0; i < Q; ++i)
46              {
47                      L = rand()%(maxN+T);
48                      R = rand()%(maxN+T);
49                      K = rand()%(maxN+T);
50                      G = rand()%(maxN+T);
51                      D = rand()%(maxN+T);
52
53                      if(N<R) R = R%N;
54
55                      if(R==0) L=0;
56                      else if(R<L) L = L%R;
57
58                      if(K>R-L) K=K%(R-L+1);
59
60                      if(N<G) G = G%N;
61
62                      if(N<D) D = D%N;
63
64                      data.L = L;
65                      data.R = R;
66                      data.K = K;
67                      data.G = G;
68                      data.D = D;
```

Kode Sumber 4.15 Potongan sumber kode data *generator bagian* 2

```
70        dataset.push_back(data);
71                }
72
73                for(int i=0;i<dataset.size();i++)
74                {
75                        file << dataset[i].L << " " <<
    dataset[i].R << " " << dataset[i].K << " " << dataset[i].G
    << " " << dataset[i].D << endl;
76                }
77                file.close();
78                Sleep(50);
79        }
80
81        return 0;
82
83  }
```

Kode Sumber 4.16 Potongan kode sumber *data generator* bagian 3

Cara kerja program data generator adalah sistem akan membuat sebuah file bernama “berkas_masukan_T” dimana T adalah nomor berkas.

BAB V
UJI COBA

Pada bab ini dijelaskan tentang uji coba dan evaluasi dari implementasi yang telah dilakukan pada Tugas Akhir ini.

5.1 Lingkungan Uji Coba

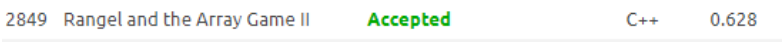
Perangkat keras yang digunakan memiliki prosesor *Intel Core* i5-6200U 4 *cores* 2.3 GHz up to 2.4 GHz dengan random access memory (RAM) sebesar 8 GB. Dengan sistem operasi Windows 10 *Education* 64 bit. Bahasa yang digunakan adalah bahasa C++ dengan bantuan *integrated development environment* (IDE) Orwell Bloodshed Dev-C++ 5.41 serta compiler: g++ (TDM-GCC 4.7.2 32-bit).

5.2 Skenario Uji Coba

Pada subbab ini akan dijelaskan skenario uji coba yang dilakukan pada sistem.

5.2.1 Uji Coba Kebenaran Penyelesaian Permasalahan

Uji coba kebenaran dilakukan dengan mengirimkan kode sumber ke situs penilaian daring *URI judge*. Dari hasil uji yang ditunjukkan pada Gambar 5.1, dapat dilihat bahwa kode sumber penyelesaian yang menggunakan *segment tree* mendapat keluaran *Accepted* dengan waktu eksekusi program adalah 0,628 detik.



Gambar 5.1 Hasil uji coba penyelesaian pada situs daring *URI judge* menggunakan struktur data *segment tree*

Sedangkan uji coba kebenaran yang dilakukan pada program yang menggunakan algoritma *quickselect* mendapatkan keluaran *Time Limit Exceeded* yang menyatakan bahwa kode sumber yang dikirimkan waktu eksekusinya melebihi batas waktu, yaitu tiga detik. Hasil keluaran program dapat dilihat pada Gambar 5.2.

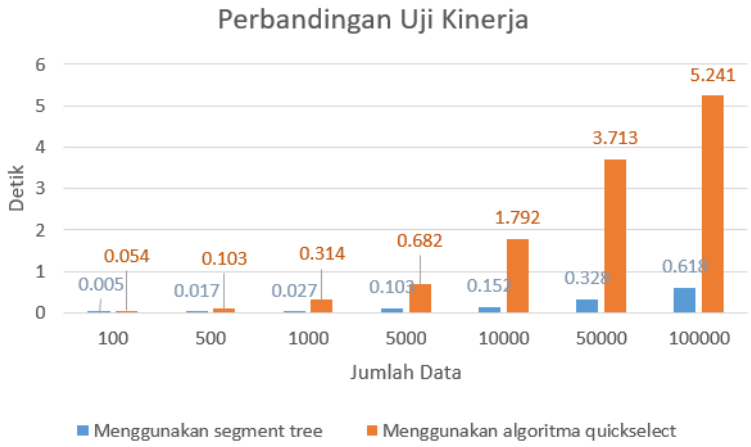
Gambar 5.2 Hasil uji coba penyelesaian pada situs daring URI judge menggunakan algoritma quickselect

Walaupun penyelesaian yang menggunakan algoritma *quickselect* melebihi batas waktu yang ditentukan, namun hasil keluaran yang dihasilkan sama dengan penyelesaian yang menggunakan struktur data *segment tree* apabila diberikan masukan data dari implmentasi kode *data generator*.

5.2.2 Uji Coba Kinerja

Uji coba perbandingan kinerja yang dilakukan pada masing-masing metode penyelesaian menggunakan 5 berkas data masukan yang sama, dan rata-rata waktu eksekusi hingga proses selesai diambil sebagai variabel pembanding pada Gambar 5.3.

Grafik bewarna biru menunjukan kinerja dari program yang mengimplementasikan pendekatan struktur data *segment tree*, sedangkan yang bewarna oranye menunjukan kinerja dari program dengan pendekatan algoritma *quickselect* dan struktur data *array*.



Gambar 5.3 Hasil uji coba program menggunakan data dari generator

Berdasarkan hasil uji coba yang ditunjukan pada Gambar 5.3 dapat dilihat bahwa program yang mengimplementasikan algoritma *quickselect* dan struktur data *array* memiliki waktu eksekusi yang lebih besar daripada program yang mengimplementasikan struktur data *segment tree*, dan bila semakin besar jumlah data yang diproses, maka semakin besar pula perbedaan rata-rata waktu eksekusinya. Hal ini disebabkan karena pada program yang mengimplementasikan *quickselect* dilakukann proses pencarian bilangan terkecil ke-K yang memiliki kompleksitas waktu eksekusi rata-rata O(n), dengan kemungkinan terburuk O(n^2). Kompleksitas waktu algoritma *quickselect* ini terjadi *bottleneck* pada proses partisi dan proses pencarian frekuensi bilangan. Hal ini diperparah oleh dilakukannya proses tersebut berkali-kali untuk sesuai jumlah *query* pada berkas masukan. Sedangkan pada program yang mengimplementasikan

segment tree proses pengolahan data hanya dilakukan sekali saja pada saat mengkonstruksi *segment tree*.

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini dijelaskan mengenai kesimpulan dan saran dari hasil uji coba yang telah dilakukan.

6.1 Kesimpulan

Dari hasil uji coba yang dilakukan terhadap implementasi penyelesaian permasalahan *Rangel and the Array Game II* pada situs penilaian daring URI *Online Judge* dapat diambil kesimpulan sebagai berikut:

1. Implementasi struktur data segment tree dapat menyelesaikan permasalahan Rangel and the Array Game II pada situs penilaian daring URI Online Judge dengan benar dan dengan waktu eksekusi 0,628 detik yang memenuhi batas waktu URI Online Judge, yaitu 3 detik.
2. Implementasi algoritma quickselect menghasilkan luaran Time Limit Exceeded yang berarti implementasi algoritma quickselct tidak memenuhi batas waktu 3 detik yang ditentukan pada permasalahan Rangel and the Array Game II pada situs penilaian daring URI Online Judge.

6.2 Saran

Pada Tugas Akhir kali ini tentunya terdapat kekurangan serta nilai-nilai yang dapat penulis ambil. Berikut adalah saran-saran yang dapat diambil melalui Tugas Akhir ini:

1. Struktur data *segment tree* cukup layak bila diimplementasikan untuk menyelesaikan permasalahan tentang rentang dinamis pada suatu *array*.

2. Penyelesaian menggunakan algoritma *quickselect* yang digunakan pada buku ini kurang layak digunakan untuk menyelesaikan permasalahan tentang rentang dinamis pada suatu *array*, karena kompleksitas waktu terburuk dari algoritma *quickselect* adalah $O(n^2)$, walaupun memiliki rata-rata sebesar $O(n)$. Optimasi algoritma *quickselect* diperlukan guna mengurangi kompleksitas waktu yang dibutuhkan.

DAFTAR PUSTAKA

[1] D. Rangel, "2849 - Rangel and the Array Game II," URI Online Judge, [Online]. Available: https://www.urionlinejudge.com.br/judge/en/problems/view/2849. [Accessed 10 October 2018].

[2] GeeksforGeeks, "Quickselect Algorithm - GeeksforGeeks," GeeksforGeeks, 2017. [Online]. Available: https://www.geeksforgeeks.org/quickselect-algorithm/. [Accessed 12 Desember 2018].

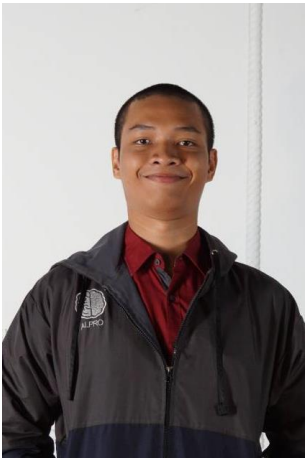
[3] P. P. Kube, "Traversing Data Structure," Department of Computer Science and Engineering , 2012. [Online]. Available: http://cseweb.ucsd.edu/~kube/cls/100/Lectures/lec3.huffman2/lec3-5.html. [Accessed Desember 2018].

[4] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, "More Geometric Data Structures," in *Computational Geometry: algorithms and applications (2nd ed.)*, New York, Springer, 2000, p. 227.

[5] GeeksforGeeks, "Segment Tree | Set 1 (Sum of given range) - GeeksforGeeks," GeeksforGeeks, 2013. [Online]. Available: https://www.geeksforgeeks.org/segment-tree-set-1-sum-of-given-range/. [Accessed November 2018].

Halaman ini sengaja dikosongkan

BIODATA PENULIS



Aldi Febriansyah, lahir di Surabaya tanggal 12 September 1996. Penulis merupakan anak ketiga dari 3 bersaudara. Penulis telah menempuh pendidikan formal TK Mangku Negoro Surabaya, SD Negeri Sememi 2 Surabaya (2002-2008), SMP Negeri 26 Surabaya (2008-2011) dan SMA Negeri 5 Surabaya (2011-2014). Penulis melanjutkan studi kuliah program sarjana di Jurusan Teknik Informatika ITS.

Selama kuliah di Teknik Informatika ITS, penulis pernah menjadi asisten dosen dan praktikum untuk mata kuliah Dasar Pemrogramman (2015 dan 2016), dan Struktur Data (2015). Selama menempuh perkuliahan penulis juga aktif di kegiatan organisasi dan kepanitiaan diantaranya menjadi staff Departemen Kaderisasi dan Pemetaan HMTC ITS (2015), staff ahli Departemen Kaderisasi dan Pemetaan HMTC ITS (2016), Wakil Ketua II Schematics 2015. Penulis juga kerap mengikuti lomba-lomba tingkat nasional, salah satunya Mandiri Hackathon (2016) dan meraih juara II, serta menjadi finalis pada Gemastik 10 yang diselenggarakan di Universitas Indonesia. Penulis juga sering melakukan pekerjaan proyek pembuatan aplikasi web dan android *mobile* (2015-2017). Hingga akhirnya penulis memutuskan untuk membangun sebuah *start up*, PT Global Medika Digitama (2017-sekarang),

yang berfokus pada perkembangan teknologi informasi pada fasilitas kesehatan dan penunjang kinerja petugas medis. Penulis dapat dihubungi melalui surel aldi@medify.id.