



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**

**OBTAINING K-MOST PROMISING PRODUCTS BASED ON  
TIME INTERVAL 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 : 0511154000043  
Departemen : Informatika FTIK-ITS  
Pembimbing I : Bagus Jati Santoso, S.Kom., Ph.D.  
Pembimbing II : Henning Titi Ciptaningtyas, S.Kom., M.Kom.

## **Abstrak**

*Kemajuan ilmu pengetahuan dan teknologi telah mempengaruhi cara produsen dalam melakukan bisnis, yaitu dengan memanfaatkan data preferensi pelanggan untuk mendapatkan informasi, misalnya untuk mencari produk yang paling banyak diminati oleh pelanggan sehingga produsen dapat memilih produk dengan tepat untuk menarik lebih banyak pelanggan dan bertahan lama di pasar global.*

*Saat ini, sudah ada komputasi yang dapat menyelesaikan permasalahan tersebut. k-Most Promising Products (k-MPP) adalah sebuah strategi pemilihan produk dengan melakukan pencarian k-produk yang paling banyak diminati oleh pelanggan. Namun, hasil pencarian k-produk tidak mungkin tetap dari waktu ke waktu. Oleh karena itu, interval waktu adalah salah satu faktor yang harus dipertimbangkan dalam proses kueri karena sangat berpengaruh terhadap hasil pencarian. Permasalahan ini kemudian didefinisikan dalam penelitian ini sebagai "k-Most Promising Products berbasis interval waktu (k-MPPTI)".*

*Pada penelitian ini akan dirancang dan diimplementasikan struktur data dan algoritme yang tepat untuk menyelesaikan permasalahan tersebut. Data yang digunakan adalah data multidimensi, sehingga menggunakan struktur data yang memiliki fitur indexing, yaitu*

*array. Algoritme k-MPPTI menggunakan dua tipe kueri skyline, yaitu dynamic skyline dan reverse skyline. Selain itu, algoritme k-MPPTI juga mengimplementasikan teknik komputasi paralel supaya pemrosesan data menjadi lebih cepat.*

*Hasil uji coba menunjukkan bahwa algoritme k-MPPTI dapat memberikan hasil dengan waktu eksekusi ... kali lebih cepat dan penggunaan memori ... kali lebih hemat dibandingkan dengan algoritme konvensional Brute Force.*

**Kata Kunci:** *Strategi pemilihan produk, Kueri, Dynamic skyline, Reverse skyline, Interval waktu*

# **OBTAINING K-MOST PROMISING PRODUCTS BASED ON TIME INTERVAL 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**

*Kemajuan ilmu pengetahuan dan teknologi telah mempengaruhi cara produsen dalam melakukan bisnis, yaitu dengan memanfaatkan data preferensi pelanggan untuk mendapatkan informasi, misalnya untuk mencari produk yang paling banyak diminati oleh pelanggan sehingga produsen dapat memilih produk dengan tepat untuk menarik lebih banyak pelanggan dan bertahan lama di pasar global.*

*Saat ini, sudah ada komputasi yang dapat menyelesaikan permasalahan tersebut. k-Most Promising Products (k-MPP) adalah sebuah strategi pemilihan produk dengan melakukan pencarian k-produk yang paling banyak diminati oleh pelanggan. Namun, hasil pencarian k-produk tidak mungkin tetap dari waktu ke waktu. Oleh karena itu, interval waktu adalah salah satu faktor yang harus dipertimbangkan dalam proses kueri karena sangat berpengaruh terhadap hasil pencarian. Permasalahan ini kemudian didefinisikan dalam penelitian ini sebagai "k-Most Promising Products berbasis interval waktu (k-MPPTI)".*

*Pada penelitian ini akan dirancang dan diimplementasikan struktur data dan algoritme yang tepat untuk menyelesaikan permasalahan tersebut. Data yang digunakan adalah data multidimensi, sehingga menggunakan struktur data yang memiliki fitur indexing, yaitu*

*array. Algoritme k-MPPTI menggunakan dua tipe kueri skyline, yaitu dynamic skyline dan reverse skyline. Selain itu, algoritme k-MPPTI juga mengimplementasikan teknik komputasi paralel supaya pemrosesan data menjadi lebih cepat.*

*Hasil uji coba menunjukkan bahwa algoritme k-MPPTI dapat memberikan hasil dengan waktu eksekusi ... kali lebih cepat dan penggunaan memori ... kali lebih hemat dibandingkan dengan algoritme konvensional Brute Force.*

**Keywords:** *Product selection strategy, Query, Dynamic skyline, Reverse skyline, Time interval*

## **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 senantiasa memberikan perhatian, dukungan, serta kasih sayang yang menjadi semangat dan motivasi bagi diri penulis untuk menyelesaikan Tugas Akhir.
2. Bapak Bagus Jati Santoso, S.Kom., Ph.D. selaku dosen pembimbing yang telah banyak meluangkan waktu untuk membimbing dan memberikan ilmu, saran, dan motivasi

kepada penulis baik selama 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. Teman-teman *Sempol Bunda*, Ajeng, Salma, Napik, Bela, dan Yola, yang telah menemani dan mewarnai masa-masa perkuliahan penulis sejak jaman mahasiswa baru.
6. Seluruh teman-teman Laboratorium Arsitektur dan Jaringan Komputer (AJK), Mas Syukron, Mas Fatih, Nahda, Satria, Awan, Mas Penyok, Fuad, Didin, Hana, Raldo, Aguel, Khawari, Tamtam, Haura, Lia, Sulton, Mail, Yoga, dan Fawwaz, yang telah menemani, mengganggu, dan membantu penulis selama mengerjakan Tugas Akhir di lab.
7. Teman-teman *Penguasa Kosan*, Ajeng, Salma, Napik, Prames, Kikik, Balqis, Tije, Nilam, dan Rini, yang pernah mengajarkan cara bersenang-senang.
8. Emak kos terbaik, Mak Ju, atas segala bantuannya selama ini dan teman-teman kosan 36, Jakiya, Mutek, Marisa, Mbak Tatak, Alya, Firda, dan Anca.
9. Teman-teman *Data Engineers*, Hana dan Rio, sebagai teman seperjuangan dan seperbimbingan.
10. Seluruh teman-teman TC 2015 yang tidak bisa penulis sebutkan satu persatu namanya.

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

*Halaman ini sengaja dikosongkan*

## DAFTAR ISI

<b>SAMPUL</b> . . . . .	<b>i</b>
<b>LEMBAR PENGESAHAN</b> . . . . .	<b>viii</b>
<b>ABSTRAK</b> . . . . .	<b>ix</b>
<b>ABSTRACT</b> . . . . .	<b>xi</b>
<b>KATA PENGANTAR</b> . . . . .	<b>xiii</b>
<b>DAFTAR ISI</b> . . . . .	<b>xvii</b>
<b>DAFTAR TABEL</b> . . . . .	<b>xxi</b>
<b>DAFTAR GAMBAR</b> . . . . .	<b>xxiii</b>
<b>DAFTAR KODE SUMBER</b> . . . . .	<b>xxv</b>
<b>BAB I PENDAHULUAN</b> . . . . .	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	3
1.3 Batasan Masalah . . . . .	3
1.4 Tujuan . . . . .	4
1.5 Manfaat . . . . .	4
1.6 Metodologi . . . . .	4
1.7 Sistematika Penulisan . . . . .	6
<b>BAB II TINJAUAN PUSTAKA</b> . . . . .	<b>9</b>
2.1 Daftar Notasi . . . . .	9
2.2 Data . . . . .	10
2.2.1 Data Multidimensi . . . . .	11
2.2.2 Data Serial Waktu . . . . .	11
2.2.3 Data Multidimensi dengan Serial Waktu . . . . .	12
2.3 <i>Skyline</i> . . . . .	13
2.4 Dominansi Dinamis . . . . .	15
2.5 <i>Dynamic Skyline</i> . . . . .	17
2.6 <i>Reverse Skyline</i> . . . . .	18
2.7 Kueri <i>k</i> -Most Promising Products ( <i>k</i> -MPP) . . . . .	20
2.7.1 <i>Uniform Product Adoption</i> (UPA) . . . . .	20

2.7.2	Strategi Pemilihan Produk . . . . .	22
2.8	Python . . . . .	22
2.9	Flask . . . . .	23
<b>BAB III ANALISIS DAN PERANCANGAN SISTEM . . . . .</b>	<b>25</b>	
3.1	Daftar Notasi . . . . .	25
3.2	Analisis Sistem . . . . .	27
3.2.1	Analisis Permasalahan . . . . .	27
3.2.2	Deskripsi Umum Sistem . . . . .	27
3.2.3	Fungsi Sistem . . . . .	28
3.2.4	Analisis Kebutuhan Fungsional . . . . .	28
3.3	Perancangan Sistem . . . . .	30
3.3.1	Struktur Data . . . . .	30
3.3.2	Algoritme Utama . . . . .	34
3.3.3	Algoritme <i>Brute Force</i> . . . . .	47
3.3.4	Arsitektur Aplikasi . . . . .	47
<b>BAB IV IMPLEMENTASI . . . . .</b>	<b>51</b>	
4.1	Lingkungan Implementasi . . . . .	51
4.2	Rancangan Data . . . . .	51
4.2.1	Data Masukan . . . . .	52
4.2.2	Data Keluaran . . . . .	52
4.3	Implementasi Algoritma . . . . .	52
4.3.1	<i>Header</i> yang Diperlukan . . . . .	52
4.3.2	Variabel Global . . . . .	53
4.3.3	Implementasi Fungsi Main . . . . .	53
4.3.4	Implementasi Struct Node . . . . .	55
4.3.5	Implementasi Fungsi Newnode . . . . .	55
4.3.6	Implementasi Fungsi Build . . . . .	56
4.3.7	Implementasi Fungsi Getsize . . . . .	56
4.3.8	Implementasi Fungsi Split . . . . .	57
4.3.9	Implementasi Fungsi Random . . . . .	58
4.3.10	Implementasi Fungsi Concate . . . . .	59
4.3.11	Implementasi Fungsi Insert . . . . .	60
4.3.12	Implementasi Fungsi Mutable Begin . . . . .	60

4.3.13 Implementasi Fungsi Mutable End . . . . .	61
4.3.14 Implementasi Fungsi Print . . . . .	63
<b>BAB V UJI COBA DAN EVALUASI . . . . .</b>	<b>65</b>
5.1 Lingkungan Uji Coba . . . . .	65
5.2 Uji Coba Kebenaran . . . . .	65
5.3 Uji Coba Kinerja . . . . .	72
5.3.1 Operasi 1 Menggabungkan <i>Rope</i> pada Posisi Awal . . . . .	72
5.3.2 Operasi 2 Menggabungkan <i>Rope</i> pada Posisi Akhir . . . . .	74
5.3.3 Operasi 3 Mencetak Karakter pada Indeks ke-Y . . . . .	75
5.4 Analisis Hasil Uji Coba . . . . .	77
<b>BAB VI KESIMPULAN . . . . .</b>	<b>79</b>
6.1 Kesimpulan . . . . .	79
6.2 Saran . . . . .	80
<b>DAFTAR PUSTAKA . . . . .</b>	<b>81</b>
<b>BIODATA PENULIS . . . . .</b>	<b>83</b>

*Halaman ini sengaja dikosongkan*

## DAFTAR TABEL

Tabel 2.1	Daftar Notasi (1) . . . . .	9
Tabel 2.2	Contoh Data <i>Time Series</i> . . . . .	12
Tabel 2.3	Contoh Data Multidimensi dengan Serial Waktu (1) . . . . .	12
Tabel 2.4	Contoh Data Multidimensi dengan Serial Waktu (2) . . . . .	13
Tabel 2.5	Contoh <i>Dataset</i> (a) Produk $P$ dan (b) Preferensi Pelanggan $C$ . . . . .	16
Tabel 3.1	Daftar Notasi (2) . . . . .	25
Tabel 3.2	Kebutuhan Fungsional . . . . .	29
Tabel 3.3	<i>Key</i> dari <i>Data Storage</i> . . . . .	32
Tabel 3.4	Atribut dari <i>Event Queue</i> . . . . .	33
Tabel 3.5	Contoh <i>Pandora Box</i> dari <i>Dataset 3.6</i> . . . . .	34
Tabel 3.6	Contoh <i>Dataset</i> (a) Produk $P$ dan (b) Preferensi Pelanggan $C$ . . . . .	37
Tabel 3.7	<i>Event Queue</i> . . . . .	38
Tabel 3.8	Hasil Perhitungan <i>Dynamic Skyline</i> . . . . .	41
Tabel 3.9	Hasil Perhitungan <i>Reverse Skyline</i> . . . . .	42
Tabel 3.10	Hasil Perhitungan Probabilitas . . . . .	43
Tabel 3.11	Hasil Perhitungan Kontribusi Pasar . . . . .	44
Tabel 5.1	Kecepatan Maksimal, Minimal dan Rata-Rata dari Hasil Uji Coba Sebanyak 15 Kali pada Situs Pengujian SPOJ . . . . .	72

*Halaman ini sengaja dikosongkan*

## DAFTAR GAMBAR

Gambar 2.1	Titik Skyline dari Data Produk pada Tabel 2.5	14
Gambar 2.2	(a) Komputasi <i>Dynamic Skyline</i> dari Pelanggan $c_4$ dan (b) <i>Dynamic Skyline</i> dari Pelanggan $c_{10}$	18
Gambar 2.3	Komputasi <i>Reverse Skyline</i> dari Produk $p_8$	19
Gambar 3.1	Struktur Data <i>Dictionary</i> Produk	31
Gambar 3.2	Struktur Data <i>Dictionary</i> Pelanggan	31
Gambar 3.3	Diagram Alur Algoritme k-MPPTI	35
Gambar 3.4	Ilustrasi Data Multidimensi dengan Serial Waktu	37
Gambar 3.5	Ilustrasi Pemrosesan Data	38
Gambar 3.6	Perancangan Arsitektur Aplikasi	48
Gambar 4.1	Ilustrasi Penyimpanan Hasil Operasi <i>Split Rope</i> pada Struktur Data Pair	57
Gambar 4.2	Ilustrasi Operasi <i>Concat</i> . Setiap <i>Node</i> Berisi Sebuah Karakter dan Nilai Berat Masing-Masing <i>Node</i>	59
Gambar 4.3	Ilustrasi Operasi <i>Mutable Begin</i> . Setiap <i>Node</i> Berisi Sebuah Karakter dan Nilai Prioritas Masing-Masing <i>Node</i>	62
Gambar 4.4	Ilustrasi Operasi <i>Mutable End</i> . Setiap <i>Node</i> Berisi Sebuah Karakter dan Nilai Prioritas Masing-Masing <i>Node</i>	64
Gambar 5.1	Contoh kasus uji permasalahan <i>Alphabetic Rope</i>	66
Gambar 5.2	Pembentukan <i>Root Rope</i>	66
Gambar 5.3	Pembentukan <i>Rope</i> pada Tingkat ke-2	67
Gambar 5.4	Pembentukan <i>Rope</i> pada Tingkat ke-3	67
Gambar 5.5	Struktur <i>Rope</i> yang Terbentuk	68

Gambar 5.6 Konfigurasi <i>Rope</i> Setelah Operasi 2 0 5 . . . . .	69
Gambar 5.7 Hasil Luaran Program pada Contoh Kasus Uji Alphabetic Rope . . . . .	70
Gambar 5.8 Hasil Uji Coba pada Situs Penilaian SPOJ . . . . .	71
Gambar 5.9 Hasil Uji Coba pada Situs Penilaian SPOJ . . . . .	71
Gambar 5.10 Hasil Uji Coba pada Operasi 1 dengan Jumlah <i>Query</i> Tetap dan Panjang <i>String</i> Bertambah . . . . .	73
Gambar 5.11 Hasil Uji Coba pada Operasi 1 dengan Jumlah <i>String</i> Tetap dan Jumlah <i>Query</i> Bertambah . . . . .	73
Gambar 5.12 Hasil Uji Coba pada Operasi 2 dengan Jumlah <i>Query</i> Tetap dan Jumlah <i>String</i> Bertambah . . . . .	74
Gambar 5.13 Hasil Uji Coba pada Operasi 2 dengan Panjang <i>String</i> Tetap dan Jumlah <i>Query</i> Bertambah . . . . .	75
Gambar 5.14 Hasil Uji Coba pada Operasi 3 dengan Jumlah <i>Query</i> Tetap dan Panjang <i>String</i> Bertambah . . . . .	76
Gambar 5.15 Hasil Uji Coba pada Operasi 3 dengan Jumlah <i>String</i> Tetap dan Jumlah <i>Query</i> Bertambah . . . . .	76

## DAFTAR KODE SUMBER

Kode Sumber 4.1	<i>Header</i> yang diperlukan . . . . .	53
Kode Sumber 4.2	Variabel Global . . . . .	53
Kode Sumber 4.3	Fungsi Main . . . . .	54
Kode Sumber 4.4	Fungsi Struct Node . . . . .	55
Kode Sumber 4.5	Fungsi Newnode . . . . .	55
Kode Sumber 4.6	Fungsi Build . . . . .	56
Kode Sumber 4.7	Fungsi Getsize . . . . .	56
Kode Sumber 4.8	Fungsi Split(1) . . . . .	57
Kode Sumber 4.9	Fungsi Split(2) . . . . .	58
Kode Sumber 4.10	Fungsi Random . . . . .	58
Kode Sumber 4.11	Fungsi Concate(1) . . . . .	59
Kode Sumber 4.12	Fungsi Concate(2) . . . . .	60
Kode Sumber 4.13	Fungsi Insert . . . . .	60
Kode Sumber 4.14	Fungsi Mutable Begin . . . . .	61
Kode Sumber 4.15	Fungsi Mutable End . . . . .	63
Kode Sumber 4.16	Fungsi Print . . . . .	63

*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 membawa dampak yang cukup besar di bidang bisnis, termasuk mempengaruhi cara produsen dalam melakukan bisnis secara lebih efisien. Produsen dapat mengumpulkan data preferensi pelanggan terhadap produk dan fitur produk dari data penjualan mereka. Selain itu, maraknya penggunaan situs web untuk menjual produk secara *online* juga memungkinkan produsen mendapatkan data preferensi pelanggan terhadap fitur produk lain.

Data preferensi pelanggan yang terkumpul dapat dimanfaatkan untuk mendapatkan informasi-informasi penting yang menguntungkan, misalnya untuk mencari produk apa saja yang paling banyak diminati oleh pelanggan sehingga produsen dapat menentukan produk mana yang harus dipilih untuk strategi *targeted marketing* supaya lebih tepat sasaran dan bertahan lama 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* [3]. Kueri *dynamic skyline* digunakan untuk mengambil data produk terbaik berdasarkan sudut pandang pelanggan, sedangkan kueri *reverse skyline* digunakan untuk mengambil data pelanggan potensial berdasarkan sudut pandang produsen.

Kemudian muncul sebuah pertanyaan, “*Apakah produk yang paling banyak diminati pelanggan akan selalu sama dari waktu ke waktu?*”. Tentu saja para produsen lain tidak akan berdiam diri. Produk-produk baru akan terus bermunculan seiring dengan berjalannya waktu, sehingga produk yang paling diminati pelanggan pun ikut berubah karena adanya produk-produk baru yang dapat mengungguli produk 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 [4].

Untuk menjawab pertanyaan tersebut, dibutuhkan pendekatan lain untuk menyelesaikan kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data

multidimensi dengan serial waktu. Algoritme yang dibangun juga mengimplementasikan teknik komputasi paralel supaya pemrosesan data menjadi lebih cepat [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 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 untuk mengetahui struktur data dan algoritme yang tepat untuk menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktusecara optimal dan efisien.

Selain itu, Tugas Akhir ini juga diharapkan dapat memberikan kontribusi pada perkembangan ilmu pengetahuan dan teknologi informasi karena algoritme ini dapat digunakan dalam berbagai hal, khususnya bagi produsen untuk membuat bisnisnya menjadi lebih baik dan tepat sasaran.

## 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 menjawab kueri *k-Most Promising Products* (*k-MPP*) berbasis interval waktu pada data multidimensi dengan serial waktu. Proposal ini berisi tentang deskripsi pendahuluan dari Tugas Akhir yang akan dibuat, terdiri atas hal yang menjadi latar belakang diajukannya usulan Tugas Akhir, rumusan masalah yang diangkat, batasan masalah, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu, dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir.

## 2. Studi literatur

Pada tahap ini dilakukan pencarian informasi dan literatur mengenai metode yang dapat digunakan dalam 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. Informasi-informasi tersebut bisa didapatkan dari buku, jurnal, maupun internet.

## 3. Analisis dan perancangan perangkat lunak

Pada tahap ini dilakukan analisis dan perancangan struktur data dan algoritme yang digunakan untuk menjawab kueri *k-Most Promising Products* (*k-MPP*) berbasis interval waktu pada data multidimensi dengan serial waktuberdasarkan literatur yang telah dipelajari.

## 4. Implementasi perangkat lunak

Pada tahap ini dilakukan implementasi atau realiasi dari hasil analisis dan perancangan struktur data dan algoritme yang telah dibuat ke dalam bentuk program.

## 5. Uji coba dan evaluasi

Pada tahap ini dilakukan uji coba dari struktur data dan algoritme yang telah diimplementasikan. Pengujian akan

dilakukan dengan dua cara, yaitu:

(a) Pengujian waktu eksekusi (*runtime*)

Pengujian yang berfokus pada waktu eksekusi 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.

(b) Pengujian penggunaan memori (*memory usage*)

Pengujian yang berfokus pada konsumsi memori 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.

Setelah dilakukan uji coba, maka dilakukan evaluasi terhadap kinerja struktur data dan algoritme yang telah diimplementasikan, dengan harapan dapat diperbaiki ke depannya.

6. Penyusunan buku Tugas Akhir

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

## 1.7 Sistematika Penulisan

Berikut adalah sistematika penulisan buku Tugas Akhir:

1. BAB I: PENDAHULUAN

Bab ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi dan sistematika penulisan Tugas Akhir.

2. BAB II: TINJAUAN PUSTAKA

Bab ini berisi dasar teori mengenai permasalahan dan algoritme penyelesaian yang digunakan dalam Tugas Akhir

3. BAB III: ANALISIS DAN PERANCANGAN SISTEM

Bab ini berisi analisis dan perancangan struktur data dan algoritme yang digunakan dalam penyelesaian permasalahan.

#### 4. BAB IV: IMPLEMENTASI

Bab ini berisi implementasi berdasarkan analisis dan perancangan struktur data dan algortime yang telah dilakukan pada tahap analisis dan perancangan sistem.

#### 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 dan evaluasi yang telah dilakukan.

*Halaman ini sengaja dikosongkan*

## **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 Notasi**

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

Tabel 2.1 Daftar Notasi (1)

Notasi	Deskripsi
$P$	<i>Dataset</i> produk
$C$	<i>Dataset</i> pelanggan (preferensi pelanggan)
$D$	$P \cup C$
$ob$	Sebuah objek data pada $D$
$ob_1 \prec ob_2$	Objek data $ob_1$ mendominasi $ob_2$
$ob_1 \prec_{ob_3} ob_2$	Objek data $ob_1$ mendominasi $ob_2$ berdasarkan $ob_3$
$p$	Sebuah produk dalam $P$ , $p \in P$
$c$	Seorang pelanggan dalam $C$ , $c \in C$
$d$	Jumlah dimensi pada $D$

Notasi	Deskripsi
$i$	Dimensi ke-1, ..., $d$
$j$	Timestamp ke-1, 2, ..., dst
$O$	<i>Orthant</i> atau daerah pada komputasi <i>reverse skyline</i>
$m$	<i>Midpoint</i> antar produk pada komputasi <i>reverse skyline</i>
$DSL(c)$	Hasil <i>dynamic skyline</i> dari pelanggan $c$
$RSL(p)$	Hasil <i>reverse skyline</i> dari produk $p$
$Pr(c, p P)$	Probabilitas produk $p$ dibeli oleh pelanggan $c$
$E(C, p P)$	Kontribusi pasar $p$
$E(C, P' P)$	Kontribusi pasar subset $P'$ dari $P$
$k - MPP$	$k$ -Most Promising Products

## 2.2 Data

Data merupakan elemen yang esensial dalam sebuah sistem informasi. Data adalah informasi faktual (seperti pengukuran atau statistik) yang digunakan sebagai dasar untuk analisis, diskusi, maupun perhitungan [10].

Meski begitu, data mentah tidaklah berarti dan harus diproses terlebih dahulu supaya menghasilkan informasi yang bermanfaat. Sehingga, dibutuhkanlah sebuah algoritme pemrosesan data yang menerima data sebagai *input*, kemudian memprosesnya 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].

### 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 Tabel 2.2, diberikan contoh sebuah data *time series*  $S$ . Supaya sederhana, 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 [4].

Tabel 2.2 Contoh Data *Time Series*

id	timestamp				
	1	2	3	4	5
$s_1$	8	2	5	10	12
$s_2$	14	4	10	7	8
$s_3$	15	6	11	7	3
$s_4$	3	8	12	9	13
$s_5$	15	9	10	2	7

### 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 *timestamp* dan berurutan menurut waktu. Pada Tabel 2.3, diberikan contoh sebuah data produk yang memiliki nilai atribut dan *timestamp*.

Tabel 2.3 Contoh Data Multidimensi dengan Serial Waktu (1)

id	timestamp									
	1		2		3		4		5	
	$d_1$	$d_2$	$d_1$	$d_2$	$d_1$	$d_2$	$d_1$	$d_2$	$d_1$	$d_2$
$p_1$	2	8	2	8	2	8	2	8	2	8
$p_2$	-	-	-	-	-	-	4	10	4	10
$p_3$	6	11	6	11	6	11	-	-	-	-
$p_4$	-	-	-	-	-	-	-	-	8	12
$p_5$	9	10	9	10	9	10	9	10	-	-

Contoh data pada Tabel 2.3 sekilas hampir sama dengan data pada Tabel 2.2, namun memiliki atribut atau dimensi lebih dari satu. Jika asumsinya nilai pada atribut data selalu tetap, maka data pada Tabel 2.3 dapat ditulis menjadi Tabel 2.4. Timestamp yang banyak dan berurutan dapat ditulis menjadi interval waktu (*timestamp in - timestamp out*), dinotasikan dengan  $[i : j]$ . Interval waktu menggambarkan bahwa sebuah data memiliki waktu hidup tertentu.

Tabel 2.4 Contoh Data Multidimensi dengan Serial Waktu (2)

<b>id</b>	<b>ts_in</b>	<b>ts_out</b>	<b>dim1</b>	<b>dim2</b>
$p_1$	1	8	2	8
$p_2$	4	14	4	10
$p_3$	1	3	6	11
$p_4$	5	15	8	12
$p_5$	1	4	9	10

Data yang digunakan pada Tugas Akhir ini adalah dataset produk  $P$  dan pelanggan  $C$ . Ilustrasinya, setiap produk  $p \in P$  memiliki waktu kapan ia pertama kali diproduksi dan kapan ia tidak diproduksi lagi, sedangkan setiap pelanggan  $c \in C$  memiliki waktu kapan ia lahir dan kapan ia meninggal dunia.

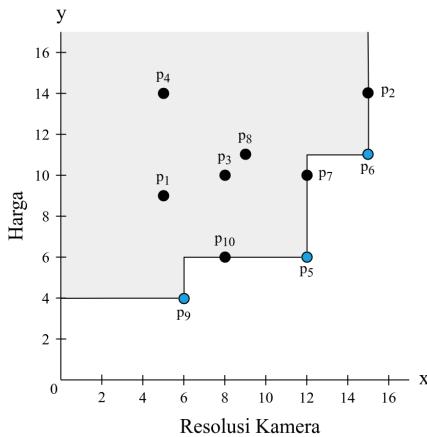
### 2.3 *Skyline*

Komputasi *skyline* telah menarik perhatian yang cukup besar dari peneliti sejak diperkenalkan pada komunitas basis data [6], 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 [6], yaitu data yang tidak didominasi oleh data lain atau data yang paling unggul.

Diberikan 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  $p_1 \prec p_2$  dapat terbentuk jika dan hanya jika:

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

Misalnya, seseorang ingin mencari produk *smartphone* terbaik, yaitu *smartphone* yang memiliki harga termurah dan memiliki resolusi kamera terbesar. Pada Tabel 2.5, diberikan data produk  $P$  yang memiliki atribut resolusi kamera ( $dim_1$ ) dan harga ( $dim_2$ ). Setiap datanya direpresentasikan sebagai titik pada bidang dua dimensi, yakni sumbu  $x$  adalah resolusi kamera dan sumbu  $y$  adalah harga *smartphone*.



Gambar 2.1 Titik Skyline dari Data Produk pada Tabel 2.5

Berdasarkan Gambar 2.1, produk *smartphone* yang terbaik

adalah  $p_5$ ,  $p_6$ , dan  $p_9$  karena tidak ada titik yang lebih baik dari titik-titik tersebut pada semua dimensi, sedangkan produk  $p_{10}$  tidak dapat menjadi *skyline* karena didominasi oleh produk  $p_5$  pada dimensi  $x$ . Begitu juga produk  $p_7$  yang didominasi  $p_5$  pada dimensi  $y$  dan produk  $p_2$  yang didominasi  $p_6$  pada dimensi  $y$ . Produk  $p_5$ ,  $p_6$ , dan  $p_9$  disebut dengan titik *skyline* atau *skyline point*.

Saat ini, komputasi *skyline* telah banyak digunakan sebagai operator pengambil keputusan multikriteria dan perencanaan bisnis [7]. Ada beberapa pengembangan dari komputasi *skyline*, seperti *dynamic skyline* dan *reverse skyline*.

## 2.4 Dominansi Dinamis

Berdasarkan definisi "Skyline" yang telah dijelaskan pada sub-bagian sebelumnya, jika diberikan *dataset* yang sama, maka hasil *skyline* dari *dataset* tersebut pasti akan selalu sama. Oleh karena itu, para ahli juga menyebut *original skyline* sebagai *static skyline* [7].

Ada suatu kasus ketika perhitungan *skyline* didasarkan pada titik kueri. Jika diberikan *dataset* yang sama, namun titik kuerinya berbeda, maka hasil *skyline*-nya pun berbeda tergantung pada titik kueri. *Skyline* ini disebut dengan *dynamic skyline* karena memiliki sifat dominansi dinamis.

Diberikan *dataset* produk  $P$  dan *dataset* pelanggan (preferensi pelanggan)  $C$  yang setiap datanya direpresentasikan sebagai objek data  $d$ -dimensi dan hanya dapat menyimpan nilai numerik pada setiap dimensinya. Data produk dan pelanggan pada dimensi ke- $i$  dinotasikan sebagai  $p^i$  dan  $c^i$ ,  $i \leq d$ . Untuk menggambarkan objek data secara umum 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:

- $$(a) \quad |ob_3^i - ob_1^i| \leq |ob_3^i - ob_2^i|, \forall i \in [1, \dots, d]$$
- $$(b) \quad |ob_3^i - ob_1^i| < |ob_3^i - ob_2^i|, \exists i \in [1, \dots, d] \quad (2.1)$$

Pada Tabel 2.5, diberikan contoh *dataset* produk dan preferensi pelanggan. Berdasarkan preferensi pelanggan  $c_1$ , produk  $p_1$  dikatakan mendominasi produk  $p_2$ , dinotasikan dengan  $p_1 \prec_{c_1} p_2$ , karena memenuhi kedua syarat dominansi dinamis yakni (a)  $|c_1^1 - p_1^1| = |5 - 5| = 0 \leq |c_1^1 - p_2^1| = |5 - 15| = 10$  dan (b)  $|c_1^2 - p_1^2| = |2 - 9| = 7 < |c_1^2 - p_2^2| = |2 - 14| = 12$ .

Tabel 2.5 Contoh *Dataset*  
(a) Produk  $P$  dan (b) Preferensi Pelanggan  $C$

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

Sebaliknya, jika berdasarkan preferensi pelanggan  $c_6$ , maka

produk  $p_2$ -lah yang mendominasi  $p_1$ , dinotasikan dengan  $p_2 \prec_{c_6} p_1$ , karena (a)  $|c_6^1 - p_2^1| = |12 - 15| = 3 \leq |c_6^1 - p_1^1| = |12 - 5| = 7$  dan (b)  $|c_6^2 - p_2^2| = |14 - 14| = 0 < |c_6^2 - p_1^2| = |14 - 9| = 5$ . Dalam hal ini, preferensi pelanggan disebut dengan titik kueri karena dapat mempengaruhi sifat dominansi antar produk.

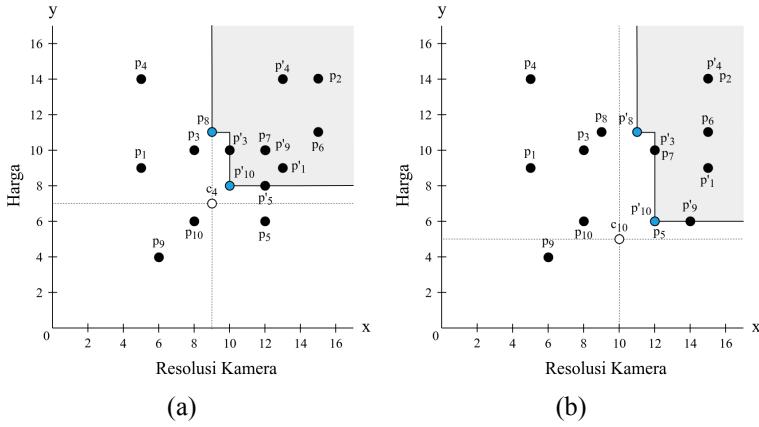
Mengambil contoh lain, produk  $p_1$  tidak mendominasi  $p_2$  berdasarkan pelanggan  $c_3$  karena ada salah satu syarat dominansi dinamis yang tidak terpenuhi, (a)  $|c_3^1 - p_1^1| = |15 - 5| = 10 \not\leq |c_3^1 - p_2^1| = |15 - 15| = 0$  dan (b)  $|c_3^2 - p_1^2| = |10 - 9| = 1 < |c_3^2 - p_2^2| = |10 - 14| = 4$ . Produk  $p_1$  dan  $p_2$  dikatakan saling mendominasi berdasarkan pelanggan  $c_2$ .

## 2.5 Dynamic Skyline

Kueri *dynamic skyline* dalam komputasi  $k$ -MPP digunakan untuk mencari produk terbaik dari sudut pandang pelanggan [1], sehingga yang menjadi titik kueri adalah pelanggan. *Dynamic skyline* [2] dari seorang pelanggan  $c_1 \in C$ , dinotasikan dengan  $DSL(c_1)$ , berisi semua produk  $p_1 \in P$  yang tidak didominasi oleh produk lain  $p_2 \in P$  berdasarkan preferensi pelanggan  $c_1$ ,  $p_2 \not\prec_{c_1} p_1$ .

*Dynamic skyline* dapat dihitung menggunakan algoritme komputasi *skyline* tradisional [6], yaitu mentransformasikan semua titik  $p \in P$  ke ruang data baru dengan menganggap titik kueri  $c$  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  $f^i(p^i) = |c^i - p^i|$ .

Menggunakan *dataset* pada Tabel 2.5, *dynamic skyline* dari pelanggan  $c_4$  adalah  $DSL(c_4) = \{p_8, p_{10}\}$ , karena produk tersebut tidak didominasi oleh produk lain berdasarkan preferensi pelanggan  $c_4$ . Berbeda halnya dengan  $c_{10}$  yang memiliki hasil *dynamic skyline*  $DSL(c_{10}) = \{p_5, p_8, p_{10}\}$ .



Gambar 2.2 (a) Komputasi *Dynamic Skyline* dari Pelanggan  $c_4$  dan (b) *Dynamic Skyline* dari Pelanggan  $c_{10}$

## 2.6 Reverse Skyline

Dalam komputasi  $k$ -MPP, kueri *reverse skyline* digunakan untuk mencari pelanggan potensial dari sudut pandang produsen [1], sehingga yang menjadi titik kueri adalah produk. *Reverse skyline* [3] 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.

Ada beberapa tahapan yang harus dilakukan dalam komputasi *reverse skyline* [1]. Pertama, menentukan *orthant* dari produk, dinotasikan dengan  $O$ . Setiap produk  $p$  memiliki  $2^d$  *orthant* pada data  $d$ -dimensi. Kedua, menghitung *midpoint* atau titik tengah antara produk kueri dan produk lainnya, misalnya  $p_1$  (sebagai titik kueri) dan  $p_2$ , dihitung menggunakan rumus berikut:

$$m_2^i = \frac{(p_1^i + p_2^i)}{2} \quad (2.2)$$

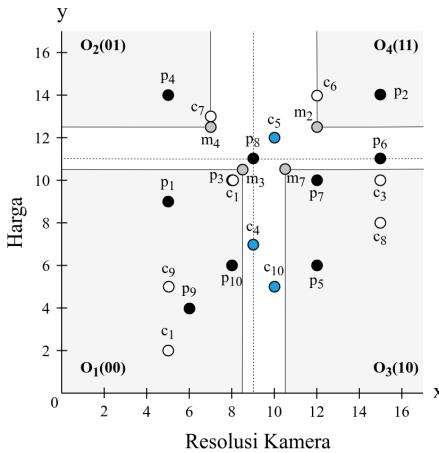
Kemudian menentukan *midpoint skyline* (juga dikenal sebagai

*mid-skyline* [8]) pada setiap *orthant*.

Langkah ketiga, mengecek apakah pelanggan  $c \in C$  didominasi oleh *midpoint skyline*  $m$  berdasarkan produk  $p_1$  atau tidak. Pelanggan  $c$  dikatakan didominasi oleh *midpoint skyline*  $m$  jika dan hanya jika:

- $|p_1^i - m^i| \leq |p_1^i - c^i|, \forall i \in [1, \dots, d]$
  - $|p_1^i - m^i| < |p_1^i - c^i|, \exists i \in [1, \dots, d]$
- (2.3)

Apabila  $c$  tidak didominasi oleh *midpoint skyline*  $m$  berdasarkan produk  $p_1$ , maka  $c$  menjadi hasil dari *reverse skyline*  $p_1$ , dinotasikan dengan  $RSL(p_1)$ . Untuk lebih jelasnya, komputasi *reverse skyline* ditunjukkan pada Gambar 2.3.



Gambar 2.3 Komputasi *Reverse Skyline* dari Produk  $p_8$

Sebagai contoh, berdasarkan *dataset* yang diberikan pada Tabel 2.5, *reverse skyline* dari produk  $p_8$  adalah pelanggan  $c_4$ ,  $c_5$ , dan  $c_{10}$ , dinotasikan dengan  $RSL(p_8) = \{c_4, c_5, c_{10}\}$  karena masing-masing pelanggan tersebut memiliki  $p_8$  pada hasil *dynamic*

*skyline*-nya.

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

Kueri *k*-Most Promising Products (*k*-MPP) adalah sebuah strategi pemilihan produk yang dikenalkan oleh Islam dan Liu dalam penelitiannya [1].

### 2.7.1 Uniform Product Adoption (UPA)

*Uniform Product Adoption* (UPA) mengasumsikan bahwa semua produk  $p \in P$  yang muncul pada hasil *dynamic skyline* pelanggan  $c \in C$  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:

$$Pr(c, p|P) = \begin{cases} \frac{1}{|DSL(c)|} & \text{if } p \in DSL(c) \\ 0 & \text{otherwise} \end{cases} \quad (2.4)$$

Berdasarkan Persamaan 2.4, 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  $c$ .

Sebagai contoh menggunakan *dataset* pada Tabel 2.5, probabilitas produk  $p_8$  dibeli oleh pelanggan  $c_4$  adalah  $Pr(c_4, p_8|P) = \frac{1}{|DSL(c_4)|} = \frac{1}{2}$ , sedangkan probabilitas produk  $p_8$  dibeli oleh pelanggan  $c_{10}$  adalah  $\frac{1}{|DSL(c_{10})|} = \frac{1}{3}$ .

### 2.7.1.1 Market Contribution

*Market contribution* atau kontribusi pasar sebuah produk  $p \in P$  diukur dari total jumlah pelanggan 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 2.4. Sehingga, kontribusi pasar sebuah produk dihitung dari hasil akumulasi bobot yang didapatkan dari semua pelanggan  $c \in C$ .

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

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

Karena probabilitas produk  $p$  dipilih oleh pelanggan yang tidak memiliki  $p$  pada hasil *dynamic skyline*-nya adalah nol (pada Persamaan 2.4), maka kita hanya perlu mengakumulasikan probabilitas produk dari setiap pelanggan  $c$  pada hasil  $RSL(p)$ . Sehingga, Persamaan 2.5 dapat disederhanakan menjadi:

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

Sebagai contoh menggunakan *dataset* pada Tabel 2.5, kontribusi pasar dari produk  $p_8$  adalah  $E(C, p_8|P) = Pr(c_4, p_8|P) + Pr(c_5, p_8|P) + Pr(c_{10}, p_8|P) = \frac{1}{2} + 1 + \frac{1}{3} = \frac{11}{6}$  atau 1.833.

Perhitungan kontribusi pasar juga dapat dilakukan pada sekumpulan produk atau *subset* produk  $P'$ , dinotasikan dengan

$E(C, P'|P)$ , yang dijelaskan pada Persamaan 2.7.

$$E(C, P'|P) = \sum_{\forall p \in P'} E(C, p|P) \quad (2.7)$$

### 2.7.2 Strategi Pemilihan Produk

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), 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 [1], sebagaimana yang dijelaskan pada Persamaan 2.7.

Jika merangkum semua penjelasan di atas, langkah-langkah yang harus dilakukan untuk memproses kueri  $k$ -MPP adalah: (1) menghitung *reverse skyline* dari setiap produk  $p \in P$ , (2) menghitung *dynamic Skyline* dari setiap pelanggan  $c \in RSL(p)$ , dan (3) memilih  $k$  produk dari  $P$  yang memiliki kontribusi pasar terbesar.

## 2.8 Python

Python adalah bahasa pemrograman tingkat tinggi, *interpreted*, dan berorientasi objek yang didukung oleh struktur data *built-in* tingkat tinggi dan semantik yang dinamis [9]. Python dikembangkan oleh Guido van Rossum pada akhir 1980-an dan dikelola oleh *Python Software Foundation*. Saat ini, Python sudah tersedia dalam dua versi, yakni 2.x dan 3.x.

Kelebihan bahasa pemrograman Python adalah pada keterbacaannya karena memiliki sintaksis yang sederhana, sehingga dapat mengurangi biaya pemeliharaan (*maintenance*). Python mendukung banyak modul dan *package*, serta memiliki

banyak *standard library* yang didistribusikan secara gratis. Selain itu, karena Python adalah bahasa *interpreted*, Python tidak memakan biaya untuk kompilasi sehingga proses pengubahan, pengujian, dan debug menjadi lebih cepat.

Melakukan debug pada program Python sangatlah mudah karena tidak akan mengakibatkan *segmentation fault*. Sebagai gantinya, ia akan menimbulkan *Exception* apabila menemukan suatu *error* atau kesalahan. Ketika program tidak menangkap *Exception*, maka Python akan menampilkan *stack trace* yang dapat digunakan untuk menganalisis dan memperbaiki kesalahan yang terjadi [9].

Pada Tugas Akhir ini, bahasa pemrograman Python digunakan untuk mengimplementasikan struktur data dan algoritme pada sistem perangkat lunak yang akan dibangun.

## 2.9 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 *web server* yang digunakan pada Tugas Akhir ini karena ringan dan lebih mudah digunakan dibandingkan 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 Notasi**

Tabel 3.1 menunjukkan daftar notasi yang digunakan dalam bab ini beserta deskripsinya.

Tabel 3.1 Daftar Notasi (2)

Notasi	Deskripsi
$P$	<i>Dataset</i> produk
$C$	<i>Dataset</i> pelanggan (preferensi pelanggan)
$D$	$P \cup C$
$E$	Himpunan <i>event</i>
$p$	Sebuah produk dalam $P$ , $p \in P$
$c$	Seorang pelanggan dalam $C$ , $c \in C$
$e$	Sebuah <i>event</i> dalam $E$ , $e \in E$
$p_{in}$	Produk masuk
$p_{out}$	Produk keluar
$c_{in}$	Pelanggan masuk
$c_{out}$	Pelanggan keluar

Notasi	Deskripsi
$P_{active}$	Himpunan produk yang sedang aktif (di dalam lini masa)
$C_{active}$	Himpunan pelanggan yang sedang aktif (di dalam lini masa)
$D_{active}$	$P_{active} \cup C_{active}$
$d$	Jumlah dimensi pada $D$
$i$	Dimensi ke-1, ..., $d$
$j$	Timestamp ke-1, 2, ..., dst
$diff$	Selisih nilai
$p_s$	Produk sebagai subjek yang membandingkan
$p_o$	Produk sebagai objek pembanding
$O$	<i>Orthant</i>
$pos_x(p)$	Posisi $p$ pada sumbu $x$
$max_x$	Nilai maksimum pada sumbu $x$
$m$	<i>Midpoint</i> antar produk
$DSL(c)$	Hasil <i>dynamic skyline</i> dari pelanggan $c$
$RSL(p)$	Hasil <i>reverse skyline</i> dari produk $p$
$Pr(c, p P)$	Probabilitas produk $p$ dibeli oleh pelanggan $c$
$E(C, p P)$	Kontribusi pasar $p$
$E(C, P' P)$	Kontribusi pasar subset $P'$ dari $P$
$k - MPPTI$	<i>k-Most Promising Products in Time Intervals</i>
$k$	Jumlah data
$[t_i : t_e]$	Interval waktu
$PBox$	<i>Pandora Box</i>

Notasi	Deskripsi
$ts$	<i>Timestamp</i>

## 3.2 Analisis Sistem

Analisis sistem dijelaskan dalam empat bagian, yakni analisis permasalahan, deskripsi umum sistem, fungsi sistem, dan analisis kebutuhan fungsional.

### 3.2.1 Analisis Permasalahan

Permasalahan yang ingin diselesaikan pada Tugas Akhir ini adalah bagaimana menjawab kueri *k-Most Promising Products* berbasis interval waktu (*k-MPPTI*). Interval waktu, dinotasikan dengan  $[t_i : t_e](t_i \leq t_e)$ , digunakan untuk menentukan rentang waktu pencarian.

Permasalahan ini tidak dapat langsung diselesaikan menggunakan metode dan algoritme yang sudah ada [1]. Sehingga, diperlukan pendekatan lain yang akan dijelaskan pada bagian perancangan sistem.

### 3.2.2 Deskripsi Umum Sistem

Secara umum, sistem yang akan dibangun adalah sebuah sistem berbasis web yang dapat membantu pengguna untuk memilih  $k$ -produk yang paling menjanjikan. Dikatakan "menjanjikan" jika produk tersebut memiliki kontribusi pasar yang besar.

Sistem ini memiliki dua proses utama, yaitu (1) *data precomputing* untuk menghitung kontribusi pasar masing-masing produk dan (2) proses utama (selanjutnya akan disebut dengan *query processing*) untuk memproses dan menampilkan hasil kueri pencarian yang dimasukkan oleh pengguna.

Sistem ini dibangun menggunakan arsitektur *client-server*. Aplikasi *client* didesain berbasis web dengan memanfaatkan Flask *microframework*, HTML, CSS, dan JavaScript. Selain itu, Flask juga digunakan sebagai *web server*.

### 3.2.3 Fungsi Sistem

Sistem yang akan dibangun memiliki beberapa fungsi utama sebagai berikut:

1. Dapat menerima masukan data berupa file dari pengguna
2. Dapat menampilkan informasi dan pratinjau data yang dimasukkan oleh pengguna
3. Dapat menampilkan visualisasi data
4. Dapat melakukan proses *data precomputing* menggunakan algoritme yang dipilih oleh pengguna
5. Dapat menerima masukan kueri pencarian
6. Dapat memproses kueri pencarian
7. Dapat menampilkan hasil kueri
8. Dapat menampilkan waktu eksekusi

### 3.2.4 Analisis Kebutuhan Fungsional

Sistem yang dibuat harus mampu memenuhi beberapa fungsi utama yang telah dijelaskan pada sub-bagian sebelumnya. Fungsi-fungsi ini merupakan hasil dari analisis kebutuhan fungsional dari pengguna yang dijelaskan pada Tabel 3.2.

Tabel 3.2 Kebutuhan Fungsional

Kode	Deskripsi Kebutuhan
F-001	Mengunggah data
F-002	Melihat informasi dan pratinjau data
F-003	Melihat visualisasi data
F-004	Memilih algoritme yang digunakan untuk <i>data precomputing</i>
F-005	Memasukkan kueri pencarian
F-006	Melihat hasil kueri
F-007	Melihat waktu eksekusi

Penjelasan rinci dari masing-masing kebutuhan fungsional pada tabel 3.2 dijelaskan sebagai berikut:

1. Mengunggah data

Pengguna dapat mengunggah data produk dan preferensi pelanggan dalam bentuk file berekstensi csv.

2. Melihat informasi dan pratinjau data

Pengguna dapat melihat informasi dan pratinjau dari data yang dimasukkan berupa tabel sebanyak dua puluh baris. Informasi yang ditampilkan antara lain jumlah baris, jumlah kolom, dan nama kolom.

3. Melihat visualisasi data

Pengguna juga dapat melihat visualisasi dari data yang di-*input*-kan berupa *timeline* sederhana.

4. Memilih algoritme yang digunakan untuk *data precomputing*

Pengguna dapat memilih algoritme yang akan digunakan untuk *data precomputing*, yaitu algoritme *k-MPPTI* dan *Brute Force*.

5. Memasukkan kueri pencarian

Pengguna dapat memasukkan kueri pencarian berupa jumlah

produk ( $k$ ) dan interval waktu.

6. Melihat hasil kueri

Pengguna dapat melihat hasil kueri pencarian berupa  $k$ -produk dengan jumlah kontribusi pasar terbesar beserta skor kontribusi pasar-nya.

7. Melihat waktu eksekusi

Pengguna dapat melihat informasi terkait waktu eksekusi.

### 3.3 Perancangan Sistem

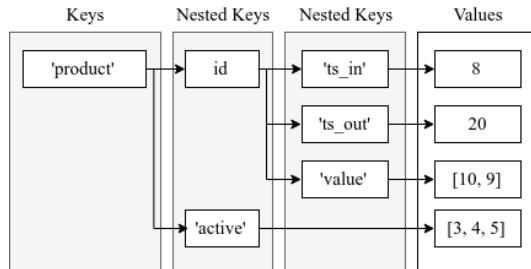
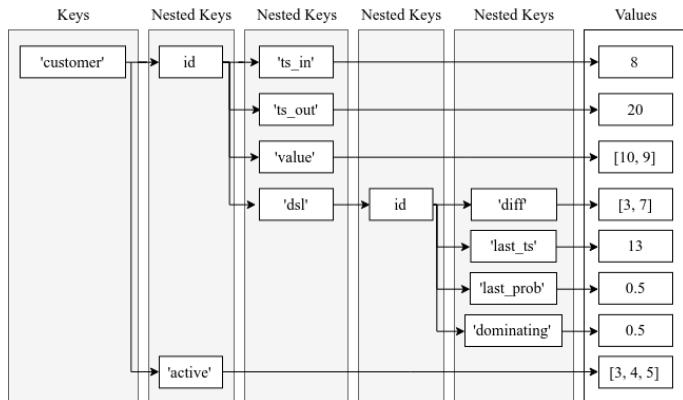
Perancangan sistem akan dibagi menjadi empat bagian, yakni struktur data, algoritme utama, algoritme pembanding menggunakan metode *brute force*, dan arsitektur aplikasi.

#### 3.3.1 Struktur Data

Struktur data adalah suatu cara untuk menyimpan, menyusun, mengelompokkan, dan merepresentasikan suatu data. Ada tiga struktur data utama yang digunakan dalam komputasi  $k$ -MPPTI, yaitu *Data Storage*, *Event Queue*, dan *Pandora Box*.

##### 3.3.1.1 *Data Storage*

*Data Storage* adalah sebuah struktur data *dictionary* yang digunakan untuk menyimpan data produk dan pelanggan. Struktur data *dictionary* lebih efisien untuk pencarian data karena menggunakan konsep *key-value pairs*, berbeda dengan struktur data *list* atau *array* yang menggunakan indeks untuk mengakses nilai suatu data.

Gambar 3.1 Struktur Data *Dictionary* ProdukGambar 3.2 Struktur Data *Dictionary* Pelanggan

Struktur data *dictionary* yang digunakan berbentuk *nested dictionary* yang terdiri dari dua *key* utama, yaitu '*product*' yang menyimpan data produk dan '*customer*' yang menyimpan data pelanggan. Struktur *nested key* masing-masing data dijelaskan pada Gambar 3.1 dan 3.2 dan deskripsinya dijelaskan pada Tabel 3.3.

Tabel 3.3 *Key* dari *Data Storage*

<b>Key</b>	<b>Deskripsi</b>
'product'	Menyimpan data produk
'customer'	Menyimpan data pelanggan
<i>id</i>	ID data produk atau pelanggan dijadikan sebagai <i>key</i>
'active'	Menyimpan ID data produk atau pelanggan yang sedang aktif dalam bentuk <i>array</i>
'ts_in'	Menyimpan <i>timestamp</i> atau waktu masuk
'ts_out'	Menyimpan <i>timestamp</i> atau waktu keluar
'value'	Menyimpan nilai data produk atau pelanggan pada semua dimensi dalam bentuk <i>array</i>
'dsl'	Menyimpan hasil <i>dynamic skyline</i> dalam bentuk <i>dictionary</i> dengan <i>id</i> produk sebagai <i>key</i>
'diff'	Menyimpan selisih antara nilai data produk dan pelanggan pada masing-masing dimensi
'last_ts'	Menyimpan <i>timestamp</i> terakhir saat diperbarui ke <i>Pandora Box</i>
'last_prob'	Menyimpan probabilitas terakhir saat diperbarui ke <i>Pandora Box</i>
'dominating'	Menyimpan ID produk lain yang pernah didominasi

### 3.3.1.2 *Event Queue*

*Event* adalah titik tempat terjadinya perubahan di dalam himpunan data. Sebuah himpunan data dikatakan berubah jika ada data yang masuk atau keluar. Sehingga, ada empat jenis *event* yang terjadi dalam penelitian ini, yaitu:

1. Data Produk masuk
2. Data Produk keluar
3. Data Pelanggan masuk
4. Data Pelanggan keluar

Produk dan pelanggan disebut dengan pemilik *event*, sedangkan masuk dan keluar disebut dengan aksi *event*.

*Event Queue* adalah sebuah struktur data *queue* yang berfungsi untuk menyimpan *event-event* yang terjadi di dalam himpunan data. *Queue* memiliki prinsip FIFO (*First In First Out*), sehingga *event* akan diproses secara berurutan menurut antrian waktu. *Event Queue* menyimpan empat informasi, yaitu *timestamp*, pemilik *event*, ID pemilik *event*, dan aksi *event*. Untuk lebih jelasnya, atribut *Event Queue* dijelaskan pada Tabel 3.4.

Tabel 3.4 Atribut dari *Event Queue*

Atribut	Deskripsi
<i>timestamp</i>	Waktu terjadinya <i>event</i>
<i>owner</i>	Pemilik <i>event</i> (produk = 0, pelanggan = 1)
<i>ownerId</i>	ID pemilik <i>event</i>
<i>action</i>	Jenis aksi yang dilakukan (masuk = 0, keluar = 1)

### 3.3.1.3 *Pandora Box*

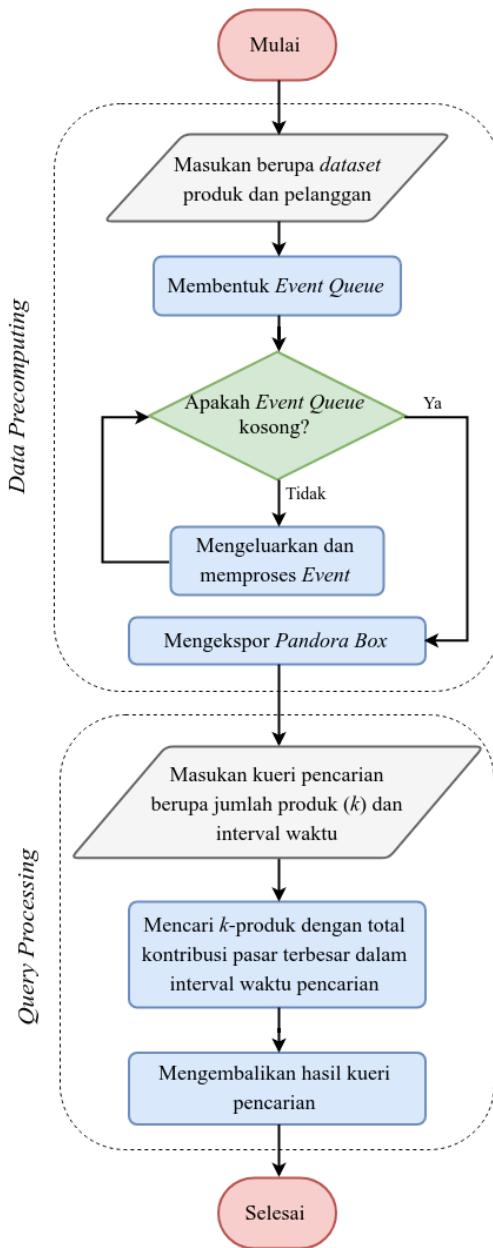
*Pandora Box* adalah sebuah struktur data *array* dua dimensi yang terdiri dari sumbu *x* (*time series*) dan sumbu *y* (produk). Struktur data ini digunakan untuk menyimpan skor kontribusi pasar produk setiap waktu. Menggunakan contoh *dataset* pada Tabel 3.6, maka model *Pandora Box* yang terbentuk adalah seperti pada Tabel 3.5.

Tabel 3.5 Contoh *Pandora Box* dari *Dataset* 3.6

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$p_1$															
$p_2$															
$p_3$															
$p_4$															
$p_5$															

### 3.3.2 Algoritme Utama

Sebagaimana yang telah dijelaskan sebelumnya bahwa algoritme k-MPPTI terdiri dari dua tahap pemrosesan, yaitu *data precomputing* dan *query processing*. Secara garis besar, alur kerja sistem secara umum disajikan dalam bentuk diagram alur yang dapat dilihat pada Gambar 3.3.



Gambar 3.3 Diagram Alur Algoritme k-MPPTI

Tahap *data precomputing* bertujuan untuk menghitung kontribusi pasar masing-masing produk berdasarkan preferensi pelanggan. Diawali dengan pembentukan *Event Queue* untuk mencatat semua *event* yang terjadi selama pemrosesan data. Kemudian, memproses *event-event* tersebut menggunakan algoritme pemrosesan berdasarkan jenis *event*-nya. Terakhir adalah menghitung kontribusi pasar dan menyimpannya ke dalam struktur data *array* bernama *Pandora Box*.

*Pandora Box* kemudian digunakan sebagai masukan pada tahap *query processing*. Diawali dengan masukan kueri pencarian berupa jumlah produk ( $k$ ) dan interval waktu pencarian. Kemudian, mencari produk sejumlah  $k$  yang memiliki total skor kontribusi pasar terbesar selama interval waktu pencarian. Terakhir adalah mengembalikan hasil kueri pencarian berupa  $k$ -produk yang paling menjanjikan kepada pengguna.

Untuk memudahkan interaksi antara pengguna dan sistem, dibuatlah aplikasi berbasis web yang memudahkan pengguna memasukkan data produk dan pelanggan, melihat pratinjau dan visualisasi data, memasukkan kueri pencarian, serta melihat hasil kueri pencarian.

### 3.3.2.1 Data Precomputing

*Data precomputing* adalah sebuah proses yang dapat menunjang performa algoritme *query processing* supaya dapat bekerja lebih efektif dan efisien. Tidak adanya proses *data precomputing* menyebabkan pengulangan komputasi data setiap kali seseorang memasukkan kueri pencarian, sehingga komputasi data cukup dilakukan satu kali di awal (*precomputing*) karena data yang digunakan adalah *historical data*, yaitu data yang dikumpulkan dari kejadian yang telah lalu. Berbeda halnya jika data yang digunakan adalah *streaming data* yang nilainya terus berubah dalam periode waktu tertentu.

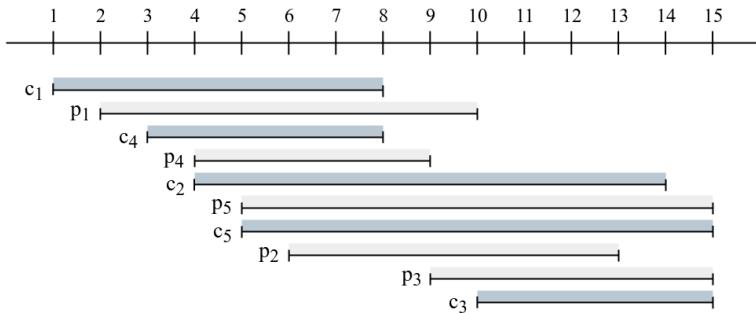
Tabel 3.6 Contoh *Dataset*  
 (a) Produk *P* dan (b) Preferensi Pelanggan *C*

<b>ID</b>	<b>Timestamp</b>		<b>Nilai</b>	
	$t_i$	$t_e$	$d_1$	$d_2$
$p_1$	2	10	6	3
$p_2$	6	13	4	12
$p_3$	9	15	6	15
$p_4$	4	9	9	5
$p_5$	5	15	12	10

<b>ID</b>	<b>Timestamp</b>		<b>Nilai</b>	
	$t_i$	$t_e$	$d_1$	$d_2$
$c_1$	1	8	2	8
$c_2$	4	14	4	10
$c_3$	10	15	6	11
$c_4$	3	8	8	12
$c_5$	5	15	9	10

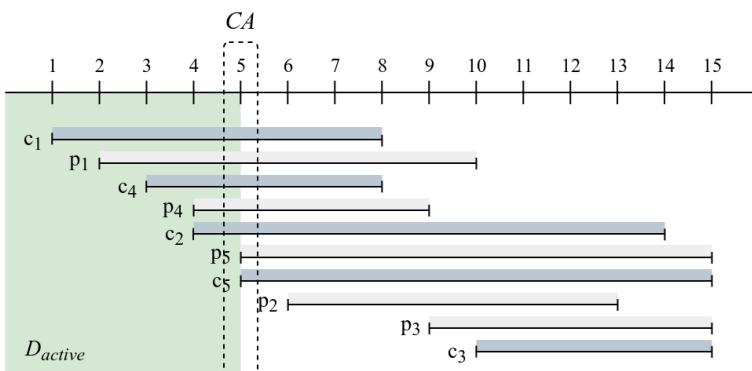
Pada Tabel 3.6, diberikan contoh *dataset* produk *P* dan preferensi pelanggan *C* yang setiap datanya direpresentasikan sebagai titik  $d$ -dimensi dengan serial waktu  $[t_i : t_e]$ . Data multidimensi dengan serial waktu dimodelkan sebagai lini masa yang diilustrasikan pada Gambar 3.4.



Gambar 3.4 Ilustrasi Data Multidimensi dengan Serial Waktu

Pada komputasi  $k$ -MPP [1], semua data dikomputasi sekaligus dalam satu waktu, namun hal ini tidak dapat diterapkan pada komputasi  $k$ -MPPTI. Pendekatan yang digunakan adalah

membagi lini masa menjadi beberapa area komputasi. Sebagaimana yang ditunjukkan oleh Gambar 3.5, *CA* adalah area komputasi saat ini dan semua data di sebelah kiri *CA* disebut sebagai data aktif, dinotasikan dengan  $D_{active}$ . Area *CA* nantinya akan dikomputasi menggunakan algoritme yang mengadopsi komputasi *k*-MPP [1] dengan berbagai penyesuaian.



Gambar 3.5 Ilustrasi Pemrosesan Data

Dalam *k*-MPPTI, tidak semua *timestamp* dianggap sebagai area *CA* karena akan menghabiskan waktu komputasi. Hasil komputasi akan berubah jika ada perubahan susunan data, yakni ada data yang masuk atau keluar. Setiap ada data yang masuk atau keluar akan dicatat sebagai *event e* dan dimasukkan ke dalam *Event Queue*. *Event-event* ini nantinya akan diproses satu persatu secara berurutan sebagai area *CA*. Contoh *Event Queue* yang terbentuk dari *dataset* pada Tabel 3.6 ditunjukkan pada Tabel 3.7.

Tabel 3.7 *Event Queue*

ID Event	Timestamp	ID Data	Aksi
$e_1$	1	$c_1$	Masuk

<b>ID Event</b>	<b>Timestamp</b>	<b>ID Data</b>	<b>Aksi</b>
$e_2$	2	$p_1$	Masuk
$e_3$	3	$c_4$	Masuk
$e_4$	4	$p_4$	Masuk
$e_5$	4	$c_2$	Masuk
$e_6$	5	$p_5$	Masuk
$e_7$	5	$c_5$	Masuk
$e_8$	6	$p_2$	Masuk
$e_9$	8	$c_1$	Keluar
$e_{10}$	8	$c_4$	Keluar
$e_{11}$	9	$p_3$	Masuk
$e_{12}$	9	$p_4$	Keluar
$e_{13}$	10	$c_3$	Masuk
$e_{14}$	10	$p_1$	Keluar
$e_{15}$	13	$p_2$	Keluar
$e_{16}$	14	$c_2$	Keluar
$e_{17}$	15	$p_3$	Keluar
$e_{18}$	15	$p_5$	Keluar
$e_{19}$	15	$c_3$	Keluar
$e_{20}$	15	$c_5$	Keluar

Ada empat jenis algoritme pemrosesan berdasarkan jenis *event*-nya, yaitu:

1. *Product Insertion*
2. *Product Deletion*
3. *Customer Insertion*
4. *Customer Deletion*

Terlepas dari empat jenis proses di atas, ada dua jenis komputasi *skyline* yang digunakan dalam *data precomputing*, yaitu *dynamic skyline* dan *reverse skyline*. Dua komputasi tersebut

digunakan sebagai metode perhitungan probabilitas dan kontribusi pasar [1].

### 3.3.2.1.1 Komputasi *Dynamic Skyline*

Sebagaimana yang telah dijelaskan pada Bab Tinjauan Pustaka, komputasi *dynamic skyline* digunakan untuk mencari produk terbaik dari sudut pandang pelanggan [1]. *Dynamic skyline* [2] dari seorang pelanggan  $c_1 \in C$ , dinotasikan dengan  $DSL(c_1)$ , berisi semua produk  $p_1 \in P$  yang tidak didominasi oleh produk lain  $p_2 \in P$  berdasarkan preferensi pelanggan  $c_1$ ,  $p_2 \not\prec_{c_1} p_1$ .

Proses komputasi *dynamic skyline* diawali dengan perhitungan selisih absolut dari nilai masing-masing dimensi antara pelanggan dan produk, dinotasikan dengan:

$$diff^i = |c_1^i - p^i| \quad (3.1)$$

Selanjutnya, mengecek dominansi dinamis antar produk dengan membandingkan selisih absolut-nya. Misalnya, ada dua produk yang akan dibandingkan, dinotasikan dengan  $p_s$  sebagai subjek yang dibandingkan dan  $p_o$  sebagai objek pembanding. Berdasarkan syarat dominansi dinamis (Persamaan 2.1),  $p_s$  dikatakan mendominasi  $p_o$  jika dan hanya jika:

$$\begin{aligned} (a) \quad & diff_s^i \leq diff_o^i, \forall i \in [1, \dots, d] \\ (b) \quad & diff_s^i < diff_o^i, \exists i \in [1, \dots, d] \end{aligned} \quad (3.2)$$

Pengecekan dominansi dinamis ini dilakukan secara iteratif sampai dipastikan suatu  $p_1$  tidak didominasi oleh  $p_2$  lain sama sekali. Jika  $p_1$  pernah didominasi, maka  $p_1$  tidak dapat menjadi hasil *dynamic skyline*.

Menggunakan contoh *dataset* pada Tabel 3.6, dengan

mengabaikan *timestamp*-nya akan didapatkan perhitungan hasil *dynamic skyline* seperti pada Tabel 3.8.

Tabel 3.8 Hasil Perhitungan *Dynamic Skyline*

$DSL(c_1)$	$\{2, 4, 5\}$
$DSL(c_2)$	$\{2, 5\}$
$DSL(c_3)$	$\{2, 3\}$
$DSL(c_4)$	$\{2, 3, 4\}$
$DSL(c_5)$	$\{4, 5\}$

### 3.3.2.1.2 Komputasi *Reverse Skyline*

Komputasi *reverse skyline* digunakan untuk mencari pelanggan potensial dari sudut pandang produsen [1]. *Reverse skyline* [3] 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.

Sebagaimana yang telah dijelaskan pada Tinjauan Pustaka, komputasi *reverse skyline* diawali dengan menentukan *orthant* dari produk, dinotasikan dengan  $O$ . Dalam geometri, *orthant* adalah analog dalam ruang data  $d$ -dimensi atau biasa dikenal sebagai kuadran dalam bidang dua dimensi. Setiap produk  $p$  memiliki  $2^d$  *orthant* pada data  $d$ -dimensi.

*Orthant* ditandai menggunakan bilangan biner. Sebagai contoh, terdapat empat *orthant* pada bidang dua dimensi, yaitu  $O_{00}$ ,  $O_{01}$ ,  $O_{10}$ , dan  $O_{11}$ , dan delapan *orthant* pada bidang tiga dimensi, yaitu  $O_{000}$ ,  $O_{001}$ ,  $O_{010}$ ,  $O_{011}$ ,  $O_{100}$ ,  $O_{101}$ ,  $O_{110}$  dan  $O_{111}$ . Penggunaan bilangan biner bertujuan untuk menandai batas wilayah sebuah *orthant*. Misalnya, *orthant*  $O_{010}$  dari produk  $p_1$  memiliki wilayah dengan batas-batas sebagai berikut: sumbu  $x[0 : pos_x(p_1)]$ , sumbu  $y[pos_y(p_1) : max_y]$ , dan sumbu  $z[0 : pos_z(p_1)]$ .

Langkah selanjutnya adalah menghitung *midpoint* atau titik tengah antara produk kueri dan produk lainnya, misalnya  $p_1$  (sebagai titik kueri) dan  $p_2 \in P$ , menggunakan rumus berikut:

$$m_2^i = \frac{(p_1^i + p_2^i)}{2} \quad (3.3)$$

Kemudian, menentukan *midpoint skyline* atau *mid-skyline* [8] pada setiap *orthant*.

Langkah terakhir adalah mengecek setiap pelanggan  $c \in C$  apakah didominasi oleh hasil *mid-skyline* pada masing-masing *orthant* atau tidak (Persamaan 2.3). Jika  $c$  didominasi, maka  $c$  tidak dapat menjadi hasil *reverse skyline*.

Menggunakan contoh *dataset* pada Tabel 3.6, dengan mengabaikan *timestamp*-nya akan didapatkan perhitungan hasil *reverse skyline* seperti pada Tabel 3.9.

Tabel 3.9 Hasil Perhitungan *Reverse Skyline*

$RSL(p_1)$	$\{\}$
$RSL(p_2)$	$\{1, 2, 3, 4\}$
$RSL(p_3)$	$\{3, 4\}$
$RSL(p_4)$	$\{1, 4, 5\}$
$RSL(p_5)$	$\{1, 2, 5\}$

### 3.3.2.1.3 Perhitungan Probabilitas

Setelah mendapatkan hasil *dynamic skyline* dan *reverse skyline* pada Tabel 3.8 dan 3.9, selanjutnya adalah menghitung probabilitas masing-masing produk  $p$  dipilih oleh pelanggan  $c$ , dinotasikan dengan  $Pr(c, p|P), \forall c \in C$ , yang telah dijelaskan pada Persamaan 2.4.

Probabilitas  $p$  dipilih oleh  $c$  yang tidak menjadi hasil

$RSL(p)$  pasti 0 karena  $p$  tersebut tidak menjadi hasil  $DSL(c)$ , sehingga untuk mempersingkat komputasi, perhitungan probabilitas dinotasikan menjadi:

$$Pr(c, p|P), \forall c \in RSL(p) \quad (3.4)$$

Sehingga, akan didapatkan hasil perhitungan probabilitas sebagai berikut.

Tabel 3.10 Hasil Perhitungan Probabilitas

$p_1$	-	0
$p_2$	$Pr(c_1, p_2 P)$	$\frac{1}{3} = 0.33$
	$Pr(c_2, p_2 P)$	$\frac{1}{2} = 0.5$
	$Pr(c_3, p_2 P)$	$\frac{1}{2} = 0.5$
	$Pr(c_4, p_2 P)$	$\frac{1}{3} = 0.33$
$p_3$	$Pr(c_3, p_3 P)$	$\frac{1}{2} = 0.5$
	$Pr(c_4, p_3 P)$	$\frac{1}{3} = 0.33$
$p_4$	$Pr(c_1, p_4 P)$	$\frac{1}{3} = 0.33$
	$Pr(c_4, p_4 P)$	$\frac{1}{3} = 0.33$
	$Pr(c_5, p_4 P)$	$\frac{1}{2} = 0.5$
$p_5$	$Pr(c_1, p_5 P)$	$\frac{1}{3} = 0.33$
	$Pr(c_2, p_5 P)$	$\frac{1}{2} = 0.5$
	$Pr(c_5, p_5 P)$	$\frac{1}{2} = 0.5$

### 3.3.2.1.4 Perhitungan Kontribusi Pasar (*Market Contribution*)

Hasil perhitungan probabilitas produk  $p$  pada Tabel 3.10 kemudian diakumulasikan menjadi skor kontribusi pasar sebagaimana yang dijelaskan pada Persamaan 2.6. Sehingga akan didapatkan hasil pada Tabel 3.11.

Tabel 3.11 Hasil Perhitungan Kontribusi Pasar

$E(C, p_1 P)$	0
$E(C, p_2 P)$	1.66
$E(C, p_3 P)$	0.83
$E(C, p_4 P)$	1.16
$E(C, p_5 P)$	1.33

### 3.3.2.1.5 Proses *Product Insertion*

Proses *Product Insertion* adalah proses yang dijalankan ketika ada produk yang masuk, dinotasikan dengan  $p_{in}$ , ke dalam data aktif  $D_{active}$ . Proses ini sangat penting dilakukan karena ada kemungkinan jika produk baru dapat mendominasi produk lama, sehingga hasil *dynamic skyline* seorang pelanggan  $c \in C$  dan perhitungan probabilitasnya ikut berubah.

Secara garis besar, algoritme pemrosesan diawali dengan (1) menambah produk  $p_{in}$  ke dalam daftar produk aktif  $P_{active}$ . Kemudian, (2) menghitung  $RSL(p_{in})$ , (3) menghitung  $DSL(c), \forall c \in RSL(p_{in})$  dan diakhiri dengan (4) memperbarui *Pandora Box*. Mekanisme pembaruan *Pandora Box* adalah dengan cara mengakumulasikan skor hasil probabilitas setiap  $c \in RSL(p_{in})$  di dalam *Pandora Box* dengan indeks ID produk masuk dan *timestamp event*, dinotasikan dengan  $PBox[p_{in}][ts]$ .

### 3.3.2.1.6 Proses *Product Deletion*

Proses *Product Deletion* adalah proses yang dijalankan ketika ada produk yang keluar, dinotasikan dengan  $p_{out}$ , ke dalam data aktif  $D_{active}$ . Proses ini sangat penting dilakukan karena ada kemungkinan jika sebuah produk yang pernah menjadi hasil

*dynamic skyline* seorang pelanggan  $c \in C$  keluar, maka produk lain yang pernah didominasi akan menjadi hasil  $DSL(c)$  yang baru.

Secara garis besar, algoritme pemrosesan diawali dengan (1) memperbarui *Pandora Box* dengan cara mengecek *last\_ts* dan *last\_prob*-nya (Tabel 3.3) untuk mengisi indeks *PBox* sebelumnya yang kosong. Kemudian (2) menghitung  $RSL(p_{out})$  dan (3) menghitung  $DSL(c), \forall c \in RSL(p_{out})$ . Terakhir adalah menghapus produk  $p$  dari daftar produk aktif  $P_{active}$ .

### 3.3.2.1.7 Proses *Customer Insertion*

Proses *Customer Insertion* adalah proses yang dijalankan ketika ada pelanggan yang masuk, dinotasikan dengan  $c_{in}$ , ke dalam data aktif  $D_{active}$ .

Secara garis besar, algoritme pemrosesan diawali dengan (1) menambah pelanggan  $c_{in}$  ke dalam daftar pelanggan aktif  $C_{active}$ . Kemudian (2) menghitung *Initial DSL*( $c_{in}$ ) untuk mendapatkan hasil *dynamic skyline* awal dan diakhiri dengan (3) memperbarui *Pandora Box*.

### 3.3.2.1.8 Proses *Customer Deletion*

Proses *Customer Deletion* adalah proses yang dijalankan ketika ada pelanggan yang keluar, dinotasikan dengan  $c_{out}$ , dari data aktif  $D_{active}$ . Secara garis besar, algoritme pemrosesan diawali dengan (1) memperbarui *Pandora Box* dengan cara mengecek *last\_ts* dan *last\_prob*-nya (Tabel 3.3) untuk mengisi indeks *PBox* sebelumnya yang kosong. Kemudian (2) menghapus pelanggan  $c$  dari daftar pelanggan aktif  $C_{active}$ .

### 3.3.2.1.9 Pemrosesan Paralel

Untuk meningkatkan efisiensi waktu komputasi, penelitian ini memanfaatkan pemrosesan paralel, yaitu pemrosesan yang dilakukan secara bersamaan dalam satu waktu. Pemrosesan paralel hanya dapat dilakukan jika suatu proses tidak membutuhkan masukan dari keluaran proses sebelumnya, misalnya komputasi  $DSL(c), \forall c \in RSL(p)$  yang diilustrasikan pada Gambar ...

### 3.3.2.2 *Query Processing*

*Query processing* adalah algoritme pencarian produk sejumlah  $k$  yang paling menjanjikan, yaitu  $k$ -produk yang memiliki kontribusi pasar terbesar. Sesuai namanya, algoritme ini memproses kueri pencarian yang dimasukkan oleh pengguna.

$$k = MPPTI(PBox, k, [t_i : t_e]) \quad (3.5)$$

Kueri  $k$ -MPPTI, dimodelkan sebagai Persamaan 3.5, membutuhkan tiga jenis masukan saja, yaitu *Pandora Box* yang merupakan hasil keluaran dari proses *data precomputing*, bilangan bulat positif  $k$  yang lebih kecil dari  $|P|$  sebagai jumlah produk yang dicari, dan interval waktu pencarian yang terdiri dari waktu awal dan waktu akhir  $[t_i : t_e]$ .

Lain halnya jika tidak melalui proses *data precomputing*, maka kueri  $k$ -MPPTI dimodelkan menjadi Persamaan 3.6 yang selalu membutuhkan *dataset* produk  $P$  dan preferensi pelanggan  $C$  setiap kali diproses.

$$k = MPPTI(P, C, k, [t_i : t_e]) \quad (3.6)$$

Algoritme ini mengadopsi strategi pemilihan produk  $k$ -MPP,

yaitu memilih *subset*  $k$  produk  $P'$  dari  $P$  yang memiliki kontribusi pasar lebih besar dibandingkan dengan *subset*  $k$  produk  $P''$  dari  $P$  yang lain [1]. Perbedaannya adalah adanya pembatasan berupa interval waktu pencarian.

Sebagai contoh, seorang pengguna melakukan kueri  $k - MPPTI(PBox, 3, [4 : 14])$  yang diilustrasikan pada Gambar .... Pada interval waktu pencarian  $[4 : 14]$ , skor kontribusi pasar setiap produk diakumulasi, kemudian diurutkan dari yang terbesar. Produk sejumlah 3 teratas akan dikembalikan sebagai hasil dari kueri pencarian.

### 3.3.3 Algoritme *Brute Force*

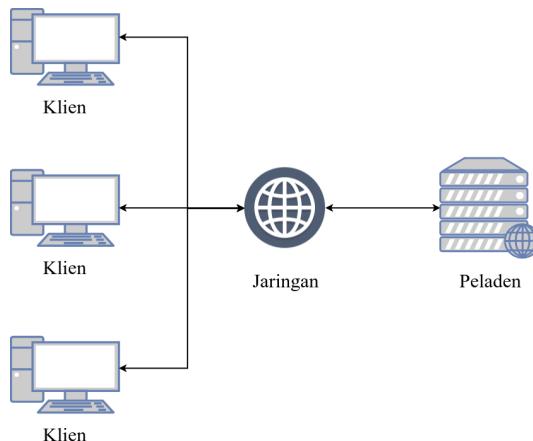
Algoritme *brute force* adalah algoritme yang mengimplementasikan metode *brute force*, yaitu melakukan percobaan terhadap semua kemungkinan dan hanya mengandalkan kekuatan pemrosesan komputer. Algoritme ini digunakan sebagai banding dari algoritme  $k$ -MPPTI.

Secara garis besar, struktur data dan pendekatan yang digunakan hampir sama dengan  $k$ -MPPTI, namun tidak ada tahap komputasi *RSL*, sehingga komputasi *DSL* dilakukan pada semua  $c \in C$  dengan cara membandingkan semua produk  $p \in P$  satu persatu. Selain itu, komputasi tidak dilakukan dengan secara paralel.

### 3.3.4 Arsitektur Aplikasi

Sistem akan diimplementasikan menggunakan arsitektur *client-server* seperti yang diilustrasikan pada Gambar 3.6.

Terdapat dua komponen utama dalam arsitektur ini, yaitu klien (*client*), pihak yang meminta atau menerima layanan, dan peladen (*server*), pihak yang memberikan atau mengirim layanan.



Gambar 3.6 Perancangan Arsitektur Aplikasi

Komponen-komponen ini terhubung ke jaringan, baik melalui kabel maupun nirkabel untuk melakukan transmisi data.

Klien mengimplementasikan antarmuka pengguna grafis atau APG (Inggris: *graphical user interface* atau GUI) berbasis web, sedangkan peladen mengimplementasikan *back-end service* yang berisi algoritme komputasi *k-MPPTI* yang terdiri atas algoritme *data precomputing* dan *query processing*. Web dibangun menggunakan Flask *microframework* dan bahasa Python, HTML, CSS, serta Javascript, sedangkan *back-end service* diimplementasikan menggunakan bahasa Python. Flask juga sekaligus berperan sebagai peladen web (*web server*).

Pengguna melakukan masukan data melalui web, kemudian data tersebut dikirimkan ke *back-end service* untuk dilakukan proses *data precomputing*. Setelah hasil *data precomputing* selesai, pengguna melakukan masukan kueri pencarian. Kueri pencarian tersebut akan dikirimkan ke *back-end service* untuk dilakukan proses *query processing*. Hasil yang didapatkan akan dikembalikan

ke klien, disertai dengan visualisasi data.

*Halaman ini sengaja dikosongkan*

## **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
11    Node* update() {
12        size = 1;
13        if ( left ) size += left->size;
14        if ( right ) size += right->size;
15        return this;
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

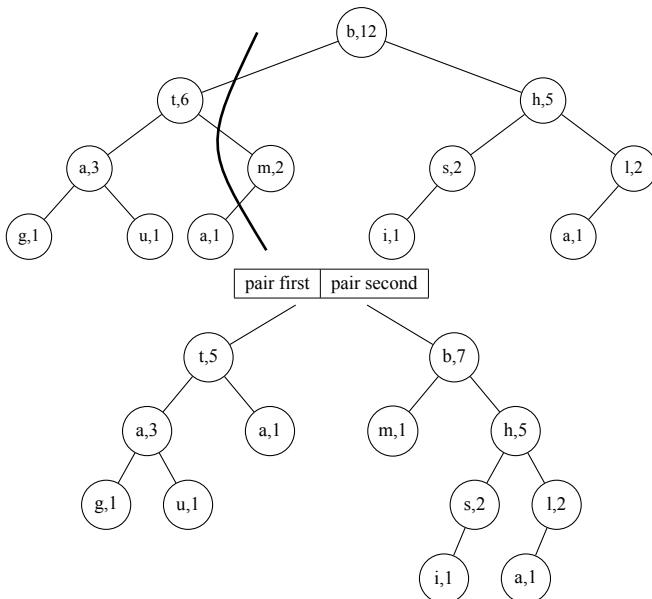
#### 4.3.7 Implementasi Fungsi Getsize

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

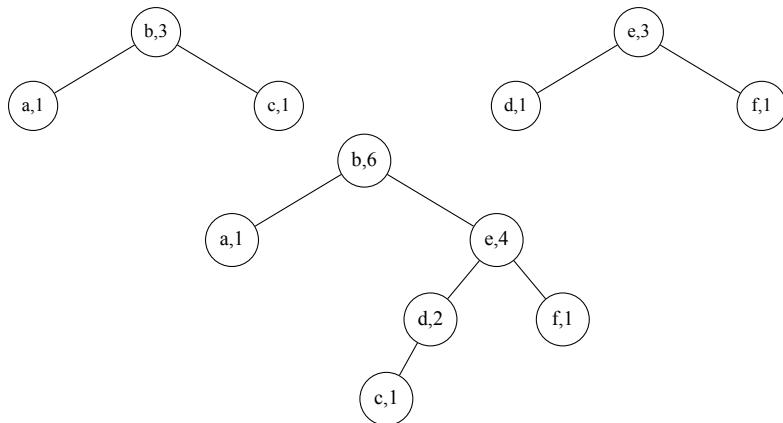
Fungsi Random digunakan untuk menentukan posisi *node* pada saat penggabungan dua buah *rope*. Digunakan fungsi Random agar posisi *node* tidak konstan dan berat suatu *rope* tidak selalu sama. Implementasi fungsi Random ditunjukkan pada Kode Sumber 4.10.

```

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.

```

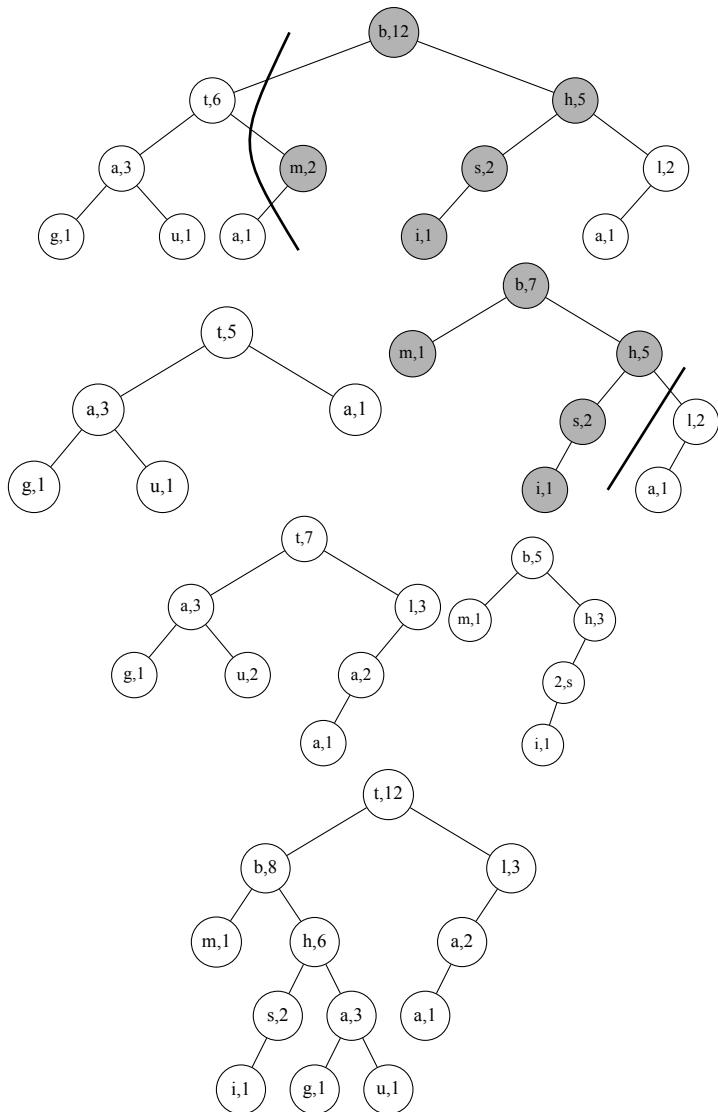
1 Node* mutable_begin(Node* r, int x, int len) {
2     pair<Node*, Node*> fir = split(r, x);
3     pair<Node*, Node*> sec = split(fir.second, len);
4     return concat(sec.first, concat(fir.first,\n
5         sec.second));
6 }
```

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

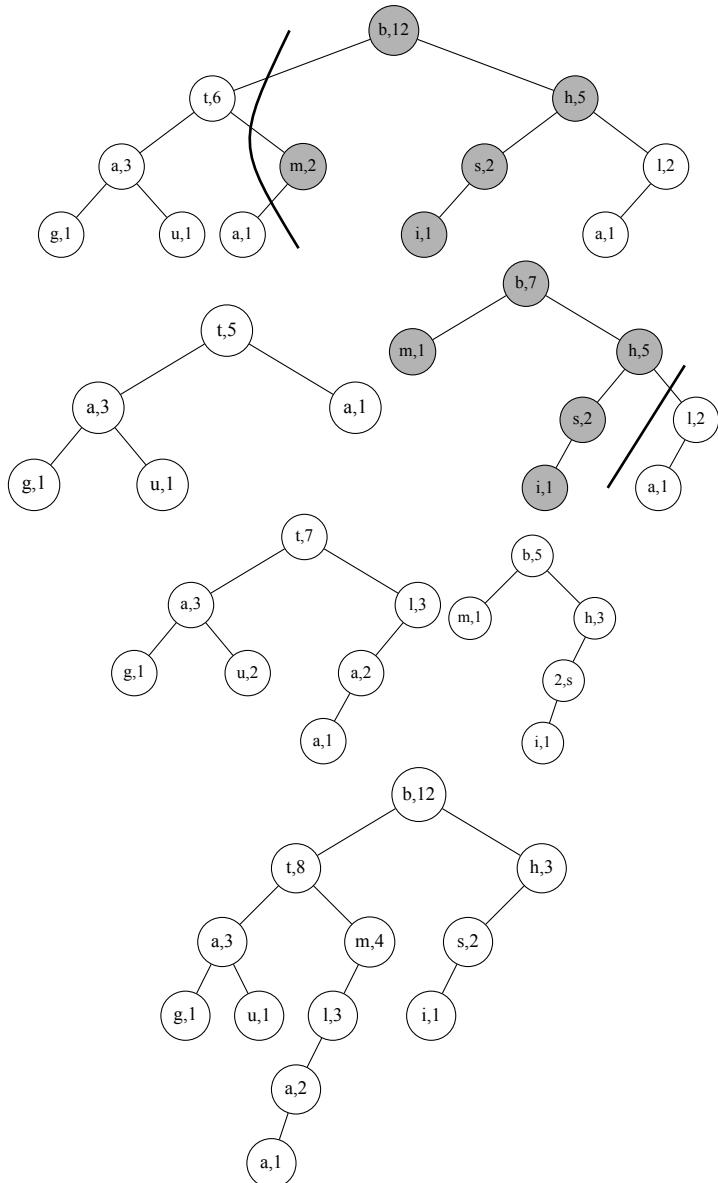
#### 4.3.14 Implementasi Fungsi Print

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*

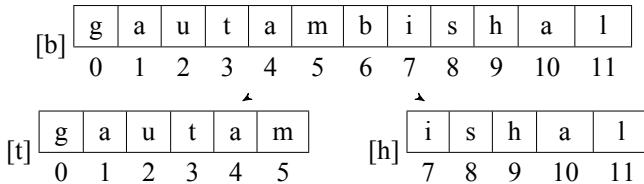
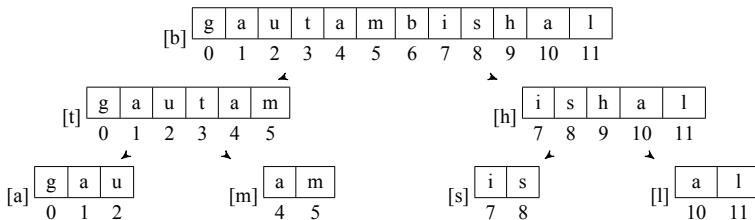
[b]	g	a	u	t	a	m	b	i	s	h	a	l
	0	1	2	3	4	5	6	7	8	9	10	11

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,

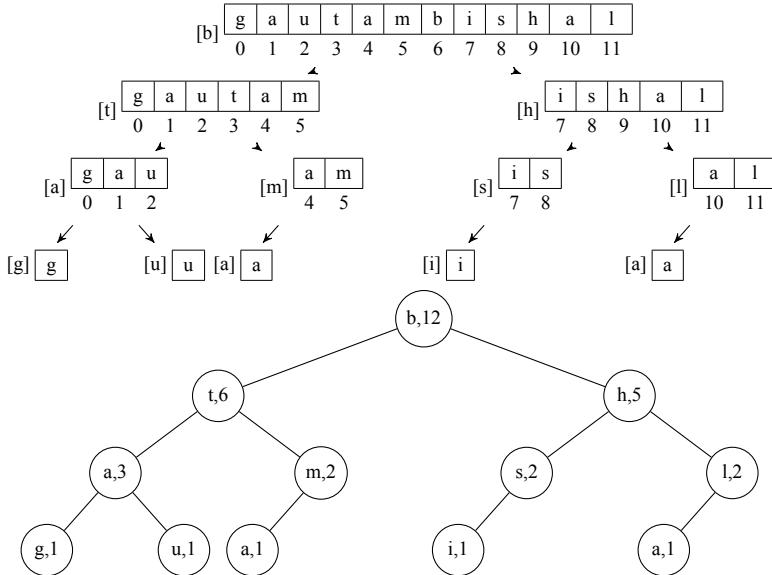
$y = 1$ ,  $left.weight = 6$ ,  $idx = left.weight$ ;  $idx \leq y$ ; dilanjutkan ke anak kiri;

$y = 1$ ,  $left.weight = 3$ ,  $idx = left.weight$ ;  $idx \leq y$ ; 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  $type = 2$ ,  $x = 0$ ,  $y = 5$ .

Gambar 5.5 Struktur *Rope* yang Terbentuk

Proses perubahan pada *rope* adalah sebagai berikut,

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

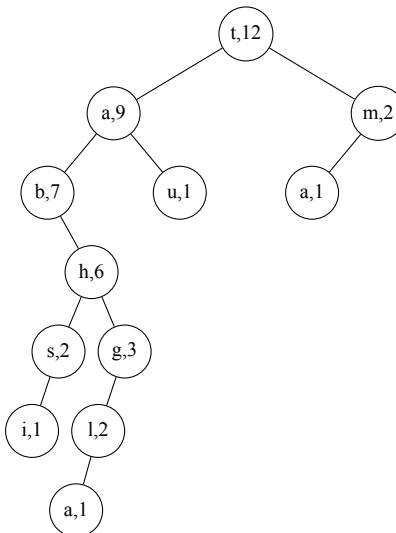
Dikarenakan  $x = 0$  operasi Split menghasilkan *rope*  $R_1$  yang berisi NULL dan  $R_2$  sisa dari potongan *rope* yang berarti keseluruhan *rope* saat ini.

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  $type = 3$ ,  $y = 0$ . 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
```

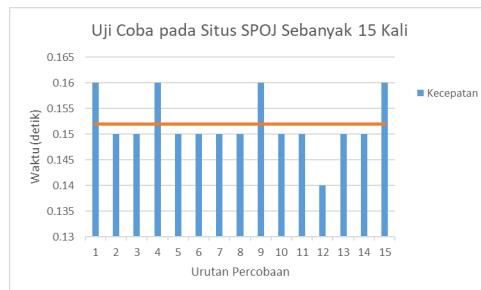
Gambar 5.7 Hasil Luaran Program pada Contoh Kasus Uji Alphabetic Rope

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 16:18:03	Alphabetic Rope	accepted edit ideone.it	0.15	5.0M	C++ 4.3.2
----------	------------------------	-----------------	----------------------------	------	------	--------------

Gambar 5.8 Hasil Uji Coba pada Situs Penilaian SPOJ

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.

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

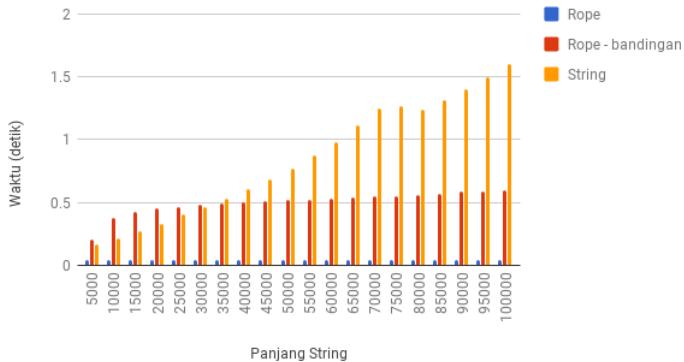
### 5.3 Uji Coba Kinerja

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

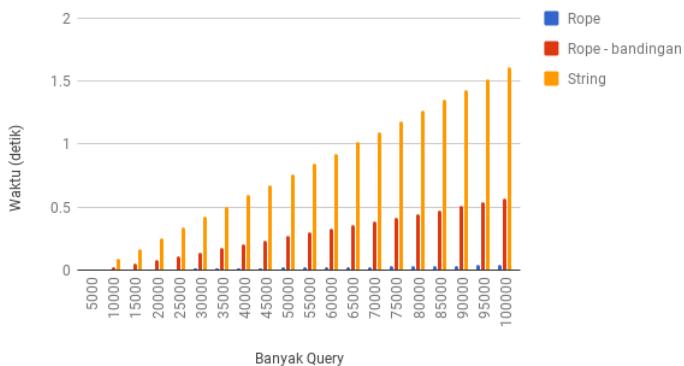
Pada Gambar 5.10 terlihat bahwa untuk  $N = 5000$  sampai  $N = 10^5$  dengan banyak  $Q = 10^5$ , struktur *Rope* yang dibuat untuk penyelesaian menjawab kueri *k-Most Promising Products* (*k-MPP*) berbasis interval waktu pada data multidimensi dengan serial waktu menunjukkan waktu yang lebih cepat dibandingan dengan algoritma *String* dan stuktur 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 ??.

Perbandingan Performa Operasi 1

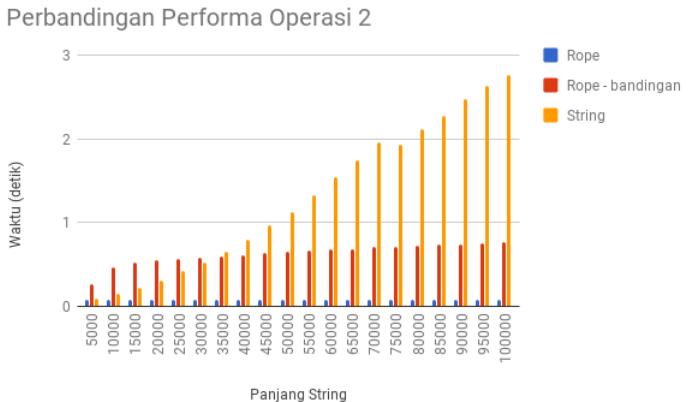


Gambar 5.10 Hasil Uji Coba pada Operasi 1 dengan Jumlah *Query* Tetap dan Panjang *String* Bertambah

Perbandingan performa Operasi 1



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

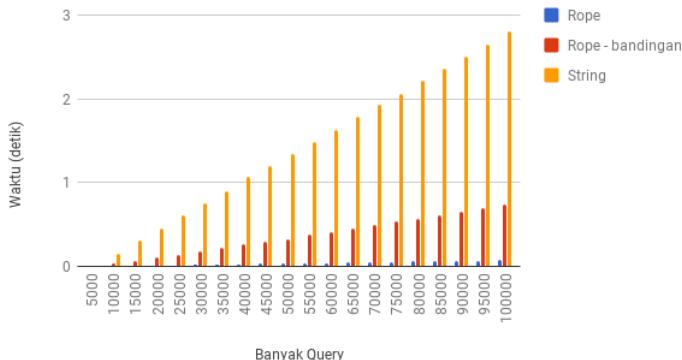


Gambar 5.12 Hasil Uji Coba pada Operasi 2 dengan Jumlah *Query* Tetap dan Jumlah *String* Bertambah

### 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 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 ??.

Perbandingan Performa Operasi 2



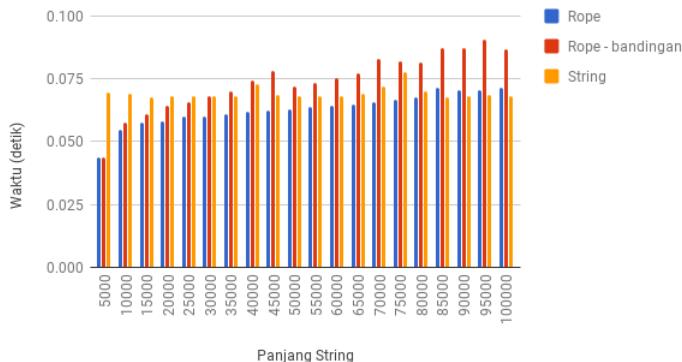
Gambar 5.13 Hasil Uji Coba pada Operasi 2 dengan Panjang *String* Tetap dan Jumlah *Query* Bertambah

### 5.3.3 Operasi 3 Mencetak Karakter pada Indeks ke-Y

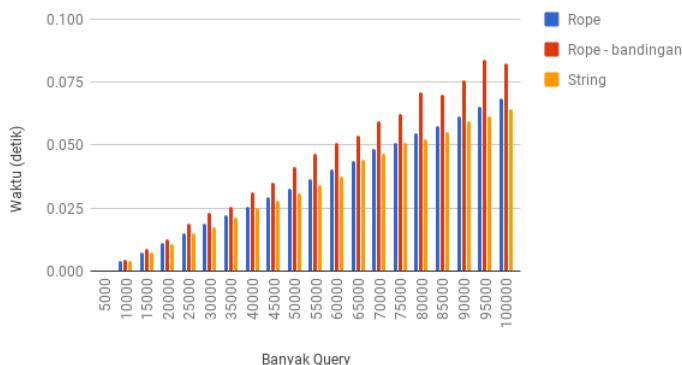
Pada Gambar 5.14 terlihat bahwa untuk  $N = 5000$  sampai  $N = 10^5$  dengan banyak  $Q = 10^5$ , algoritma String untuk penyelesaian menjawab kueri *k-Most Promising Products (k-MPP)* berbasis interval waktu pada data multidimensi dengan serial waktu menunjukkan waktu yang lebih cepat dibandingkan dengan struktur data Rope yang dibuat dan struktur 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*-bandingan terletak pada operasi *insert*. Pada *rope* bandingan operasi *insert* dilakukan sama seperti algoritma *insert* pada BST. Sedangkan pada *rope*, *insert* dilakukan dengan membagi dua panjang *string* kemudian dimasukkan ke dalam *tree*.

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

## 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 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] E. Dellis and B. Seeger, "Efficient Computation of Reverse Skyline Queries," *VLDB Endowment*, pp. 291-302, 2007.
- [4] B. Jiang and J. Pei, "Online Interval Skyline Queries on Time Series," *IEEE International Conference on Data Engineering*, pp. 1036-1047, 2009.
- [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] S. Borzsonyi, D. Kossmann and K. Stocker, "The Skyline Operator," In: *ICDE*, pp. 421-430, 2001.
- [7] 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.
- [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] Python.org. "What is Python? Executive Summary". [Online]. Available: <https://www.python.org/doc/essays/blurb/>. [Diakses: 18 Mei 2019].
- [10] Merriam-webster.com. "Definition of DATA". [Online]. Available: <https://www.merriam-webster.com/dictionary/data>. [Diakses: 13 Mei 2019]

## **BIODATA PENULIS**



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](mailto:hafarafirdausi@gmail.com).