ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - IF184952

DESAIN DAN ANALISIS ALGORITMA GENERALIZED MULTI-PARAMETER RANKING PADA SISTEM TER-DISTRIBUSI PADA STUDI KASUS PPDB 2019

JONATHAN REHUEL LEWERISSA
NRP 05111640000105

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

Dosen Pembimbing 2
Yudhi Purwananto, S.Kom., M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya, 2020

[*Halaman ini sengaja dikosongkan*]

TUGAS AKHIR - IF184952

DESAIN DAN ANALISIS ALGORITMA GENERALIZED MULTI-PARAMETER RANKING PADA SISTEM TER-DISTRIBUSI PADA STUDI KASUS PPDB 2019

JONATHAN REHUEL LEWERISSA
NRP 05111640000105

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

Dosen Pembimbing 2
Yudhi Purwananto, S.Kom., M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya, 2020

[*Halaman ini sengaja dikosongkan*]

UNDERGRADUATE THESIS - IF184952

DESIGN AND ANALYSIS FOR GENERALIZED MULTI-PARAMETER RANKING ALGORITHM IN DISTRIBUTED SYSTEM WITH PPDB 2019 STUDY CASE

JONATHAN REHUEL LEWERISSA
NRP 05111640000105

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

Supervisor 2
Yudhi Purwananto, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2020

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

DESAIN DAN ANALISIS ALGORITMA GENERALIZED MULTI-PARAMETER RANKING PADA SISTEM TERDISTRIBUSI PADA STUDI KASUS PPDB 2019

TUGAS AKHIR

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

Oleh:
Jonathan Rehuel Lewerissa
NRP. 05111640000105

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Rully Soelaiman, S.Kom., M.Kom. .....

NIP. 19700213194021001 (Pembimbing 1)

Yudhi Purwananto, S.Kom., M.Kom. .....

NIP. 197007141997031002 (Pembimbing 2)

SURABAYA
AGUSTUS 2020

[*Halaman ini sengaja dikosongkan*]

ABSTRAK

DESAIN DAN ANALISIS ALGORITMA GENERALIZED MULTI-PARAMETER RANKING PADA SISTEM TERDISTRIBUSI PADA STUDI KASUS PPDB 2019

Nama	:	Jonathan Rehuel Lewerissa
NRP	:	05111640000105
Departemen	:	Departemen Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, ITS
Pembimbing I	:	Rully Soelaiman, S.Kom., M.Kom.
Pembimbing II	:	Yudhi Purwananto, S.Kom., M.Kom.

ABSTRAK

Permasalahan pemeringkatan merupakan sebuah permasalahan yang umum ditemui dalam dunia pendidikan, khususnya dalam melakukan pemilihan calon peserta didik. Parameter pemeringkatan yang digunakan dapat berubah sesuai dengan kriteria yang dibutuhkan serta kondisi yang terjadi, sehingga dibutuhkan pemeringkatan yang bersifat general agar mampu melakukan perubahan parameter tanpa perlu mengubah kode sumber.

Tugas akhir ini bertujuan untuk memodelkan Generalized Multiparameter Ranking pada sistem terdistribusi. Model yang didesain diambil dari studi kasus PPDB tingkat pendidikan dasar dan menengah tahun 2019. Model yang kemudian didesain bersifat general untuk berbagai studi kasus/parameter serta mampu diimplementasikan pada sistem terdistribusi. Hasil dari pemodelan kemudian diimplementasikan serta diuji dengan himpunan data pada stu-

di kasus PPDB tahun 2019 dengan hasil implementasi yang mampu melakukan pemeringkatan dengan parameter yang berbeda-beda.

Kata Kunci: generalized multiparameter ranking, pemeringkatan, sistem terdistribusi, PPDB

ABSTRACT

DESIGN AND ANALYSIS FOR GENERALIZED MULTIPARAMETER RANKING ALGORITHM IN DISTRIBUTED SYSTEM WITH PPDB 2019 STUDY CASE

Name	:	Jonathan Rehuel Lewerissa
Student ID	:	05111640000105
Department	:	Department of Informatics, Faculty of Intelligent Electrical and Informatics Technology, ITS
Supervisor I	:	Rully Soelaiman, S.Kom., M.Kom.
Supervisor II	:	Yudhi Purwananto, S.Kom., M.Kom.

ABSTRACT

Ranking problem is a problem commonly found in the education world, especially in a student registration and selection activity. Parameters used in ranking problem may differ in each unique situation based on the requirements and conditions, therefore requiring a general ranking model that can use multiple and different parameters without changing the underlying source code.

This thesis aims to model a Generalized Multiparameter Ranking model in a distributed system. The model is designed based on the elementary and middle school student selection case study (PPDB 2019). The designed model is then generalized to handle different cases/parameters in a distributed system. The implemented model is then tested using PPDB 2019 dataset study case, resulting in a implementation that is able to rank with differing parameters.

Keywords: generalized multiparameter ranking, ranking, distributed system, PPDB

[*Halaman ini sengaja dikosongkan*]

PRAKATA

Puji syukur kepada Tuhan Yang Maha Esa atas segala berkat, karunia, dan rahmat-Nya penulis dapat menyelesaikan tugas akhir dengan judul **Desain dan Analisis Algoritma Generalized Multi-Parameter Ranking pada Sistem Terdistribusi pada Studi Kasus PPDB 2019**.

Pengerjaan tugas akhir ini dilakukan untuk memenuhi syarat meraih gelar Sarjana di Departemen Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember.

Dengan selesainya tugas akhir ini, penulis berharap bahwa apa yang telah dikerjakan oleh penulis dapat memberikan manfaat bagi perkembangan ilmu pengetahuan, terutama di bidang teknologi informasi serta bagi penulis selaku peneliti.

Penulis ingin menyampaikan terima kasih kepada seluruh pihak yang telah mendukung penulis baik secara langsung maupun tidak langsung dalam pengerjaan tugas akhir antara lain:

1. Tuhan Yesus Kristus atas segala hikmat, kesehatan, kesempatan, serta penyertaan-Nya selama ini.
2. Ayah, ibu, adik, serta keluarga penulis yang selalu memberikan dukungan, perhatian, serta kasih sayang kepada penulis yang menjadi penyemangat pengerjaan tugas akhir ini.
3. Bapak Rully Soelaiman, S.Kom., M.Kom., selaku dosen pembimbing I yang telah meluangkan waktu memberikan ilmu, nasihat, bimbingan, wejangan, serta didikan bagi penulis selama penulis berkuliah maupun saat pengerjaan tugas akhir ini.
4. Bapak Yudhi Purwananto, S.Kom., M.Kom., selaku pembimbing II penulis yang telah memberikan pengetahuan dan arah-

an bagi pengerjaan tugas akhir ini serta mengenalkan penulis kepada dunia pekerjaan yang berkaitan dengan ilmu yang penulis pelajari.
5. Seluruh tenaga pengajar dan karyawan Departemen Teknik Informatika ITS yang telah memberikan waktu dan tenaganya dalam mengajar penulis selama ini.
6. Yoshima yang telah membantu dalam memeriksa penulisan laporan tugas akhir ini.
7. Chendrasena, Nurlita, Rizaldi, serta kakak-kakak administrator Laboratorium Pemrograman 2 yang telah menerima penulis dan menjadi keluarga selama penulis berkuliah ITS.
8. Teman-teman angkatan 2016 yang telah memberikan dukungan kepada penulis selama penulis berkuliah di ITS.
9. Alvin, Ferdinand, Dandy, Fadli, dan Vidya yang telah membantu penulis dalam membuat laporan tugas akhir ini.
10. Serta pihak-pihak lain yang tidak dapat disebutkan di sini yang telah membantu penulisan laporan tugas akhir ini.

Penulis sangat menyadari bahwa tugas akhir serta buku laporan ini jauh dari kata sempurna, sehingga penulis memohon maaf apabila terdapat kesalahan serta mengharapkan kritik/saran dari pembaca. Akhir kata, semoga tugas akhir ini mampu memberikan kontribusi serta manfaat yang sebaik-baiknya.

Surabaya, Agustus 2020

Jonathan Rehuel Lewerissa

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE SUMBER

[*Halaman ini sengaja dikosongkan*]

BAB I
PENDAHULUAN

Pada bab ini, akan dijelaskan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi pengerjaan, dan sistematika penulisan tugas akhir.

1.1 Latar Belakang

Penerimaan Peserta Didik Baru (PPDB) merupakan sebuah kegiatan tahunan yang dilaksanakan oleh Kementerian Pendidikan dan Budaya untuk menyeleksi peserta didik yang akan masuk ke institusi pendidikan negeri. Pada kegiatan ini, calon peserta didik akan dipilih dan diurutkan peringkatnya berdasarkan kuota pagu sekolah tujuan calon peserta didik serta kriteria-kriteria yang dimiliki oleh peserta didik, seperti nilai, jarak rumah, serta usia peserta didik. Masalah yang kemudian dihadapi dalam kegiatan ini adalah sering berubahnya kriteria pemeringkatan peringkat calon peserta didik, sehingga membutuhkan perubahan pada kode sumber agar sesuai dengan kriteria yang diminta. Hal ini dirasakan tidak efisien, karena sumber daya yang harus dikeluarkan seringkali tidak berbanding lurus dengan waktu yang ada.

Metode yang saat ini digunakan adalah dengan melakukan pengurutan peringkat secara rekursif untuk mencoba memenuhi pagu sekolah yang diinginkan. Metode ini cukup baik, tetapi belum mampu menangani perubahan parameter dengan cepat. Metode yang ingin dicari dan diimplementasikan melalui pembuatan tugas akhir ini adalah metode pemeringkatan yang efisien yang bersifat general dan dapat menangani perubahan parameter pengurutan tanpa perlu melakukan banyak perubahan pada kode sumber serta dapat diimplementasikan pada sistem yang terdistribusi.

Riset mengenai analisa dan metode pemeringkatan pernah dilakukan oleh beberapa peneliti. Salah satu peneliti yang paling awal

melakukan riset dan analisa adalah Bradley dan Terry pada permasalahan *paired comparison* [2]. Mereka melakukan analisa prediksi terhadap suatu hasil perbandingan data berdasarkan data yang diperoleh sebelumnya. Peneliti lain yang telah melakukan riset adalah Beaudoin dan Swartz di mana mereka melakukan riset terhadap metode pemeringkatan pada studi kasus pertandingan olahraga bola basket yang diadakan oleh *National Collegiate Athletic Association* (NCAA) [1]. Para peneliti tersebut berhasil menghasilkan metode yang mampu melakukan pemeringkatan dengan baik serta bersifat spesifik pada masing-masing studi kasus.

Pembuatan tugas akhir ini diharapkan dapat mengefisienkan operasional pelaksanaan kegiatan PPDB melalui kecepatan dan kemudahan perubahan kriteria pemeringkatan. Pembuatan tugas akhir ini juga diharapkan dapat meningkatkan efisiensi komputasi pada basis data terdistribusi, sehingga diharapkan dapat meningkatkan kinerja pelaksanaan kegiatan PPDB.

1.2 Rumusan Masalah

Permasalahan yang akan diselesaikan pada tugas akhir ini adalah sebagai berikut:

1. Bagaimana merancang metode yang tepat guna menyelesaikan permasalahan *Generalized Multiparameter Ranking* pada sistem terdistribusi?
2. Bagaimana mengimplementasikan solusi pada permasalahan *Generalized Multiparameter Ranking* pada sistem terdistribusi?

1.3 Batasan Masalah

Masalah yang akan diselesaikan memiliki batasan-batasan sebagai berikut:

1. Implementasi dilakukan menggunakan bahasa pemrograman *Python*.
2. Basis data yang digunakan pada tugas akhir ini adalah basis data relasional.
3. Parameter yang digunakan pada permasalahan *Generalized Multiparameter Ranking* adalah parameter yang bersifat primitif (*integer, double*) maupun kompleks (*string, date, timestamp*) serta dapat dilakukan operasi perbandingan.

1.4 Tujuan

Tujuan tugas akhir ini adalah sebagai berikut:

1. Merancang metode yang tepat guna menyelesaikan permasalahan *Generalized Multiparameter Ranking* pada sistem terdistribusi.
2. Mengimplementasikan serta menguji solusi yang telah dirancang pada studi kasus guna menyelesaikan permasalahan *Generalized Multiparameter Ranking* pada sistem terdistribusi.

1.5 Metodologi

Metodologi pengerjaan yang digunakan pada tugas akhir ini memiliki beberapa tahapan. Tahapan-tahapan tersebut yaitu:

1.5.1 Penyusunan Proposal Tugas Akhir

Pada tahapan ini penulis memberikan penjelasan mengenai apa yang penulis akan kerjakan serta mengapa tugas akhir ini dilakukan. Penjelasan juga mencakup solusi yang diusulkan sebagai solusi penyelesaian permasalahan. Penjelasan tersebut dituliskan dalam bentuk proposal tugas akhir.

1.5.2 Studi Literatur

Pada tahapan ini penulis mengumpulkan referensi yang diperlukan guna mendukung pengerjaan tugas akhir. Referensi yang digunakan dapat berupa hasil penelitian yang sudah pernah dilakukan, buku, artikel, atau sumber lain yang bisa dipertanggungjawabkan.

1.5.3 Desain dan Analisis Sistem

Pada tahapan ini penulis melakukan analisis pada solusi yang dirancang, merancang sistem dari solusi yang diusulkan.

1.5.4 Implementasi

Pada tahapan ini penulis melakukan implementasi algoritma yang telah dirancang untuk menyelesaikan permasalahan *Generalized Multiparameter Ranking* pada sistem terdistribusi.

1.5.5 Pengujian dan evaluasi

Pada tahapan ini penulis menguji performa implementasi algoritma yang telah dirancang. Hasil pengujian kemudian dievaluasi untuk mendapatkan hasil yang dapat digunakan sebagai bahan pertimbangan apakah algoritma masih bisa ditingkatkan lagi atau tidak.

1.5.6 Penyusunan buku

Pada tahapan ini penulis menyusun hasil pengerjaan tugas akhir mengikuti format penulisan tugas akhir.

1.6 Sistematika Penulisan

Sistematika laporan tugas akhir yang akan digunakan adalah sebagai berikut:

1. **Bab I - Pendahuluan**. Bab ini berisi penjelasan mengenai konteks dari tugas akhir yang dikerjakan. Penjelasan men-

cakup latar belakang, tujuan, rumusan dan batasan masalah, serta metodologi yang digunakan.

2. **Bab II - Dasar Teori**. Bab ini berisi penjelasan mengenai dasar teori yang digunakan untuk mendukung pembuatan tugas akhir ini.
3. **Bab III - Desain dan Analisis Sistem**. Bab ini berisi penjelasan mengenai rancangan solusi dari sistem yang akan dibangun.
4. **Bab IV - Implementasi**. Bab ini berisi penjelasan implementasi dari rancangan pada bab III. Implementasi dijelaskan serta disajikan dalam bentuk kode sumber serta penjelasannya.
5. **Bab V - Pengujian dan Evaluasi**. Bab ini berisi penjelasan mengenai percobaan yang dilakukan terhadap rancangan solusi serta implementasi pada bab IV.
6. **Bab VI - Kesimpulan dan Saran**. Bab ini berisi penjelasan mengenai kesimpulan yang menjawab rumusan masalah berdasarkan hasil pengujian serta saran untuk pengembangan sistem kedepannya.

[*Halaman ini sengaja dikosongkan*]

BAB II
DASAR TEORI

Pada bab ini akan dijelaskan mengenai dasar-dasar teori yang akan digunakan untuk menyelesaikan permasalahan *Generalized Multiparameter Ranking* pada sistem terdistribusi. Dasar-dasar teori yang digunakan meliputi deskripsi permasalahan secara umum, *order theory*, struktur data *priority queue*, serta algoritma *sorting*.

2.1 Deskripsi Permasalahan Generalized Multiparameter Ranking

Permasalahan *Generalized Multiparameter Ranking* awalnya muncul dari permasalahan pemeringkatan calon peserta didik pada kegiatan penerimaan calon peserta didik pada beberapa sekolah.

Terdapat sebuah kegiatan pemilihan calon-calon peserta didik dalam suatu institusi pendidikan. Institusi pendidikan tersebut membuka beberapa jalur pendaftaran calon peserta didik yang masing-masing memiliki kriteria pemeringkatannya sendiri. Setiap jalur pendaftaran memiliki batas jumlah calon siswa, sehingga jumlah peserta didik masing-masing jalur dibatasi oleh jumlah tersebut.

Dalam melakukan pemilihan sekolah, masing-masing calon peserta didik mampu menentukan beberapa pilihan sekolah, biasanya terdiri dari dua sampai tiga sekolah pilihan. Pilihan-pilihan tersebut kemudian diberi prioritas berdasarkan keinginan calon peserta didik, mulai dari prioritas tertinggi sampai prioritas terendah.

Dalam melakukan pemeringkatan, terdapat kriteria-kriteria perbandingan yang digunakan untuk membandingkan peringkat setiap calon peserta didik. Kriteria-kriteria tersebut harus memiliki kemampuan untuk dibandingkan hasilnya; apakah lebih besar atau lebih kecil. Sebagai contoh, beberapa pelaksanaan PPDB memiliki kriteria-kriteria perbandingan sebagai berikut:

1. Pada kegiatan PPDB SMK Provinsi Jawa Timur, kriteria pemeringkatan disusun berdasarkan urutan nilai ujian nasional, diikuti nilai mata pelajaran Bahasa Indonesia, nilai mata pelajaran Ilmu Pengetahuan Alam, nilai mata pelajaran Matematika, serta nilai mata pelajaran Bahasa Inggris[7].
2. Pada kegiatan PPDB SMA Provinsi Jawa Timur, kriteria pemeringkatan dibagi menjadi dua jenis pemeringkatan yang dilakukan secara bersamaan, yaitu pemeringkatan berdasarkan jarak (jarak, nilai ujian nasional) diikuti dengan pemeringkatan berdasarkan nilai (sama dengan PPDB SMK Provinsi Jawa Timur) pada masing-masing pilihan sekolah[7].
3. Pada kegiatan PPDB SMP dan SD Kabupaten Sidoarjo, pemeringkatan dilakukan berdasarkan skor jarak peserta ke sekolah tujuan diikuti dengan nomor urut pendaftaran[5].
4. Pada kegiatan PPDB SMP Kota Surabaya, pemeringkatan pada jalur zonasi dilakukan berdasarkan skor jarak peserta ke sekolah tujuan diikuti dengan nomor urut pendaftaran, sedangkan pemeringkatan pada jalur prestasi dilakukan berdasarkan nilai ujian nasional, nilai mata pelajaran Bahasa Indonesia, Matematika, dan IPA, diikuti dengan nomor urut pendaftaran[6].

Pemeringkatan dilakukan berdasarkan kriteria-kriteria yang ditentukan dimulai dari pilihan pertama masing-masing calon peserta didik. Jika calon peserta didik tidak mampu menjadi calon peserta didik pada sekolah pilihan pertama, maka calon peserta didik akan mencoba lagi masuk pada sekolah pilihan kedua, ketiga, dst. Jika sampai akhir calon peserta didik tidak mampu masuk sekolah manapun, maka calon peserta didik tersebut dinyatakan tidak lolos.

2.2 Sorting

Algoritma *sorting* merupakan sebuah algoritma yang mampu mengurutkan elemen-elemen pada suatu kumpulan data berdasarkan urutan tertentu. Urutan yang biasanya digunakan sebagai metode pengurutan adalah urutan secara numerik serta leksikografik. Tantangan yang biasanya muncul dari implementasi algoritma ini adalah mengenai kompleksitas waktu eksekusi.

Ada beberapa metode untuk melakukan *sorting*. Terdapat metode sorting secara naif yang biasa dikenal dengan metode *bubble sort*. Metode ini memiliki kompleksitas komputasi sebesar $O(n^2)$, di mana nilai tersebut cukup pelan. Berdasarkan permasalahan tersebut, maka telah dirancang beberapa algoritma lain yang memiliki kompleksitas komputasi sebesar $O(n \log n)$.

Salah satu konsep penting dalam algoritma ini adalah konsep *stable sort*. *Stable sort* merupakan suatu konsep di mana jika ditemukan nilai yang sama pada kumpulan data, maka urutan relatif nilai nilai tersebut pada kumpulan data tidak akan berubah. Contoh dari ini adalah jika terdapat dua siswa dengan hasil perbandingan parameter yang sama, maka urutan relatif dari dua siswa tersebut tidak boleh berubah. Konsep ini penting dalam solusi yang akan dirancang karena calon peserta didik harus diurutkan berdasarkan urutan asli dari kumpulan data, sehingga urutan relatif pada nilai yang sama tidak boleh berubah. Cara untuk mengimplementasikan konsep ini adalah dengan memastikan pada perbandingan bertingkat sebuah nilai pembanding tambahan yang merupakan nilai urut pada kumpulan data.

2.3 Order Theory

Dalam melakukan perbandingan entitas, dalam hal ini perbandingan calon peserta didik baru, maka kita perlu mendefinisikan bagaimana entitas-entitas tersebut dapat dibandingkan.

Order theory merupakan salah satu cabang matematika yang

mempelajari tentang bagaimana membandingkan dan mengurutkan entitas pada suatu kumpulan berdasarkan relasi yang dimiliki suatu entitas terhadap entitas lainnya. Relasi yang digunakan harus mampu diformulasikan secara formal agar dapat menentukan hasil relasi yang tepat. Contoh dari relasi ini adalah relasi kurang dari ($<$) atau relasi mendahului.

Contoh kecil dari teori ini adalah melakukan perbandingan angka pada bilangan asli. Secara intuitif, angka-angka pada bilangan asli dapat diketahui perbandingannya berdasarkan nilai yang dimilikinya, seperti angka 2 pasti lebih kecil dari angka 5.

Pertanyaan yang selanjutnya muncul adalah bagaimana melakukan perbandingan pada data lain seperti *string*, atau pada data siswa seperti pada kasus tugas akhir ini. Perbandingan-perbandingan entitas seperti demikian dapat dilakukan dengan melakukan suatu formulasi relasi yang mampu menentukan hasil perbandingan berdasarkan relasi yang dimiliki.

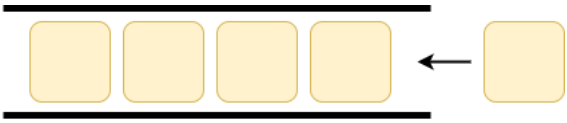
2.4 Struktur Data Queue

Struktur data *Queue* merupakan sebuah struktur data abstrak di mana struktur data ini mampu menyimpan sekumpulan data berdasarkan urutan masuk masing-masing data. Struktur data ini juga dikenal dengan struktur data yang memiliki konsep *First In First Out* (FIFO).

Struktur data ini memiliki operasi-operasi sebagai berikut:

1. ***isEmpty***, yaitu operasi untuk memeriksa apakah struktur data memiliki data didalamnya atau tidak. Kompleksitas dari operasi ini sebesar $O(1)$.
2. ***peekFront***, yaitu operasi untuk melihat data terdepan pada struktur data. Kompleksitas dari operasi ini sebesar $O(1)$.
3. ***enqueue***, yaitu operasi untuk memasukkan data pada posisi terakhir pada struktur data. Kompleksitas dari operasi ini sebesar $O(1)$.

4. ***dequeue***, yaitu operasi untuk mengambil data terdepan pada struktur data. Kompleksitas dari operasi ini sebesar $O(1)$.



Gambar 2.1 Operasi *enqueue* pada struktur data *Queue*

Gambar 2.2 Operasi *dequeue* pada struktur data *Queue*

Cara kerja dari struktur data ini dapat dilihat pada Gambar 2.1-2.2. Pertama data dimasukkan ke dalam struktur data dengan operasi *enqueue*. Data kemudian akan berada di dalam struktur data sampai data berada di posisi terdepan dari struktur data dan dilakukan pemanggilan operasi *dequeue*.

2.5 Struktur Data Priority Queue

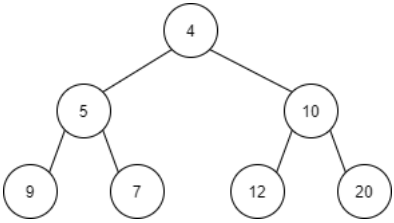
Struktur data *Priority Queue* merupakan sebuah struktur data abstrak di mana struktur data ini mampu mengambil data dengan prioritas tertinggi berdasarkan urutan atau prioritas tertentu. Pengambilan data berdasarkan prioritas pada struktur data ini memiliki kompleksitas waktu logaritmik $O(\log n)$.

Struktur data *Priority Queue* merupakan pengembangan lanjutan dari struktur data *Queue*, yaitu struktur data abstrak yang telah dijelaskan pada subbab 2.4.

Sebelum membahas cara kerja struktur data *Priority Queue*, maka akan dibahas terlebih dahulu mengenai operasi-operasi yang dapat dilakukan terhadap struktur data ini, yaitu sebagai berikut:

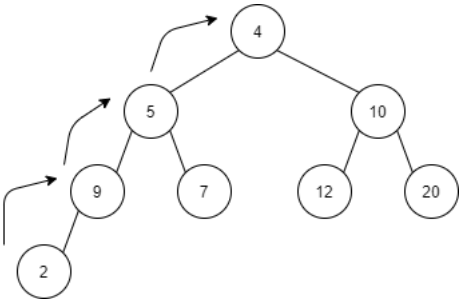
1. ***isEmpty***, yaitu operasi untuk memeriksa apakah struktur data ini memiliki data didalamnya atau tidak.
2. ***enqueueWithPriority***, yaitu operasi untuk memasukkan suatu data ke dalam struktur data ini. Data yang dimasukkan harus mampu ditentukan prioritasnya. Kompleksitas dari operasi ini sebesar $O(\log n)$.
3. ***peekHighestPriorityElement***, yaitu melihat data dengan prioritas tertinggi didalam struktur data. Kompleksitas dari operasi ini sebesar $O(1)$.
4. ***getHighestPriority***, yaitu operasi untuk mengambil data dengan prioritas tertinggi dari struktur data ini. Kompleksitas dari operasi ini sebesar $O(\log n)$

Biasanya, struktur data ini diimplementasikan dengan struktur data *Heap*, yaitu struktur data berbentuk *tree* di mana element teratas (*root*) pada *tree* merupakan elemen dengan prioritas tertinggi (nilai maksimum atau minimum) serta seluruh *node* pada *tree* memiliki prioritas yang lebih tinggi jika dibandingkan dengan *children node* [3]. Cara kerja struktur data ini dijelaskan pada Gambar 2.3-2.5.

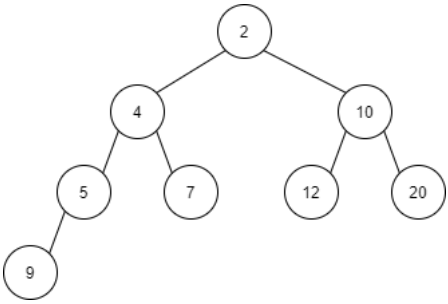


Gambar 2.3 Ilustrasi kondisi *Heap* awal

Semisal dimiliki struktur data *Heap* dengan komposisi data seperti ditunjukkan pada Gambar 2.3, kemudian akan dilakukan operasi *insertWithPriority* dengan angka 2. Pertama, angka tersebut dimasukkan kedalam *tree* pada posisi terakhir pada tingkat paling akhir. Kemudian, dilakukan operasi berulang seperti pada Gam-



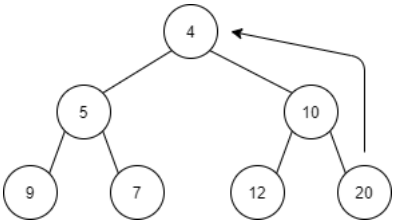
Gambar 2.4 Operasi *insertWithPriority*



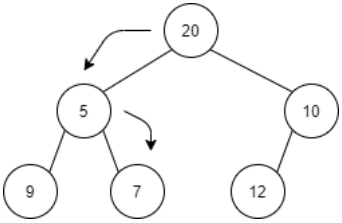
Gambar 2.5 Kondisi *heap* akhir

bar 2.4 untuk membandingkan nilai *node* dengan *parent node*. Jika prioritas pada *node* lebih tinggi dibanding dengan *parent node*, maka nilai kedua *node* tersebut ditukar. Operasi ini dilakukan secara berulang sampai kondisi akhir tercapai, yaitu apabila *node* mencapai *root* atau perbandingan prioritas sudah tepat, seperti ditunjukkan pada Gambar 2.5.

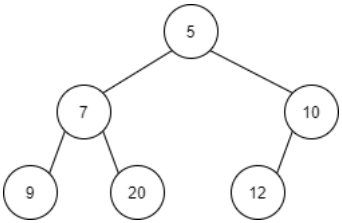
Operasi *getHighestPriorityElement* dilakukan dengan cara sebagai berikut. Semisal dimiliki struktur data Heap seperti pada Gambar 2.3, kemudian kita ingin mengambil elemen dengan prioritas tertinggi, dalam hal ini angka 4. Langkah pertama adalah dengan memasukkan *node* paling akhir ke *root node*, seperti ditunjukkan pada Gambar 2.6. Jika pada *root node* dilakukan perbandingan ke-



Gambar 2.6 Operasi awal *getHighestPriorityElement*



Gambar 2.7 Iterasi berulang untuk memeriksa kondisi optimal



Gambar 2.8 Kondisi *Heap* akhir setelah operasi *getHighestPriorityElement*

pada kedua *children node* yang menghasilkan nilai prioritas *root node* yang lebih rendah daripada kedua *children node*, maka dilakukan penukaran nilai dengan *child node* yang perbedaan prioritasnya paling jauh. Iterasi ini dilakukan berulang seperti ditunjukkan pada Gambar 2.7 sampai kondisi dari struktur data *Heap* sudah optimal, seperti ditunjukkan pada Gambar 2.8.

2.6 Sistem Basis Data Terdistribusi

Sistem basis data terdistribusi merupakan kumpulan basis data yang saling terhubung dalam suatu sistem terdistribusi [8]. Sistem ini merupakan pengembangan dari sistem basis data, yaitu sistem untuk melakukan penyimpanan data secara terstruktur pada suatu sistem komputer. Untuk melakukan interaksi dengan sistem basis data terdistribusi, maka dibutuhkan sistem manajemen khusus, yaitu sistem manajemen basis data terdistribusi. Sistem manajemen basis data terdistribusi merupakan sebuah perangkat lunak yang memiliki kemampuan untuk mengolah data pada basis data terdistribusi serta menghasilkan data berdasarkan basis data yang terhubung.

Sistem ini digunakan untuk menjawab beberapa permasalahan seperti jumlah data yang sangat besar, skalabilitas dan performa basis data, serta distribusi data kepada pengguna. Kendala utama dari sistem basis data ini adalah bagaimana merepresentasikan data ke dalam bentuk yang mampu didistribusikan dengan baik. Terdapat dua metode umum yang digunakan dalam menyimpan data, yaitu replikasi data dan partisi data. Replikasi data merupakan metode untuk melakukan replikasi data ke beberapa basis data sekaligus, sedangkan partisi data merupakan metode untuk membagi data ke beberapa basis data sekaligus.

[*Halaman ini sengaja dikosongkan*]

BAB III
DESAIN DAN ANALISIS

Pada bab ini akan dijelaskan desain dan analisis algoritma yang akan digunakan untuk mencapai tujuan tugas akhir ini, yaitu menyelesaikan permasalahan *Generalized Multiparameter Ranking* pada sistem terdistribusi.

3.1 Deskripsi Umum Desain Sistem

Pada subbab ini akan dijelaskan mengenai gambaran secara umum dari sistem yang akan dirancang. Gambaran umum yang dimaksud adalah alur kerja sistem secara umum mulai dari masukan, pemrosesan data, pemeringkatan data, sampai luaran yang dihasilkan.

Pertama, sistem akan memilah berdasarkan jalur pendaftaran yang akan diurutkan. Setiap jalur akan memiliki kriteria pemeringkatannya serta kuota calon peserta didiknya sendiri, sehingga pemeringkatan harus dilakukan secara terpisah berdasarkan masing-masing jalur.

Setelah melakukan pemilahan jalur pendaftaran, maka dilakukan pengambilan dan pemodelan data calon peserta didik berdasarkan masing-masing jalur pendaftaran. Setelah dilakukan pengambilan data, maka dilakukan pemeringkatan sambil melakukan pemenuhan pagu sekolah berdasarkan jumlah pagu yang tersedia. Pemeringkatan dilakukan berdasarkan pemodelan perbandingan calon peserta didik.

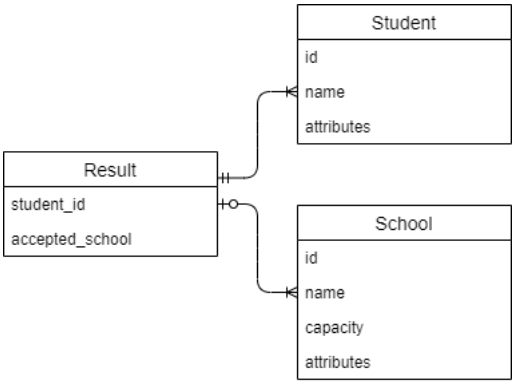
Luaran dari sistem adalah sejumlah data calon peserta didik yang lolos pada sekolah tertentu berdasarkan hasil pemeringkatan serta data calon peserta didik yang tidak lolos proses pemeringkatan.

3.2 Analisis Sistem

Sistem yang akan dibuat adalah sistem yang menggunakan metode pemeringkatan berbasis prioritas. Dalam hal ini prioritas yang digunakan adalah prioritas pemilihan tujuan sekolah digabungkan dengan peringkat calon peserta didik. Perbandingan prioritas calon peserta didik dilakukan menggunakan metode perbandingan bertingkat berdasarkan konfigurasi pemeringkatan agar mampu memfasilitasi perbandingan yang bersifat umum. Sedangkan pemeringkatan berbasis prioritas dilakukan menggunakan struktur data *priority queue* yang mampu melakukan pemeringkatan prioritas data dengan kompleksitas waktu $O(n \log n)$.

3.3 Perancangan Struktur Basis Data

Pada subbab ini akan dijelaskan mengenai perancangan basis data yang digunakan pada tugas akhir ini. Terdapat tiga kumpulan data utama yang digunakan pada tugas akhir ini, yaitu data *input* calon peserta didik, data sekolah, serta data *output* penerimaan. Perancangan ini dijabarkan pada Gambar 3.1



Gambar 3.1 Skema *Data Model*

3.4 Perancangan Fungsi Utama

Subbab ini menjelaskan mengenai perancangan fungsi utama, di mana seluruh proses komputasi dieksekusi. Proses pada fungsi ini mencakup proses pengambilan dan pemodelan data calon siswa dan sekolah dari basis data, proses pengambilan dan pemodelan peraturan perbandingan bertingkat, proses pemeringkatan, serta proses penulisan data ke basis data. Perancangan tersebut dijabarkan pada Gambar 3.2.

```
   MAIN
 1 rules = getRules()
 2 results = []
 3 schools = getSchoolsFromDatabase()
 4
 5 foreach database
 6     students = getStudentsFromDatabase()
 7
 8     for i = 0 to students.length - 1
 9         students[i] =
               generateStudentObject(students[i],
               rules)
10
11     results = RANK(schools, students)
12     append results to array
13
14 writeToDatabase(results)
```

Gambar 3.2 Perancangan fungsi *main*

3.5 Perancangan Model Peraturan Pemeringkatan

Pemeringkatan dilakukan secara khusus pada masing-masing jalur atau kategori, sehingga setiap jalur atau kategori pemeringkatan memiliki peraturannya tersendiri serta perancangan konfigurasi dilakukan secara khusus pada masing-masing jalur atau kategori pemeringkatan.

Kemudian, perancangan peraturan pemeringkatan dibagi menjadi dua jenis variabel, yaitu variabel global serta variabel yang spesifik pada pilihan. Variabel global adalah variabel yang digunakan pada seluruh pilihan, sedangkan variabel spesifik adalah variabel yang hanya digunakan secara spesifik pada satu pilihan. Gambar 3.3 menjelaskan mengenai perancangan peraturan pemeringkatan pada suatu jalur atau kategori.

```
   RULE-GENERATOR
 1 specific rule = getSpecificRule()
 2 global rule = getGlobalRule()
 3
 4 rules = []
 5
 6 for i in specific rule.length
 7     combine specific rule[i] with global rule
 8     append result to rules
 9
10 return rules
```

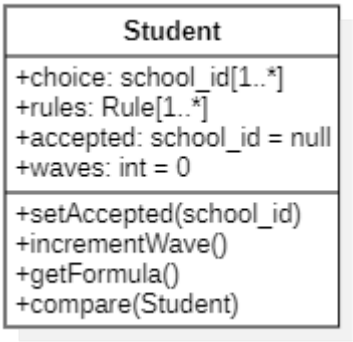
Gambar 3.3 Perancangan fungsi peraturan pemeringkatan

3.6 Perancangan Model Calon Peserta Didik

Sebelum masuk ke model pengurutan calon peserta didik, maka perlu dibahas terlebih dahulu mengenai pemodelan calon peserta didik beserta dengan pemodelan kriteria perbandingan.

Gambar 3.4 merupakan diagram kelas yang menjelaskan pemodelan calon peserta didik. Calon peserta didik memiliki atribut sekolah yang dipilih, serangkaian kriteria perbandingan beserta dengan nilainya, berapa kali calon peserta didik telah mencoba, serta sekolah yang diterima.

Setelah melakukan pemodelan peserta didik, maka akan dibahas pemodelan perbandingan peserta didik. Semisal kita memiliki kriteria-kriteria sebagai berikut:



Gambar 3.4 Diagram kelas calon peserta didik

1. Jarak rumah ke sekolah. Semakin dekat ke sekolah, semakin baik.
2. Akumulasi nilai rapor. Semakin tinggi, semakin baik.
3. Waktu pendaftaran. Semakin cepat, semakin baik.

Kriteria 1 dan 3 memiliki perbandingan yang bersifat *lower-better* (<), sedangkan kriteria 2 memiliki perbandingan yang bersifat *higher-better* (>). Kemudian perbandingan dilakukan secara berurutan mulai dari kriteria pertama sampai akhir, sehingga jika pada kriteria pertama hasil perbandingannya sama (=), maka perbandingan dilanjutkan pada kriteria kedua. Jika masih ditemukan kesamaan, maka perbandingan dilanjutkan pada kriteria ketiga dan seterusnya.

Setelah diturunkan dari contoh pada bagian sebelumnya, maka didapatkan algoritma pada Gambar 3.5. Algoritma tersebut merupakan algoritma untuk mengetahui calon peserta didik yang memiliki kriteria lebih rendah jika dibandingkan peserta didik lain. Perbandingan kriteria dilakukan secara bertingkat mulai dari kriteria pertama sampai kriteria terakhir. Agar perbandingan yang nantinya dilakukan bersifat stabil, maka perbandingan pada nilai terakhir haruslah merupakan nilai yang unik serta memiliki pengurutan yang

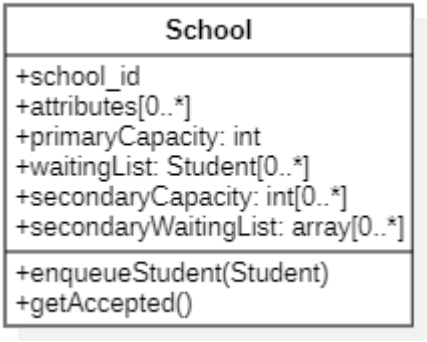
```
   COMPARE-LESS-THAN(a, b, rules)
1  for i = 0 to rules.length
2      Rule = rules[i]
3      if Rule(a) != Rule(b)
4          if Rule.type == descending
5              return Rule(a) > Rule(b)
6          else return Rule(a) < Rule(b)
```

Gambar 3.5 Desain fungsi perbandingan (<) calon peserta didik

menaik. Contoh dari kriteria ini adalah atribut nomor urutan calon peserta didik pada basis data atau waktu pemasukan data.

Karena pemeringkatan nantinya dilakukan secara *ascending* serta menggunakan *min-heap*, maka kita perlu mengubah perbandingan agar hasil yang lebih kecil mendapatkan prioritas yang lebih tinggi. Cara ini dilakukan dengan mengubah perbandingan *higher-better* menjadi *lower-better*.

3.7 Perancangan Model Sekolah



Gambar 3.6 Diagram kelas model sekolah

Gambar 3.6 merupakan diagram kelas hasil pemodelan seko-

lah. Atribut yang dimiliki berupa atribut identifikasi (berupa nomor identifikasi), kapasitas, serta antrian pendaftaran sekolah.

Kapasitas sendiri dibagi menjadi dua jenis kapasitas, yaitu kapasitas primer dan kapasitas sekunder. Kapasitas primer merupakan kapasitas utama dari suatu sekolah, sedangkan kapasitas sekunder merupakan kapasitas batasan bagi beberapa jenis jalur pendaftaran. Contoh dari kapasitas sekunder adalah pembatasan kapasitas siswa yang mendaftar dari luar kota menuju suatu sekolah, yang kemudian dibatasi jumlahnya menjadi 5% saja.

Gambar 3.7 merupakan rancangan fungsi antrian pendaftaran sekolah. Dalam melakukan antrian pendaftaran sekolah, digunakan struktur data *Priority Queue* dalam melakukan pemeringkatan calon peserta didik. Struktur data ini digunakan agar posisi calon peserta didik dengan peringkat prioritas terendah mampu diurutkan dan dilacak posisinya pada suatu waktu, sehingga mampu dibandingkan dengan siswa lain yang ingin masuk ke sekolah tersebut.

Terdapat kasus khusus jika terdapat calon peserta didik yang mendaftar pada jalur khusus, seperti pada pembatasan kuota pada bagian sebelumnya. Calon peserta didik pertama akan dibandingkan seperti biasa, kemudian dilakukan pemeriksaan tambahan pada jumlah kuota jalur khusus. Alur kemudian dilanjutkan seperti yang telah dijelaskan.

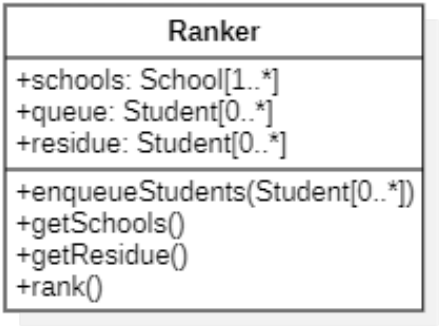
3.8 Perancangan Fungsi Pemeringkatan

Subbab ini merupakan penjelasan secara umum mengenai rancangan fungsi pemeringkatan yang telah dirancang.

Pemeringkatan dilakukan secara satu persatu pada setiap siswa sampai setiap siswa diterima atau siswa telah mencoba masuk pada seluruh pilihan sekolah di mana hal ini ditandai dengan *Queue* yang kosong. Jika pada pemeringkatan terdapat beberapa jalur sekaligus, maka perlu dipastikan urutan peringkat setiap siswa selalu dijaga untuk memastikan urutan masuk dan pemeringkatan yang

```
   ENQUEUE-STUDENT-TO-SCHOOL(student, school)
 1 waiting list = BUILD-PRIORITY-QUEUE()
 2 if school capacity is less than waiting
       list.length
 3     if special lane, then check special capacity
           first
 4     waiting list.enqueue(student)
 5 else
 6     minimum priority student = minimum from
           waiting list
 7     if student has more priority than minimum
           priority student
 8         remove minimum priority student from
               waiting list
 9         enqueue student to waiting list
10         if special lane
11             then enqueue to special lane
12         return minimum priority student
13     else
14         return student
```

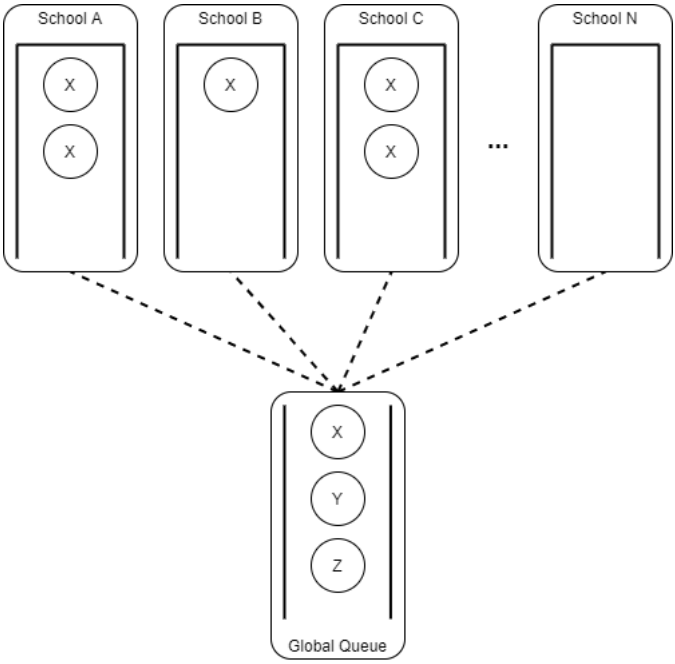
Gambar 3.7 Desain fungsi antrian calon peserta didik



Gambar 3.8 Diagram kelas pemeringkat

benar. Untuk mendapatkan calon peserta didik dengan peringkat terbaik pada suatu waktu, digunakan struktur data *Priority Queue* dengan metode perbandingan peringkat yang telah dijelaskan pada subbab 3.6.

Gambaran fungsi secara umum dapat dilihat pada Gambar 3.9, sedangkan perancangan fungsi dapat dilihat pada Gambar 3.10.



Gambar 3.9 Gambaran Umum Desain Sistem Pemeringkatan

3.9 Perancangan Fungsi Penulisan Hasil

Subbab ini menjelaskan mengenai perancangan fungsi penulisan hasil pemeringkatan ke dalam basis data. Penulisan hasil pemeringkatan dilakukan untuk setiap calon peserta didik di setiap sekolah. Penulisan hasil pemeringkatan juga dilakukan untuk setiap

```
   RANK(students, schools)
 1 queue = BUILD-QUEUE(students)
 2 repeat
 3     current student = queue.dequeue()
 4     current school = current school choice from
           current student
 5
 6     popped student =
           ENQUEUE-STUDENT-TO-SCHOOL(current
           student, current school)
 7
 8     if any popped student
 9         queue.enqueue(popped student)
10 until queue is empty
```

Gambar 3.10 Desain fungsi pemeringkatan

calon peserta didik yang tidak diterima. Perancangan fungsi ini dapat dilihat pada Gambar 3.11.

```
   WRITE-RESULT(schools, residue)
 1 results = []
 2
 3 for i = 0 to schools.length
 4     school = schools[i]
 5     for j = 0 to school waiting list.length
 6         student = waiting list[j]
 7         append student to result
 8
 9 for i = 0 to residue.length
10     student = residue[i]
11     append student to result
```

Gambar 3.11 Desain fungsi penulisan hasil

3.10 Perancangan Fungsi Pemeriksa Kebenaran Hasil

Subbbab ini menjelaskan mengenai perancangan fungsi pemeriksa hasil pemeringkatan. Hasil pemeringkatan diperiksa dengan membandingkan data pada tabel asli dengan data pada tabel hasil fungsi pemeringkatan. Adapun atribut penting yang perlu dibandingkan adalah sekolah di mana calon peserta didik diterima serta pada pilihan dan urutan berapakah calon peserta didik diterima. Perancangan fungsi ini dapat dilihat pada Gambar 3.12.

```
  CHECK-RESULT(original_table, result_table)
1 for i = 0 to original_table.length
2     student = original_table[i]
3     if student in result_table
4         then assert equality
5     else
6         record inequality
```

Gambar 3.12 Desain fungsi pemeriksa kebenaran hasil

[*Halaman ini sengaja dikosongkan*]

BAB IV
IMPLEMENTASI

4.1 Lingkungan Implementasi

Pada pengerjaan tugas akhir ini, terdapat dua lingkungan implementasi yang digunakan sebagai lingkungan pengembangan, yaitu lingkungan implementasi basis data relasional serta lingkungan implementasi program utama. Desain yang telah dijabarkan pada bab 3 akan diimplementasikan pada lingkungan program utama. Lingkungan implementasi yang digunakan adalah sebagai berikut.

1. Perangkat Keras (*Host*)
    (a) Processor Intel® Core™ i5-6500 CPU @ 3.20GHz (4 CPUs), 3.2GHz
    (b) Random Access Memory (RAM) 8192MB
    (c) VirtualBox 6.1
    (d) Vagrant 2.2.9
2. Perangkat Lunak (*Virtual Machine*)
    (a) Basis Data
        i. Sistem Operasi Ubuntu 18.04.4 LTS (Bionic Beaver)
        ii. Alokasi RAM 1024MB
        iii. MySQL 5.7.24
        iv. Klien SQL DBeaver 7.1.0
    (b) Program Utama
        i. Sistem Operasi Ubuntu 20.04 LTS (Focal Fossa)
        ii. Alokasi RAM 1024MB
        iii. Bahasa pemrograman Python 3.8.2
        iv. Memory-profiler 0.57.0
        v. IDE PyCharm Professional 2020.1

4.2 Implementasi Program Utama

Pada subbab ini akan dijelaskan mengenai implementasi program utama yang telah dirancang guna menyelesaikan permasalahan *Generalized Multiparamater Ranking*.

4.2.1 Pengunaan Library

Pada bagian ini dijabarkan beberapa *library* yang digunakan pada implementasi. Terdapat tiga library eksternal yang digunakan, yaitu *connector* MySQL, PyPika (*SQL query builder*), serta *heapq*. *Connector MySQL* merupakan *library* yang digunakan untuk menghubungkan program dengan basis data MySQL. PyPika merupakan *library* yang digunakan untuk membantu konstruksi *query* pada basis data [4]. *Heapq* merupakan *library* standar pada bahasa pemrograman Python yang digunakan sebagai implementasi struktur data *heap*. Penggunaannya dapat dilihat pada Kode Sumber 4.1

```
1 import MySQLdb
2 from pypika import Table, MySQLQuery, Parameter,
      Field
3 import heapq
```

Kode Sumber 4.1 Library pada Program Utama

4.2.2 Implementasi Konfigurasi

Pada bagian ini dijabarkan konfigurasi yang digunakan pada program utama. Konfigurasi mencakup konfigurasi koneksi basis data, konfigurasi nama dan atribut pada tabel, serta konfigurasi peraturan pengurutan. Konfigurasi ditulis menggunakan format *JavaScript Object Notation* (JSON). Implementasi konfigurasi dapat dilihat pada Kode Sumber 4.2

```
{
  "input_database": {
    "host": "hostname",
    "database": "database",
    "user": "username",
    "password": "password",
    "port": 3306
  },
  "output_database": {
    "host": "hostname",
    "database": "database",
    "user": "username",
    "password": "password",
    "port": 3306
  },
  "student": {
    "table_name": "input_sma_umum",
    "student_id": "input_uasbn",
    "attributes": [ "input_nama_siswa" ],
    "filter": []
  },
  "school": {
    "table_name": "sekolah",
    "school_id": "id_sekolah",
    "attributes": [ "nama_sekolah" ],
    "primary_capacity": [
      {
        "capacity": "pagu_jarak",
        "label": "zonasi",
        "secondary_capacity": []
      },
      {
        "capacity": "pagu_nilai",
        "label": "prestasi",
        "secondary_capacity": []
      }
    ]
  },
```

Kode Sumber 4.2 Dokumen konfigurasi pada program utama

```
39   "choice_rules": [
40     {
41       "school_choice": "input_pilihan1",
42       "lane_label": "input_jalur",
43       "rule": [
44         {
45           "field": "first_criteria",
46           "descending": false
47         }
48       ]
49     }
50   ],
51   "global_rules": [
52     {
53       "field": "input_timestamp",
54       "descending": false
55     }
56   ],
57   "output": {
58     "table_name": "output_table",
59     "accepted": "output_accepted",
60     "choice": "output_choice",
61     "rank": "output_rank",
62     "score": "output_score",
63     "write_residue": true,
64     "truncate": true,
65     "input_column": [ "input_field" ],
66     "output_column": [ "output_field" ]
67   }
68 }
```

Kode Sumber 4.2 Dokumen konfigurasi pada program utama (sambungan)

4.2.3 Implementasi Fungsi Utama

Pada bagian ini dijabarkan implementasi fungsi utama yang telah dijelaskan pada subbab 3.4. Implementasi dapat dilihat pada Kode Sumber 4.3

```
def ranker(config_path):
  with open(config_path) as rf:
    config = json.load(rf)
  if not isinstance(config['input_database'],
      list):
    config['input_database'] =
        [config['input_database']]
  main_ranker =
      PriorityQueueRanker(config['school'])
  rule_generator =
      RuleGenerator(config['choice_rules'],
      config['global_rules'])
  rules = rule_generator.generate_rules()

  for i, database in
      enumerate(config['input_database']):
    connection =
        repository.create_connection(database)
    if i == 0:
      schools =
          repository.get_schools(connection,
          config['school'])
      mapped_schools =
          map_schools(config['school'], schools)
      main_ranker.add_schools(mapped_schools)
    students =
        repository.get_students(connection,
        config['student'],
        config['choice_rules'],
        config['global_rules'])

    mapped_students =
        map_students(config['choice_rules'],
        rules, students)
    main_ranker.add_students(mapped_students)

    connection.close()
    main_ranker.rank()
```

Kode Sumber 4.3 Fungsi Utama

```
23
24   school_result = main_ranker.get_schools()
25   residue_students = main_ranker.get_residue() if
         config['output']['write_residue'] else []
26
27   connection = repository.create_connection(
         config['output_database'])
28   if config['output']['truncate']:
29     repository.clean_table(connection,
           config['output']['table_name'])
30   connection.close()
31
32   workers = repository.write_students(
         config['output_database'],
         config['output'], school_result,
         residue_students)
33   for x in workers:
34     x.start()
35   for x in workers:
36     x.join()
37
38 def map_students(config, rules, students):
39   return [StudentWaveWrapper(Student(x, rules,
         config)) for x in students]
40
41 def map_schools(school_config, schools):
42   return {school[school_config['school_id']]:
         School(school, school_config) for school in
         schools}
```

Kode Sumber 4.3 Fungsi Utama (sambungan)

4.2.4 Implementasi Fungsi Pengambilan Data

Pada bagian ini dijabarkan implementasi fungsi pengambilan seluruh data yang dibutuhkan dari basis data menuju program utama. Implementasi dapat dilihat pada Kode Sumber 4.4-4.6.

```
1 def create_connection(configuration: Dict):
2   return MySQLdb.connect(**configuration)
```

Kode Sumber 4.4 Fungsi Pembuatan Koneksi kepada Basis Data

```
1 def get_schools(connection, configuration: Dict):
2   table = Table(configuration['table_name'])
3   query = MySQLQuery.from_(table).select(
        configuration['school_id'],
        *configuration['attributes'])
4   query = query.select(*[x['capacity'] for x in
        configuration['primary_capacity']])
5   for x in configuration['primary_capacity']:
6     if 'secondary_capacity' in x and
          x['secondary_capacity']:
7       query = query.select(*[y['capacity'] for y
            in x['secondary_capacity']])
8
9   cursor = connection.cursor(
        MySQLdb.cursors.DictCursor)
10  cursor.execute(query.get_sql())
11  schools = list(cursor)
12  cursor.close()
13  return schools
```

Kode Sumber 4.5 Fungsi Pengambilan Data Sekolah

```
 1 def get_students(connection,
       student_configuration, choice_rules,
       global_rules):
 2   rules_column = []
 3   for choice in choice_rules:
 4     rules_column.extend([
           choice['school_choice'],
           choice['lane_label'] ])
 5     for rule in choice['rule']:
 6       rules_column.append(rule['field'])
 7   for rule in global_rules:
 8     rules_column.append(rule['field'])
 9
10   table =
         Table(student_configuration['table_name'])
11   args = [student_configuration['student_id'],
         *student_configuration['attributes'],
         *rules_column]
12   query = MySQLQuery.from_(table).select(*args)
13   if 'filter' in student_configuration and
         student_configuration['filter']:
14     for filter in student_configuration['filter']:
15       query = query.select(filter['field'])
16       if len(filter['values']) > 1:
17         query = query.where(Field(
               filter['field']).isin(
               filter['values']))
18       else:
19         query = query.where(Field(
               filter['field']) ==
               filter['values'][0])
20
21   cursor = connection.cursor(
         MySQLdb.cursors.DictCursor)
22   cursor.execute(query.get_sql())
23   students = list(cursor)
24   cursor.close()
25   return students
```

Kode Sumber 4.6 Fungsi Pengambilan Data Calon Peserta Didik

4.2.5 Implementasi Pemodelan dan Perbandingan Calon Peserta Didik

Pada bagian ini dijabarkan implementasi pemodelan dan perbandingan calon peserta didik yang telah dijelaskan pada subbab 3.6. Implementasi dapat dilihat pada Kode Sumber 4.7-4.8.

```
class Student:
  __slots__ = '__dict__'

  def __init__(self, fields, rules, choices):
    self.choices = []
    self.label = []
    self.accepted = None
    self.wave = 0
    self.student_rule = []
    self.attributes = fields
    for choice in choices:
      self.choices.append(self.attributes[
          choice['school_choice']])
      self.label.append(self.attributes[
          choice['lane_label']])
    for wave in rules:
      a = [(self.attributes[rule[0]], rule[1])
          for rule in wave]
      self.student_rule.append(tuple(a))

  def set_accepted(self, accepted):
    self.accepted = accepted
    return self.accepted

  def increment_wave(self):
    self.wave = self.wave + 1
    return self.wave

  def get_formula(self):
    return "|".join([str(x[0]) for x in
        self.student_rule[self.wave]]) if
        self.wave >= 0 else ""
```

Kode Sumber 4.7 Implementasi *Class Student*

```
 1  def __eq__(self, other):
 2     current_rules = self.student_rule[self.wave]
 3     other_rules = other.student_rule[other.wave]
 4
 5     for i in range(len(current_rules)):
 6       if current_rules[i][0] != other_rules[i][0]:
 7         return False
 8     return True
 9
10   def __lt__(self, other):
11     current_rules = self.student_rule[self.wave]
12     other_rules = other.student_rule[other.wave]
13     for i in range(len(current_rules)):
14       if current_rules[i][0] == other_rules[i][0]:
15         continue
16       else:
17         if current_rules[i][1]:
18           return current_rules[i][0] <
                 other_rules[i][0]
19         else:
20           return current_rules[i][0] >
                 other_rules[i][0]
21
22   def __le__(self, other):
23     current_rules = self.student_rule[self.wave]
24     other_rules = other.student_rule[other.wave]
25     for i in range(len(current_rules)):
26       if current_rules[i][0] == other_rules[i][0]:
27         continue
28       else:
29         if current_rules[i][1]:
30           return current_rules[i][0] <=
                 other_rules[i][0]
31         else:
32           return current_rules[i][0] >=
                 other_rules[i][0]
```

Kode Sumber 4.8 Implementasi perbandingan bertingkat pada *Class Student*

```
33   def __gt__(self, other):
34     current_rules = self.student_rule[self.wave]
35     other_rules = other.student_rule[other.wave]
36     for i in range(len(current_rules)):
37       if current_rules[i][0] == other_rules[i][0]:
38         continue
39       else:
40         # check if comparator behavior is
               ascending or descending
41         if current_rules[i][1]:
42           return current_rules[i][0] >
                 other_rules[i][0]
43         else:
44           return current_rules[i][0] <
                 other_rules[i][0]
45
46   def __ge__(self, other):
47     current_rules = self.student_rule[self.wave]
48     other_rules = other.student_rule[other.wave]
49     for i in range(len(current_rules)):
50       if current_rules[i][0] == other_rules[i][0]:
51         continue
52       else:
53         # check if comparator behavior is
               ascending or descending
54         if current_rules[i][1]:
55           return current_rules[i][0] >=
                 other_rules[i][0]
56         else:
57           return current_rules[i][0] <=
                 other_rules[i][0]
```

Kode Sumber 4.8 Implementasi perbandingan bertingkat pada *Class Student* (sambungan)

4.2.6 Implementasi Pemodelan Sekolah

Pada bagian ini dijabarkan implementasi pemodelan sekolah serta metode pemenuhan pagu pada sekolah yang telah dijelaskan pada subbab 3.7. Implementasi dapat dilihat pada Kode Sumber 4.9

```
class School:
  def __init__(self, fields: Dict, school_config):
    for key in fields:
      self.__setattr__(key, fields[key])

    self.school_config = school_config
    self.multilane_flag = 0

    for x in school_config['primary_capacity']:
      if len(x['secondary_capacity']):
        self.multilane_flag = 1

    self.parent_label = {}
    self.waiting_list = {}
    self.capacity = {}

    for x in school_config['primary_capacity']:
      self.parent_label[x['label']] = x['label']
      self.waiting_list[x['label']] = []
      self.capacity[x['label']] =
          self.__getattribute__(x['capacity'])
      for y in x['secondary_capacity']:
        self.parent_label[y['label']] = x['label']
        self.waiting_list[y['label']] = []
        self.capacity[y['label']] =
            self.__getattribute__(y['capacity'])

  def check_lane(self, student: Student):
    lane = student.label[student.wave]
    return lane
```

Kode Sumber 4.9 Implementasi *Class School*

```
29   def swap(self, lane, student):
30     min_check = self.waiting_list[lane][0]
31     student.set_accepted(self.__getattribute__(
           self.school_config['school_id']))
32     swapped = heapq.heapreplace(
           self.waiting_list[lane], student)
33     assert swapped == min_check
34     swapped.set_accepted(-1)
35     return swapped
36
37   def enqueue_student(self, student: Student) ->
         Optional[Student]:
38     student_lane = self.check_lane(student)
39     student_parent_lane =
           self.parent_label[student_lane]
40
41     if self.capacity[student_parent_lane] <= 0:
42       return student
43     if
           len(self.waiting_list[student_parent_lane])
           < self.capacity[student_parent_lane]:
44       if self.multilane_flag and student_lane !=
             student_parent_lane:  # jalur khusus
45         if len(self.waiting_list[student_lane]) <
               self.capacity[student_lane]:
46           student.set_accepted(
                 self.__getattribute__(
                 self.school_config['school_id']))
47           heapq.heappush(
                 self.waiting_list[student_lane],
                 student)
48           heapq.heappush( self.waiting_list[
                 student_parent_lane], student)
49           return
50         else:
51           return student
```

Kode Sumber 4.9 Implementasi *Class School* (sambungan)

```
52        else:
53          student.set_accepted(
                self.__getattribute__(
                self.school_config['school_id']))
54          heapq.heappush( self.waiting_list[
                student_parent_lane], student)
55          return
56      else:
57        min_check =
              self.waiting_list[student_parent_lane][0]
58        if min_check < student:
59          min_lane = self.check_lane(min_check)
60          min_parent_lane =
                self.parent_label[min_lane]
61
62          if not self.multilane_flag or (min_lane
                == min_parent_lane and student_lane
                == student_parent_lane):
63            swapped =
                  self.swap(student_parent_lane,
                  student)
64            return swapped
65          else:
66            if min_lane != min_parent_lane and
                  student_lane == student_parent_lane:
67              swapped =
                    self.swap(student_parent_lane,
                    student)
68              t = heapq.heappop(
                    self.waiting_list[min_lane])
69              assert swapped == t
70              return swapped
```

Kode Sumber 4.9 Implementasi *Class School* (sambungan)

```
71            elif min_lane == min_parent_lane and
                  student_lane != student_parent_lane:
72              if len(self.waiting_list[
                    student_lane]) + 1 <=
                    self.capacity[student_lane]:
73                heapq.heappush( self.waiting_list[
                      student_lane], student)
74                swapped =
                      self.swap(student_parent_lane,
                      student)
75                return swapped
76              else:
77                return student
78            else:
79              if student_lane == min_lane or
                    len(self.waiting_list[
                    student_lane]) + 1 <=
                    self.capacity[student_lane]:
80                heapq.heappush( self.waiting_list[
                      student_lane], student)
81                t = heapq.heappop(
                      self.waiting_list[ min_lane])
82                swapped =
                      self.swap(student_parent_lane,
                      student)
83                assert swapped == t
84                return swapped
85              else:
86                return student
87        else:
88          student.set_accepted(-1)
89          return student
```

Kode Sumber 4.9 Implementasi *Class School* (sambungan)

4.2.7 Implementasi Fungsi Pemeringkatan

Pada bagian ini dijabarkan implementasi dari fungsi pemeringkatan siswa ke dalam sekolah seperti yang telah dijelaskan pada subbab 3.8. Implementasi dapat dilihat pada Kode Sumber 4.10

```
class PriorityQueueRanker:
  def __init__(self, school_config, students:
      List[StudentWaveWrapper] = None, schools:
      Dict[Any, School] = None):
    self.schools = {}
    self.residue = []

    self.use_heap = 0
    for x in school_config['primary_capacity']:
      if len(x['secondary_capacity']) > 0:
        self.use_heap = 1
    self.queue = [] if self.use_heap else deque()

    if students:
      self.add_students(students)
    if schools:
      self.add_schools(schools)

  def add_students(self, students:
      List[StudentWaveWrapper]):
    self.queue.extend(students)
    if self.use_heap:
      heapq.heapify(self.queue)

  def get_residue(self):
    res = self.residue
    return res

  def add_schools(self, schools: Dict[Any,
      School]):
    self.schools.update(schools)

  def get_schools(self):
    return self.schools
```

Kode Sumber 4.10 Implementasi Fungsi Pemeringkatan

```
31   def rank(self):
32     while len(self.queue):
33       if self.use_heap:
34         current_student =
               heapq.heappop(self.queue).student
35       else:
36         current_student =
               self.queue.popleft().student
37
38       school_choice_id = current_student.choices[
             current_student.wave]
39
40       if current_student.wave >=
             len(current_student.choices):
41         print("Fill error")
42         continue
43
44       if int(school_choice_id) == 0:
45         student = current_student
46       else:
47         school_choice =
               self.schools[school_choice_id]
48         student = school_choice.enqueue_student(
               current_student)
49
50       if student:
51         student.increment_wave()
52         if student.wave < len(student.choices):
53           if self.use_heap:
54             heapq.heappush(self.queue,
                   StudentWaveWrapper(student))
55           else:
56             self.queue.append(
                   StudentWaveWrapper(student))
57         else:
58           student.wave = -1
59           self.residue.append(student)
```

Kode Sumber 4.10 Implementasi Fungsi Pemeringkatan (sambungan)

4.2.8 Implementasi Fungsi Penulisan Hasil

Pada bagian ini dijabarkan implementasi fungsi penulisan hasil ke dalam basis data seperti yang telah dijelaskan pada subbab 3.9. Implementasi dapat dilihat pada Kode Sumber 4.11

```
 1 class WriterWorker(Thread):
 2   __lock = Lock()
 3
 4   def __init__(self, db_config, statement, data):
 5     Thread.__init__(self)
 6     self.db_config = db_config
 7     self.statement = statement
 8     self.data = data
 9
10   def run(self) -> None:
11     connection = create_connection(self.db_config)
12     cursor = connection.cursor()
13     cursor.executemany(self.statement, self.data)
14     connection.commit()
15     cursor.close()
16     connection.close()
17
18 def write_students(db_config, output_config,
       schools, residue):
19
20   def generate_student_tuple(rank: int, student:
         Student):
21     st = [student.attributes[x] for x in
           columns_input]
22     st.append(student.accepted if
           student.accepted != -1 else "")
23     st.append(student.wave + 1 if student.wave >=
           0 else -1)
24     st.extend([rank, student.get_formula()])
25     return tuple(st)
```

Kode Sumber 4.11 Implementasi Fungsi Penulisan Hasil

```
26   columns_output = output_config['output_column']
27   columns_output = [*columns_output,
         output_config['accepted'],
         output_config['choice'],
         output_config['rank'],
         output_config['score']]
28   columns_input = output_config['input_column']
29   output = Table(output_config['table_name'])
30   params = [Parameter('%s') for i in
         columns_output]
31   statement = MySQLQuery.into(output).columns(
         *columns_output).insert(*params)
32
33   output_students = []
34   for key in schools:
35     current_school = schools[key]
36     for rank, student in enumerate(sorted(
           current_school.waiting_list,
           reverse=True), start=1):
37       output_students.append(
             generate_student_tuple( rank, student))
38   for rank, student in enumerate(residue,
         start=1):
39     output_students.append(
           generate_student_tuple( rank, student))
40   output_students = chunks(output_students, 25000)
41   output_students.reverse()
42
43   worker_list = []
44   for chunk in output_students:
45     w = WriterWorker(db_config,
           statement.get_sql(), chunk)
46     worker_list.append(w)
47     w.start()
48   for w in worker_list:
49     w.join()
```

Kode Sumber 4.11 Implementasi Fungsi Penulisan Hasil (sambungan)

4.3 Implementasi Program Penguji

Pada subbab ini akan dijelaskan mengenai implementasi program penguji yang telah dirancang guna memeriksa hasil fungsi pemeringkatan yang telah dirancang. Pemeriksaan dilakukan pada data-data penentu penerimaan, seperti nomor identifikasi calon peserta didik, nomor identifikasi sekolah diterima, serta pada pilihan berapa calon peserta didik diterima. Pemeriksaan dilakukan dengan metode yang dijelaskan pada subbab 3.10. Implementasi dapat dilihat pada Kode Sumber 4.12.

```
from pypika import Table, MySQLQuery

from repository import create_connection

if __name__ == '__main__':
  config = {
    'host': '192.168.17.10',
    'database': 'database',
    'user': 'user',
    'password': 'password',
    'port': 3306
  }

  conn = create_connection(config)

  table = Table('output_real')
  query = MySQLQuery.from_( table ).select(
      'output_uasbn', 'output_diterima',
      'output_pilihan')

  cursor = conn.cursor(MySQLdb.cursors.DictCursor)
  cursor.execute(query.get_sql())

  a = {x['output_uasbn']: x for x in cursor}
  cursor.close()
```

Kode Sumber 4.12 Implementasi Program Penguji

```
24   table = Table('output_test')
25   query = MySQLQuery.from_( table ).select(
         'output_uasbn', 'output_diterima',
         'output_pilihan')
26
27   cursor = conn.cursor(MySQLdb.cursors.DictCursor)
28   cursor.execute(query.get_sql())
29
30   b = {x['output_uasbn']: x for x in cursor}
31   cursor.close()
32
33   i = 0
34   j = 0
35
36   for key in b:
37     if key in a:
38       if a[key] != b[key]:
39         print('mismatch!')
40         print(a[key])
41         print(b[key])
42         i+=1
43     else:
44       print('missing!')
45       print(b[key])
46       j+=1
47   print("mismatch:{} missing:{}".format(i,j))
```

Kode Sumber 4.12 Implementasi Program Penguji (sambungan)

[*Halaman ini sengaja dikosongkan*]

BAB V
UJI COBA DAN EVALUASI

5.1 Lingkungan Uji Coba

Uji coba dilakukan pada perangkat dengan spesifikasi sebagai berikut.

1. Perangkat Keras
   (a) Server *DigitalOcean*
   (b) 1 vCPU
2. Perangkat Lunak
   (a) Basis Data
      i. Sistem Operasi Ubuntu 18.04.4 LTS (Bionic Beaver)
      ii. Alokasi memori 1024 MB
      iii. MySQL 5.7.24
   (b) Program Utama
      i. Sistem Operasi Ubuntu 18.04.4 LTS (Bionic Beaver)
      ii. Alokasi memori 1024 MB
      iii. Python 3.8.2
      iv. Memory-profiler 0.57.0

5.2 Skenario Uji Coba

Pada subbab ini dijabarkan skenario uji coba yang digunakan untuk menguji kebenaran dan performa implementasi dari solusi yang telah dirancang dalam menyelesaikan permasalahan *Generalized Multiparameter Ranking* pada sistem terdistribusi.

Uji coba kebenaran dilakukan dengan cara menguji imple-

mentasi yang telah dirancang dengan himpunan data PPDB 2019, kemudian membandingkan kesamaan hasil yang dihasilkan implementasi dengan himpunan data asli. Sepertinya yang telah dijelaskan pada subbab 2.1, terdapat tiga data uji yang digunakan sebagai masukkan pengujian, yaitu data PPDB Jawa Timur tahun 2019, PPDB Sidoarjo tahun 2019, dan PPDB Surabaya tahun 2019.

Uji coba performa dilakukan dengan cara menguji implementasi yang telah dirancang dengan himpunan data PPDB 2019 sebanyak 10 kali yang kemudian dilakukan analisa performa berdasarkan waktu eksekusi dan memori yang digunakan oleh implementasi.

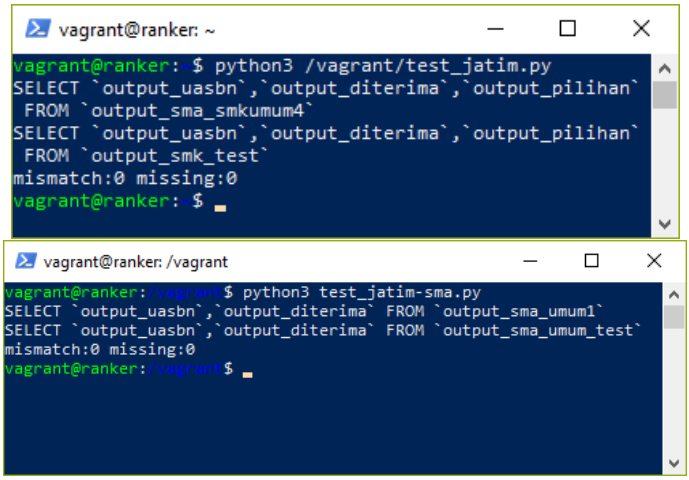
5.3 Uji Coba Kebenaran

Kebenaran metode dan implementasi diuji dengan cara memasukkan himpunan data sebagai parameter masukan dengan konfigurasi yang telah ditentukan untuk masing-masing himpunan data. Terdapat tiga himpunan data uji yang akan digunakan dengan konfigurasinya masing-masing, yaitu data PPDB Jawa Timur tahun 2019, PPDB Sidoarjo tahun 2019, dan PPDB Surabaya tahun 2019.

5.3.1 Kasus Uji PPDB Jawa Timur 2019

Pada kasus uji ini, data yang digunakan adalah data pendaftaran dan penerimaan calon peserta didik Sekolah Menengah Kejuruan (SMK) dan Sekolah Menengah Atas (SMA) tahun 2019 di provinsi Jawa Timur. Jumlah data yang terdapat pada himpunan data SMK adalah sebesar 40,6 MB dengan 122.952 baris data sedangkan pada himpunan data SMA adalah sebesar 16 MB dengan 112.943 baris data. Konfigurasi pemeringkatan dapat dilihat pada Lampiran A.

Setelah pemeringkatan dilakukan, maka dilakukan pengujian menggunakan fungsi dan implementasi yang telah dijabarkan pada subbab 4.3. Hasil perbandingan antara data asli dengan data hasil implementasi dapat dilihat pada Gambar 5.1 dengan hasil kesesuaian pada seluruh perbandingan data.

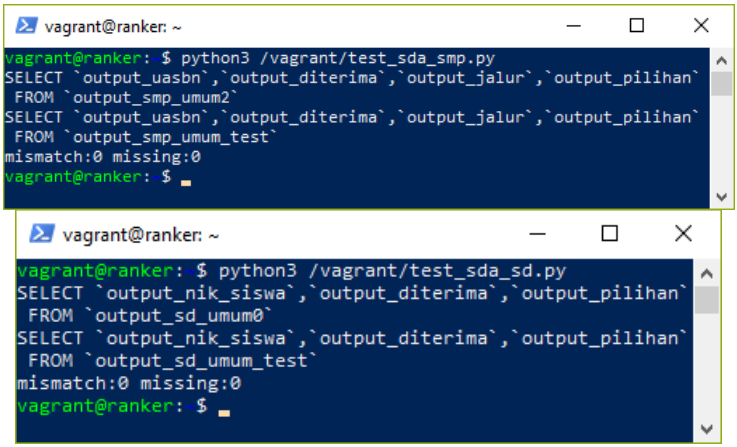


Gambar 5.1 Hasil pengujian perbandingan data pada *dataset* PPDB SMK dan SMA Jawa Timur tahun 2019

5.3.2 Kasus Uji PPDB Sidoarjo 2019

Pada kasus uji ini, data yang digunakan adalah data pendaftaran dan penerimaan calon peserta didik Sekolah Dasar (SD) dan Sekolah Menengah Pertama (SMP) tahun 2019 di kabupaten Sidoarjo, provinsi Jawa Timur. Jumlah data yang terdapat pada himpunan data ini sebesar 14,8 MB dengan 22.419 baris data. Konfigurasi pemeringkatan dapat dilihat pada Lampiran A.

Setelah pemeringkatan dilakukan, maka dilakukan pengujian menggunakan fungsi yang telah dijabarkan pada subbab 4.3. Hasil perbandingan antara data asli dengan data hasil implementasi dapat dilihat pada Gambar 5.2 dengan hasil kesesuaian pada seluruh perbandingan data.



Gambar 5.2 Hasil pengujian perbandingan data pada *dataset* PPDB Sidoarjo tahun 2019

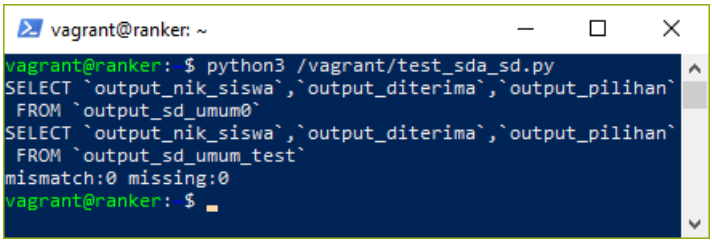
5.3.3 Kasus Uji PPDB Surabaya 2019

Pada kasus uji ini, data yang digunakan adalah data pendaftaran dan penerimaan calon siswa SMP tahun 2019 di kota Surabaya, provinsi Jawa Timur. Jumlah data yang terdapat pada himpunan data ini sebesar 15,4 MB dengan 23.471 baris data. Konfigurasi pemeringkatan dapat dilihat pada Lampiran A.

Setelah pemeringkatan dilakukan, maka dilakukan pengujian menggunakan fungsi yang telah dijabarkan pada subbab 4.3. Hasil perbandingan antara data asli dengan data hasil implementasi dapat dilihat pada Gambar 5.3 dengan hasil kesesuaian pada seluruh perbandingan data.

5.4 Uji Coba Kinerja

Subbab ini akan menjabarkan kinerja implementasi rancangan solusi dihadapkan dengan kasus uji. Uji coba kinerja dilakukan pada lingkungan uji coba yang dijabarkan pada subbab 5.1.



Gambar 5.3 Hasil pengujian perbandingan data pada *dataset* PPDB Surabaya tahun 2019

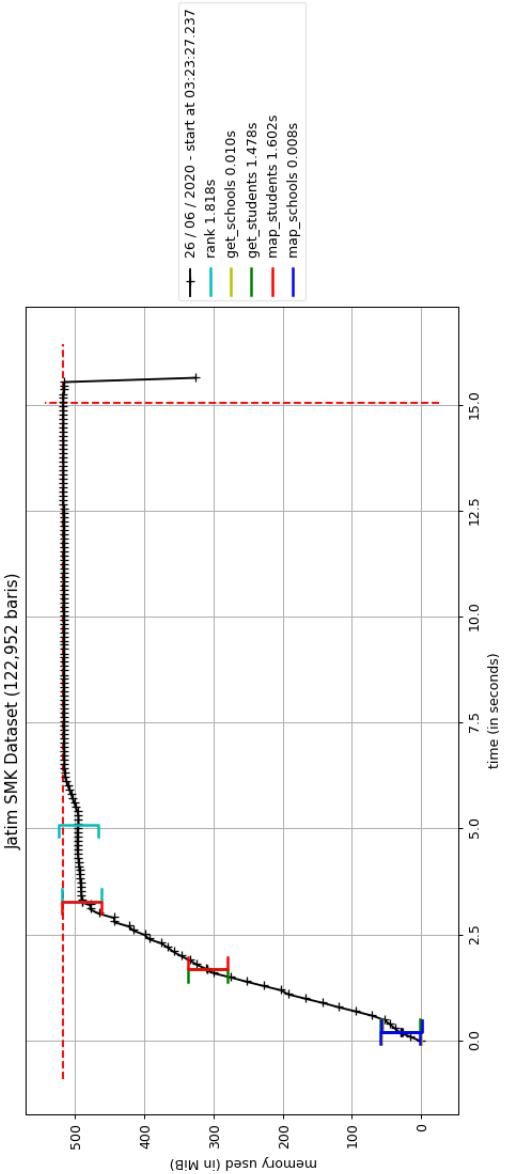
Uji coba kinerja dilakukan dengan cara mengeksekusi program sebanyak sepuluh kali, kemudian diukur waktu dan memori yang dibutuhkan untuk menyelesaikan setiap kasus uji. Hasil waktu dan memori setiap eksekusi program dicatat menggunakan bantuan modul *memory-profiler*.

5.4.1 Kinerja Kasus Uji PPDB Jawa Timur 2019

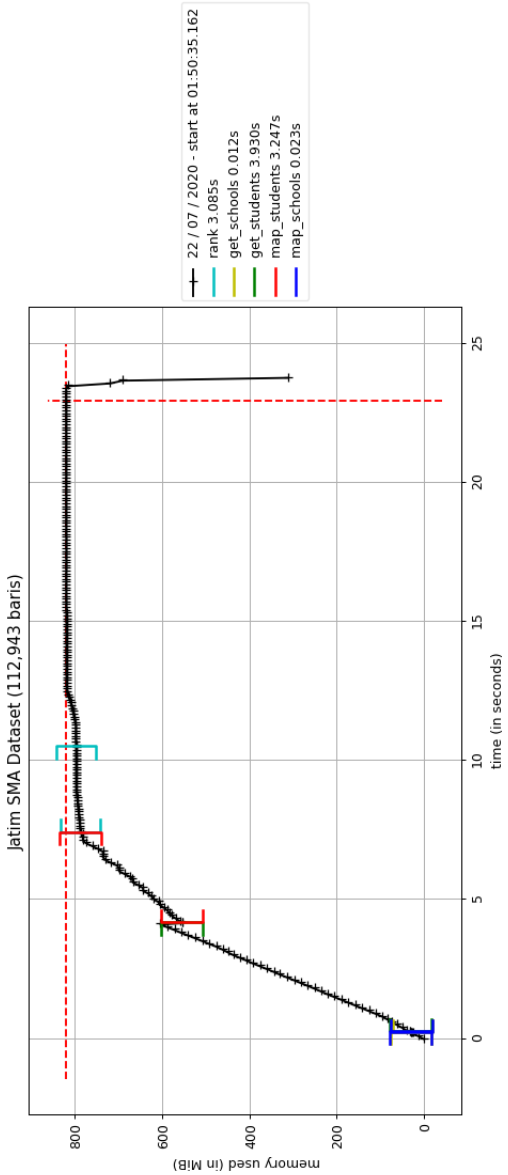
Tabel 5.1 dan 5.2 menjabarkan waktu dan memori yang dibutuhkan oleh program setelah dijalankan sebanyak sepuluh kali.

Gambar 5.4 menjelaskan waktu eksekusi dan memori dari program pada kasus uji PPDB SMK Jawa Timur. Eksekusi fungsi *rank* dijalankan dengan waktu rata-rata sebesar 2,83 detik. Eksekusi program utama menghasilkan waktu rata-rata sebesar 16,46 detik dengan memori rata-rata sebesar 538 MB, dengan porsi waktu eksekusi terbesar pada fungsi pengambilan dan penulisan data.

Gambar 5.5 menjelaskan waktu eksekusi dan memori dari program pada kasus uji PPDB SMA Jawa Timur. Eksekusi fungsi *rank* dijalankan dengan waktu rata-rata sebesar 3,42 detik. Eksekusi program utama menghasilkan waktu rata-rata sebesar 24,58 detik dengan memori rata-rata sebesar 819 MB, dengan porsi waktu eksekusi terbesar pada fungsi pengambilan dan penulisan data.



Gambar 5.4 Gambar Kinerja Kasus Uji PPDB SMK Jawa Timur 2019



Gambar 5.5 Gambar Kinerja Kasus Uji PPDB SMA Jawa Timur 2019

Tabel 5.1 Tabel Kinerja Kasus Uji PPDB SMK Jawa Timur 2019

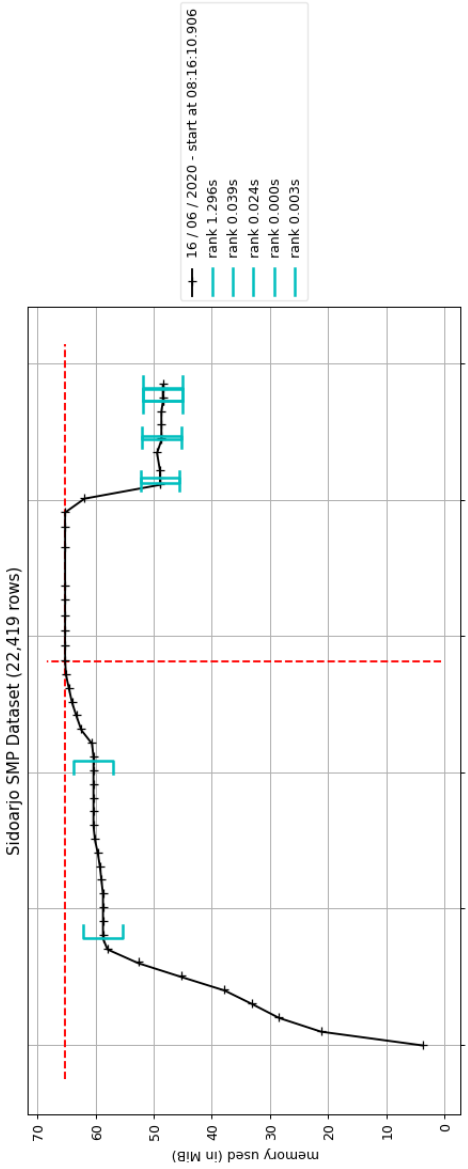
No	Waktu eksekusi fungsi rank (s)	Waktu eksekusi program (s)	Memori (MB)
1	2.03	16.08	517.29
2	1.88	15.89	517.67
3	1.98	15.99	517.68
4	2.02	16.17	517.53
5	1.86	15.67	517.47
6	1.81	15.38	517.60
7	1.95	15.57	517.45
8	1.82	15.68	517.56
9	1.86	15.57	517.71
10	1.97	15.88	517.45

Tabel 5.2 Tabel Kinerja Kasus Uji PPDB SMA Jawa Timur 2019

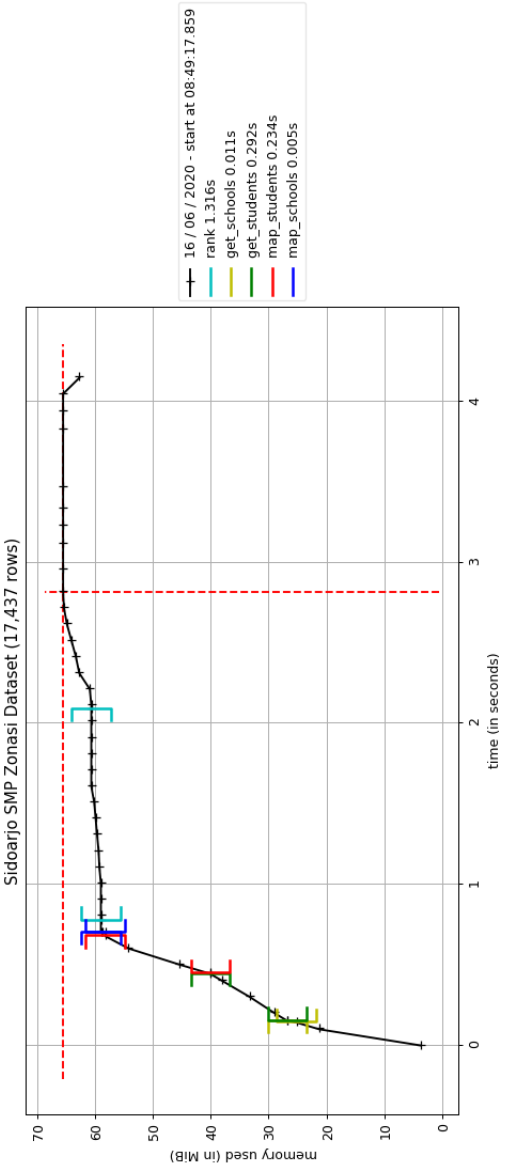
No	Waktu eksekusi fungsi rank (s)	Waktu eksekusi program (s)	Memori (MB)
1	3.08	23.78	818.84
2	3.61	26.19	818.78
3	3.83	25.80	819.59
4	3.36	24.77	819.39
5	3.35	23.67	819.55
6	3.27	23.98	819.70
7	3.25	23.48	819.64
8	3.38	24.09	819.60
9	3.52	25.39	819.65
10	3.52	24.68	819.53

5.4.2 Kinerja Kasus Uji PPDB Sidoarjo 2019

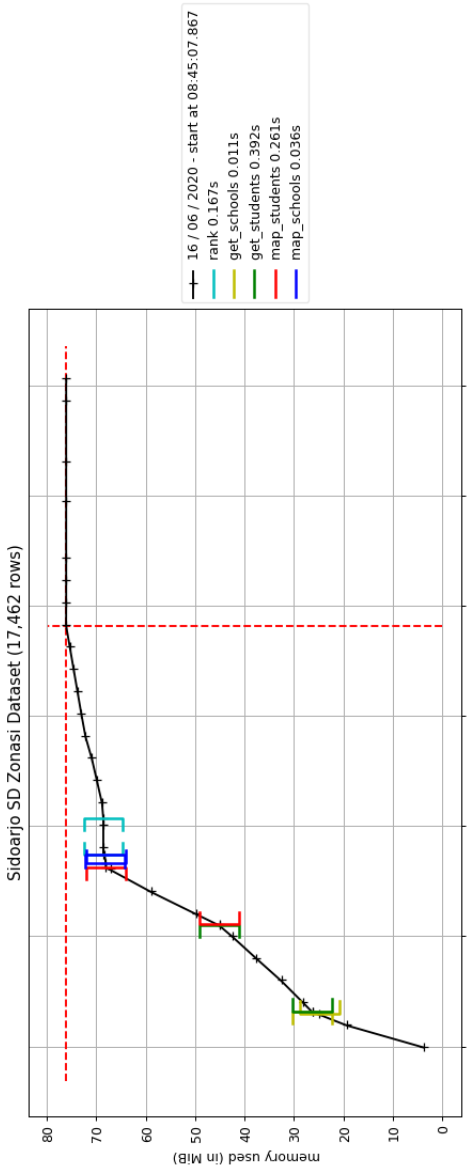
Tabel 5.3 dan 5.4 menjabarkan waktu dan memori yang dibutuhkan oleh program setelah dijalankan sebanyak sepuluh kali.



Gambar 5.6 Gambar Kinerja Kasus Uji PPDB SMP Sidoarjo 2019 pada seluruh jalur



Gambar 5.7 Gambar Kinerja Kasus Uji PPDB SMP Sidoarjo 2019 Jalur Zonasi



Gambar 5.8 Gambar Kinerja Kasus Uji PPDB SD Sidoarjo 2019

Tabel 5.3 Tabel Kinerja Kasus Uji PPDB SMP Sidoarjo 2019

No	Waktu eksekusi fungsi rank (s)	Waktu eksekusi program (s)	Memori (MB)
1	1,36	4,98	65,57
2	1,35	5,32	65,25
3	1,34	4,97	65,28
4	1,35	4,97	65,10
5	1,42	5,44	65,03
6	1,36	5,15	65,27
7	1,37	5,35	65,00
8	1,36	5,27	65,02
9	1,36	4,85	65,27
10	1,37	4,93	65,39

Tabel 5.4 Tabel Kinerja Kasus Uji PPDB SD Sidoarjo 2019

No	Waktu eksekusi fungsi rank (s)	Waktu eksekusi program (s)	Memori (MB)
1	0,17	3,16	76,34
2	0,17	3,26	76,37
3	0,19	3,45	76,49
4	0,17	3,43	76,37
5	0,17	3,43	76,27
6	0,17	3,03	76,09
7	0,17	3,22	76,20
8	0,17	3,44	76,27
9	0,18	3,40	76,39
10	0,17	3,39	76,49

Gambar 5.6 dan 5.7 menjelaskan waktu eksekusi dan memori dari program pada kasus uji PPDB SMP Sidoarjo. Berdasarkan Tabel 5.3, eksekusi fungsi *rank* dijalankan dengan waktu rata-rata

sebesar 1,36 detik. Eksekusi program utama menghasilkan waktu rata-rata sebesar 5,12 detik dengan memori rata-rata sebesar 65 MB.

Gambar 5.8 menjelaskan waktu eksekusi dan memori dari program pada kasus uji PPDB SD Sidoarjo. Berdasarkan Tabel 5.4, eksekusi fungsi *rank* dijalankan dengan waktu rata-rata sebesar 0,17 detik. Eksekusi program utama menghasilkan waktu rata-rata sebesar 3,32 detik dengan memori rata-rata sebesar 76 MB.

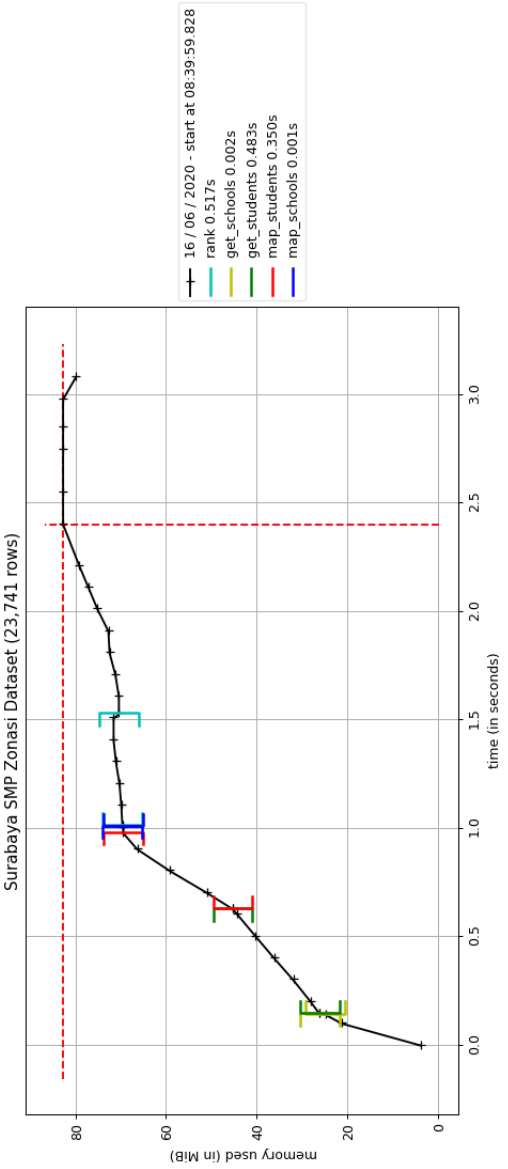
5.4.3 Kinerja Kasus Uji PPDB Surabaya 2019

Tabel 5.5 menjabarkan waktu dan memori yang dibutuhkan oleh program setelah dijalankan sebanyak sepuluh kali.

Tabel 5.5 Tabel Kinerja Kasus Uji PPDB SMP Surabaya 2019

No	Waktu eksekusi fungsi rank (s)	Waktu eksekusi program (s)	Memori (MB)
1	0,51	3,39	82,95
2	0,50	3,12	82,89
3	0,48	3,15	82,91
4	0,52	3,08	82,73
5	0,51	3,22	82,71
6	0,50	3,14	82,70
7	0,49	3,13	82,98
8	0,51	3,21	82,96
9	0,54	3,23	82,69
10	0,50	3,14	83,09

Gambar 5.9 menjelaskan waktu eksekusi dan memori dari program pada kasus uji PPDB SD Sidoarjo. Berdasarkan Tabel 5.5, eksekusi fungsi *rank* dijalankan dengan waktu rata-rata sebesar 0,50 detik. Eksekusi program utama menghasilkan waktu rata-rata sebesar 3,18 detik dengan memori rata-rata sebesar 82 MB.



Gambar 5.9 Gambar Kinerja Kasus Uji PPDB SMP Sidoarjo 2019 Jalur Zonasi

5.5 Analisis Pengujian dan Kesimpulan Umum

Hasil uji coba kebenaran pada seluruh kasus uji menghasilkan kesesuaian pada seluruh perbandingan data. Hal ini menunjukkan bahwa metode yang telah dirancang dengan perubahan pada tingkat konfigurasi mampu bekerja dengan tepat pada seluruh kasus uji. Kemudian, hasil uji coba kinerja menghasilkan waktu pemeringkatan yang sesuai dengan jumlah ukuran data yang ada. Uji coba kinerja juga sejalan dengan analisis sistem pemeringkatan yang telah dijelaskan pada 3.2 yang menunjukkan kompleksitas waktu sebesar $O(n \log n)$.

Kesimpulan secara umum dari pengujian diatas adalah bahwa rancangan solusi *Generalized Multiparameter Ranking* mampu diimplementasikan serta diuji kebenarannya melalui berbagai macam kasus uji. Hal ini dibuktikan dengan hasil perbandingan setiap kasus uji yang cocok pada setiap perbandingan data.

[*Halaman ini sengaja dikosongkan*]

BAB VI
KESIMPULAN

Pada bab ini dijelaskan mengenai kesimpulan dan saran terkait perancangan solusi permasalahan *Generalized Multiparameter Ranking* pada sistem terdistribusi berdasarkan penjabaran pada bab 2 dan bab 3.

6.1 Kesimpulan

Dari hasil uji coba yang dilakukan terhadap implementasi rancangan solusi, maka didapatkan hasil sebagai berikut:

1. Permasalahan *Generalized Multiparameter Ranking* pada sistem terdistribusi mampu dimodelkan dan diselesaikan menggunakan metode pemeringkatan bertingkat menggunakan struktur data Queue dan Priority Queue.
2. Permasalahan *Generalized Multiparameter Ranking* pada sistem terdistribusi mampu diimplementasikan serta diujikan pada seluruh studi kasus (dalam hal ini studi kasus PPDB 2019) tanpa ketidakcocokan hasil data.

6.2 Saran

Penyelesaian permasalahan *Generalized Multiparameter Ranking* masih memiliki ruang untuk dikembangkan. Pengembangan pada bagian pemodelan solusi dapat dilakukan dengan mengembangkan metode yang mampu mengurutkan data dengan lebih cepat. Pengembangan juga dapat dilakukan pada perancangan model untuk mendukung pemodelan pemeringkatan yang lebih kompleks. Selain itu, pengembangan implementasi juga dapat dilakukan dengan menggunakan bahasa pemrograman atau basis data lain yang bisa menjalankan solusi pemodelan dengan lebih cepat.

[*Halaman ini sengaja dikosongkan*]

DAFTAR PUSTAKA

[1] D. Beaudoin **and** T. Swartz, "A computationally intensive ranking system for paired comparison data," *Operations Research Perspectives*, **jourvol** 5, **pages** 105 –112, 2018, ISSN: 2214-7160. DOI: https://doi.org/10.1016/j.orp.2018.03.002. **url**: http://www.sciencedirect.com/science/article/pii/S2214716017302208.

[2] R. A. Bradley **and** M. E. Terry, "Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons," *Biometrika*, **jourvol** 39, **number** 3/4, **pages** 324–345, 1952, ISSN: 00063444. **url**: http://www.jstor.org/stable/2334029.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest **and** C. Stein, *Introduction to Algorithms, Third Edition*, 3rd. The MIT Press, 2009, ISBN: 0262033844.

[4] KAYAK Germany, GmbH, *PyPika - Python Query Builder — PyPika 0.35.16 documentation*, 2020. **url**: https://pypika.readthedocs.io/en/latest/.

[5] Pemerintah Kabupaten Sidoarjo, *Peraturan Bupati Sidarjo Nomor 26 Tahun 2019 tentang Penerimaan Peserta Didik Baru pada Taman Kanak-kanak Negeri, Sekolah Dasar Negeri dan Sekolah Menengah Pertama Negeri di Kabupaten Sidoarjo*, 2019.

[6] Pemerintah Kota Surabaya, *Peraturan Walikota Surabaya Nomor 25 Tahun 2019 tentang Petunjuk Teknis Pelaksanaan Penerimaan Peserta Didik Baru (PPDB) Jenjang Sekolah Dasar Negeri dan Sekolah Menengah Pertama Negeri Tahun Ajaran 2019/2020*, 2019.

[7] Pemerintah Provinsi Jawa Timur, *Keputusan Kepala Dinas Pendidikan Provinsi Jawa Timur Nomor: 188.4/3112/101.7.1/KP-TS/2019 tentang Petunjuk Teknis Pelaksanaan Penerimaan Peserta Didik Baru (PPDB) jenjang SMA Negeri, SMK Negeri dan SLB Negeri Provinsi Jawa Timur Tahun Pelajaran 2019/2020*, 2019.

[8] M. T. Özsu **and** P. Valduriez, *Principles of Distributed Database Systems*. Springer International Publishing, 2020. DOI: 10.1007/978-3-030-26253-2. **url**: https://doi.org/10.1007/978-3-030-26253-2.

Lampiran A: Konfigurasi Pemeringkatan Kasus Uji

```
{
  "input_database": {
    "host": "hostname",
    "database": "ppdb_jatim_2019",
    "user": "username",
    "password": "password",
    "port": 3306
  },
  "output_database": {
    "host": "hostname",
    "database": "ppdb_jatim_2019",
    "user": "username",
    "password": "password",
    "port": 3306
  },
  "student": {
    "table_name": "input_smk_umum",
    "student_id": "input_uasbn",
    "attributes": [
      "input_nama_siswa"
    ],
    "filter": [
      {
        "field": "input_status_validasi",
        "values": [
          1
        ]
      }
    ]
  },
  "school": {
    "table_name": "sekolah",
    "school_id": "id_sekolah",
    "attributes": [
      "nama_sekolah"
    ],
```

```
    "primary_capacity": [
      {
        "capacity": "pagu_un_non_zi",
        "label": "umum",
        "secondary_capacity": []
      }
    ]
  },
  "choice_rules": [
    {
      "school_choice": "input_pilihan1",
      "lane_label": "input_jalur",
      "rule": []
    },
    {
      "school_choice": "input_pilihan2",
      "lane_label": "input_jalur",
      "rule": []
    }
  ],
  "global_rules": [
    {
      "field": "input_nilai_uan",
      "descending": true
    },
    {
      "field": "input_nilai_bind",
      "descending": true
    },
    {
      "field": "input_nilai_ipa",
      "descending": true
    },
    {
      "field": "input_nilai_mat",
      "descending": true
    },
    {
      "field": "input_nilai_bing",
      "descending": true
    },
```

```
    {
      "field": "input_timestamp",
      "descending": false
    }
  ],
  "output": {
    "table_name": "output_smk_test",
    "accepted": "output_diterima",
    "choice": "output_pilihan",
    "rank": "output_urut",
    "score": "output_formula",
    "write_residue": false,
    "truncate": true,
    "input_column": [
      "input_uasbn",
      "input_nama_siswa",
      "input_pilihan1",
      "input_pilihan2",
      "input_nilai_uan",
      "input_nilai_bind",
      "input_nilai_ipa",
      "input_nilai_mat",
      "input_nilai_bing",
      "input_timestamp"
    ],
    "output_column": [
      "output_uasbn",
      "output_nama_siswa",
      "output_pilihan1",
      "output_pilihan2",
      "output_nilai_uan",
      "output_nilai_bind",
      "output_nilai_ipa",
      "output_nilai_mat",
      "output_nilai_bing",
      "output_timestamp"
    ]
  }
}
```

Gambar A.1 Konfigurasi Pemeringkatan PPDB SMK Jawa Timur

```
{
  "input_database": {
    "host": "hostname",
    "database": "ppdb_jatim_2019",
    "user": "username",
    "password": "password",
    "port": 3306
  },
  "output_database": {
    "host": "hostname",
    "database": "ppdb_jatim_2019",
    "user": "username",
    "password": "password",
    "port": 3306
  },
  "student": {
    "table_name": "input_sma_umum",
    "student_id": "input_uasbn",
    "attributes": [
      "input_nama_siswa",
      "input_nilai_bind",
      "input_nilai_ipa",
      "input_nilai_mat",
      "input_nilai_bing",
      "jarak_pilihan1",
      "jarak_pilihan2"
    ],
    "filter": []
  },
  "school": {
    "table_name": "sekolah",
    "school_id": "id_sekolah",
    "attributes": [
      "nama_sekolah"
    ],
    "primary_capacity": [
      {
        "capacity": "pagu_un_non_zi",
        "label": "ZD-N",
        "secondary_capacity": []
```

```
      },
      {
        "capacity": "pagu_un_zi",
        "label": "ZI-N",
        "secondary_capacity": []
      },
      {
        "capacity": "pagu_jarak_non_zi",
        "label": "ZD-J",
        "secondary_capacity": []
      },
      {
        "capacity": "pagu_jarak_zi",
        "label": "ZI-J",
        "secondary_capacity": []
      }
    ]
  },
  "choice_rules": [
    {
      "school_choice": "input_pilihan1",
      "lane_label": "status_zona_jarak1",
      "rule": [
        {
          "field": "jarak_pilihan1",
          "descending": false
        },
        {
          "field": "input_nilai_uan",
          "descending": true
        }
      ]
    },
    {
      "school_choice": "input_pilihan1",
      "lane_label": "status_zona_nilai1",
      "rule": [
        {
          "field": "input_nilai_uan",
          "descending": true
        },
```

```
 82          {
 83            "field": "input_nilai_bind",
 84            "descending": true
 85          },
 86          {
 87            "field": "input_nilai_ipa",
 88            "descending": true
 89          },
 90          {
 91            "field": "input_nilai_mat",
 92            "descending": true
 93          },
 94          {
 95            "field": "input_nilai_bing",
 96            "descending": true
 97          }
 98        ]
 99      },
100      {
101        "school_choice": "input_pilihan2",
102        "lane_label": "status_zona_jarak2",
103        "rule": [
104          {
105            "field": "jarak_pilihan2",
106            "descending": false
107          },
108          {
109            "field": "input_nilai_uan",
110            "descending": true
111          }
112        ]
113      },
114      {
115        "school_choice": "input_pilihan2",
116        "lane_label": "status_zona_nilai2",
117        "rule": [
118          {
119            "field": "input_nilai_uan",
120            "descending": true
121          },
122          {
```

```
123           "field": "input_nilai_bind",
124           "descending": true
125         },
126         {
127           "field": "input_nilai_ipa",
128           "descending": true
129         },
130         {
131           "field": "input_nilai_mat",
132           "descending": true
133         },
134         {
135           "field": "input_nilai_bing",
136           "descending": true
137         }
138       ]
139     }
140   ],
141   "global_rules": [
142     {
143       "field": "input_timestamp",
144       "descending": false
145     }
146   ],
147   "output": {
148     "table_name": "output_sma_umum_test",
149     "accepted": "output_diterima",
150     "choice": "output_pilihan",
151     "rank": "output_rank",
152     "score": "output_nilai",
153     "write_residue": true,
154     "truncate": true,
155     "input_column": [
156       "input_uasbn",
157       "input_nama_siswa",
158       "input_pilihan1",
159       "input_pilihan2",
160       "input_nilai_uan",
161       "input_nilai_bind",
162       "input_nilai_ipa",
163       "input_nilai_mat",
```

```
164       "input_nilai_bing",
165       "jarak_pilihan1",
166       "jarak_pilihan2",
167       "input_timestamp"
168     ],
169     "output_column": [
170       "output_uasbn",
171       "output_nama_siswa",
172       "output_pilihan1",
173       "output_pilihan2",
174       "output_nilai_uan",
175       "output_nilai_bind",
176       "output_nilai_ipa",
177       "output_nilai_mat",
178       "output_nilai_bing",
179       "output_jarak_pilihan1",
180       "output_jarak_pilihan2",
181       "output_timestamp"
182     ]
183   }
184 }
```

Gambar A.2 Konfigurasi Pemeringkatan PPDB SMA Jawa Timur

```
{
  "input_database": {
    "host": "hostname",
    "database": "ppdb_sda_2019",
    "user": "username",
    "password": "password",
    "port": 3306
  },
  "output_database": {
    "host": "hostname",
    "database": "ppdb_sda_2019",
    "user": "username",
    "password": "password",
    "port": 3306
  },
  "student": {
    "table_name": "input_smp_umum",
    "student_id": "input_uasbn",
    "attributes": [ "input_nama_siswa" ],
    "filter": []
  },
  "school": {
    "table_name": "sekolah",
    "school_id": "kode_sekolah",
    "attributes": [ "nama_sekolah" ],
    "primary_capacity": [
      {
        "capacity": "pagu_zonasi",
        "label": "murni",
        "secondary_capacity": [
          {
            "capacity": "pagu_luarkota",
            "label": "luar"
          }
        ]
      },
      {
        "capacity": "pagu_tendik",
        "label": "tendik",
        "secondary_capacity": []
```

```
      },
      {
        "capacity": "pagu_teladan",
        "label": "teladan",
        "secondary_capacity": []
      },
      {
        "capacity": "pagu_tidak_mampu",
        "label": "miskin",
        "secondary_capacity": []
      },
      {
        "capacity": "pagu_inklusif",
        "label": "inklusi",
        "secondary_capacity": []
      }
    ]
  },
  "choice_rules": [
    {
      "school_choice": "input_kode_pilihan1",
      "lane_label": "input_jalur_rank",
      "rule": [
        {
          "field": "input_skor1",
          "descending": true
        }
      ]
    },
    {
      "school_choice": "input_kode_pilihan2",
      "lane_label": "input_jalur_rank",
      "rule": [
        {
          "field": "input_skor2",
          "descending": true
        }
      ]
    }
  ],
  "global_rules": [
```

```
    {
      "field": "input_timestamp",
      "descending": false
    }
  ],
  "output": {
    "table_name": "output_smp_umum_test",
    "accepted": "output_diterima",
    "choice": "output_pilihan",
    "rank": "output_rank",
    "score": "output_nilai",
    "write_residue": true,
    "truncate": true,
    "input_column": [
      "input_uasbn",
      "input_nama_siswa",
      "input_kode_pilihan1",
      "input_skor1",
      "input_kode_pilihan2",
      "input_skor2",
      "input_jalur_rank",
      "input_timestamp"
    ],
    "output_column": [
      "output_uasbn",
      "output_nama_siswa",
      "output_kode_pilihan1",
      "output_skor1",
      "output_kode_pilihan2",
      "output_skor2",
      "output_jalur",
      "output_timestamp"
    ]
  }
}
```

Gambar A.3 Konfigurasi Pemeringkatan PPDB SMP Zonasi Sidoarjo

```
{
  "input_database": {
    "host": "hostname",
    "database": "ppdb_sda_2019",
    "user": "username",
    "password": "password",
    "port": 3306
  },
  "output_database": {
    "host": "hostname",
    "database": "ppdb_sda_2019",
    "user": "username",
    "password": "password",
    "port": 3306
  },
  "student": {
    "table_name": "input_sd_umum",
    "student_id": "input_nik_calon_siswa",
    "attributes": [
      "input_nama_calon_siswa",
      "input_tempat_lahir_calon_siswa",
      "input_tanggal_lahir_calon_siswa"
    ],
    "filter": []
  },
  "school": {
    "table_name": "sekolah",
    "school_id": "kode_sekolah",
    "attributes": [
      "nama_sekolah"
    ],
    "primary_capacity": [
      {
        "capacity": "pagu_zonasi",
        "label": "murni",
        "secondary_capacity": []
      }
    ]
  },
  "choice_rules": [
```

```
    {
      "school_choice": "input_sd_id",
      "lane_label": "input_jalur",
      "rule": [
        {
          "field": "is_cerdas_istimewa",
          "descending": false
        },
        {
          "field": "input_skor1",
          "descending": true
        }
      ]
    },
    {
      "school_choice": "input_sd2_id",
      "lane_label": "input_jalur",
      "rule": [
        {
          "field": "is_cerdas_istimewa",
          "descending": false
        },
        {
          "field": "input_skor2",
          "descending": true
        }
      ]
    }
  ],
  "global_rules": [
    {
      "field": "input_tanggal_lahir_calon_siswa",
      "descending": false
    },
    {
      "field": "input_timestamp",
      "descending": false
    },
    {
      "field": "input_pendaftaran_id",
      "descending": false
```

```
    }
  ],
  "output": {
    "table_name": "output_sd_umum_test",
    "accepted": "output_diterima",
    "choice": "output_pilihan",
    "rank": "output_rank",
    "score": "output_nilai",
    "write_residue": true,
    "truncate": true,
    "input_column": [
      "input_nik_calon_siswa",
      "input_nama_calon_siswa",
      "input_tempat_lahir_calon_siswa",
      "input_tanggal_lahir_calon_siswa",
      "input_sd_id",
      "input_sd2_id",
      "input_timestamp"
    ],
    "output_column": [
      "output_nik_siswa",
      "output_nama_siswa",
      "output_tempat_lahir",
      "output_tanggal_lahir",
      "output_pilihan1",
      "output_pilihan2",
      "output_timestamp"
    ]
  }
}
```

Gambar A.4 Konfigurasi Pemeringkatan PPDB SD Sidoarjo

```
{
  "input_database": {
    "host": "hostname",
    "database": "ppdb_sby_2019",
    "user": "username",
    "password": "password",
    "port": 3306
  },
  "output_database": {
    "host": "hostname",
    "database": "ppdb_sby_2019",
    "user": "username",
    "password": "password",
    "port": 3306
  },
  "student": {
    "table_name": "input_smp_zonasi",
    "student_id": "input_nik",
    "attributes": [
      "input_nama_siswa"
    ],
    "filter": [
      {
        "field": "input_status_validasi",
        "values": [
          1
        ]
      }
    ]
  },
  "school": {
    "table_name": "sekolah",
    "school_id": "id_sekolah",
    "attributes": [
      "nama_sekolah"
    ],
    "primary_capacity": [
      {
        "capacity": "pagu_zonasi",
        "label": "zonasi",
```

```
41          "secondary_capacity": []
42        }
43      ]
44    },
45    "choice_rules": [
46      {
47        "school_choice": "input_pilihan1",
48        "lane_label": "zonasi",
49        "rule": [
50          {
51            "field": "input_jarak1",
52            "descending": false
53          }
54        ]
55      },
56      {
57        "school_choice": "input_pilihan2",
58        "lane_label": "zonasi",
59        "rule": [
60          {
61            "field": "input_jarak2",
62            "descending": false
63          }
64        ]
65      }
66    ],
67    "global_rules": [
68      {
69        "field": "input_timestamp",
70        "descending": false
71      },
72      {
73        "field": "input_smp_id",
74        "descending": false
75      }
76    ],
77    "output": {
78      "table_name": "output_smp_zonasi_test",
79      "accepted": "output_diterima",
80      "choice": "output_pilihan",
81      "rank": "output_urut",
```

```
 82      "score": "output_formula",
 83      "write_residue": false,
 84      "truncate": true,
 85      "input_column": [
 86        "input_nik",
 87        "input_nama_siswa",
 88        "input_pilihan1",
 89        "input_jarak1",
 90        "input_pilihan2",
 91        "input_jarak2",
 92        "input_timestamp"
 93      ],
 94      "output_column": [
 95        "output_nik",
 96        "output_nama_siswa",
 97        "output_pilihan1",
 98        "output_jarak1",
 99        "output_pilihan2",
100        "output_jarak2",
101        "output_timestamp"
102      ]
103    }
104 }
```

Gambar A.5 Konfigurasi Pemeringkatan PPDB SMP Zonasi Surabaya

[*Halaman ini sengaja dikosongkan*]

BIODATA PENULIS

Penulis bernama Jonathan Rehuel Lewerissa, lahir di Tebing Tinggi pada tanggal 25 Mei 1998. Penulis merupakan anak pertama dari dua bersaudara. Penulis telah menempuh pendidikan di SD Kr. Pelangi Kristus Surabaya, SMP Kr. Pelangi Kristus Surabaya, dan SMA Kr. Kalam Kudus Surabaya. Penulis melanjutkan pendidikan tingkat sarjana di Departemen Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember Surabaya. Selama masa studi tingkat sarjana, penulis mengikuti dan aktif di beberapa organisasi dan kepanitiaan seperti Persekutuan Mahasiswa Kristen ITS, Schematics, dan Himpunan Mahasiswa Teknik Computer-Informatika ITS. Selain itu, penulis juga menjadi administrator Laboratorium Pemrograman 2, Departemen Teknik Informatika mulai tahun 2018. Penulis memiliki ketertarikan secara khusus pada bidang perancangan dan optimasi algoritma, pengembangan sistem informasi, serta perancangan perangkat lunak. Dalam meningkatkan kemampuan, penulis telah memiliki pengalaman pengerjaan proyek perangkat lunak pertandingan olahraga air di Fluidra Malaysia yang juga digunakan pada kegiatan Asian Games 2018 dan Asian Para Games 2018. Selain itu, penulis juga pernah mengikuti kegiatan PPDB Kabupaten Sidoarjo tahun 2019-2020 dan PPDB Kabupaten Ponorogo tahun 2020 sebagai *developer*. Penulis bisa dihubungi melalui *jonathan.rehuel16@gmail.com*.