



TUGAS AKHIR - KI141502

**STUDI PERMASALAHAN K-MOST PROMISING PRODUCTS
BERBASIS INTERVAL WAKTU PADA DATA MULTIDIMENSI
DENGAN SERIAL WAKTU**

HAFARA FIRDAUSI
NRP 05111540000043

Dosen Pembimbing 1
Bagus Jati Santoso, S.Kom., Ph.D.

Dosen Pembimbing 2
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

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

Halaman ini sengaja dikosongkan



TUGAS AKHIR - KI141502

**STUDI PERMASALAHAN K-MOST PROMISING PRODUCTS
BERBASIS INTERVAL WAKTU PADA DATA MULTIDIMENSI
DENGAN SERIAL WAKTU**

HAFARA FIRDAUSI
NRP 05111540000043

Dosen Pembimbing 1
Bagus Jati Santoso, S.Kom., Ph.D.

Dosen Pembimbing 2
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

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

Halaman ini sengaja dikosongkan



UNDERGRADUATE THESES - KI141502

**ANSWERING K-MOST PROMISING PRODUCTS QUERY
BASED ON TIME INTERVALS ON MULTIDIMENSIONAL TIME
SERIES DATA**

HAFARA FIRDAUSI
NRP 05111540000043

Supervisor 1
Bagus Jati Santoso, S.Kom., Ph.D.

Supervisor 2
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

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

Halaman ini sengaja dikosongkan

**IMPLEMENTASI STRUKTUR DATA ROPE PADA STUDI
KASUS PERMASALAHAN SPOJ ALPHABETIC ROPE**

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:

Desy Nurbaiti Rahmi
NRP: 5114 100 030

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Rully Soelaiman, S.Kom., M.Kom.
NIP: 197002131994021001

Abdul Munif, S.Kom., M.Sc.
NIP: 198608232015041004



**SURABAYA
JANUARI 2018**
VII

Halaman ini sengaja dikosongkan

STUDI PERMASALAHAN K-MOST PROMISING PRODUCTS BERBASIS INTERVAL WAKTU PADA DATA MULTIDIMENSI DENGAN SERIAL WAKTU

Nama : HAFARA FIRDAUSI
NRP : 05111540000043
Departemen : Informatika FTIF-ITS
Pembimbing I : Bagus Jati Santoso, S.Kom., Ph.D.
Pembimbing II : Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Abstrak

Permasalahan alphabetic rope merupakan sebuah permasalahan yang melibatkan sebuah rentang pencarian. Tipe query secara umum dibagi menjadi dua yaitu, operasi pencarian dan perubahan. Operasi perubahan pada suatu rentang akan menyebabkan perubahan hasil pencarian selanjutnya. Untuk menangani berbagai permasalahan pada operasi alphabetic rope yang harus dilakukan sehingga dibutuhkan struktur data yang mampu mendukung operasi-operasi tersebut dengan efisien.

Pada Tugas Akhir ini akan dirancang penyelesaian permasalahan alphabetic rope antara lain operasi pencarian karakter pada indeks ke-y pada konfigurasi rope saat ini, operasi memotong segmen rope pada indeks ke-x sampai y dan menggabungkan pada bagian depan rope, dan operasi memotong segmen rope pada indeks ke-x sampai y dan menggabungkan pada bagian belakang rope. Struktur data klasik yang biasa digunakan dalam penyelesaian permasalahan ini adalah stuktur data String. Namun penggunaan algoritma String masih kurang efisien dalam hal kecepatan dan kebutuhan memori.

Pada Tugas Akhir ini digunakan struktur data Rope untuk menyelesaikan operasi-operasi tersebut. Hasil uji coba menunjukkan program menghasilkan jalur yang benar dan memiliki pertumbuhan waktu secara logaritmik dengan kompeksitas waktu sebesar $O(\log N)$ per query.

X

Kata Kunci: concatenation, rope, split, string

ANSWERING K-MOST PROMISING PRODUCTS QUERY BASED ON TIME INTERVALS ON MULTIDIMENSIONAL TIME SERIES DATA

Name	:	HAFARA FIRDAUSI
NRP	:	05111540000043
Major	:	Informatics Department Faculty of IT-ITS
Supervisor I	:	Bagus Jati Santoso, S.Kom., Ph.D.
Supervisor II	:	Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Abstract

Alphabetic rope computation is a problem which involves a specific range of data. Generally it divide by two type of query, range search and update. An update operation would have an impact for the following range search query. To handle various problem in alphabetic rope, an efficient data structure is needed to support the operations.

This undergraduate thesis will be designed problem solving for alphabetic rope query variation such as find the y-th position on current rope, cut the rope segment from x-th to y-th to join at the front of rope and cut the rope segment from x-th to y-th and join at the back of rope. Well known data structures, e.g string are commonly used for solving the this problem. But those algorithm were not efficient enough in the term of running time and memory usage.

In this undergraduate thesis Rope data structure will be used instead for solving those alphabetic rope variations. The program had been tested and proved that it produces correct result and running in $O(\log N)$ per query.

Keywords: concatenation, rope, split, string

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah Swt. atas pertolongan dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

STUDI PERMASALAHAN K-MOST PROMISING PRODUCTS BERBASIS INTERVAL WAKTU PADA DATA MULTIDIMENSI DENGAN SERIAL WAKTU.

Penelitian Tugas Akhir ini dilakukan untuk memenuhi salah satu syarat meraih gelar Sarjana di Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya.

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

Penulis juga mengucapkan banyak 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:

1. Ibu, bapak, kedua adik, Akbar dan Alam, serta segenap keluarga yang selalu menunggu kedatangan penulis di rumah dan senantiasa memberikan perhatian, dukungan, serta kasih sayang yang menjadi semangat dan motivasi bagi diri penulis.
2. Bapak Bagus Jati Santoso, S.Kom., Ph.D. selaku dosen pembimbing yang telah banyak meluangkan waktu untuk memberikan ilmu, nasihat, motivasi, pandangan dan

bimbingan kepada Penulis baik selama Penulis menempuh masa kuliah maupun selama pengerjaan Tugas Akhir ini.

3. Ibu Henning Titi Ciptaningtyas, S.Kom., M.Kom. selaku dosen pembimbing yang telah memberikan ilmu, dan masukan kepada Penulis.
4. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku Kepala Departemen Informatika ITS pada masa pengerjaan Tugas Akhir, Bapak Radityo Anggoro, S.Kom., M.Sc. selaku koordinator Tugas Akhir, dan segenap dosen dan karyawan Informatika yang telah memberikan ilmu, waktu, dan pengalamannya.
5. Seluruh teman-teman Laboratorium Arsitektur dan Jaringan Komputer angkatan 2015, Nahda, Satria, Hana, Fuad, Didin, Awan, dan Penyok, yang telah menjadi teman berbagi cerita suka dan duka, serta selalu

Penulis mohon maaf apabila masih ada kekurangan pada Tugas Akhir ini. Penulis juga mengharapkan kritik dan saran yang membangun untuk pembelajaran dan perbaikan di kemudian hari. Semoga melalui Tugas Akhir ini Penulis dapat memberikan kontribusi dan manfaat yang sebaik-baiknya.

Surabaya, Juni 2019

Hafara Firdausi

DAFTAR ISI

SAMPUL	i
LEMBAR PENGESAHAN	viii
ABSTRAK	ix
ABSTRACT	xi
KATA PENGANTAR	xiii
DAFTAR ISI	xv
DAFTAR TABEL	xix
DAFTAR GAMBAR	xxi
DAFTAR KODE SUMBER	xxiii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah	3
1.4 Tujuan	3
1.5 Manfaat	4
1.6 Metodologi	4
1.7 Sistematika Penulisan	5
BAB II TINJAUAN PUSTAKA	7
2.1 Daftar Simbol	7
2.2 Data	8
2.2.1 Data Multidimensi	8
2.2.2 Data Serial Waktu	9
2.2.3 Data Multidimensi dengan Serial Waktu	9
2.3 <i>Skyline</i>	10
2.4 Dominansi Dinamis	11
2.5 <i>Dynamic Skyline</i>	13
2.6 <i>Reverse Skyline</i>	14
2.7 Kueri <i>k</i> -Most Promising Products (<i>k</i> -MPP)	16
2.7.1 <i>Uniform Product Adoption</i> (UPA)	16

2.7.2	Strategi Pemilihan Produk	17
2.8	Kueri <i>k-Most Promising Products</i> (k-MPP) Berbasis Interval Waktu	18
2.9	Python	18
2.10	Flask	19
BAB III ANALISIS DAN PERANCANGAN SISTEM		21
3.1	Daftar Istilah	21
3.2	Analisis Sistem	21
3.2.1	Analisis Permasalahan	21
3.2.2	Deskripsi Umum Sistem	21
3.3	Perancangan Sistem	22
3.3.1	<i>Data Preprocessing</i>	22
3.3.2	Proses Utama	22
BAB IV IMPLEMENTASI		23
4.1	Lingkungan Implementasi	23
4.2	Rancangan Data	23
4.2.1	Data Masukan	24
4.2.2	Data Keluaran	24
4.3	Implementasi Algoritma	24
4.3.1	<i>Header</i> yang Diperlukan	24
4.3.2	Variabel Global	25
4.3.3	Implementasi Fungsi Main	25
4.3.4	Implementasi Struct Node	27
4.3.5	Implementasi Fungsi Newnode	27
4.3.6	Implementasi Fungsi Build	28
4.3.7	Implementasi Fungsi Getsize	28
4.3.8	Implementasi Fungsi Split	29
4.3.9	Implementasi Fungsi Random	30
4.3.10	Implementasi Fungsi Concate	31
4.3.11	Implementasi Fungsi Insert	32
4.3.12	Implementasi Fungsi Mutable Begin	32
4.3.13	Implementasi Fungsi Mutable End	33
4.3.14	Implementasi Fungsi Print	35

BAB V UJI COBA DAN EVALUASI	37
5.1 Lingkungan Uji Coba	37
5.2 Uji Coba Kebenaran	37
5.3 Uji Coba Kinerja	44
5.3.1 Operasi 1 Menggabungkan <i>Rope</i> pada Posisi Awal	44
5.3.2 Operasi 2 Menggabungkan <i>Rope</i> pada Posisi Akhir	44
5.3.3 Operasi 3 Mencetak Karakter pada Indeks ke-Y	46
5.4 Analisis Hasil Uji Coba	47
BAB VI KESIMPULAN	51
6.1 Kesimpulan	51
6.2 Saran	52
DAFTAR PUSTAKA	53
BIODATA PENULIS	55

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Tabel 2.1	Daftar Simbol (Bagian 1)	7
Tabel 2.2	(a) <i>Dataset</i> produk P dan (b) <i>dataset</i> preferensi pelanggan C	13
Tabel 5.1	Kecepatan Maksimal, Minimal dan Rata-Rata dari Hasil Uji Coba Sebanyak 15 Kali pada Situs Pengujian SPOJ	43

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Gambar 2.1	<i>Dataset</i> produk <i>smartphone</i> P dan hasil <i>skyline</i> -nya	11
Gambar 2.2	Komputasi <i>dynamic skyline</i> dari pelanggan c_5	14
Gambar 4.1	Ilustrasi Penyimpanan Hasil Operasi <i>Split Rope</i> pada Struktur Data Pair	29
Gambar 4.2	Ilustrasi Operasi <i>Concat</i> . Setiap <i>Node</i> Berisi Sebuah Karakter dan Nilai Berat Masing-Masing <i>Node</i>	31
Gambar 4.3	Ilustrasi Operasi <i>Mutable Begin</i> . Setiap <i>Node</i> Berisi Sebuah Karakter dan Nilai Prioritas Masing-Masing <i>Node</i>	34
Gambar 4.4	Ilustrasi Operasi <i>Mutable End</i> . Setiap <i>Node</i> Berisi Sebuah Karakter dan Nilai Prioritas Masing-Masing <i>Node</i>	36
Gambar 5.1	Contoh kasus uji permasalahan <i>Alphabetic Rope</i>	38
Gambar 5.2	Pembentukan <i>Root Rope</i>	38
Gambar 5.3	Pembentukan <i>Rope</i> pada Tingkat ke-2	39
Gambar 5.4	Pembentukan <i>Rope</i> pada Tingkat ke-3	39
Gambar 5.5	Struktur <i>Rope</i> yang Terbentuk	40
Gambar 5.6	Konfigurasi <i>Rope</i> Setelah Operasi 2 0 5	41
Gambar 5.7	Hasil Luaran Program pada Contoh Kasus Uji <i>Alphabetic Rope</i>	42
Gambar 5.8	Hasil Uji Coba pada Situs Penilaian SPOJ	42
Gambar 5.9	Hasil Uji Coba pada Situs Penilaian SPOJ	43
Gambar 5.10	Hasil Uji Coba pada Operasi 1 dengan Jumlah <i>Query</i> Tetap dan Panjang <i>String</i> Bertambah	45

Gambar 5.11 Hasil Uji Coba pada Operasi 1 dengan Jumlah <i>String</i> Tetap dan Jumlah <i>Query</i> Bertambah	45
Gambar 5.12 Hasil Uji Coba pada Operasi 2 dengan Jumlah <i>Query</i> Tetap dan Jumlah <i>String</i> Bertambah	46
Gambar 5.13 Hasil Uji Coba pada Operasi 2 dengan Panjang <i>String</i> Tetap dan Jumlah <i>Query</i> Bertambah	47
Gambar 5.14 Hasil Uji Coba pada Operasi 3 dengan Jumlah <i>Query</i> Tetap dan Panjang <i>String</i> Bertambah	48
Gambar 5.15 Hasil Uji Coba pada Operasi 3 dengan Jumlah <i>String</i> Tetap dan Jumlah <i>Query</i> Bertambah	48

DAFTAR KODE SUMBER

Kode Sumber 4.1	<i>Header</i> yang diperlukan	25
Kode Sumber 4.2	Variabel Global	25
Kode Sumber 4.3	Fungsi Main	26
Kode Sumber 4.4	Fungsi Struct Node	27
Kode Sumber 4.5	Fungsi Newnode	27
Kode Sumber 4.6	Fungsi Build	28
Kode Sumber 4.7	Fungsi Getsize	28
Kode Sumber 4.8	Fungsi Split(1)	29
Kode Sumber 4.9	Fungsi Split(2)	30
Kode Sumber 4.10	Fungsi Random	30
Kode Sumber 4.11	Fungsi Concate(1)	31
Kode Sumber 4.12	Fungsi Concate(2)	32
Kode Sumber 4.13	Fungsi Insert	32
Kode Sumber 4.14	Fungsi Mutable Begin	33
Kode Sumber 4.15	Fungsi Mutable End	35
Kode Sumber 4.16	Fungsi Print	35

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

Kemajuan ilmu pengetahuan dan teknologi telah merevolusi cara produsen dalam melakukan bisnis. Produsen dapat mengumpulkan data preferensi pelanggan terhadap produk dan fitur produk dari data penjualan mereka. Selain itu, maraknya penggunaan *World Wide Web* untuk menjual produk secara *online* juga memungkinkan produsen mengumpulkan data preferensi pelanggan terhadap fitur produk perusahaan lain.

Tak hanya itu, produsen dapat memanfaatkan data preferensi pelanggan tersebut secara cerdas, misalnya untuk mengidentifikasi produk pesaing dan pembeli potensial untuk produk mereka. Produsen juga dapat mencari produk apa saja yang paling banyak diminati oleh pelanggan sehingga ia dapat menentukan produk mana yang harus dipilih untuk strategi *targeted marketing* supaya lebih tepat sasaran dan bertahan di pasar global.

Saat ini, sudah ada komputasi yang dapat menyelesaikan masalah pemilihan produk dengan memanfaatkan data preferensi pelanggan. *k-Most Promising Products* (*k-MPP*) [1] adalah sebuah strategi pemilihan produk dengan melakukan pencarian *k* produk yang paling banyak diminati oleh pelanggan.

Komputasi *k-MPP* menggunakan dua tipe kueri *skyline*, yaitu *dynamic skyline* [2] dan *reverse skyline* [7]. Kueri *dynamic skyline* digunakan untuk mengambil data produk berdasarkan sudut

pandang pelanggan, sedangkan kueri *reverse skyline* digunakan untuk mengambil data pelanggan berdasarkan sudut pandang produsen.

Muncul sebuah pertanyaan, “*Apakah produk yang paling banyak diminati pelanggan selalu sama dari waktu ke waktu?*”. Tentu saja para produsen tidak akan berdiam diri. Produk-produk baru akan terus bermunculan seiring dengan berjalaninya waktu. Begitu pula produk yang paling diminati pelanggan juga ikut berubah karena produk-produk baru tersebut kemungkinan dapat mengungguli produk top sebelumnya.

Sebagai contoh, *smartphone* A adalah produk yang paling banyak diminati oleh pelanggan pada bulan Januari hingga Juni 2018, namun pada bulan Juli posisinya tergeser oleh *smartphone* B yang fitur-fiturnya lebih disukai oleh pelanggan. Dari ilustrasi tersebut, dapat diketahui bahwa interval waktu juga merupakan faktor penting yang harus dipertimbangkan dalam proses pencarian k produk yang paling banyak diminati oleh pelanggan karena sangat berpengaruh terhadap hasil pencarian.

Pertanyaan baru yang mungkin akan diajukan oleh produsen atau analis pemasaran adalah “*k produk apa saja yang paling banyak diminati oleh pelanggan pada bulan Januari hingga Desember 2018?*”. Dalam hal ini, bulan Januari hingga Desember 2018 disebut dengan interval waktu kueri dan data produk yang berbasis interval waktu disebut dengan data *time series* atau serial waktu [?].

Untuk menjawab pertanyaan tersebut, dibutuhkan penyesuaian dan modifikasi terhadap kueri *k-Most Promising Products (k-MPP)* dan kerangka kerja algoritme pemrosesan kueri yang sudah ada supaya dapat diimplementasikan pada data multidimensi dengan serial waktu. Kerangka kerja tersebut menggunakan struktur data *grid-based index* dan teknik komputasi paralel supaya pemrosesan data dapat dilakukan secara cepat dan akurat [1].

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana desain dan implementasi struktur data dan algoritme untuk menjawab kueri *k-Most Promising Products* (*k-MPP*) berbasis interval waktu pada data multidimensi dengan serial waktu?
2. Bagaimana kinerja dari struktur data dan algoritme yang dibangun untuk menjawab kueri *k-Most Promising Products* (*k-MPP*) berbasis interval waktu pada data multidimensi dengan serial waktu?
3. Bagaimana strategi yang optimal untuk meningkatkan efisiensi komputasi *k-Most Promising Products* (*k-MPP*) berbasis interval waktu pada data multidimensi dengan serial waktu?

1.3 Batasan Masalah

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Struktur data dan algoritme dalam komputasi *k-Most Promising Products* (*k-MPP*) hanya dapat menyimpan dan memproses nilai numerik.
2. Implementasi struktur data dan algoritme menggunakan bahasa pemrograman Python.
3. *Dataset* yang digunakan adalah data asli dan sintetis.

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Merancang dan mengimplementasikan struktur data dan algoritme untuk menjawab kueri *k-Most Promising Products*

(*k*-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu.

2. Mengevaluasi kinerja dari struktur data dan algoritme yang dibangun untuk menjawab kueri *k*-Most Promising Products (*k*-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu.
3. Mengimplementasikan strategi yang optimal untuk meningkatkan efisiensi komputasi *k*-Most Promising Products (*k*-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu.

1.5 Manfaat

Manfaat yang diharapkan dari penulisan Tugas Akhir ini adalah dapat mendesain dan mengimplementasikan struktur data dan algoritme yang tepat untuk menjawab kueri *k*-Most Promising Products (*k*-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu. Tugas Akhir ini juga diharapkan dapat memberikan kontribusi pada perkembangan ilmu pengetahuan dan teknologi informasi.

1.6 Metodologi

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

1. Penyusunan proposal Tugas Akhir

Tahap awal untuk memulai penggerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir yang berisi gagasan untuk menyelesaikan permasalahan *rope* pada studi kasus menjawab kueri *k*-Most Promising Products (*k*-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu.

2. Studi literatur

Pada tahap ini dilakukan pencarian informasi dan studi literatur mengenai pengetahuan atau metode yang dapat digunakan dalam penyelesaian masalah. Informasi didapatkan dari materi-materi yang berhubungan dengan algoritma yang digunakan untuk penyelesaian permasalahan ini, materi-materi tersebut didapatkan dari buku, jurnal, maupun internet.

3. Desain

Pada tahap ini dilakukan desain rancangan algoritma yang digunakan dalam solusi untuk pemecahan menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktu.

4. Implementasi perangkat lunak

Pada tahap ini dilakukan implementasi atau realiasi dari rancangan desain algoritma yang telah dibangun pada tahap desain ke dalam bentuk program.

5. Uji coba dan evaluasi

Pada tahap ini dilakukan uji coba kebenaran implementasi. Pengujian kebenaran dilakukan pada sistem penilaian daring SPOJ sesuai dengan masalah yang dikerjakan untuk diuji apakah luaran dari program telah sesuai.

6. Penyusunan buku Tugas Akhir

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

1.7 Sistematika Penulisan

Berikut adalah sistematika penulisan buku Tugas Akhir ini:

1. BABI: PENDAHULUAN

Bab ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, 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 algoritma 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 dan saran yang didapat dari hasil uji coba yang telah dilakukan.

BAB II

TINJAUAN PUSTAKA

Bab ini menjelaskan dasar teori yang digunakan dalam analisis, perancangan, dan implementasi struktur data dan algoritme untuk menjawab permasalahan *k-Most Promising Products* (k-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu yang diangkat dalam Tugas Akhir ini.

2.1 Daftar Simbol

Tabel 2.1 menunjukkan daftar simbol yang digunakan untuk memudahkan beberapa penjelasan pada bab ini berikut dengan deskripsinya.

Tabel 2.1 Daftar Simbol (Bagian 1)

Simbol	Deskripsi
ob	Objek data
$ob_1 \prec ob_2$	Objek data ob_1 mendominasi ob_2
$ob_1 \prec_{ob_3} ob_2$	Objek data ob_1 mendominasi ob_2 secara dinamis berdasarkan ob_3
P	<i>Dataset</i> produk
C	<i>Dataset</i> pelanggan (preferensi pelanggan)
p	Sebuah data produk dalam P , dinotasikan sebagai $p \in P$
c	Sebuah data pelanggan dalam C , dinotasikan sebagai $c \in C$
d	Jumlah dimensi/atribut data
i	Dimensi ke-1, ..., d
j	Timestamp

Simbol	Deskripsi
$DSL(c)$	<i>Dynamic skyline</i> dari pelanggan c

2.2 Data

Data merupakan elemen yang esensial dalam sebuah sistem informasi. Menurut Checkland dan Holwell [?], data adalah representasi dari fakta, konsep, ataupun instruksi secara formal yang digunakan untuk komunikasi, interpretasi, maupun pemrosesan. Sehingga, sebuah sistem informasi harus menerima *input* berupa data supaya dapat diproses menjadi informasi tertentu sesuai dengan kebutuhan pengguna dan mengeluarkannya sebagai *output*.

2.2.1 Data Multidimensi

Model data multidimensi adalah sebuah cara pandang yang melihat data dari berbagai sudut pandang atau dimensi. Model data ini memiliki struktur yang disesuaikan untuk mengoptimalkan analisis berdasarkan data dari *relational database* dan diolah sehingga informasi dapat dikategorikan. Model data multidimensi merupakan variasi dari model relasional yang menggunakan struktur multidimensi untuk menyusun data dan menjelaskan relasi antar data.

Struktur multidimensi merepresentasikan dimensi-dimensi data dalam bentuk kubus. Jika sebuah data multidimensi memiliki lebih dari tiga dimensi, maka disebut dengan *hypercube* [5]. Dalam implementasinya, data multidimensi disajikan dalam bentuk *array* multidimensi yang masing-masing nilai dalam selnya dapat diakses menggunakan sebuah indeks.

Data multidimensi banyak digunakan untuk analisis. Selama beberapa tahun terakhir, konsep data multidimensi telah menjadi hal yang fundamental dalam sistem pengambil keputusan, seperti sistem *data warehouse* [5].

Contoh data multidimensi

2.2.2 Data Serial Waktu

Data *time series* atau serial waktu adalah nilai-nilai suatu variabel yang berurutan menurut waktu. Data *time series* memiliki nilai dan *timestamp*, sehingga data diurutkan berdasarkan waktu atau *timestamp*-nya. Pada Gambar 5, diberikan contoh sebuah *time series dataset* S yang setiap datanya diindeks ke dalam suatu *array*. Pada contoh ini, kita asumsikan bahwa *timestamp* adalah bilangan bulat positif. Nilai $s_1 \in S$ pada *timestamp* j dinotasikan sebagai $s_1[j]$, sehingga *time series* s_1 jika ditulis secara berurutan menjadi $s_1[1], s_1[2], \dots$, dan seterusnya [6].

Contoh data serial waktu

2.2.3 Data Multidimensi dengan Serial Waktu

Untuk menjawab permasalahan yang diangkat pada Tugas Akhir ini, data yang digunakan merupakan penggabungan dari kedua jenis data di atas, yaitu data multidimensi dengan serial waktu.

Data multidimensi dengan serial waktu adalah data *multi-attribute* yang memiliki interval waktu berupa *timestamp* awal dan akhir, dinotasikan dengan $[i : j]$. Interval waktu menggambarkan bahwa setiap data memiliki waktu hidup tertentu, misalnya pada dataset produk (P), setiap produk $p \in P$ memiliki waktu kapan ia pertama kali diproduksi dan kapan ia tidak diproduksi lagi. Sebagai contoh lain, pada dataset pelanggan (C), setiap pelanggan $c \in C$ memiliki waktu kapan ia lahir dan kapan ia meninggal dunia.

Contoh data multidimensi dengan serial waktu

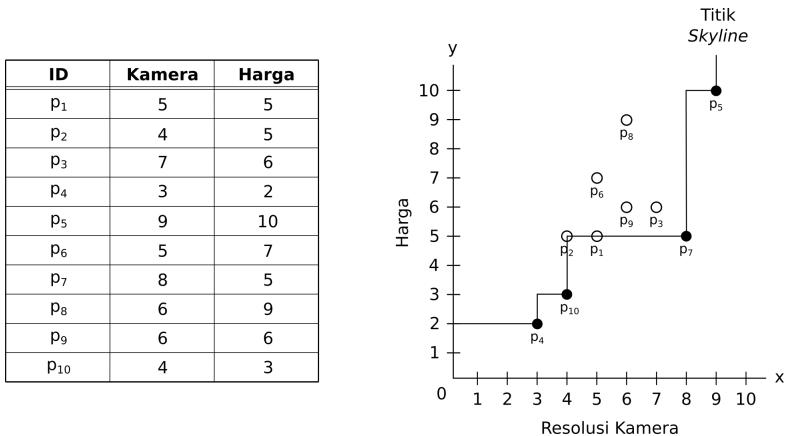
2.3 Skyline

Komputasi *skyline* telah menarik perhatian yang cukup besar dari peneliti sejak diperkenalkan pada komunitas basis data [4], terutama mengenai metode progresif yang dapat mengembalikan hasil kueri dengan cepat tanpa perlu membaca keseluruhan data [2]. Tujuan dari komputasi *skyline* adalah mencari data yang “menarik” dari suatu himpunan data [4], yaitu data yang tidak didominasi oleh data lain atau data yang paling unggul.

Pada *dataset* produk P yang setiap datanya direpresentasikan sebagai titik d -dimensi, sebuah titik p_1 dikatakan mendominasi titik lain p_2 , dinotasikan dengan $p_1 \prec p_2$, jika nilai p_1 tidak lebih besar dari p_2 pada semua dimensi dan ada nilai p_1 yang lebih kecil dari p_2 minimal pada satu dimensi. Secara matematika, relasi $p_1 \prec p_2$ terbentuk jika dan hanya jika:

- (a) $p_1^i \leq p_2^i, \forall i \in [1, \dots, d]$
- (b) $p_1^i < p_2^i, \exists i \in [1, \dots, d]$

Sebagai contoh, seseorang ingin mencari produk *smartphone* terbaik, yaitu *smartphone* yang memiliki harga termurah dan resolusi kamera terbesar. Pada Gambar 2.1, diberikan *dataset* produk *smartphone* P yang memiliki atribut harga dan resolusi kamera. Setiap datanya direpresentasikan sebagai titik pada bidang dua dimensi, yakni sumbu x adalah resolusi kamera dan sumbu y adalah harga *smartphone*.



Gambar 2.1 *Dataset* produk *smartphone* P dan hasil *skyline*-nya

Berdasarkan Gambar 2.1, produk *smartphone* yang terbaik adalah produk p_4 , p_5 , p_7 , dan p_{10} karena tidak ada titik yang lebih baik dari titik-titik tersebut pada semua dimensi, sedangkan *smartphone* p_2 dan p_1 tidak dapat menjadi *skyline* karena didominasi oleh *smartphone* p_7 pada dimensi x . *Smartphone* p_4 , p_5 , p_7 , dan p_{10} disebut juga dengan titik *skyline*.

Saat ini, komputasi *skyline* telah banyak digunakan sebagai operator pengambilan keputusan multikriteria dan perencanaan bisnis [3]. Ada beberapa pengembangan dari komputasi *skyline*, seperti *dynamic skyline* dan *reverse skyline*.

2.4 Dominansi Dinamis

Berdasarkan definisi "Skyline" yang telah dijelaskan pada poin sebelumnya, jika diberikan *dataset* yang sama, maka hasil *skyline* dari *dataset* tersebut pasti akan selalu sama (statis). Oleh karena itu, para ahli juga menyebut *original skyline* sebagai *static skyline* [3]. Ada suatu kasus ketika perhitungan *skyline* didasarkan pada titik kueri. Jika diberikan *dataset* yang sama, namun titik

kuerinya berbeda, maka hasil *skyline*-nya akan berubah tergantung pada titik kuerinya. Inilah yang kemudian disebut dengan *dynamic skyline* yang memiliki sifat dominansi dinamis.

Sebagai ilustrasi, diberikan *dataset* produk P dan *dataset* pelanggan (preferensi pelanggan) C yang setiap datanya direpresentasikan sebagai objek data d -dimensi. Data produk dan pelanggan pada dimensi ke- i dinotasikan sebagai p^i dan c^i , $i \leq d$. Setiap objek data hanya dapat menyimpan nilai numerik pada setiap dimensinya. Untuk menggambarkan objek data secara umum, kecuali dispesifikkan menjadi data produk atau pelanggan, digunakan notasi ob .

Suatu objek data ob_1 dikatakan mendominasi objek data ob_2 secara dinamis berdasarkan objek data ob_3 , dinotasikan dengan $ob_1 \prec_{ob_3} ob_2$, jika nilai ob_1 dekat dengan ob_3 pada semua dimensi dan ada nilai ob_1 yang lebih dekat dengan ob_3 dibandingkan nilai ob_2 dengan ob_3 minimal pada satu dimensi. Secara matematika, relasi $ob_1 \prec_{ob_3} ob_2$ terbentuk jika dan hanya jika:

$$(a) \quad |ob_3^i - ob_1^i| \leq |ob_3^i - ob_2^i|, \forall i \in [1, \dots, d]$$

$$(b) \quad |ob_3^i - ob_1^i| < |ob_3^i - ob_2^i|, \exists i \in [1, \dots, d]$$

Pada Tabel 2.2, diberikan contoh *dataset* produk dan preferensi pelanggan. Berdasarkan definisi dominansi dinamis yang telah dijelaskan sebelumnya, produk p_1 mendominasi produk p_4 secara dinamis berdasarkan pelanggan c_1 , dinotasikan dengan $p_1 \prec_{c_1} p_4$, karena memenuhi kedua syarat dominansi dinamis yakni (a) $|c_1^1 - p_1^1| = |5 - 5| = 0 \leq |c_1^1 - p_4^1| = |5 - 5| = 0$ dan (b) $|c_1^2 - p_1^2| = |2 - 9| = 7 < |c_1^2 - p_4^2| = |2 - 14| = 12$.

Tabel 2.2 (a) *Dataset* produk P dan (b) *dataset* preferensi pelanggan C

(a)			(b)		
id	dim1	dim2	id	dim1	dim2
p_1	5	9	c_1	5	2
p_2	10	16	c_2	8	10
p_3	8	10	c_3	17	10
p_4	5	14	c_4	9	7
p_5	12	6	c_5	10	12
p_6	20	6	c_6	16	14
p_7	16	10	c_7	20	13
p_8	9	11	c_8	15	8
p_9	20	5	c_9	20	17
p_{10}	6	20	c_{10}	12	4

Sebaliknya, jika berdasarkan preferensi pelanggan c_5 , maka produk p_4 -lah yang mendominasi p_1 secara dinamis, dinotasikan dengan $p_4 \prec_{c_5} p_1$, karena (a) $|c_5^1 - p_4^1| = |10 - 5| = 5 \leq |c_5^1 - p_1^1| = |10 - 5| = 5$ dan (b) $|c_5^2 - p_4^2| = |12 - 14| = 2 < |c_5^2 - p_1^2| = |12 - 9| = 3$. Preferensi pelanggan inilah yang disebut dengan titik kueri.

Dominansi dinamis memiliki peran yang sangat penting dalam komputasi *dynamic skyline* [2], sehingga dipelajari secara luas untuk membangun hubungan antara pelanggan dan produk yang baik [1].

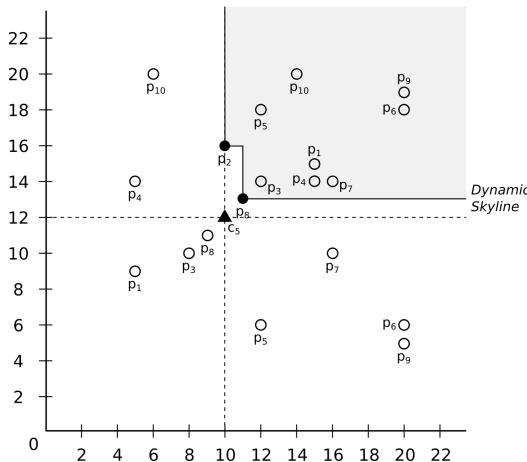
2.5 Dynamic Skyline

Kueri *dynamic skyline* dalam komputasi k-MPP digunakan untuk mencari produk terbaik dari sudut pandang pelanggan [1]. *Dynamic skyline* [2] dari seorang pelanggan $c \in C$, dinotasikan dengan $DSL(c)$, berisi semua produk $p_1 \in P$ yang tidak didominasi oleh produk lain $p_2 \in P$ berdasarkan preferensi

pelanggan c , $p_2 \not\prec_c p_1$.

Menggunakan contoh *dataset* pada Tabel 2.2, *dynamic skyline* dari pelanggan c_5 adalah produk p_2 dan p_8 karena produk tersebut tidak didominasi oleh produk lain berdasarkan preferensi pelanggan c_5 . Jika titik kueri diubah, maka hasil *skyline*-nya juga akan berubah.

Dynamic skyline dapat dihitung menggunakan algoritme komputasi *skyline* tradisional [4], yaitu mentransformasikan semua titik $p \in P$ ke *data space* baru dengan menganggap titik c sebagai titik asal dan jarak absolut titik p ke c digunakan sebagai fungsi pemetaan seperti yang ditunjukkan pada Gambar 2.2. Fungsi pemetaan f^i didefinisikan sebagai $f^i(p^i) = |c^i - p^i|$.



Gambar 2.2 Komputasi *dynamic skyline* dari pelanggan c_5

2.6 Reverse Skyline

Dalam komputasi k-MPP, kueri *reverse skyline* digunakan untuk mencari pelanggan potensial dari sudut pandang produsen [1]. *Reverse skyline* [7] dari sebuah produk $p_1 \in P$, dinotasikan dengan $RSL(p_1)$, berisi semua pelanggan $c \in C$ yang memiliki p_1

pada hasil *dynamic skyline*-nya.

Sebagai contoh, berdasarkan dataset yang diberikan pada Tabel 2.2, *reverse skyline* dari produk p_4 adalah pelanggan c_3 , c_4 , dan c_5 karena masing-masing pelanggan tersebut memiliki p_4 dalam *dynamic skyline*-nya.

** Gambar **

Ada beberapa tahapan yang harus dilakukan dalam komputasi *reverse skyline* [1]. Pertama, menghitung *midpoint skyline* (juga dikenal sebagai *mid-skyline* [8]) pada setiap *orthant* dari produk p_4 . Setiap produk p memiliki 2^d *orthant* pada data d -dimensi. *Mid-point* atau titik tengah antar produk, misalnya p_4 (yang menjadi titik kueri) dan p_2 , dihitung menggunakan rumus berikut:

$$m_2^i = \frac{(p_1^i + p_2^i)}{2} \quad (2.1)$$

Setelah itu, menentukan *midpoint skyline* setiap *orthant*.

Langkah kedua, mengecek apakah pelanggan $c \in C$ didominasi oleh *midpoint skyline* m berdasarkan produk p_4 atau tidak. Pelanggan c dikatakan didominasi oleh *midpoint skyline* m jika dan hanya jika:

- (a) $|p_4^i - m^i| \leq |p_4^i - c^i|, \forall i \in [1, \dots, d]$
- (b) $|p_4^i - m^i| < |p_4^i - c^i|, \exists i \in [1, \dots, d]$

Apabila c tidak didominasi oleh *midpoint skyline* m berdasarkan produk p_4 , maka c menjadi hasil dari *reverse skyline* p_4 , dinotasikan dengan $RSL(p_4)$. Komputasi *reverse skyline* pada p_4 ditunjukkan pada Gambar ??.

2.7 Kueri *k*-Most Promising Products (k-MPP)

Islam dan Liu dalam penelitiannya [1] telah memodelkan kueri *k*-Most Promising Products (k-MPP) dan kerangka kerja algoritme untuk memproses kueri tersebut.

2.7.1 Uniform Product Adoption (UPA)

Uniform Product Adoption (UPA) adalah sebuah pemodelan yang mengasumsikan bahwa semua produk $p \in P$ yang muncul pada hasil *dynamic skyline* pelanggan $c \in C$, dinotasikan $DSL(c)$ akan saling berkompetisi satu sama lain untuk menarik pelanggan c , sehingga produk-produk tersebut memiliki probabilitas yang sama untuk dibeli oleh pelanggan c .

Probabilitas produk p dibeli oleh pelanggan c , dinotasikan dengan $Pr(c, p|P)$ dapat dijelaskan oleh persamaan berikut:

$$Pr(c, p|P) = \begin{cases} \frac{1}{|DSL(c)|} & \text{if } p \in DSL(c) \\ 0 & \text{otherwise} \end{cases} \quad (2.2)$$

Berdasarkan persamaan 2.2, dapat dipastikan bahwa setiap produk yang muncul dalam $DSL(c)$ memiliki kesempatan yang sama untuk dipilih oleh pelanggan c . Sebaliknya, produk yang tidak muncul dalam $DSL(c)$ tidak memiliki kesempatan sama sekali untuk dipilih oleh c .

$$\sum_{\forall p \in P} Pr(c, p|P) = 1 \quad (2.3)$$

Sebagai contoh, probabilitas produk p_8 dibeli oleh pelanggan c_5 , dinotasikan dengan $Pr(c_5, p_8|P)$ adalah $\frac{1}{|DSL(c_5)|} = \frac{1}{2}$.

2.7.1.1 Market Contribution

Market contribution atau kontribusi pasar sebuah produk $p \in P$ diukur dari total jumlah pelanggan yang diharapkan

yang mungkin lebih memilih membeli produk p dibandingkan produk lain p' . Asumsinya jika seorang pelanggan memiliki dua produk atau lebih dalam hasil *dynamic skyline*-nya, maka ia akan memberikan bobot yang sama pada produk-produk tersebut sebagaimana yang sudah dijelaskan pada persamaan 2.2. Sehingga, kontribusi pasar sebuah produk didapatkan dari hasil akumulasi bobot yang didapatkan dari semua pelanggan $c \in C$.

Kontribusi pasar produk p , dinotasikan dengan $E(C, p|P)$, diperoleh dengan menambahkan probabilitas produk $Pr(c, p|P)$ dari setiap pelanggan $c \in C$ sebagai berikut:

$$E(C, p|P) = \sum_{\forall c \in C} Pr(c, p|P) \quad (2.4)$$

Karena pelanggan c yang muncul pada hasil *reverse skyline* produk p , dinotasikan dengan $RSL(p)$, dan probabilitas produk p dipilih oleh pelanggan yang tidak memiliki p pada hasil *dynamic skyline*-nya adalah nol (pada persamaan 2.2), maka persamaan 2.4 dapat disederhanakan menjadi:

$$E(C, p|P) = \sum_{\forall c \in RSL(p)} Pr(c, p|P) \quad (2.5)$$

Sebagai contoh, ...

Perhitungan kontribusi pasar juga dapat dilakukan pada sekumpulan produk atau *subset* produk P' , dinotasikan dengan $E(C, P'|P)$, yang dijelaskan pada persamaan 2.6.

$$E(C, P'|P) = \sum_{\forall p \in P'} E(C, p|P) \quad (2.6)$$

2.7.2 Strategi Pemilihan Produk

Strategi pemilihan produk yang diusulkan oleh Islam dan Liu dalam penelitiannya [1] bernama kueri *k-Most Promising Products*

(k-MPP).

Diberikan *dataset* produk P , *dataset* preferensi pelanggan C , dan bilangan bulat positif k yang lebih kecil dari $|P|$. Kueri k-Most Promising Products (k-MPP) [1], dinotasikan dengan $k - MPP(P, C, k)$, akan memilih *subset* k produk P' dari P yang memiliki kontribusi pasar lebih besar dibandingkan dengan *subset* k produk P'' dari P yang lain, sebagaimana yang dijelaskan pada persamaan 2.6.

2.8 Kueri *k*-Most Promising Products (k-MPP) Berbasis Interval Waktu

Interval waktu $[t_i : t_e](t_i \leq t_e)$ yang ditambahkan pada kueri *k*-Most Promising Products (k-MPP) pada Tugas Akhir ini digunakan untuk menentukan rentang waktu pencarian. Sehingga, kueri k-MPP yang dimodifikasi dinotasikan menjadi:

$$k - MPPTS(P, C, k, [t_i : t_e]) \quad (2.7)$$

2.9 Python

Python adalah bahasa pemrograman tingkat tinggi dan *open source* yang dikembangkan oleh Guido van Rossum pada akhir 1980-an. Saat ini, Python dikelola oleh *Python Software Foundation*. Python adalah bahasa multi-fungsi yang dapat digunakan untuk membuat *game*, *GUI (Graphical User Interface)*, dan mengembangkan aplikasi web. Karena Python adalah bahasa tingkat tinggi, membaca dan menulis kode Python semudah membaca dan menulis bahasa Inggris biasa [9].

Python adalah bahasa pemrograman *interpreter*, artinya setiap kali program berbahasa Python dijalankan, *interpreter*-nya akan berjalan dan mengeksekusi kode program baris demi baris, kemudian menterjemahkannya ke dalam bahasa mesin.

Python adalah bahasa pemrograman berorientasi objek

yang memungkinkan pengguna untuk mengelola dan mengontrol struktur data atau objek untuk membuat program. Semua objek, tipe data, fungsi, metode, dan kelas memiliki posisi yang sama dalam Python [9].

Pada Tugas Akhir ini, bahasa pemrograman Python digunakan untuk mengimplementasikan struktur data dan algoritme dari sistem perangkat lunak yang akan dibangun. Bahasa ini juga digunakan untuk membuat aplikasi web dan layanan *web server*.

2.10 Flask

Flask adalah kerangka kerja web berbahasa Python yang sederhana, ringan, dan mudah dikembangkan, sehingga Flask kerap disebut dengan *microframework*. Flask dibangun dari dua pustaka utama, yaitu Jinja *template engine* dan Werkzeug WSGI *toolkit*, serta memiliki lisensi BSD. Saat ini, Flask dikembangkan dan dikelola oleh *Pallets team* dan kontributor komunitas.

Kerangka kerja Flask digunakan untuk mengimplementasikan aplikasi web dan layanan *webserver* pada Tugas Akhir ini karena ringan dan lebih mudah digunakan dibanding dengan *framework* Python Django. Selain itu, Flask juga memiliki banyak dokumentasi dan tutorial yang dapat diikuti.

Halaman ini sengaja dikosongkan

BAB III

ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini akan dijelaskan mengenai analisis dan perancangan sistem perangkat lunak yang akan dibangun, meliputi struktur data, algoritme, dan arsitektur aplikasi.

3.1 Daftar Istilah

Beberapa daftar istilah yang digunakan dalam bab ini dapat dilihat pada Tabel ? untuk melengkapi definisi yang telah dijelaskan pada Tabel 2.1.

3.2 Analisis Sistem

Analisis sistem dibagi menjadi dua bagian, yaitu analisis permasalahan yang diangkat pada tugas akhir ini dan deskripsi umum sistem perangkat lunak yang akan dibangun.

3.2.1 Analisis Permasalahan

Permasalahan yang diangkat dalam tugas akhir ini adalah pencarian k produk yang paling menjanjikan pada interval waktu tertentu. Sebuah produk dikatakan ”menjanjikan” jika ia memiliki nilai kontribusi pasar yang besar, dinilai dari tingkat kedekatannya dengan preferensi masing-masing pelanggan.

3.2.2 Deskripsi Umum Sistem

3.3 Perancangan Sistem

3.3.1 *Data Preprocessing*

3.3.1.1 Struktur Data

3.3.1.2 Komputasi *Dynamic Skyline*

3.3.2 Proses Utama

3.3.2.1 Pemrosesan Kueri *k-Most Promising Products* (*k-MPP*)

BAB IV

IMPLEMENTASI

Pada bab ini dijelaskan mengenai implementasi dari desain dan algoritma penyelesaian menjawab kueri *k-Most Promising Products* (*k-MPP*) berbasis interval waktu pada data multidimensi dengan serial waktu.

4.1 Lingkungan Implementasi

Lingkungan implementasi dalam pembuatan Tugas Akhir ini meliputi perangkat keras dan perangkat lunak yang digunakan untuk penyelesaian menjawab kueri *k-Most Promising Products* (*k-MPP*) berbasis interval waktu pada data multidimensi dengan serial waktu adalah sebagai berikut:

1. Perangkat Keras:
 - Processor Intel(R) Core(TM) i3 - M 330 CPU @ 2.13GHz x 4
 - Memori 4 GB.
2. Perangkat Lunak:
 - Sistem operasi Ubuntu 14.04 LTS 64 bit.
 - *Text editor* Sublime Text 3.
 - *Compiler* g++ versi 4.3.2.

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 Data Masukan

Data masukan adalah data yang akan diproses oleh program sebagai masukan menggunakan 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 menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktu. Pada masing-masing berkas data masukan, baris pertama berupa sebuah bilangan bulat yang merepresentasikan jumlah kasus uji yang ada pada berkas tersebut. Untuk setiap kasus uji dengan tipe 1 dan 2, masukan berupa sebuah baris masukan yang terdiri dari dua buah parameter posisi berupa *x* dan *y*. Sedangkan pada kasus uji dengan tipe 3, masukan berupa sebuah parameter *y* yang merupakan indeks *string* yang dicari pada konfigurasi *rope* saat ini.

4.2.2 Data Keluaran

Data keluaran yang dihasilkan oleh program hanya berupa satu nilai, yaitu huruf pada indeks ke-*y* untuk setiap kasus uji dengan tipe 3.

4.3 Implementasi Algoritma

Pada subbab ini akan dijelaskan tentang implementasi proses algoritma secara keseluruhan berdasarkan desain yang telah dijelaskan pada Bab ??.

4.3.1 *Header* yang Diperlukan

Implementasi algoritma dengan pemanfaatan struktur data Rope untuk menyelesaikan menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktu membutuhkan empat buah *header* yaitu stdio.h,

cstdlib, cstring dan utility, seperti yang terlihat pada Kode Sumber 4.1.

```

1 #include <stdio.h>
2 #include <cstdlib>
3 #include <cstring>
4 #include <utility>
```

Kode Sumber 4.1 *Header* yang diperlukan

Header stdio.h berisi modul untuk menerima masukan dan memberikan keluaran. *Header* cstdlib berisi modul untuk manajemen memori dinamis, generasi bilangan acak, pemilahan dan konversi. *Header* cstring berisi modul yang memiliki fungsi-fungsi untuk melakukan pemrosesan *string*. *Header* utility mencakup berbagai modul yang menyediakan fungsionalitas mulai dari aplikasi perhitungan bit hingga aplikasi fungsi parsial. Contoh implementasinya penggunaan *pair*.

4.3.2 Variabel Global

Variabel global digunakan untuk memudahkan dalam mengakses data yang digunakan lintas fungsi. Kode sumber implementasi variabel global dapat dilihat pada Kode Sumber 4.2.

```

1 using namespace std;
2
3 const int N = 1e5 + 10;
```

Kode Sumber 4.2 Variabel Global

4.3.3 Implementasi Fungsi Main

Fungsi Main adalah implementasi algoritma yang dirancang pada Gambar ???. Setiap tipe memiliki operasi yang berbeda-beda. Untuk

operasi dengan tipe 3, masukan hanya berupa parameter y yang merupakan nilai posisi indeks yang dicari pada konfigurasi *rope* saat ini. Pada operasi dengan tipe 1 dan 2, baris masukan berupa nilai x dan y yang merupakan posisi indeks dari *string* pada *rope*. Setiap masukan dengan tipe 3 akan ditampilkan sebagai jawaban akhir dari permasalahan. Implementasi fungsi Main dapat dilihat pada Kode Sumber 4.3.

```
1 int main() {
2     char st[N];
3     scanf("%s", st);
4     Node* root = 0;
5     int Q, type;
6     root = insert(root, st);
7     scanf("%d", &Q);
8     while ( Q-- ) {
9         int x, y;
10        scanf("%d", &type);
11        if (type == 3) {
12            scanf("%d", &y);
13            printf("%c\n", print(root, y));
14        } else {
15            scanf("%d%d", &x, &y);
16            if(type == 1) {
17                root = mutable_begin(root, x, y - x + 1);
18            } else {
19                root = mutable_end(root, x, y - x + 1);
20            }
21        }
22    }
23    return 0;
24 }
```

Kode Sumber 4.3 Fungsi Main

4.3.4 Implementasi Struct Node

Fungsi Struct Node berisi atribut yang dimiliki *node* pada *tree*. Implementasi dari fungsi Struct Node dapat dilihat pada Kode Sumber 4.4.

```
1 struct Node {  
2     Node *left, *right;  
3     int size;  
4     char value;  
5     Node(char v) {  
6         left = right = 0;  
7         size = 1;  
8         value = v;  
9     }  
10    Node* update() {  
11        size = 1;  
12        if (left) size += left->size;  
13        if (right) size += right->size;  
14        return this;  
15    }  
16}  
17};
```

Kode Sumber 4.4 Fungsi Struct Node

4.3.5 Implementasi Fungsi Newnode

Fungsi Newnode digunakan untuk membentuk sebuah *node* yang berisikan karakter yang diberikan dan relasi terhadap karakter pada posisi yang bersebelahan. Sehingga membentuk sebuah *tree* seperti pada Gambar ???. Implementasi fungsi newnode dapat dilihat pada Kode Sumber 4.5.

```
1 Node* newnode(char c, Node* left, Node* right) {  
2     Node* r = new Node(c);  
3     r->left = left;  
4     r->right = right;  
5     r->update();
```

```

6     return r;
7 }
```

Kode Sumber 4.5 Fungsi Newnode

4.3.6 Implementasi Fungsi Build

Fungsi Build membangun struktur *tree* dari *rope* yang dilakukan dari karakter yang berada pada posisi indeks di tengah sampai pada karakter paling awal maupun akhir. Setiap *node* akan memiliki anak kiri dan anak kanan jika dan hanya jika masih terdapat *string* yang tersisa. Ilustrasinya dapat dilihat pada Gambar ???. Implementasi dari Fungsi Build dapat dilihat pada Kode Sumber 4.6.

```

1 Node* build(char* start, char* end) {
2     if ( start == end ) return NULL;
3     char* mid = start + (end - start)/2;
4     return newnode(*mid, build(start, mid), \
5         build(mid+1, end));
6 }
```

Kode Sumber 4.6 Fungsi Build

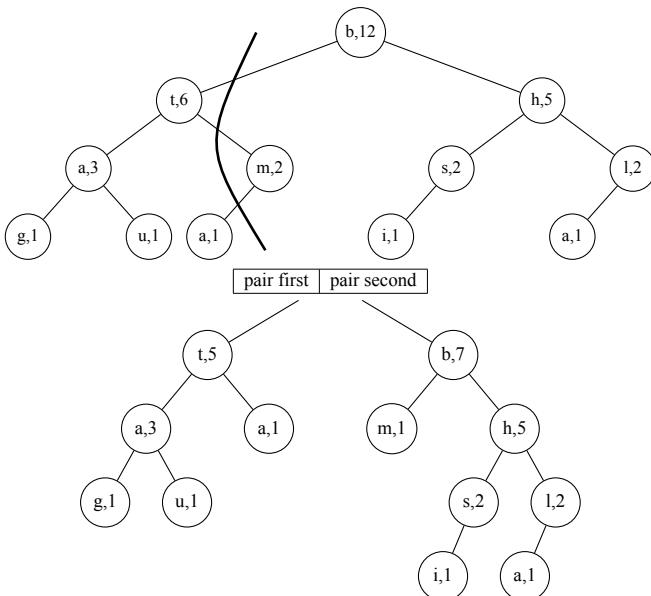
4.3.7 Implementasi Fungsi Getsize

Fungsi Getsize digunakan untuk mendapatkan berat *node* yang diperlukan. Implementasi dari fungsi Getsize dapat dilihat pada Kode Sumber 4.7.

```

1 int getSize(Node* o) {
2     return o ? o->size : 0;
3 }
```

Kode Sumber 4.7 Fungsi Getsize



Gambar 4.1 Ilustrasi Penyimpanan Hasil Operasi *Split Rope* pada Struktur Data Pair

4.3.8 Implementasi Fungsi Split

Fungsi *Split* memanfaatkan struktur data *Pair* yang berupa pasangan dari dua *pointer* menuju *node*. *Pointer node* pertama akan berisi semua *node* yang bernilai lebih kecil dari posisi indeks parameter *split*, sedangkan *pointer node* kedua berisi semua *node* yang bernilai lebih besar sama dengan posisi indeks parameter *split*.

Misalkan dilakukan *split* pada indeks ke-5 dari *rope* pada Gambar 4.1, *pointer node* pertama akan menyimpan semua *node* dari indeks ke-0 sampai dengan 4. Sedangkan *pointer node* kedua menyimpan *node* dari indeks ke-5 sampai akhir. Implementasi dari fungsi *split* dapat dilihat pada Kode Sumber 4.8 dan 4.9.

```
1 pair<Node*, Node*> split(Node* r, int pos) {
```

```

2     Node *R1 = 0, *R2 = 0;
3     if ( !r ) return make_pair(R1, R2);
4     int idx = getSize(r->left);
5     if ( idx < pos ) {

```

Kode Sumber 4.8 Fungsi Split(1)

```

1      pair<Node*, Node*> temp = \
2          split(r->right, pos - idx - 1);
3      r->right = temp.first;
4      R2 = temp.second;
5      R1 = r;
6  } else {
7      pair<Node*, Node*> temp = split(r->left, pos);
8      R1 = temp.first;
9      r->left = temp.second;
10     R2 = r;
11 }
12 r->update();
13 return make_pair(R1, R2);
14 }

```

Kode Sumber 4.9 Fungsi Split(2)

4.3.9 Implementasi Fungsi Random

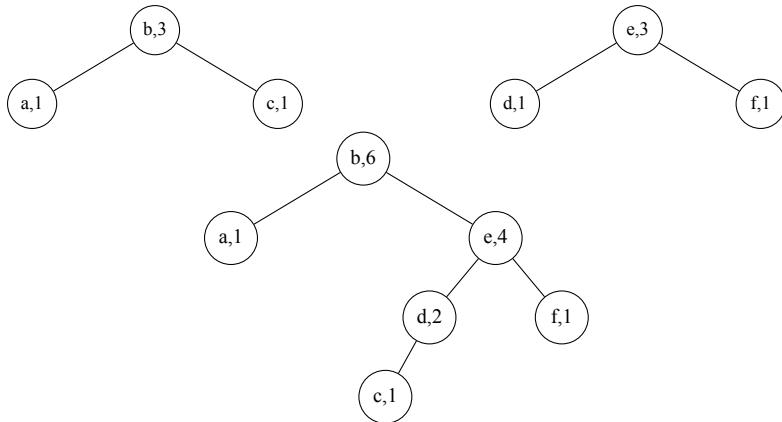
Fungsi Random digunakan untuk menentukan posisi *node* pada saat penggabungan dua buah *rope*. Digunakan fungsi Random agar posisi *node* tidak konstan dan berat suatu *rope* tidak selalu sama. Implementasi fungsi Random ditunjukkan pada Kode Sumber 4.10.

```

1 bool random(int a, int b) {
2     return rand() % ( a + b ) < a;
3 }

```

Kode Sumber 4.10 Fungsi Random



Gambar 4.2 Ilustrasi Operasi Concate. Setiap *Node* Berisi Sebuah Karakter dan Nilai Berat Masing-Masing *Node*

4.3.10 Implementasi Fungsi Concate

Fungsi Concate menggabungkan dua buah *rope* menjadi sebuah *rope* utuh. Untuk setiap *node* yang memiliki hasil modular nilai acak lebih kecil dari berat *node root rope* pertama, akan menjadi *root* dari *rope* yang baru terbentuk. Untuk *rope* kedua akan menjadi anak kanan atau anak kirinya, tergantung pada urutan BST yang seharusnya dibuat.

Misalkan terdapat dua buah *rope* R_1 yang berisi *string* "abc" dan R_2 yang berisi *string* "def" seperti pada Gambar 4.2. Apabila akan digabungkan dengan R_1 berada di posisi depan sehingga membentuk *string* "abcdef", maka semua *node* pada R_2 berada di posisi kanan *rope* R_1 . Jika nilai hasil modular R_1 lebih kecil dari berat *node root* R_1 , *node* tersebut akan menjadi *root* dari *rope* yang baru terbentuk. *Node root* R_2 akan menjadi anak kanannya. Jika *root* R_1 memiliki anak kanan, akan menjadi anak kiri dari *node* R_1 yang memiliki indeks 0. Implementasi Fungsi Concate ditunjukkan pada Kode Sumber 4.11 dan 4.12.

```

1 Node* concate(Node* R1, Node* R2) {
2     if ( !R1 || !R2) return R1 ? R1 : R2;

```

Kode Sumber 4.11 Fungsi Concate(1)

```

1     if (random(R1->size, R2->size)) {
2         R1->right = concate(R1->right, R2);
3         return R1->update();
4     } else {
5         R2->left = concate(R1, R2->left);
6         return R2->update();
7     }
8 }

```

Kode Sumber 4.12 Fungsi Concate(2)

4.3.11 Implementasi Fungsi Insert

Fungsi Insert adalah implementasi dari desain algoritma pada Gambar ???. Fungsi ini bertujuan untuk memasukkan data *string* ke dalam *rope*. Setiap *string* akan dimasukkan ke dalam suatu *node*. Setiap *node* hanya berisi oleh satu karakter pecahan dari *string* masukan. Implementasi dari Fungsi Insert dapat dilihat pada Kode Sumber 4.13.

```

1 Node* insert(Node* r, char s[]) {
2     Node* x = build(s, s + strlen(s));
3     return concate(r, x);
4 }

```

Kode Sumber 4.13 Fungsi Insert

4.3.12 Implementasi Fungsi Mutable Begin

Fungsi Mutable Begin adalah implementasi dari desain algoritma pada Gambar ???. Operasi ini dilakukan untuk menjawab

permasalahan pada subbab ?? dengan memanfaatkan dua buah operasi Split dan dua buah operasi Concat.

Misal dilakukan operasi 1 5 9 pada *rope* di Gambar 4.3. Maka *rope* akan dipotong pada posisi indeks ke-5 menghasilkan *rope* R_1 dan R_2 . Kemudian dilakukan *split* kedua kali pada indeks $y - x + 1$ pada *rope* R_2 menghasilkan *rope* R_{21} dan R_{22} . Sehingga menghasilkan tiga buah *rope* yang disimpan dalam struktur data Pair. Langkah selanjutnya dilakukan operasi Concat pada *rope* R_1 dengan R_{22} . Hasil penggabungan *rope* R_1 dengan R_{22} akan digabungkan dengan *rope* R_{21} . Pada operasi ini, *rope* R_{21} akan berada di posisi paling kiri dari keseluruhan *rope*. Dan menghasilkan sebuah *rope* utuh dengan urutan posisi *rope* saat ini adalah R_{21} , R_1 dan R_{22} . Implementasi Fungsi Mutable Begin dapat dilihat pada Kode Sumber 4.14.

```

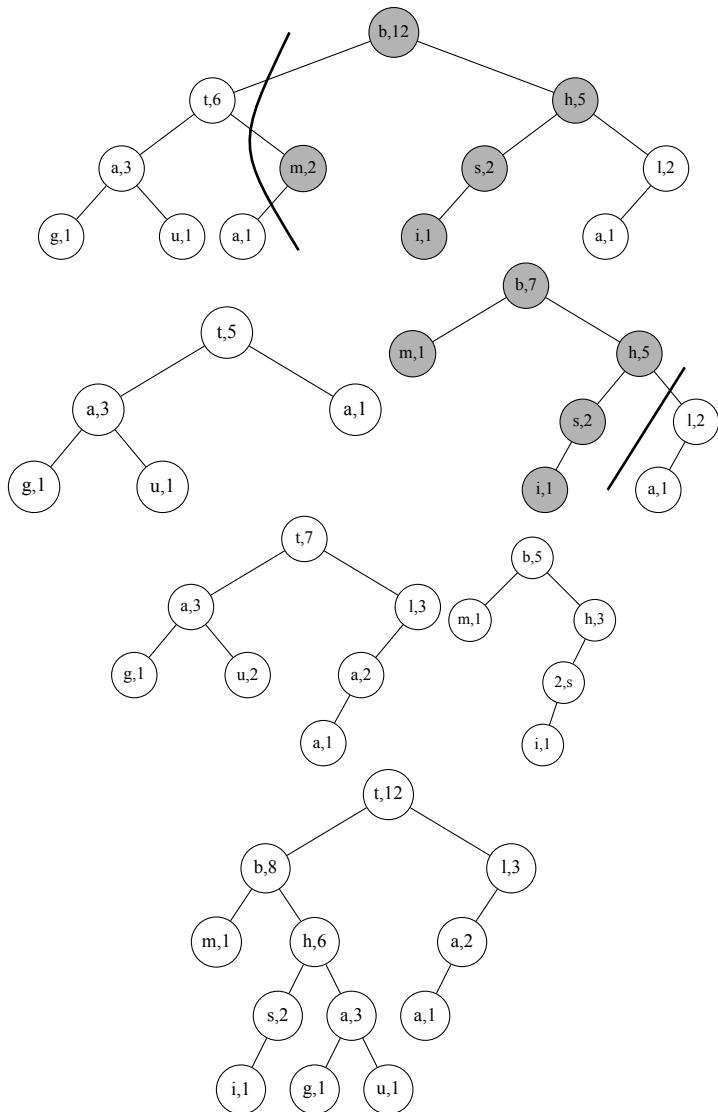
1 Node* mutable_begin(Node* r, int x, int len) {
2     pair<Node*, Node*> fir = split(r, x);
3     pair<Node*, Node*> sec = split(fir.second, len);
4     return concat(sec.first, concat(fir.first,\n
5         sec.second));
6 }
```

Kode Sumber 4.14 Fungsi Mutable Begin

4.3.13 Implementasi Fungsi Mutable End

Fungsi Mutable End dilakukan untuk menjawab permasalahan pada subbab ?? . Implementasinya dilakukan dengan memanfaatkan dua buah operasi Split dan dua buah operasi Concat.

Misal dilakukan operasi 2 5 9 pada *rope* di Gambar 4.4. Maka *rope* akan dipotong pada posisi indeks ke-5 menghasilkan *rope* R_1 dan R_2 . Kemudian dilakukan *split* kedua kali pada indeks $y - x + 1$ pada *rope* R_2 menghasilkan *rope* R_{21} dan R_{22} . Sehingga menghasilkan tiga buah *rope* yang disimpan dalam struktur data Pair. Langkah selanjutnya dilakukan operasi Concat pada *rope* R_1 dengan R_{22} .



Gambar 4.3 Ilustrasi Operasi Mutable Begin. Setiap *Node* Berisi Sebuah Karakter dan Nilai Prioritas Masing-Masing *Node*

Hasil penggabungan *rope* R_1 dengan R_{22} akan digabungkan dengan *rope* R_{21} . Pada operasi ini, *rope* R_{21} akan berada di posisi paling kanan dari keseluruhan *rope*. Dan menghasilkan sebuah *rope* utuh dengan urutan posisi *rope* saat ini adalah R_1 , R_{22} dan R_{21} . Implementasi Fungsi Mutable End dapat dilihat pada Kode Sumber 4.15.

```

1 Node* mutable_end(Node* r, int x, int len) {
2     pair<Node*, Node*> fir = split(r, x);
3     pair<Node*, Node*> sec = split(fir.second, len);
4     return concat(concat(fir.first, sec.second), \
5         sec.first);
6 }
```

Kode Sumber 4.15 Fungsi Mutable End

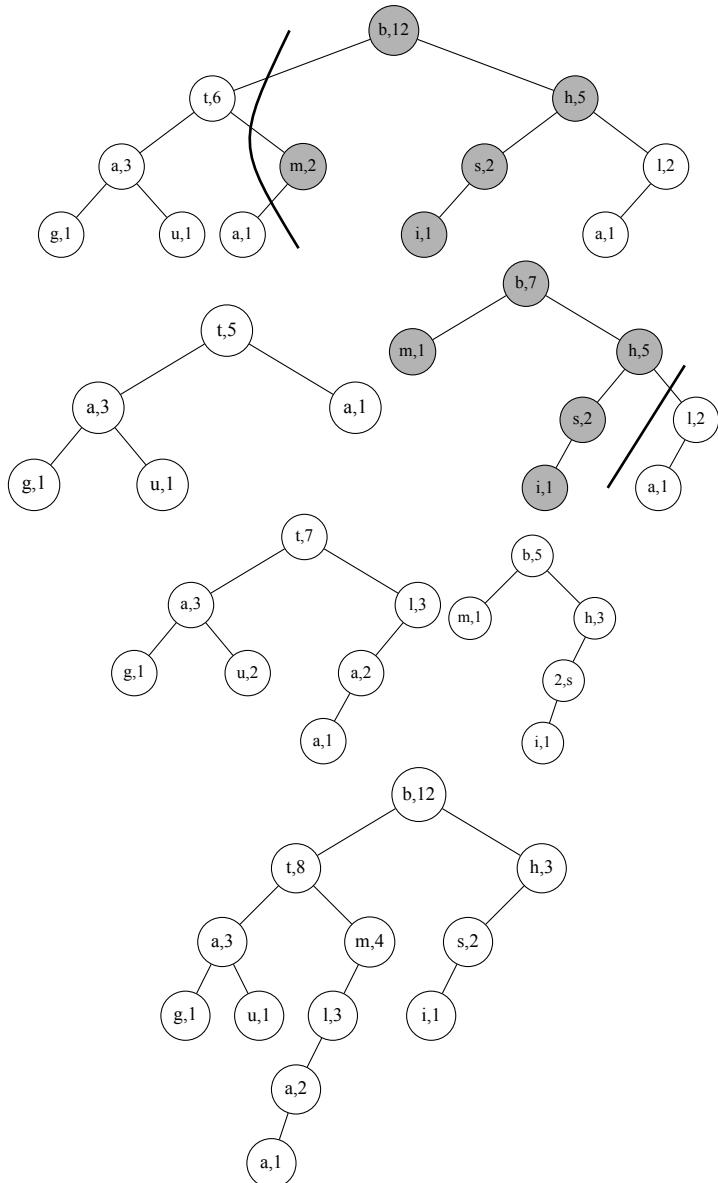
4.3.14 Implementasi Fungsi Print

Fungsi Print digunakan untuk menjawab menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktu sesuai dengan permasalahan pada subbab ???. Implementasi fungsi ini berdasarkan dari desain algorithma pada Gambar ???. Implementasi dari Fungsi Print dapat dilihat pada Kode Sumber 4.16.

```

1 char print(Node* r, int pos) {
2     int idx = getSize(r->left);
3     if ( idx == pos ) return r->value;
4     if ( pos <= idx ) return print(r->left, pos);
5     else return print(r->right, pos - idx - 1);
6 }
```

Kode Sumber 4.16 Fungsi Print



Gambar 4.4 Ilustrasi Operasi Mutable End. Setiap *Node* Berisi Sebuah Karakter dan Nilai Prioritas Masing-Masing *Node*

BAB V

UJI COBA DAN EVALUASI

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

5.1 Lingkungan Uji Coba

Linkungan uji coba yang digunakan adalah salah satu sistem yang digunakan situs penilaian daring SPOJ, yaitu kluster *Cube* dengan spesifikasi sebagai berikut:

1. Perangkat Keras:
 - Processor Intel(R) Pentium G860 CPU @ 3GHz.
 - Memory 1536 MB.
2. Perangkat Lunak:
 - Compiler g++ versi 4.3.2.

5.2 Uji Coba Kebenaran

Uji coba kebenaran dilakukan dengan analisis penyelesaian sebuah contoh kasus menggunakan pendekatan penyelesaian yang telah dijelaskan pada subbab ?? serta pengumpulan berkas kode sumber hasil implementasi ke dalam situs penilaian daring SPOJ.

Kasus yang akan digunakan sebagai bahan uji kebenaran dalam analisis penyelesaian menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktu menggunakan contoh kasus pada Gambar 5.1.

Mula-mula dalam setiap permasalahan adalah membangun struktur *rope* yang sesuai dengan *string* masukan. Operasi dasar pada sebuah *rope* adalah fungsi Insert, dimana dibentuk sebuah *tree*

```

gautambishal
3
3 1
2 0 5
3 1

```

Gambar 5.1 Contoh kasus uji permasalahan Alphabetic Rope

yang berfungsi untuk menyimpan potongan karakter pada *rope*. Selanjutnya *rope* akan dibangun pada *node root* yang dilakukan berdasarkan posisi tengah dari panjang *string*. Pembentukan ini dapat dilihat pada Gambar 5.2. Nilai pada tanda kurung siku menyatakan isi karakter pada *node* tersebut. Dikarenakan masih terdapat *string* yang tersisa maka proses pembentukan dilanjutkan untuk membentuk anak kiri dan anak kanan *root*. Rentang *string*

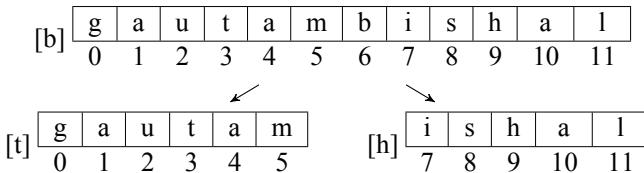
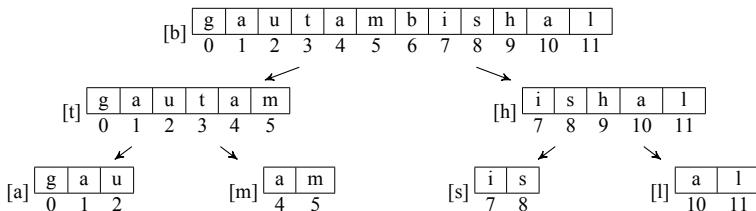
[b]	g	a	u	t	a	m	b	i	s	h	a	l
	0	1	2	3	4	5	6	7	8	9	10	11

Gambar 5.2 Pembentukan *Root Rope*

pada anak kiri diperoleh dari indeks 0 hingga nilai tengah-1 sedangkan rentang kanan diperoleh dari nilai tengah+1 hingga panjang *string*. Pembentukan anak dari *root* ditunjukkan pada Gambar 5.3. Pada setiap anak kiri dan anak kanan, posisi karakter yang berada ditengah-tengah *string* anak menjadi nilai dari *node* tersebut. Yang ditunjukkan oleh nilai di dalam tanda kurung siku. Dikarenakan masih terdapat *string* yang tersisa, proses pembentukan dilanjutkan untuk tingkat ke-3 yang ditunjukkan pada Gambar 5.4.

Untuk tahap selanjutnya dilakukan pembentukan *rope* pada tingkat ke-4 yang ditunjukkan pada Gambar 5.5.

Query pertama memiliki parameter *type* = 3, *y* = 0. Tahapan

Gambar 5.3 Pembentukan *Rope* pada Tingkat ke-2Gambar 5.4 Pembentukan *Rope* pada Tingkat ke-3

pertama dengan mencari karakter pada indeks ke- y . Untuk memperoleh jawaban dilakukan penelusuran sesuai dengan algoritma yang dijelaskan pada subbab ??.

Tahapan pencarian pada *rope* adalah sebagai berikut,

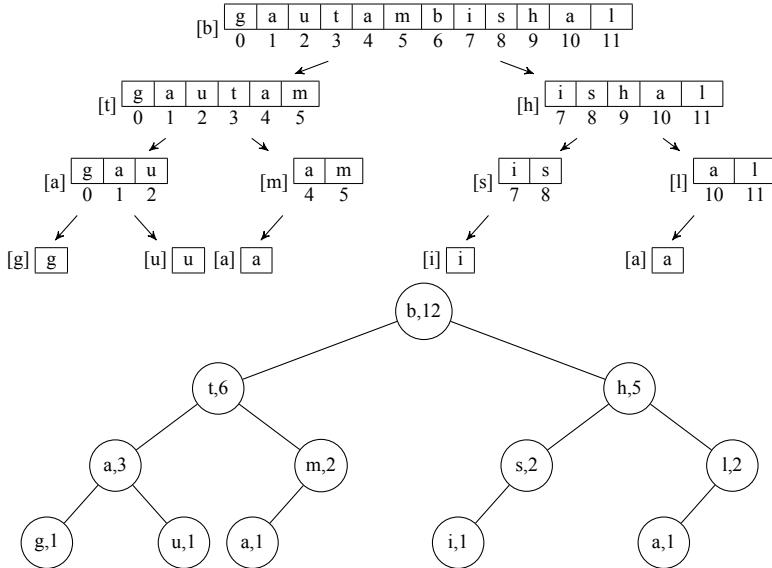
$y = 1$, $left.weight = 6$, $idx = left.weight$; $idx \leq y$;
dilanjutkan ke anak kiri;

$y = 1$, $left.weight = 3$, $idx = left.weight$; $idx \leq y$;
dilanjutkan ke anak kiri;

$y = 1$, $left.weight = 1$, $idx = left.weight$; $idx = y$;
kembalikan nilai *node* saat ini;

Maka jawaban untuk *query* ini adalah *a*.

Query kedua memiliki parameter $type = 2$, $x = 0$, $y = 5$.
Proses perubahan pada *rope* adalah sebagai berikut,

Gambar 5.5 Struktur *Rope* yang Terbentuk

$x = 0, y = 5, \text{left.weight} = 6, \text{idx} = \text{left.weight};$
 $\text{idx} \leq y;$ dilanjutkan ke anak kiri;

$x = 0, y = 5, \text{left.weight} = 3, \text{idx} = \text{left.weight};$
 $\text{idx} \leq y;$ dilanjutkan ke anak kiri;

$x = 0, y = 5, \text{left.weight} = 1, \text{idx} = \text{left.weight};$
 $\text{idx} \leq y;$ dilanjutkan ke anak kiri;

$x = 0, y = 5, \text{left.weight} = 0, \text{idx} = \text{left.weight};$
 $\text{idx} = y;$ kembalikan nilai *node* saat ini;

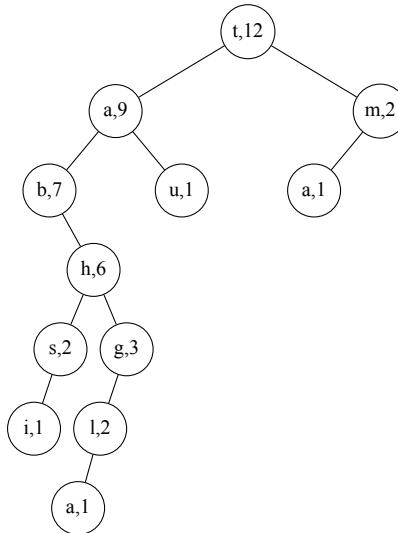
Dikarenakan $x = 0$ operasi Split menghasilkan *rope* R_1 yang berisi NULL dan R_2 sisa dari potongan *rope* yang berarti keseluruhan *rope* saat ini.

Proses perubahan selanjutnya adalah sebagai berikut,

$x = 0, y = 5, \text{left.weight} = 1, \text{len} = y - x + 1, \text{idx} = \text{left.weight}; \text{idx} = \text{len}$; kembalikan nilai *node* saat ini;

Setelah menemukan posisi *node* yang akan dipotong, hapus sambungan dari seluruh *node* yang memiliki nilai indeks kurang dari nilai indeks saat ini. Hasil potongan dari *rope* akan disimpan dengan menggunakan struktur data Pair yang bertipe *pointer node*. Sehingga menghasilkan dua buah *rope* baru R_{21} yang berisi string "gautam" dan R_{22} yang berisi string "bishal".

Potongan *rope* yang baru terbentuk akan digabungkan berdasarkan *type* = 2, dimana R_{21} berada diposisi belakang dari keseluruhan *rope*. Konfigurasi *rope* yang baru terbentuk berisi string "bishalgautam" yang ditunjukkan pada Gambar 5.6.



Gambar 5.6 Konfigurasi *Rope* Setelah Operasi 2 0 5

Query ketiga memiliki parameter $\text{type} = 3, y = 0$. Proses perubahan dan pencarian pada *rope* adalah sebagai berikut,

$y = 0, left.weight = 9, idx = left.weight; idx \leq y$; dilanjutkan ke anak kiri;

$y = 0, left.weight = 7, idx = left.weight; idx \leq y$; dilanjutkan ke anak kiri;

$y = 0, left.weight = 0, idx = left.weight; idx = y$; kembalikan nilai *node* saat ini;

Maka jawaban untuk *query* ini adalah *b*.

Secara terurut jawaban dari kedua *query* tersebut adalah *a* dan *b*. Kemudian sistem penyelesaian dijalankan dan diberi masukan sesuai kasus uji dari analisis sebelumnya dan hasil luaran sistem adalah *a* dan *b* seperti terlihat pada Gambar 5.7.

```
gautambishal
3
3 1
a
2 0 5
3 0
b
```

Gambar 5.7 Hasil Luaran Program pada Contoh Kasus Uji Alphabetic Rope

Selanjutnya dilakukan juga uji coba kebenaran dengan mengirimkan kode sumber program ke dalam situs penilaian daring SPOJ. Permasalahan yang diselesaikan adalah menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktu. Hasil uji kebenaran dan waktu eksekusi program pada saat pengumpulan kasus uji pada situs SPOJ ditunjukkan pada Gambar 5.8.

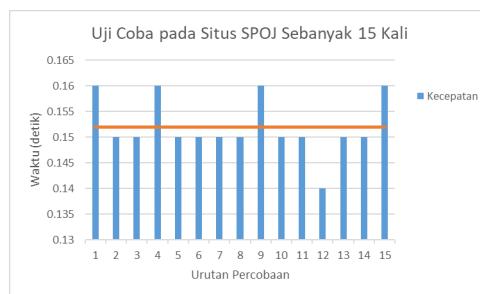
20966212	2018-01-11 18:18:01	Alphabetic Rope	accepted edit ideone.it	0.15	5.0M	C++ 4.3.2
----------	------------------------	-----------------	----------------------------	------	------	--------------

Gambar 5.8 Hasil Uji Coba pada Situs Penilaian SPOJ

Tabel 5.1 Kecepatan Maksimal, Minimal dan Rata-Rata dari Hasil Uji Coba Sebanyak 15 Kali pada Situs Pengujian SPOJ

Waktu Maksimal	0.16 detik
Waktu Minimal	0.14 detik
Waktu Rata-Rata	0.152 detik
Memori Maksimal	5.0 MB
Memori Minimal	5.0 MB
Memori Rata-rata	5.0 MB

Hal ini membuktikan bahwa implementasi yang dilakukan telah berhasil menyelesaikan menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktu dengan batasan-batasan yang telah ditetapkan. Setelah itu dilakukan pengiriman kode sumber implementasi sebanyak 15 kali untuk melihat variasi waktu dan memori yang dibutuhkan program. Hasil uji coba sebanyak 15 kali dapat dilihat pada Gambar ???. Grafik hasil uji coba sebanyak 15 kali ditunjukkan pada Gambar 5.9.



Gambar 5.9 Hasil Uji Coba pada Situs Penilaian SPOJ

Berdasarkan Tabel 5.1 dari percobaan pengujian yang dilakukan, didapat waktu rata-rata program yaitu 0.152 detik dan

penggunaan memori yang dibutuhkan program yaitu 5.0 MB. Hasil ini masih dibawah dari batas maksimal waktu dan memori pada menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktu, yaitu 1 detik dan 1536 MB.

5.3 Uji Coba Kinerja

Uji coba kinerja penyelesaian menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktu dilakukan dengan cara membandingkan waktu yang dibutuhkan untuk penyelesaian menggunakan *string* dengan struktur data *Rope* yang dibuat.

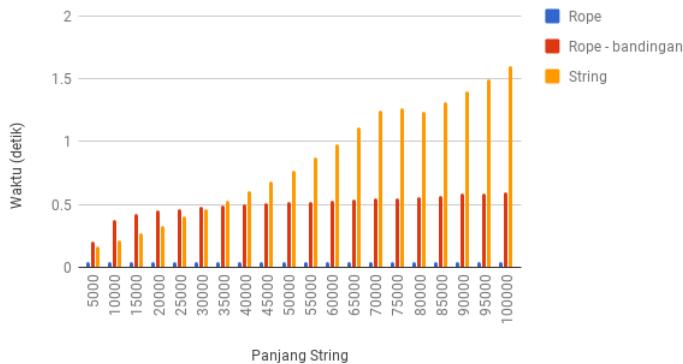
5.3.1 Operasi 1 Menggabungkan *Rope* pada Posisi Awal

Pada Gambar 5.10 terlihat bahwa untuk $N = 5000$ sampai $N = 10^5$ dengan banyak $Q = 10^5$, struktur *Rope* yang dibuat untuk penyelesaian menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktu menunjukkan waktu yang lebih cepat dibandingkan dengan algoritma *String* dan struktur data *Rope* pembanding. Sedangkan untuk $N = 10^5$ dengan $Q = 5000$ sampai $Q = 10^5$, pertumbuhan waktu secara logaritmik dan lebih kecil dibandingkan dengan algoritma *String* dan struktur data *Rope* pembanding yang ditunjukkan pada Gambar 5.11. Untuk tabel performa antara algoritma *String* dengan struktur data *Rope* yang telah dibuat dapat dilihat pada Tabel ?? dan ??.

5.3.2 Operasi 2 Menggabungkan *Rope* pada Posisi Akhir

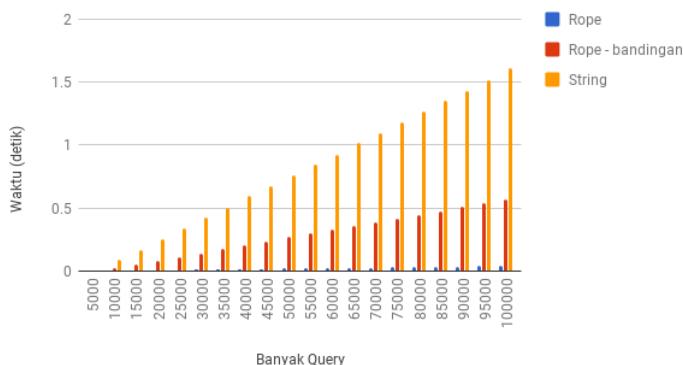
Pada Gambar 5.12 terlihat bahwa untuk $N = 5000$ sampai $N = 10^5$ dengan banyak $Q = 10^5$, struktur *Rope* yang dibuat untuk

Perbandingan Performa Operasi 1

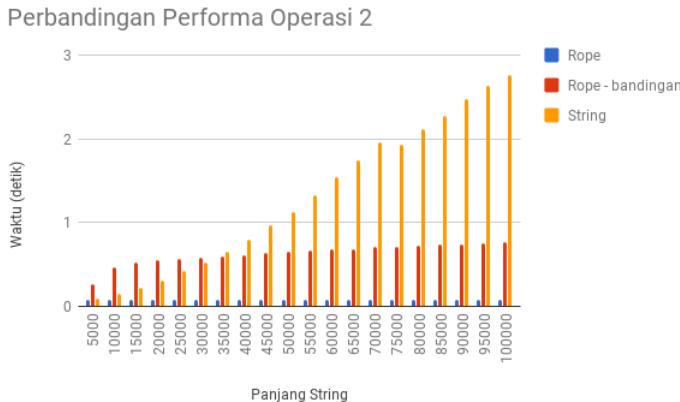


Gambar 5.10 Hasil Uji Coba pada Operasi 1 dengan Jumlah *Query* Tetap dan Panjang *String* Bertambah

Perbandingan performa Operasi 1



Gambar 5.11 Hasil Uji Coba pada Operasi 1 dengan Jumlah *String* Tetap dan Jumlah *Query* Bertambah



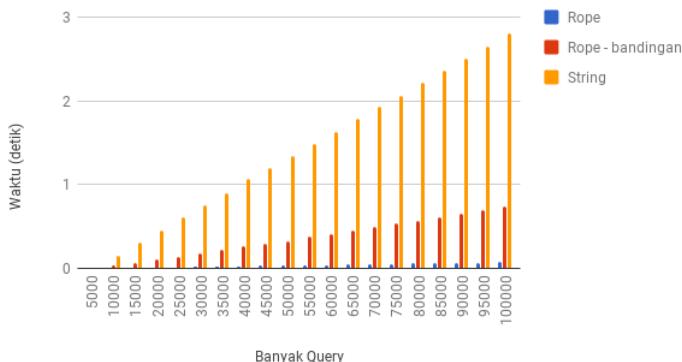
Gambar 5.12 Hasil Uji Coba pada Operasi 2 dengan Jumlah *Query* Tetap dan Jumlah *String* Bertambah

penyelesaian menjawab kueri *k-Most Promising Products* (*k-MPP*) berbasis interval waktu pada data multidimensi dengan serial waktu menunjukkan waktu yang lebih cepat dibandingkan dengan algoritma String dan struktur data Rope pembanding. Sedangkan untuk $N = 10^5$ dengan $Q = 5000$ sampai $Q = 10^5$, pertumbuhan waktu secara logaritmik dan lebih kecil dibandingkan dengan algoritma String dan struktur data Rope pembanding yang ditunjukkan pada Gambar 5.13. Untuk tabel performa antara algoritma String dengan struktur data Rope yang telah dibuat dapat dilihat pada Tabel ?? dan ??.

5.3.3 Operasi 3 Mencetak Karakter pada Indeks ke-Y

Pada Gambar 5.14 terlihat bahwa untuk $N = 5000$ sampai $N = 10^5$ dengan banyak $Q = 10^5$, algoritma String untuk penyelesaian menjawab kueri *k-Most Promising Products* (*k-MPP*) berbasis interval waktu pada data multidimensi dengan serial waktu

Perbandingan Performa Operasi 2



Gambar 5.13 Hasil Uji Coba pada Operasi 2 dengan Panjang *String* Tetap dan Jumlah *Query* Bertambah

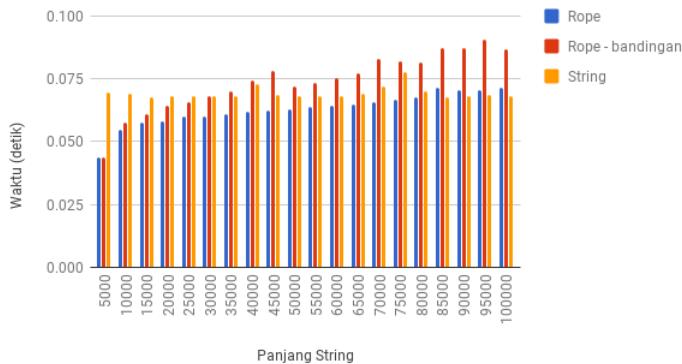
menunjukkan waktu yang lebih cepat dibandingan dengan struktur data Rope yang dibuat dan stuktur data Rope pembanding. Dan berlaku ketika menjalankan operasi untuk $N = 10^5$ dengan $Q = 5000$ sampai $Q = 10^5$ yang ditunjukkan pada Gambar 5.15. Untuk tabel performa antara algoritma String dengan struktur data Rope yang telah dibuat dapat dilihat pada Tabel ?? dan ??.

Perbedaan *rope* dengan *rope*-bandungan terletak pada operasi *insert*. Pada *rope* bandungan operasi *insert* dilakukan sama seperti algoritma *insert* pada BST. Sedangkan pada *rope*, *insert* dilakukan dengan membagi dua panjang *string* kemudian dimasukkan ke dalam *tree*.

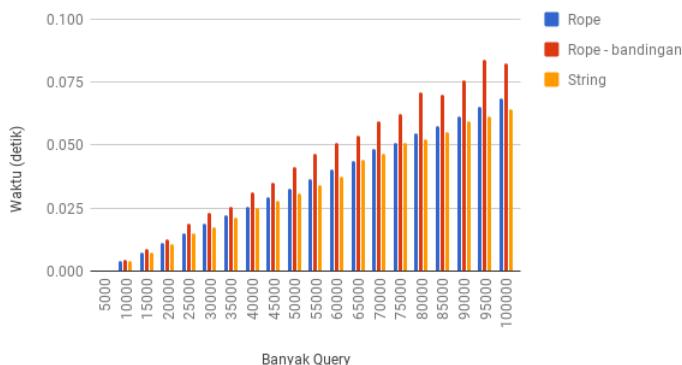
5.4 Analisis Hasil Uji Coba

Pada subbab ini akan dibahas mengenai analisis hasil uji coba yang dikerjakan pada subbab 5.2. Berdasarkan ketiga skenario yang telah dilakukan, masing-masing membuktikan kebenaran dan

Perbandingan Performa Operasi 3

Gambar 5.14 Hasil Uji Coba pada Operasi 3 dengan Jumlah *Query* Tetap dan Panjang *String* Bertambah

Perbandingan Performa Operasi 3

Gambar 5.15 Hasil Uji Coba pada Operasi 3 dengan Jumlah *String* Tetap dan Jumlah *Query* Bertambah

efisiensi hasil implementasi Tugas Akhir ini.

Dari ketiga hasil uji coba yang telah dilakukan, dapat dilihat bahwa hasil implementasi algoritma penyelesaian menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktu pada Tugas Akhir ini mengeluarkan hasil keluaran yang sama dengan jalur yang ditelusuri secara manual. Hasil uji coba kebenaran pada subbab 5.2 dapat digunakan sebagai acuan bahwa hasil keluaran program akan benar untuk segala jenis operasi pertanyaan dan perubahan.

Selanjutnya berdasarkan uji coba performa yang ditunjukkan pada Gambar ??, dapat dilihat bahwa memori dan waktu yang dibutuhkan untuk mengeksekusi program dapat dikategorikan efisien menurut situs SPOJ.

Proses *split*, *concat* dan *print* pada *rope* mengacu pada algoritma pencarian pada *tree* sehingga dapat disimpulkan bahwa kompleksitas waktu yang diperlukan sebesar $O(\log N)$. Pada proses pembuatan *rope* berdasarkan Kode Sumber 4.6 didapatkan kompleksitas waktu sebesar $O(N \log N)$. Sehingga dapat disimpulkan bahwa kompleksitas waktu untuk algoritma komputasi *string* pada Tugas Akhir ini adalah $O(\log N)$, dimana N merupakan panjang *string*. Sedangkan algoritma String membutuhkan kompleksitas sebesar $O(N)$ untuk setiap proses *split* dan *concat*[?].

Pada hasil perbandingan antara algoritma String dengan struktur Rope yang digunakan pada Tugas Akhir ini, ditunjukkan perbedaan pada kecepatan eksekusi program. Hal ini menunjukkan bahwa algoritma pada Tugas Akhir ini lebih efisien daripada algoritma String dan Rope-bandungan.

Halaman ini sengaja dikosongkan

BAB VI

KESIMPULAN

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

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan terhadap perancangan dan implementasi algoritma untuk menyelesaikan menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktu dapat diambil kesimpulan sebagai berikut:

1. Implementasi algoritma dengan menggunakan struktur data Rope dapat menyelesaikan menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktu dengan benar.
2. Kompleksitas waktu sebesar $O(\log N)$ dapat menyelesaikan menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktu.
3. Waktu yang dibutuhkan program untuk menyelesaikan menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktu minimum 0.14 detik, maksimum 0.16 detik dan rata-rata 0.152 detik. Memori yang dibutuhkan sebesar 5.0 MB.
4. Struktur data Rope yang dibuat sangat baik diaplikasikan untuk melakukan komputasi *string* yang panjang.

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 Rope adalah pendekatan yang sesuai untuk menyelesaikan permasalahan *string* yang sangat panjang.
2. Struktur data Rope dapat diimplementasikan pada bahasa pemrograman lain.

DAFTAR PUSTAKA

- [1] M. S. Islam and C. Liu, "Know Your Customer: Computing K-Most Promising Products," *The VLDB Journal*, pp. 545–570, 2016.
- [2] D. Papadias, Y. Tao, G. Fu and B. Seeger, "Progressive Skyline Computation in Database Systems," *ACM Transactions on Database Systems*, Vol. 30, No. 1, pp. 41–82, 2005.
- [3] L. Zou, L. Chen, M. T. Özsü and D. Zhao, "Dynamic Skyline Queries in Large Graphs," *DASFAA'10 Proceedings of the 15th International Conference on Database Systems for Advanced Applications - Volume Part II*, pp. 62-78, 2010.
- [4] S. Borzsonyi, D. Kossmann and K. Stocker, "The Skyline Operator," *In: ICDE*, pp. 421-430, 2001.
- [5] M. Golfarelli and S. Rizzi, "Introduction to Data Warehousing," in *Data Warehouse Design: Modern Principles and Methodologies*, New York: McGraw-Hill, 2009, pp. 1-42.
- [6] B. Jiang and J. Pei, "Online Interval Skyline Queries on Time Series," *IEEE International Conference on Data Engineering*, pp. 1036-1047, 2009.
- [7] E. Dellis and B. Seeger, "Efficient Computation of Reverse Skyline Queries," *VLDB Endowment*, pp. 291-302, 2007.
- [8] X. Wu, Y. Tao, R. C.-W. Wong, L. Ding and J. X. Yu, "Finding the Influence Set through Skylines," *EDBT*, pp. 1030-1041, 2009.

- [9] A. Johansen. (2016). *Python: The Ultimate Beginner's Guide!* [Online]. Available FTP: astronomi.erciyes.edu.tr
Directory: wp-content/uploads/astronom/pdf File:
Python%20-%20Andrew%20Johansen.pdf

BIODATA PENULIS



Hafara Firdausi, lahir di Surabaya pada tanggal 19 September 1998. Penulis merupakan anak pertama dari 3 bersaudara. Penulis telah menempuh pendidikan formal di SD Negeri Gelam 1 Candi (2004-2010), SMP Negeri 1 Sidoarjo (2010-2013), dan SMA Negeri 1 Sidoarjo (2013-2015). Kemudian, penulis melanjutkan studi kuliah program sarjana di Departemen Informatika, Institut Teknologi Sepuluh Nopember Surabaya.

Selama menempuh pendidikan di Informatika ITS, penulis pernah menjadi asisten dosen dan praktikum untuk mata kuliah Jaringan Komputer (2017-2018) dan Sistem Operasi (2019), serta menjadi Administrator Laboratorium Arsitektur dan Jaringan Komputer (AJK). Selain itu, penulis juga aktif dalam kegiatan organisasi dan kepanitiaan, antara lain menjadi Badan Pengurus Harian I 3D (Desain, Dekorasi, dan Dokumentasi) Schematics ITS 2017, staff dan staff ahli Departemen Media Informasi Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) ITS, dan Panitia Gemastik ke-11 Bidang Keamanan Jaringan dan Sistem Informasi (KJSI). Penulis dapat dihubungi melalui surel di hafarafirdausi@gmail.com.