TUGAS AKHIR - IF184802

IMPLEMENTASI PEMROGRAMAN DINAMIS DENGAN TEKNIK CONVEX HULL SEBAGAI LANGKAH OPTIMASI DALAM PENCARIAN SOLUSI OPTIMAL PADA PERMASALAHAN SPOJ CLOUDMG: *CLOUD COMPUTING*

BELLA SEPTINA IKA HARTANTI
NRP 05111740000117

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

Dosen Pembimbing 2
Ir. M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya, 2021

[*Halaman ini sengaja dikosongkan*]

TUGAS AKHIR - IF184802

IMPLEMENTASI PEMROGRAMAN DINAMIS DENGAN TEKNIK CONVEX HULL SEBAGAI LANGKAH OPTIMASI DALAM PENCARIAN SOLUSI OPTIMAL PADA PERMASALAHAN SPOJ CLOUDMG: *CLOUD COMPUTING*

BELLA SEPTINA IKA HARTANTI
NRP 05111740000117

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

Dosen Pembimbing 2
Ir. M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.

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

[*Halaman ini sengaja dikosongkan*]

UNDERGRADUATE THESIS - IF184802

IMPLEMENTATION OF DYNAMIC PROGRAMMING WITH THE CONVEX HULL TECHNIQUE AS AN OPTIMIZATION STEP IN THE SEARCH FOR OPTIMAL SOLUTIONS TO THE PROBLEM OF SPOJ CLOUDMG: CLOUD COMPUTING

BELLA SEPTINA IKA HARTANTI
NRP 05111740000117

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

Supervisor 2
Ir. M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.

DEPARTMENT OF INFORMATICS
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2021

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

IMPLEMENTASI PEMROGRAMAN DINAMIS DENGAN TEKNIK CONVEX HULL SEBAGAI LANGKAH OPTIMASI DALAM PENCARIAN SOLUSI OPTIMAL PADA PERMASALAHAN SPOJ CLOUDMG: CLOUD COMPUTING

TUGAS AKHIR

Diajukan untuk 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:
Bella Septina Ika Hartanti
NRP: 05111740000117

Disetujui oleh Pembimbing tugas akhir:

1. Rully Soelaiman, S.Kom., M.Kom.
   (NIP. 197002131994021001)
   ……
   (Pembimbing 1)

2. Ir. M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.
   (NIP. 197402092002121001)
   ……
   (Pembimbing 2)

SURABAYA
JUNI, 2021

IMPLEMENTASI PEMROGRAMAN DINAMIS DENGAN TEKNIK CONVEX HULL SEBAGAI LANGKAH OPTIMASI DALAM PENCARIAN SOLUSI OPTIMAL PADA PERMASALAHAN SPOJ CLOUDMG: *CLOUD COMPUTING*

Nama	:	Bella Septina Ika Hartanti
NRP	:	05111740000117
Departemen	:	Departemen Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, ITS
Pembimbing I	:	Rully Soelaiman, S.Kom., M.Kom.
Pembimbing II	:	Ir. M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.

Abstrak

Pemrograman Dinamis merupakan salah satu teknik optimasi yang cukup banyak diterapkan untuk menyelesaikan permasalahan. Pemrograman dinamis, seperti metode divide-and-conquer, menyelesaikan permasalahan dengan cara membagi permasalahan utama menjadi sub-permasalahan. Pemrograman dinamis secara umum diterapkan pada permasalahan optimasi (Cormen et al, 2009). Implementasi strategi ini dalam kehidupan sehari-hari biasa digunakan dalam mencari solusi permasalahan rute optimal, graf, jaringan komputer, dsb. Permasalahan optimasi lain yang dapat menggunakan strategi dynamic programming yaitu adalah mencari total biaya optimal. Penelitian Tugas Akhir merepresentasikan persoalan mencari total biaya minimum persoalan Cloud Computing pada situs Sphere Online Judge. Secara umum persoalan yang akan diselesaikan pada penelitian Tugas Akhir ini adalah bagaimana cara melakukan komputasi dengan pemrograman dinamis convex hull.

Tugas akhir ini mengulas penggunaan strategi algoritma pemrograman dinamis convex hull, yang digunakan untuk menyelesaikan permasalahan mencari total biaya minimum untuk membeli semua server yang memenuhi kebutuhan dengan jenis server yang terbatas. Lalu, dengan memanfaatkan strategi ini, permasalahan dapat diselesaikan dengan cukup efisien, yaitu dengan kompleksitas $\mathcal{O}(nlogn)$ *dimana* n *merepresentasikan jumlah server yang harus dibeli.*

Dari hasil uji coba pada studi kasus yang diberikan, didapat hasil bahwa algoritma pemrograman dinamisconvex hull memiliki efisiensi kinerja baik. Rata-rata waktu hasil uji coba adalah $0,06$ *detik. Sementara rata-rata memori hasil uji coba adalah* $4,4$ *MB.*

Kata Kunci: convex hull, pemrograman dinamis, total biaya minumum

[*Halaman ini sengaja dikosongkan*]

IMPLEMENTATION OF DYNAMIC PROGRAMMING WITH THE CONVEX HULL TECHNIQUE AS AN OPTIMIZATION STEP IN THE SEARCH FOR OPTIMAL SOLUTIONS TO THE PROBLEM OF SPOJ CLOUDMG: CLOUD COMPUTING

Name	:	Bella Septina Ika Hartanti
Student ID	:	05111740000117
Department	:	Department of Informatics, Faculty of Intelligent Electrical and Informatics Technology, ITS
Supervisor I	:	Rully Soelaiman, S.Kom., M.Kom.
Supervisor II	:	Ir. M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil.

Abstract

Dynamic Programming is one of the most widely applied optimization techniques to solve problems. The implementation of this strategy in real life is commonly used in finding solutions to problems of optimal routes, graphs, computer networks, etc. This thesis can be represented as a problem of finding the minimum total cost of cloud computing problems on the Sphere Online Judge site. In general, the problem that will be resolved in this thesis is how to do computation with convex hull dynamic programming.

This thesis represented a problem of finding the minimum total cost of cloud computing problems on the Sphere Online Judge site, i.e., the problem of buying all necessary servers against the limited server type. Convex hull dynamic programming strategy has been used to solve well and efficient this problem, i.e., by the complexity O (n log n) where n represents the number of servers that have to be purchased.

From the test results in the given case study, it has been shown that the convex hull dynamic programming algorithm has good performance efficiency. The average tests time used was 0.06 seconds, and the average memory used was 4.4 MB.

Keywords: convex hull, dynamic programming, total minimum cost

KATA PENGANTAR

Puji syukur penulis ucapkan kepada Tuhan Yang Maha Esa. Atas rahmat dan karunia-Nya, penulis dapat menyelesaikan tugas akhir beserta laporannya dalam bentuk buku tugas akhir ini.

Pengerjaan buku ini penulis tujukan untuk mengeksplorasi lebih mendalam topik-topik yang tidak diwadahi oleh kampus, namun banyak menarik perhatian penulis. Selain itu besar harapan penulis bahwa pengerjaan tugas akhir sekaligus pengerjaan buku ini dapat menjadi salah satu poin permulaan penulis dalam menimba ilmu serta memberikan manfaat kepada orang lain.

Penulis ingin menyampaikan rasa terima kasih kepada banyak pihak yang telah membimbing, menemani dan membantu penulis selama masa pengerjaan tugas akhir maupun masa studi.

1. Terima kasih yang sebesar-besarnya kepada Allah SWT, di mana penulis masih diberi berbagai nikmat dan kelancaran sehingga penulis dapat menempuh kuliah di Departemen Informatika ITS dengan baik hingga saat pengerjaan tugas akhir ini.
2. Kedua orang tua, kakak dan keluarga penulis yang selalu memberikan dukungan, perhatian, kasih sayang serta doa dan restu bagi penulis yang menjadi semangat selama perkuliahan hingga pengerjaan tugas akhir.
3. Bapak Rully Soelaiman S.Kom.,M.Kom., selaku dosen pembimbing. Ucapan terima kasih penulis sampaikan atas ilmu, nasehat, dan motivasi yang diberikan beliau hingga saat pengerjaan tugas akhir.
4. Bapak M. M. Irfan Subakti, S.Kom., M.Sc.Eng., M.Phil. selaku dosen pembimbing. Ucapan terima kasih penulis sampaikan atas bantuan dan masukan yang bermanfaat dari proses Pra-Thesis hingga Thesis.

5. Wulan Rosa Panggalih selaku sahabat saya yang telah menemani dan selalu menyemangati selama masa perkuliahan dan proses pengerjaan tugas akhir.
6. Teman-teman angkatan 2017 Departemen Teknik Informatika ITS, khususnya teman-teman bidang minat Algoritma dan Pemrograman, yang senantiasa membantu, menemani, memberi semangat, dan meninggalkan kenangan selama kurang lebih 4 tahun masa studi penulis.
7. Seluruh pihak yang tidak bisa penulis sebutkan satu-persatu yang telah memberikan dukungan selama penulis menyelesaikan tugas akhir ini.

Penulis menyadari bahwa buku ini jauh dari kata sempurna. Maka dari itu, penulis memohon maaf apabila terdapat salah kata maupun makna pada buku ini. Akhir kata, penulis berharap dapat memberikan kontribusi dan manfaat melalui buku ini.

Surabaya, Juni 2021

Bella Septina Ika Hartanti

DAFTAR ISI

KATA PENGANTAR . . . . . **xiii**
DAFTAR ISI . . . . . **xv**
DAFTAR GAMBAR . . . . . **xix**
DAFTAR TABEL . . . . . **xxi**
DAFTAR PSEUDOCODE . . . . . **xxiii**
DAFTAR KODE SUMBER . . . . . **xxv**
DAFTAR NOTASI . . . . . **xxvii**
BAB I PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan 3
1.5 Manfaat 3
1.6 Metodologi 4
1.6.1 Penyusunan proposal Tugas Akhir 4
1.6.2 Studi literatur 4
1.6.3 Analisis dan desain perangkat lunak 4
1.6.4 Implementasi algoritma 4
1.6.5 Pengujian dan evaluasi 5
1.6.6 Penyusunan buku 5
1.7 Sistematika Penulisan 5
BAB II DASAR TEORI **7**
2.1 Deskripsi Permasalahan SPOJ CLOUDMG . . . 7
2.2 Penjelasan dan Pemodelan Matematis Permasalahan CLOUDMG 8
2.3 Pemrograman Dinamis Convex Hull 10
2.4 Analisis Submasalah Optimal pada Permasaahan SPOJ CLOUDMG 12
2.5 Definisi dan Notasi 12
2.6 Penyelesaian Permasalahan SPOJ CLOUDMG . . 13
2.7 Penerapan Pemrograman Dinamis untuk Menggabungkan Submasalah 13
2.8 Penerapan Pemrograman Dinamis dengan Convex Hull 14

2.9 Contoh Penyelesaian Permasalahan SPOJ CLOU-DMG 15
BAB III DESAIN DAN ANALISIS 29
3.1 Desain Umum Sistem 29
3.1.1 Desain Tipe Data Line 29
3.1.2 Desain Fungsi Valid 31
3.1.3 Desain Fungsi Value 32
3.1.4 Desain Fungsi InsertLine 32
3.1.5 Desain Fungsi SlideLine 33
3.1.6 Desain Fungsi Main 34
BAB IV IMPLEMENTASI 37
4.1 Lingkungan Implementasi 37
4.2 Rancangan Data 37
4.2.1 Data Masukan 38
4.2.2 Data Keluaran 39
4.3 Penggunaan Header, Preprocessor, dan Variabel Global 39
4.3.1 Header yang Diperlukan 39
4.3.2 Preprocessor 40
4.3.3 Variabel Global 40
4.4 Implementasi Desain Algoritma 40
4.4.1 Implementasi Struct Line 41
4.4.2 Implementasi Fungsi valid 41
4.4.3 Implementasi Fungsi value 41
4.4.4 Implementasi Fungsi InsertLine 42
4.4.5 Implementasi Fungsi SlideLine 42
4.4.6 Implementasi Fungsi Main 43
BAB V UJI COBA DAN ANALISIS 45
5.1 Lingkungan Uji Coba 45
5.2 Skenario Uji Coba 46
5.3 Uji Coba Kebenaran 46
5.4 Uji Coba Kinerja Lokal 47
5.5 Uji Coba Kinerja Luar 49
5.6 Analisis dan Kesimpulan Umum 50
BAB VI KESIMPULAN DAN SARAN 53
6.1 Kesimpulan 53
6.2 Saran 53
DAFTAR PUSTAKA 55

LAMPIRAN A: DATA UJI **57**
LAMPIRAN B: Hasil Uji Coba pada Situs SPOJ Sebanyak 10 Kali **65**
BIODATA PENULIS **67**

[*Halaman ini sengaja dikosongkan*]

DAFTAR GAMBAR

Gambar 2.1 Deskripsi soal 7
Gambar 2.2 Ilustrasi Permasalahan 9
Gambar 2.3 Ilustrasi Convex Hull 11
Gambar 2.4 Hasil Convex Hull dengan k=1 16
Gambar 2.5 Hasil Convex Hull dengan k=2 17
Gambar 2.6 Ilustrasi k=3, x=7 18
Gambar 2.7 Ilustrasi k=3, x=6 19
Gambar 2.8 Ilustrasi k=3, x=5 20
Gambar 2.9 Ilustrasi k=3, x=5 21
Gambar 2.10 Ilustrasi k=3, x=4 22
Gambar 2.11 Ilustrasi k=3, x=3 23
Gambar 2.12 Ilustrasi k=3, x=2 24
Gambar 2.13 Ilustrasi k=3, x=1 25
Gambar 2.14 Hasil Convex Hull dengan k=3 26

Gambar 5.1 Hasil Uji Coba Kebeneran Situs SPOJ dengan strategi algoritma Pemrograman Dinamis *Convex Hull* 46
Gambar 5.2 Rata-rata *Running Time* untuk Setiap Kasus Uji dengan nilai L tetap 47
Gambar 5.3 Rata-rata *Running Time* untuk Setiap Kasus Uji dengan nilai K tetap 48
Gambar 5.4 Rata-rata *Running Time* untuk Setiap Kasus Uji dengan nilai K dan L berbanding lurus . . 49
Gambar 5.5 Grafik Waktu Hasil Pengumpulan Kode Berkas Sebanyak 10 Kali 50
Gambar 5.6 Grafik Memori Hasil Pengumpulan Kode Berkas Sebanyak 10 Kali 50

Gambar B.1 Hasil Pengumpulan Kode Program Utama . . 65

[*Halaman ini sengaja dikosongkan*]

DAFTAR TABEL

Tabel 2.1 Hasil Final Tabel $lines$ untuk iterasi k=3 14
Tabel 2.2 Hasil Final Tabel $lines$ untuk iterasi k=1 15
Tabel 2.3 Tabel DP dengan k=1 15
Tabel 2.4 Hasil Final Tabel $lines$ untuk iterasi k=2 16
Tabel 2.5 Tabel DP dengan k=2 17
Tabel 2.6 Hasil Final Tabel $lines$ untuk iterasi k=3, x=7 . 18
Tabel 2.7 Hasil Final Tabel $lines$ untuk iterasi k=3, x=6 . 19
Tabel 2.8 Hasil Final Tabel $lines$ untuk iterasi k=3, x=5 . 20
Tabel 2.9 Hasil Final Tabel $lines$ untuk iterasi k=3, x=5 . 21
Tabel 2.10 Hasil Final Tabel $lines$ untuk iterasi k=3, x=4 . 22
Tabel 2.11 Hasil Final Tabel $lines$ untuk iterasi k=3, x=3 . 23
Tabel 2.12 Hasil Final Tabel $lines$ untuk iterasi k=3, x=2 . 24
Tabel 2.13 Hasil Final Tabel $lines$ untuk iterasi k=3, x=1 . 25
Tabel 2.14 Hasil Final Tabel $lines$ untuk iterasi k=3 26
Tabel 2.15 Tabel DP dengan k = 3 26

Tabel 3.1 Nama dan Fungsi Variabel dalam Tipe Data LINE 30
Tabel 3.2 Masukan, Proses, dan Keluaran dari Fungsi VALID 31
Tabel 3.3 Masukan, Proses, dan Keluaran dari Fungsi VALUE 32
Tabel 3.4 Masukan, Proses, dan Keluaran dari Fungsi INSERTLINE 33
Tabel 3.5 Masukan, Proses, dan Keluaran dari Fungsi SLIDELINE 34
Tabel 3.6 Nama dan Fungsi Variabel dalam *Fungsi* MAIN 34

Tabel 4.1 Contoh masukan permasalahan 38
Tabel 4.2 Contoh keluaran permasalahan 39

Tabel A.1 Data Uji dengan Nilai n = 40(1) 57
Tabel A.2 Data Uji dengan Nilai n = 40(2) 58
Tabel A.3 Data Uji dengan Nilai n = 50(1) 59
Tabel A.4 Data Uji dengan Nilai n = 50(2) 60
Tabel A.5 Data Uji dengan Nilai n = 60(1) 61
Tabel A.6 Data Uji dengan Nilai n = 60(2) 62

Tabel A.7 Data Uji dengan Nilai n = 60(3) 63

DAFTAR PSEUDOCODE

Pseudocode 3.1 Tipe Data LINE 30
Pseudocode 3.2 Prosedur CONSTRUCTOR pada *tipe data* LINE 30
Pseudocode 3.3 Fungsi Valid 31
Pseudocode 3.4 Fungsi Value 32
Pseudocode 3.5 Fungsi InsertLine 33
Pseudocode 3.6 Fungsi SlideLine 33
Pseudocode 3.7 Fungsi MAIN 35

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE SUMBER

Kode Sumber 4.1 *Header* yang diperlukan 39
Kode Sumber 4.2 *Preprocessor* yang diperlukan 40
Kode Sumber 4.3 *Variabel global* yang diperlukan 40
Kode Sumber 4.4 Struct Line 41
Kode Sumber 4.5 Fungsi valid 41
Kode Sumber 4.6 Fungsi value 42
Kode Sumber 4.7 Fungsi InsertLine 42
Kode Sumber 4.8 Fungsi SlideLine 42
Kode Sumber 4.9 Fungsi main 43

[*Halaman ini sengaja dikosongkan*]

DAFTAR NOTASI

$\sum$ Notasi yang digunakan untuk menjumlahkan sejumlah bilangan terurut dengan aturan tertentu.

Δ Notasi yang digunakan untuk menghitung selisih antara dua nilai dengan aturan tertentu.

[*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 buku tugas akhir.

1.1 Latar Belakang

Pemrograman dinamis, seperti metode *divide and conquer*, menyelesaikan permasalahan dengan cara membagi permasalahan utama menjadi sub:permasalahan. Pemrograman dinamis secara umum diterapkan pada permasalahan optimasi [1]. Salah satu permasalahan optimasi adalah mencari total biaya optimal.

Topik tugas akhir yang diusulkan penulis berfokus pada permasalahan di Online Judge SPOJ dengan kode soal CLOUDMG. Permasalahan ini diawali dengan deskripsi tentang Cloud, Inc. ,penyedia IaaS (*Infrastructure as a Service*), ingin mendesain pusat data baru dengan server untuk memenuhi kebutuhan klien. Berdasarkan pengamatan, diperlukan sejumlah K server guna menangani kebutuhan klien berdasarkan level. Sedangkan diketahui bahwa harga server berbanding lurus dengan level kebutuhan klien. Agar tidak menyulitkan administrator sistem, konfigurasi *hardware* hanya diperbolehkan menggunakan sejumlah maksimum L tipe server yang berbeda. Diasumsikan apabila suatu server dapat memenuhi kebutuhan klien sebesar n, maka server tersebut dapat memenuhi kebutuhan klien yang lebih kecil dari n. Namun perlu diperhatikan bahwa setiap server hanya dapat digunakan oleh satu klien saja.

Permasalahan yang dihadapi adalah mencari total biaya minimum untuk membeli server sejumlah K untuk memenuhi seluruh permintaan kebutuhan klien dengan jumlah maksimal L tipe server berbeda yang dapat dibeli. Solusi permasalahan yang baik akan didapat dengan menggunakan pemrograman dinamis dengan teknik

convex hull. Hal ini dikarenakan batas waktu perangkat lunak solusi berjalan atau *time limit* yang relatif kecil, yaitu maksimal 0.189 detik. Persoalan dari *time limit* ini juga menjadi kemenarikan dari permasalahan SPOJ CLOUDMG: *Cloud Computing*.

Hasil tugas akhir ini diharapkan dapat memberikan gambaran mengenai implementasi strategi algoritma pemrograman dinamis dengan teknik *convex hull* untuk menemukan solusi dari permasalahan mencari total biaya optimal dengan kinerja yang baik dan dapat memberikan kontribusi pada perkembangan ilmu pengetahuan dan teknologi informasi.

1.2 Rumusan Masalah

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

1. Bagaimana cara menentukan total biaya minimum untuk membeli server untuk permasalahan SPOJ CLOUDMG : *Cloud Computing*?
2. Bagaimana analisis dan desain solusi algoritma pemrograman dinamis teknik *convex hull* untuk menyelesaikan permasalahan SPOJ CLOUDMG : *Cloud Computing*?
3. Bagaimana cara implementasi desain algoritma pemrograman dinamis teknik *convex hull* untuk menyelesaikan permasalahan SPOJ CLOUDMG : *Cloud Computing*?

1.3 Batasan Masalah

Terdapat dua topik batasan dalam tugas akhir, yaitu batasan tugas akhir dan batasan yang tertera di SPOJ. Adapun batasan dari tugas akhir adalah sebagai berikut.

1. Permasalahan yang dibahas terbatas pada solusi optimal dari studi kasus SPOJ CLOUDMG : *Cloud Computing*.

2. Metode penyelesaian menggunakan algoritma pemrograman dinamis dengan teknik *convex hull*.

Sedangkan batasan yang tertera di SPOJ adalah sebagai berikut.

1. Implementasi algoritma menggunakan bahasa C++.
2. Batas jumlah server yang dapat dibeli (K) adalah $1 \leq K \leq 2000$.
3. Batas maksimal jumlah tipe server berbeda yang dapat dibeli adalah $1 \leq L \leq K$.
4. Batas kebutuhan klien terhadap server (D) adalah $1 \leq D \leq 2000$.
5. Batas harga setiap server yang memenuhi kebutuhan klien adalah $1 \leq D \leq 100000$.
6. Batas maksimum perangkat lunak berjalan adalah 0.189 detik.
7. Batas memori perangkat lunak adalah 1536 MB.
8. Batas ukuran program adalah 50kB.

1.4 Tujuan

Tujuan dari tugas akhir adalah sebagai berikut.

1. Melakukan analisis dan mendesain algoritma pemrograman dinamis dengan teknik *convex hull* untuk mencari solusi dari permasalahan total biaya minimum dari soal SPOJ CLOUDMG : *Cloud Computing*.
2. Melakukan implementasi algoritma pemrograman dinamis dengan teknik *convex hull* untuk mencari solusi dari permasalahan total biaya minimum dari soal SPOJ CLOUDMG : *Cloud Computing*.

1.5 Manfaat

Manfaat yang diperoleh dari tugas akhir ini adalah membantu memahami penggunaan algoritma pemrograman dinamis dengan teknik *convex hull* untuk mencari solusi optimal dari permasalahan SPOJ CLOUDMG : *Cloud Computing*.

1.6 Metodologi

Metodologi pengerjaan yang digunakan pada Tugas Akhir ini memiliki beberapa tahapan. Tahapan:tahapan tersebut adalah sebagai berikut.

1.6.1 Penyusunan proposal Tugas Akhir

Tahap awal untuk memulai pengerjaan tugas akhir adalah menyusun proposal tugas akhir. Dalam penyusunannya, penulis mengusulkan gagasan yang akan digunakan untuk menyelesaikan permasalah total biaya minimum pada studi kasus SPOJ CLOUDMG : *Cloud Computing* dengan algoritma pemrograman dinamis dengan teknik *convex hull*.

1.6.2 Studi literatur

Tahap kedua adalah mencari sumber referensi berupa studi literatur maupun informasi yang relevan dengan penyelesaian tugas akhir. Adapun sumber referensi didapat dari buku, jurnal ilmiah, artikel internet, dan materi kuliah yang berkaitan.

1.6.3Analisis dan desain perangkat lunak

Tahap ini dilakukan dengan analisis dan desain yang sesuai dengan tujuan yang telah dijabarkan.

1.6.4 Implementasi algoritma

Tahap implementasi meliputi implementasi algoritma pemrograman dinamis dengan teknik *convex hull* pada perangkat lunak. Implementasi akan dilakukan dengan menggunakan bahasa pemrograman C++.

1.6.5 Pengujian dan evaluasi

Tahap pengujian dan evaluasi dilakukan dengan menggunakan *grader* dari SPOJ dan perangkat lunak akan diuji dengan kasus uji coba yang telah tersedia. Evaluasi dan perbaikan dilakukan hingga perangkat lunak yang diuji memenuhi semua kasus uji coba dan menunjukkan kinerja yang baik.

1.6.6 Penyusunan buku

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan

Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. BAB I : Pendahuluan
   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.
2. BAB II : Dasar Teori
   Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir ini.
3. BAB III : Desain
   Bab ini menjelaskan desain algoritma yang akan dibangun berdasarkan dasar teori dan digunakan dalam penyelesaian.
4. BAB IV : Implementasi
   Bab ini membahas implementasi dari perancangan yang telah dibuat pada bab sebelumnya. Terdapat juga penjelasan berupa kode yang digunakan untuk proses implementasi.
5. BAB V : Uji Coba dan Analisis

Bab ini membahas tahapan uji coba, kemudian hasil uji coba di-evaluasi dan dianalisis terhadap kinerja dari algoritma yang di-implementasikan.

6. BAB VI : Kesimpulan dan Saran
   Bab ini merupakan bab yang menjelaskan kesimpulan dari hasil uji coba yang dilakukan, masalah:masalah yang dialami pada proses dan tertulis saat pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

BAB II
DASAR TEORI

Pada bab ini, akan dijelaskan dasar teori yang digunakan sebagai landasaan pengerjaan tugas akhir ini. Pertama akan dijelaskan mengenai deskripsi permasalahan. Kemudian, pembahasan mengenai analisis soal beserta dengan beberapa dasar teori yang dapat digunakan akan dijabarkan.

2.1 Deskripsi Permasalahan SPOJ CLOUDMG

SPOJ CLOUDMG : *Cloud Computing* [4], merupakan suatu permasalahan yang terdapat pada situs penilaian Sphere Online Judge (SPOJ). Berikut adalah deskripsi soal dari sumber asli menggunakan bahasa Inggris, lihat Gambar 2.1.

CLOUDMG - Cloud Computing

no tags

The concept of cloud computing has become fairly popular lately. One of the main kinds of cloud computing is known as IaaS (*Infrastructure as a Service*), in which servers can be rented and managed via the Internet.

Cloud, Inc. is an IaaS provider. The company is designing a new data center for its clients. Through some research, they found out that their clients, as a whole, need K servers, each of which needs to be able to withstand a certain level of demand. If we assume that the cost of a server always grows with the demand which that server can process, the best solution in terms of cost is to buy K servers explicitly designed to meet the exact requirements of the corresponding client.

However, having K different hardware configurations in the data center is extremely problematic for the system administrators. To simplify, they demand that no more than L distinct server types be bought. A server that meets a demand c also meets any demand smaller than c.

A possible solution is to buy only one server type that is able to meet the highest client demand, as it will also be able to meet all other demands. On the other hand, the cost of such a solution can be prohibitively high. Considering that you can buy up to L different kinds of servers, there's likely a better option. For instance, suppose that there are 3 clients, with demands 3, 7 and 16. Assume that the cost of a server meeting demands up to 3 is of R$ 1500 (1500 brazilian reais), the cost of a server that meets demand 7 is R$ 5500 and the cost of a server that meets demand 16 is R$ 19200. If you want to buy at most 2 kinds of servers to satisfy all clients, you have four possibilities:

- Buying three servers with capacity 16, at a total cost of R$ 57600;
- Buying two servers with capacity 16 and one with capacity 7, at a total cost of R$ 43900;
- Buying two servers with capacity 16 and one with capacity 3, at a total cost of R$ 39900;
- Buying one server with capacity 16 and two servers with capacity 7, at a total cost of R$ 30200.

Among these options, the one with the least total cost is the last one.

You will receive a list of K requested demands and the price of a server type that meets the corresponding demand. Determine the lowest total cost to buy K servers in a way such that all requested demands are met and at most L different types of servers are bought.

Gambar 2.1 Deskripsi soal

Permasalahan tugas akhir yang diusulkan ini diawali dengan deskripsi tentang Cloud, Inc., penyedia IaaS (*Infrastructure as a Service*), ingin mendesain pusat data baru dengan server untuk memenuhi kebutuhan klien. Berdasarkan pengamatan, diperlukan sejumlah K server guna menangani kebutuhan klien berdasarkan level. Sedangkan diketahui bahwa harga server berbanding lurus dengan level kebutuhan klien. Agar tidak menyulitkan administrator sistem, konfigurasi *hardware* hanya diperbolehkan menggunakan sejumlah maksimum L tipe server yang berbeda.

Diasumsikan apabila suatu server dapat memenuhi kebutuhan klien sebesar n, maka server tersebut dapat memenuhi kebutuhan klien yang lebih kecil dari n. Namun perlu diperhatikan bahwa setiap server hanya dapat digunakan oleh satu klien saja. Permasalahan yang dihadapi adalah mencari total biaya minimum untuk membeli server sejumlah K untuk memenuhi seluruh permintaan kebutuhan klien dengan jumlah maksimal L tipe server berbeda yang dapat dibeli.

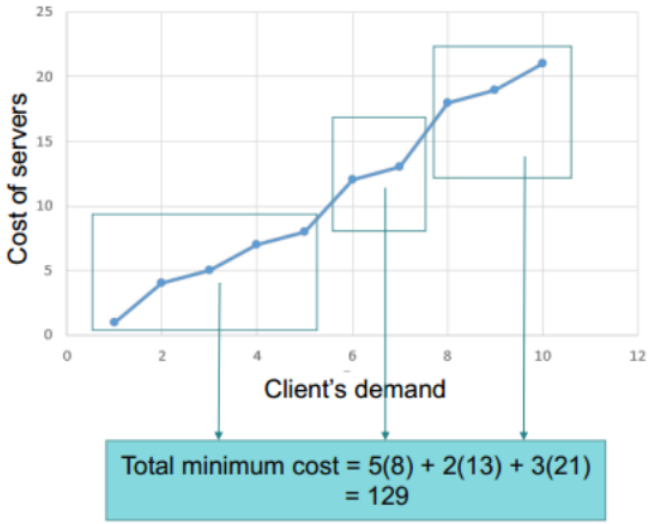
Permasalahan ini dibatasi dengan jumlah server yang dapat dibeli (K) adalah $1 \leq K \leq 2000$, maksimal jumlah tipe server berbeda yang dapat dibeli adalah $1 \leq L \leq K$, kebutuhan klien terhadap server (D) adalah $1 \leq D \leq 2000$, dan harga setiap server yang memenuhi kebutuhan klien adalah $1 \leq D \leq 100000$. Sedangkan batas waktu maksimal adalah 0.189 detik. Hasil akhir yang didapatkan berupa total biaya minimum untuk membeli server.

2.2 Penjelasan dan Pemodelan Matematis Permasalahan CLOUDMG

Permasalahan di atas menceritakan kebutuhan suatu perusahaan terhadap server untuk memenuhi kebutuhan klien dengan memperhatikan biaya yang harus dianggarkan untuk membeli server baru serta jenis server yang harus dibeli. Namun dengan catatan bahwa biaya total harus bernilai minimum dan jenis server yang

dapat dibeli dibatasi.

Adapun contoh penggambaran permasalahan ditunjukkan pada gambar 2.2.



Gambar 2.2 Ilustrasi Permasalahan

Sesuai dengan contoh permasalahan di atas, maka hasil ilustrasi pengelompokan kebutuhan klien dengan jenis server yang dapat memenuhi. Dengan contoh tersebut, maka terdapat tiga jenis server yang diperlukan untuk menghasilkan biaya total minimum.

Berdasarkan deskripsi permasalahan, variabel keputusan yang diperoleh dapat didefinisikan seperti di bawah ini.

x_i : jumlah server tipe i yang akan dibeli.

c_i : harga server tipe i.

$$a_i = \begin{cases} 1, & \text{apabila server tipe } i \text{ dipilih} \\ 0, & \text{sebaliknya} \end{cases}$$

Sedemikian hingga, permasalahan diatas dapat dimodelkan dalam bentuk matematika dalam persamaan 2.1.

$$
\begin{aligned}
&Minimize \sum_{i=1}^{K} c_i a_i \\
&subject\, to \\
&\sum_{i=1}^{K} a_i x_i = K \\
&\sum_{i=1}^{K} a_i \leq L
\end{aligned} \tag{2.1}
$$

2.3 Pemrograman Dinamis Convex Hull

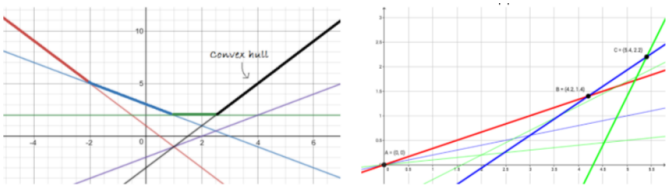
Pemrograman dinamis adalah pendekatan optimasi yang mengubah masalah kompleks menjadi urutan masalah yang lebih sederhana. Karakteristik dasarnya adalah sifat multistage dari prosedur optimasi. Pemrograman dinamis juga menyediakan kerangka kerja umum untuk menganalisis banyak jenis masalah. Dalam kerangka ini berbagai teknik optimasi dapat digunakan untuk menyelesaikan aspek-aspek tertentu dari formulasi yang lebih umum. Berikut adalah karakteristik dari permasalahan yang dapat diselesaikan dengan pemrograman dinamis:

- Adanya *optimal substructure*, yaitu suatu permasalahan dikatakan memiliki substruktur optimal apabila solusi optimal untuk keseluruhan permasalahan mencakup solusi optimal untuk sub-permasalahan.

- Adanya *overlapping subproblem*, yaitu ketika algoritma rekursif mengunjungi sub-permasalahan yang sama secara berulang [1].

Salah satu teknik optimasi pemrograman dinamis adalah teknik *convex hull*. Teknik ini dapat didefinisikan dengan suatu teknik yang digunakan untuk menentukan anggota suatu set fungsi linear suatu variabel yang mencapai nilai ekstrem untuk suatu nilai yang diberikan oleh variabel independen tertunda [2]. Ketika titik-titik ekstrem tersebut saling dihubungkan, maka akan terbentuk daerah atau lereng *convex* [3]. Agar dapat menggunakan strategi algoritma pemrograman dinamis *convex hull*, ada syarat yang harus terpenuhi, yaitu bersifat monotik.

Dengan pendekatan *convex hull*, *hull* yang menghadap ke atas (*lower hull*) mengindikasikan titik-titik solusi minimal. Sedangkan *hull* yang menghadap ke bawah (*upper hull*) mengindikasikan titik-titik solusi maksimal. Adapun ilustrasi dari *convex hull* ditunjukkan pada gambar 2.3.



Gambar 2.3 Ilustrasi Convex Hull

Untuk memelihara terbentuknya *convex hull* maka diperlukan operasi penggeseran dan penambahan garis.

- Operasi penggeseran garis *(slide)* dilakukan untuk menyeleksi garis-garis yang tidak dapat menghasilkan nilai minimum agar dikeluarkan dalam *hull*.

- Operasi penambahan garis *(insert)* dilakukan untuk menyimpan garis yang dapat memelihara *convex hull* sehingga tetap menghasilkan nilai optimal (paling minimum).

2.4 Analisis Submasalah Optimal pada Permasaahan SPOJ CLOUDMG

Subbab ini menjelaskan submasalah-submasalah yang solusinya dapat memberikan hasil minimum dari pembelian server. Pemrograman dinamis menyelesaikan masalah dengan memecahnya menjadi submasalah. Pemrograman dinamis dapat diterapkan ketika terdapat submasalah yang sama terjadi berulang-ulang. Strategi algoritma pemrograman dinamis memecahkan masing-masing sub-masalah sekali saja kemudian menyimpan jawabannya dalam sebuah tabel, dengan demikian strategi ini dapat mengurangi perhitungan ulang setiap kali menyelesaikan submasalah yang sama. Submasalah yang akan dicari solusinya yaitu mencari nilai total harga minimum yang diperlukan untuk memenuhi klien dengan memperhatikan jenis server yang akan digunakan tidak melebihi batas yang diberikan.

2.5 Definisi dan Notasi

Berikut adalah notasi yang digunakan pada tugas akhir:

1. $DP_{(i,\,k)}$ adalah notasi untuk nilai hasil perhitungan submasalah, di mana simbol i merepresentasikan server ke-i dan simbol k merepresentasikan jumlah jenis server.
2. Iterasi $i.j$ adalah notasi yang menyatakan iterasi perulangan, di mana simbol i merepresentasikan jumlah server maksimum yang dapat dibeli, dan simbol j merepresentasikan server ke-j.
3. $line(l)$ adalah notasi yang menyatakan persamaan garis lurus yang menyimpan nilai kemiringan garis dan total biaya minimum.
4. $cost_{(j)}$ adalah notasi harga server ke-j.

2.6 Penyelesaian Permasalahan SPOJ CLOUDMG

Pada subbab ini akan dibahas mengenai strategi penyelesaian permasalahan *Cloud Computing* dengan kode CLOUDMG. Strategi penyelesaian masalah akan dibagi menjadi 2 tahap, yaitu:

1. Merumuskan total biaya server yang dibeli dengan maksimal L jenis server.
2. Menggabungkan submasalah melalui pemrograman dinamis dengan memperhatikan konsep *convex hull*.

Penyelesaian utama dari tugas akhir ini adalah dengan menerapkan strategi algoritma pemrograman dinamis dengan pendekatan *convex hull* untuk mencapai hasil yang paling optimum dengan efisiensi dalam segi waktu.

2.7 Penerapan Pemrograman Dinamis untuk Menggabungkan Submasalah

Pada bagian ini dibahas mengenai penggabungan submasalah dengan cara pemrograman dinamis. $DP_{(x,y)}$ digunakan untuk menyimpan total biaya minimum guna membeli sebanyak y server dengan maksimal x jenis server. Pada awal inisiasi $DP_{(x,y)}$, seluruh nilai biaya minimum akan bernilai INF kecuali ketika $y = N$ akan bernilai 0. Perhitungan dimulai dari y bernilai N, dan bergerak menuju ke kiri hingga $y = 0$ yang memiliki arti bahwa seluruh kebutuhan klien telah dipenuhi server dimana $DP_{(x,0)}$ mewakili total biaya minimum server yang harus dipenuhi untuk membeli server dengan x jenis server. Setiap iterasinya bergantung pada banyaknya maksimum jenis server atau K.

Untuk menghitung nilai total biaya minimum untuk membeli server digunakan persamaan 2.2.

$$DP_{(k,i)} = DP_{(k-1,j)} + cost_{(j)}(j - i) \tag{2.2}$$

Dari persamaan 2.2 dapat dijabarkan menjadi persamaan 2.3

$$DP_{(k,i)} = DP_{(k-1,j)} + cost_{(j)}(j) - cost_{(j)}(i) \qquad (2.3)$$

Setiap iterasi akan melalui tahap transisi yaitu untuk mencari apakah ada nilai yang lebih kecil dari penggabungan submasalah daripada melakukan penambahan garis ke dalam *hull*.

2.8 Penerapan Pemrograman Dinamis dengan Convex Hull

Untuk menggunakan pendekatan *convex hull*, persamaan 2.3 dapat didefinisikan dalam bentuk persamaan garis yang ditunjukkan pada tabel 2.1.

Tabel 2.1 Hasil Final Tabel $lines$ untuk iterasi k=3

Notasi persamaan garis	Notasi persamaan rekurens
$m_{(j)}$	$cost_{(j)}$
x	i
$c_{(j)}$	$DP_{(k-1,j)}$ + $cost_{(j)}$(j)

Perhatikan bentuk khusus dari $m_{(j)}x$ + $c_{(j)}$ pada persamaan tabel 2.1. Hal ini identik dengan persamaan garis lurus dengan gradien $m_{(j)}$ dan titik potong $c_{(j)}$. Jadi, permasalahan ini dapat diasumsikan sama dengan diberikannya serangkaian garis dan ditanyakan nilai y minimum yang dapat diberikan oleh setiap garis tersebut pada x tertentu. Jika diilustrasikan, dari kumpulan garis lurus pada sebuah bidang akan didapat bahwa nilai minimum berada di sepanjang daerah *convex hull*.

Konsep dari *convex hull* adalah setiap garis di *hull* memberikan nilai minimum pada beberapa rentang nilai x yang berdekatan. Sebaliknya, setiap baris yang tidak berada *hull* tidak berguna karena tidak pernah memberikan nilai minimum. Semua garis di *hull* yang *convex* memiliki kemiringan yang berbeda. Urutan lereng juga menentukan posisinya *convex hull*. Sebuah garis dengan kemiringan

yang lebih rendah muncul pada *convex hull* di sebelah kiri dari lereng yang lebih tinggi.

Dalam soal ini kemiringan garis, $m_{(j)}$ dimisalkan oleh $-p_{(j)}$. Karena sifat dari pembatas setelah diurutkan berdasarkan harga server-j dengan meningkatkan p sehingga bersifat monoton naik.

2.9 Contoh Penyelesaian Permasalahan SPOJ CLOUDMG

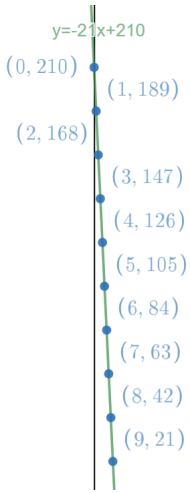
Pada subbab ini akan dibahas mengenai tahap penyelesaian permasalahan dari salah satu contoh soal. Contoh kasus adalah nilai K=10 dan L=3 serta diikuti nilai D dan P untuk 10 baris berturut-turut adalah [(1,1), (2,4), (3,5), (4,7), (5,8), (6,12), (7,13), (8,18), (9,19), (10,21)]. Iterasi dimulai dari tingkat paling bawah yaitu ketika k = 0 sebagai awal inisiasi pada tabel DP hingga $k = K$ yang merupakan jumlah jenis server yang tersedia. Adapun hasil dari kumpulan garis yang dapat membentuk *convex hull* sehingga menghasilkan nilai biaya minimum untuk k = 1 ditunjukkan pada tabel 2.2. Sedangkan hasil tabel DP untuk k = 1 ditunjukkan pada tabel 2.3.

Tabel 2.2 Hasil Final Tabel $lines$ untuk iterasi k=1

Iterasi k=1
$left$=3 , $right$=6
(-21,210)
(-1,INF)

Tabel 2.3 Tabel DP dengan k=1

	0	1	2	3	4	5	6	7	8	9	10
0	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	0
1	210	189	168	147	126	105	84	63	42	21	INF



Gambar 2.4 Hasil Convex Hull dengan k=1

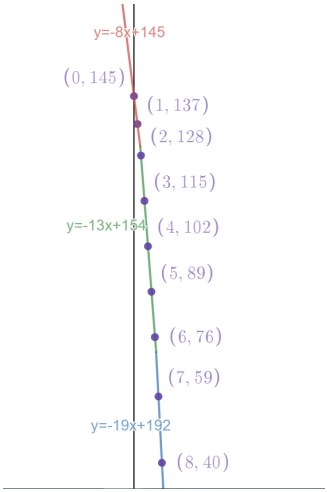
Apabila melihat dari hasil tabel 2.3., nilai minimum untuk $DP_{(j,0)}$ berada pada $DP_{(10,0)}$ (warna hijau) yaitu bernilai 0. Sehingga dengan menggunakan nilai tersebut, untuk mengisi tiap $DP_{(i,1)}$ cukup dengan menambahkan nilai $DP_{(10,0)}$ dengan perkalian dari selisih nilai j dan i dengan $cost_{(10)}$. Hingga setelah iterasi akan diperoleh bahwa total biaya minimum yang diperoleh untuk membeli 10 server dengan k = 1 adalah $DP_{(0,0)}$sebesar 210.

Tabel 2.4 Hasil Final Tabel $lines$ untuk iterasi k=2

Iterasi k=2
$left$=3 , $right$=6
(-21,INF)
(-19,192)
(-13,154)
(-8,145)
(-5,162)
(-1,190)

Tabel 2.5 Tabel DP dengan k=2

	0	1	2	3	4	5	6	7	8	9	10
0	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	0
1	210	189	168	147	126	105	84	63	42	21	INF
2	145	137	128	115	102	89	76	59	40	INF	INF



Gambar 2.5 Hasil Convex Hull dengan k=2

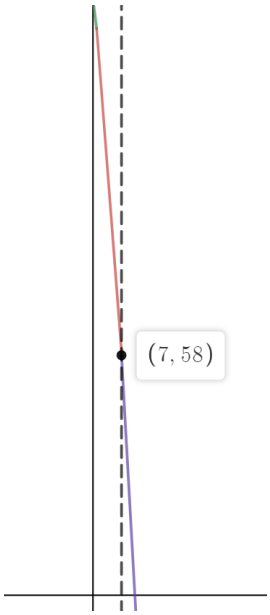
Pada tabel 2.5, terlihat bahwa untuk nilai $DP_{(i,\,1)}$ dengan 0 $\leq i < n$ diperoleh dengan mengalikan $cost_{(j)}$ dengan selisih $i \leq j < n$. Untuk mendapatkan $DP_{(0,\,2)}$, maka dicari nilai minimum dari $DP_{(j,\,1)}$ dimana $1 \leq j < n$. Berdasarkan tabel, nilai minimum diperoleh dengan menjumlahkan nilai $DP_{(5,\,1)}$ dengan perkalian dari selisih nilai j dan i yaitu 5 dengan $cost_{(5)}$. Sehingga diperoleh nilai minimum sebesar 145. Adapun kumpulan garis yang membentuk *convex hull* untuk k = 2 ditunjukkan pada tabel 2.4.

Sedangkan untuk k=3, detail hasil array yang memuat garis-garis pembentuk *convex hull* ditunjukkan dari tabel 2.6 hingga tabel

2.14. Sedangkan ilustrasi dari proses iterasi untuk membentuk *convex hull* akan ditampilkan dari gambar 2.6 hingga gambar 2.14.

Tabel 2.6 Hasil Final Tabel $lines$ untuk iterasi k=3, x=7

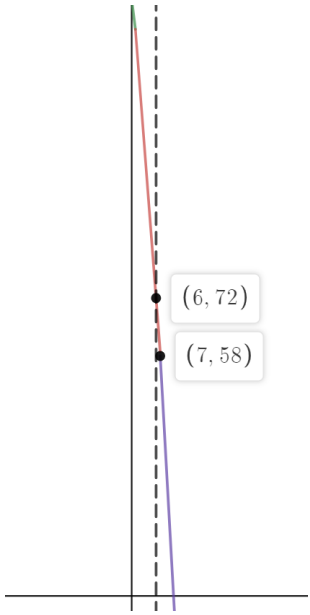
Iterasi k=3, x=7
$left$=1 , $right$=2
(-21,INF)
(-18,184)
(-13,154)
(-8,145) (updated)
(-5,162)
(-1,190)



Gambar 2.6 Ilustrasi k=3, x=7

Tabel 2.7 Hasil Final Tabel $lines$ untuk iterasi k=3, x=6

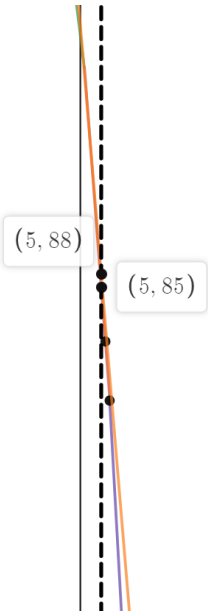
Iterasi k=3, x=6
$left$=2 , $right$=3
(-21,INF)
(-18,184)
(-13,154)
(-8,145)
(-5,162)
(-1,190)



Gambar 2.7 Ilustrasi k=3, x=6

Tabel 2.8 Hasil Final Tabel $lines$ untuk iterasi k=3, x=5

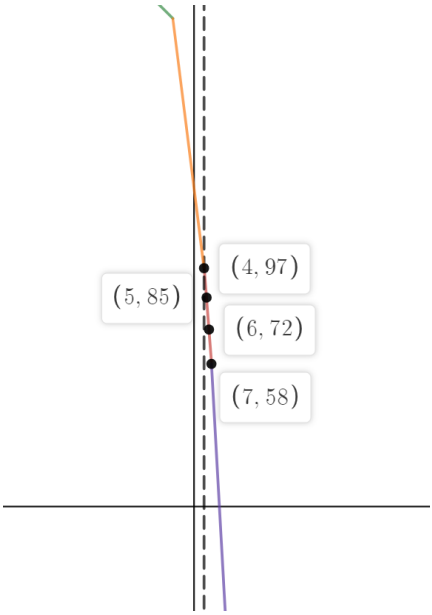
Iterasi k=3, x=5
$left$=3 , $right$=6
(-21,INF)
(-18,184)
(-13,150) (updated)
(-8,145)
(-5,162)
(-1,190)



Gambar 2.8 Ilustrasi k=3, x=5

Tabel 2.9 Hasil Final Tabel $lines$ untuk iterasi k=3, x=5

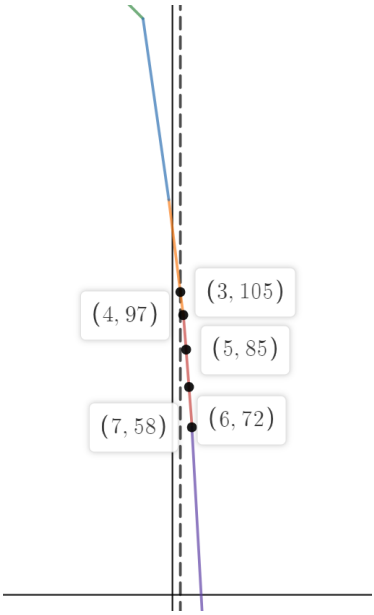
Iterasi k=3, x=5
$left$=2 , $right$=4
(-21,INF)
(-18,184)
(-13,150)
(-12,148) (updated)
(-5,162)
(-1,190)



Gambar 2.9 Ilustrasi k=3, x=5

Tabel 2.10 Hasil Final Tabel $lines$ untuk iterasi k=3, x=4

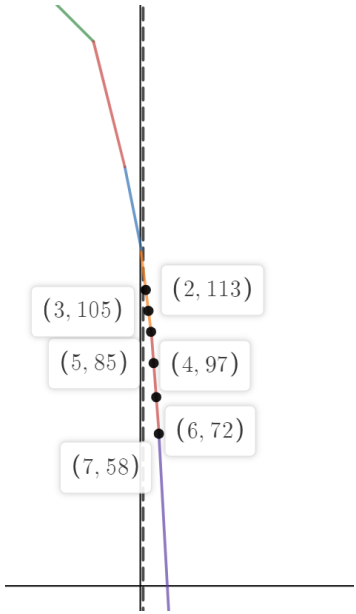
Iterasi k=3, x=4
$left$=3 , $right$=4
(-21,INF)
(-18,184)
(-13,150)
(-8,129)
(-5,162) (updated)
(-1,190)



Gambar 2.10 Ilustrasi k=3, x=4

Tabel 2.11 Hasil Final Tabel $lines$ untuk iterasi k=3, x=3

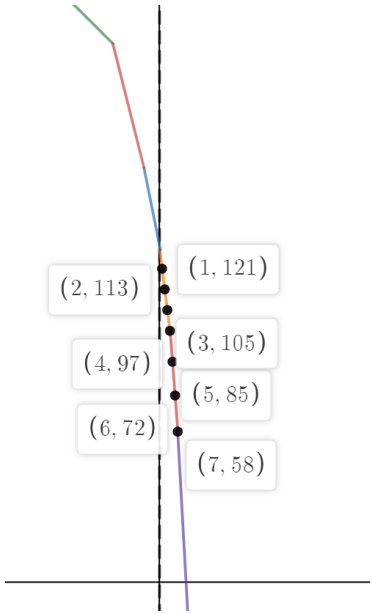
Iterasi k=3, x=3
$left$=3 , $right$=5
(-21,INF)
(-18,184)
(-13,150)
(-8,129) (updated)
(-5,162)
(-1,190)



Gambar 2.11 Ilustrasi k=3, x=3

Tabel 2.12 Hasil Final Tabel $lines$ untuk iterasi k=3, x=2

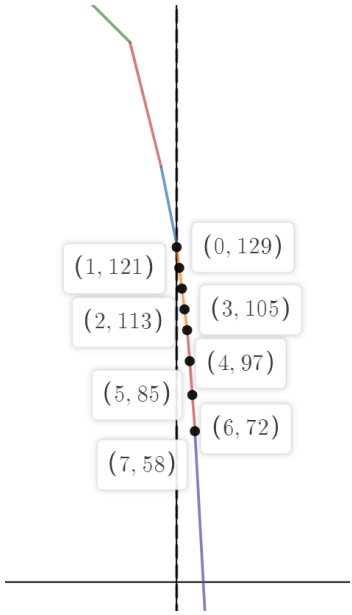
Iterasi k=3, x=2
$left$=3 , $right$=5
(-21,INF)
(-18,184)
(-13,150)
(-8,129)
(-5,130) (updated)
(-1,190)



Gambar 2.12 Ilustrasi k=3, x=2

Tabel 2.13 Hasil Final Tabel $lines$ untuk iterasi k=3, x=1

Iterasi k=3, x=1
$left$=3 , $right$=6
(-21,INF)
(-18,184)
(-13,150)
(-8,129)
(-5,130)
(-4,136) (updated)



Gambar 2.13 Ilustrasi k=3, x=1

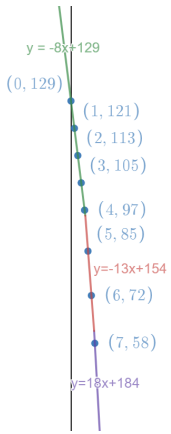
Tabel 2.14 Hasil Final Tabel $lines$ untuk iterasi k=3

Iterasi k=3
$left$=3 , $right$=6
(-21,INF)
(-18,184)
(-13,150)
(-8,129)
(-5,136)
(-1,138)

Tabel 2.15 Tabel DP dengan k = 3

	0	1	2	3	4	5	6	7	8	9	10
0	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	0
1	210	189	168	147	126	105	84	63	42	21	INF
2	145	137	128	115	102	89	76	59	40	INF	INF
3	129	121	113	105	97	85	72	58	INF	INF	INF

Gambar 2.14 Hasil Convex Hull dengan k=3



Pada gambar diatas, terlihat bahwa untuk $DP_{(i,1)}$ dengan 0 $\leq i < n$ diperoleh dengan mengalikan $cost_{(j)}$ dengan selisih $i \leq j < n$n. Untuk mendapatkan $DP_{(0,3)}$, maka dicari nilai minimum dari $DP_{(j,2)}$ dimana $1 \leq j < n$. Berdasarkan tabel, nilai minimum diperoleh dengan menjumlahkan nilai $DP_{(5,2)}$ dengan perkalian dari selisih nilai j dan i yaitu 5 dengan $cost_{(5)}$. Sehingga diperoleh nilai minimum sebesar 129.

[*Halaman ini sengaja dikosongkan*]

BAB III
DESAIN DAN ANALISIS

Pada bab ini akan dijelaskan desain dan analisis algoritma yang akan digunakan untuk menyelesaikan permasalahan pada tugas akhir ini.

3.1 Desain Umum Sistem

Pada subbab ini akan dijelaskan mengenai gambaran secara umum dari algoritma yang dirancang. Sistem menerima masukan berupa 2 bilangan yaitu N yang merepresentasikan jumlah server yang diperlukan untuk memenuhi kebutuhan tiap klien dan K yang merepresentasikan maksimal jumlah jenis server yang boleh dibeli. Kemudian, masukan diikuti sebanyak N pasangan d dan p yang masing-masing mengindikasikan *demand* atau kebutuhan klien dan harga server yang memenuhi kebutuhan klien. Nilai maksimal dari bilangan N adalah sebesar 2000. Masukan akan berhenti pada kasus uji ketika $N = 0$ dan $K = 0$. Selanjutnya, masukan d dan p akan diurutkan berdasarkan nilai d secara $ascending$. Lalu sistem memproses masukan yang akan diubah dalam representasi persamaan garis dan dilakukan komputasi dengan pemrograman dinamis *convex hull*. Pada pemrograman dinamis, nilai d dan biaya total yang mengacu pada persamaan akan dimemoisasi pada sebuah tabel DP. Pada akhirnya, didapat keluaran sistem yaitu total biaya minimum yang digunakan untuk membeli seluruh N server yang dapat memenuhi kebutuhan klien.

3.1.1 Desain Tipe Data Line

Tipe data LINE adalah sebuah tipe struktur data yang menyimpan koefisien variabel ft, variabel sc, constructor LINE, dan fungsi bool operator $<$. Adapun fungsi masing-masing variabel di-

tunjukkan pada tabel 3.1. dan pseudocode dari tipe data LINE ditunjukkan pada *pseudocode* 3.1.

Tabel 3.1 Nama dan Fungsi Variabel dalam Tipe Data LINE

Nama Variabel	Fungsi Variabel
ft	Menyimpan nilai ft untuk nilai $cost$ yang akan dihitung
sc	Menyimpan nilai sc untuk nilai total $cost$ yang akan dihitung

Pseudocode 3.1 Tipe Data LINE

```
1: TYPE LINE
2:  ft ← Integer
3:  sc ← Integer
4:  constructor LINE(_ft, _sc)
5:  procedure OPERATOR <(ft, sc)
6: END TYPE
```

Fungsi $constructor$ dari tipe data ini hanya terdiri dari satu jenis. Fungsi $constructor$ adalah fungsi dengan parameter $_ft$ dan $_sc$, menyatakan nilai ft dan sc secara berurutan. *Pseudocode procedure* CONSTRUCTOR ditunjukkan pada *pseudocode* 3.2.

Pseudocode 3.2 Prosedur CONSTRUCTOR pada *tipe data* LINE

```
Input: _ft, _sc
1: ft ← _ft
2: cs ← _sc
```

Sedangkan fungsi operator bool $<$ digunakan untuk membandingkan nilai ft sekarang dengan yang baru.

3.1.2 Desain Fungsi Valid

Fungsi Valid digunakan untuk melakukan validasi antar tiga garis yang merupakan representasi dari struct tipe LINE. Ketiga garis yang saling berpotongan akan divalidasi. Apabila perhitungan rasio masih memenuhi, maka akan mengembalikan nilai $True$ dan $False$ apabila sebaliknya.

Pseudocode 3.3 Fungsi Valid

Input: line a, b, c

1: **if** $(a.sc - c.sc)(b.ft - a.ft) < (a.sc - b.sc)(c.ft - a.ft)$ **then**
2: **return** $True$
3: **else**
4: **return** $False$
5: **end if**

Tabel 3.2 Masukan, Proses, dan Keluaran dari Fungsi VALID

Masukan	Proses	Keluaran
Line a, b, c	Melakukan komputasi perbandingan antara ketiga baris terhadap kondisi perpotongan antar garis yang terbentuk. Perhitungan mengikuti bentuk persamaan gradien pada persamaan garis	$True$ atau $False$

3.1.3 Desain Fungsi Value

Fungsi Value digunakan untuk menghitung nilai biaya atau $cost$ sementara yang tersimpan dalam struct tipe LINE. Nilai biaya sementara tersebut dihitung dengan menjumlahkan nilai biaya atau $cost$ pada struct tipe LINE ($curr.sc$) dengan biaya beli server ($curr.ft$) dengan banyaknya klien (x).

Pseudocode 3.4 Fungsi Value

Input: $curr, x$

```
if curr.sc == INF then
    return INF
else
    return (curr.ft)(x) + curr.sc
end if
```

Tabel 3.3 Masukan, Proses, dan Keluaran dari Fungsi VALUE

Masukan	Proses	Keluaran
line $curr$, x	Melakukan komputasi untuk pembaruan nilai $curr.sc$	$curr.sc$ terbaru

3.1.4 Desain Fungsi InsertLine

Fungsi InsertLine digunakan untuk menambahkan garis baru ke dalam array. Persamaan garis akan diperiksa apakah akan menghasilkan nilai biaya yang minimum atau tidak. Apabila menghasilkan nilai minimum, garis tersebut akan dimasukkan ke dalam array.

Pseudocode 3.5 Fungsi InsertLine

Input: $curr, r, l$

```
while r − l > 1 and !valid(lines[r − 2], lines[r − 1], curr) do
    decrement r
end while
return r
```

Tabel 3.4 Masukan, Proses, dan Keluaran dari Fungsi INSERTLINE

Masukan	Proses	Keluaran
line $curr, r, l$	Melakukan operasi memasukkan garis atau tidak dengan memperhatikan posisi garis dan nilai validasi	nilai r terbaru

3.1.5 Desain Fungsi SlideLine

Fungsi SlideLine digunakan untuk memilih persamaan garis yang telah tersimpan dalam array.Persamaan garis akan diperiksa apakah akan menghasilkan nilai biaya yang minimum atau tidak.

Pseudocode 3.6 Fungsi SlideLine

Input: $curr, x$

```
while l < r−1 and value(lines[l], x) > value(lines[l+1], x)
do
    increment l
end while
return l
```

Tabel 3.5 Masukan, Proses, dan Keluaran dari Fungsi SLIDELINE

Masukan	Proses	Keluaran
line $curr, r, l$	Melakukan operasi penggeseran garis dengan memperhatikan posisi garis dan value	nilai l terbaru

3.1.6 Desain Fungsi Main

Pseudocode fungsi MAIN ditunjukkan pada *pseudocode* 3.7. Pada *pseudocode* 3.7, fungsi MAIN merupakan bagian fungsi utama yang menerima masukan, memproses dan menampilkan masukan. Fungsi INPUT merupakan fungsi untuk menerima masukan, fungsi SORT(ARR, ARR+N) merupakan fungsi $sorting$ builtin, dan fungsi PRINT merupakan fungsi untuk menampilkan hasil.

Tabel 3.6 Nama dan Fungsi Variabel dalam *Fungsi* MAIN

Nama Variabel	Fungsi Variabel
n	Menyimpan nilai banyaknya klien
k	Menyimpan nilai maksimal jenis server
DP	Kontainer 2 dimensi penyimpan nilai biaya minimum
$CurSc$	Menyimpan nilai bilangan $CurSc$ yang digunakan untuk pembuatan garis
r	Menyimpan nilai bilangan r sebagai indikator garis-garis pada ruas kanan
l	Menyimpan nilai bilangan l sebagai indikator garis-garis pada ruas kiri
$lines$	Kontainer 1 dimensi penyimpan garis pembentuk *convex hull*
arr	Kontainer 1 dimensi penyimpan input harga dan level kebutuhan klien

Pseudocode 3.7 Fungsi MAIN

```
while 1 do
    n ← INPUT()
    k ← INPUT()
    for i ← 0 to n − 1 do
        arr[i].ft ← INPUT()
        arr[i].sc ← INPUT()
    end for
    SORT(ARR, ARR+N)
    DP(n, 0) ← 0
    for i ← 0 to n − 1 do
        DP(i, 0) ← INF
    end for
    for rem ← 0 to k do
        a ← a^1
        DP(n, a) ← 0
        l ← 0
        r ← 0
        CurSc ← INTEGER
        cur ← LINE
        for x ← n − 1 to 1 do
            if DP(x + 1, a^1) == INF then
                CurSc ← INF
            else
                CurSc ← DP(x + 1, a^1) + arr[x].sc(x + 1)
            end if
            cur ← line(−arr[x].sc, CurSc)
            r ← InsertLine(l, r, cur)
            l ← SlideLine(l, r, x)
            DP(x, a) ← value(lines(l), x);
        end for
    end for
    print DP(0, a)
end while
```

[*Halaman ini sengaja dikosongkan*]

BAB IV
IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai implementasi dari desain algoritma penyelesaian permasalahan mencari biaya minimum total untuk membeli server dengan jenis yang dapat dibeli maksimal sebanyak L pada soal *Cloud Computing* di situs SPOJ. Pertama akan dijelaskan implementasi secara umum, dilanjutkan dengan implementasi dari kedua metode yang digunakan.

4.1 Lingkungan Implementasi

Lingkungan implementasi dan pengembangan yang dilakukan adalah sebagai berikut.

1. Perangkat Keras
   (a) Processor Intel® Core™ i7-6700HQ CPU @ 2.60GHz (8 CPUs), 2.6GHz
   (b) *Random Access Memory* 8192MB
2. Perangkat Lunak
   (a) Sistem Operasi Windows 10 Education 64-bit
   (b) Visual Studio Code
   (c) Bahasa Pemrograman C++
   (d) *Compiler* GCC 7.4.0 (Ubuntu 7.4.0-1ubuntu1 18.04.1) untuk Windows Subsystem Linux

4.2 Rancangan Data

Subbab ini menjelaskan mengenai desain data masukan yang diperlukan untuk melakukan proses algoritma serta data keluaran yang dihasilkan oleh program.

4.2.1 Data Masukan

Data masukan adalah data yang akan diproses oleh program sebagai masukan menggunakan struktur data yang telah dirancang dan akan digunakan dalam algoritma pemrograman dinamis *convex hull* sebagai penyelesaian permasalahan ini. Jenis data masukan dapat dilihat di bawah ini.

1. Masukan berupa bilangan bulat N dan K yang masing-masing merepresentasikan jumlah server yang diperlukan untuk memenuhi kebutuhan tiap klien dan maksimal jumlah jenis server yang boleh dibeli. Nilai N dan K memiliki rentang hingga 2000 buah. Masukan akan berhenti bila nilai $N = 0$ dan $K = 0$.
2. Masukkan berupa pasangan bilangan dari rentang [1,1000] yang masing-masing merepresentasikan d (server yang memenuhi kebutuhan klien) dan p (harga server) sebanyak N.

Adapun contoh masukan dari permasalahan terdapat pada Tabel 4.1.

Tabel 4.1 Contoh masukan permasalahan

8 2
2 4
3 6
4 7
5 9
6 12
7 13
8 18
10 22
0 0

4.2.2 Data Keluaran

Data keluaran yang dihasilkan oleh program berupa bilangan integer yang merepresentasikan total biaya server minimum yang diperlukan untuk memenuhi kebutuhan klien dengan maksimal jenis server yang diperbolehkan sebersar K. Adapun contoh keluaran dari permasalahan terdapat pada Tabel 4.2.

Tabel 4.2 Contoh keluaran permasalahan

122

4.3 Penggunaan Header, Preprocessor, dan Variabel Global

Subbab ini menjelaskan mengenai header, preprocessor, dan variabel global yang digunakan dalam sistem.

4.3.1 Header yang Diperlukan

Implementasi algoritma ini membutuhkan tiga buah *header* yaitu `cstdio`, `cstring`, dan `algorithm` seperti yang terlihat pada kode sumber 4.1.

```
1 #include <cstdio>
2 #include <cstring>
3 #include <algorithm>
```

Kode Sumber 4.1 *Header* yang diperlukan

Pada baris pertama, *header* `cstdio` berisi fungsi standar input output operasi yang digunakan untuk menerima masukan dan menampilkan keluaran. Pada baris kedua, *header* `cstring` digunakan untuk memanggil `memset`. Pada baris ketiga, *header* `algorithm` digunakan untuk melakukan *sorting* dengan memanggil fungsi `std::sort`.

4.3.2 Preprocessor

Pre-processor seperti `define` digunakan untuk membuat alias dari tipe data atau fungsi sesungguhnya. Terdapat dua alias yang digunakan yaitu `INF` yang menandakan nilai tak terbatas dengan nilai defaut sebesar 1000000006 dan `getchar_unlocked` sebagai `gc`. Pre-processor dapat dilihat pada kode sumber 4.2.

```
1 using namespace std;
2
3 #define INF 1000000006
4 #define gc getchar_unlocked
5 #define ll long long
```

Kode Sumber 4.2 *Preprocessor* yang diperlukan

4.3.3 Variabel Global

Variabel global digunakan untuk memudahkan dalam mengakses data yang digunakan antar fungsi, seperti pada Kode Sumber 4.2. Variabel $lines[]$ digunakan untuk menyimpan nilai ft dan sc yang nantinya akan digunakan untuk merepresentasikan garis dan variabel $DP[]$ digunakan untuk menyimpan hasil komputasi pemrograman dinamis.

```
1 struct line arr[2001];
2 struct line lines[2001];
3 int DP[2001][2];
```

Kode Sumber 4.3 *Variabel global* yang diperlukan

4.4 Implementasi Desain Algoritma

Subbab ini menjelaskan implementasi proses algoritma secara keseluruhan berdasarkan desain yang telah dijelaskan pada Bab 3. Program ini merupakan program yang digunakan untuk menyelesaikan permasalahan total biaya minimum pada studi kasus SPOJ CLOUDMG: *Cloud Computing*.

4.4.1 Implementasi Struct Line

Struct Line adalah implementasi tipe data struktur yang digunakan untuk menyelesaikan masalah sesuai pseudocode dari subbab 3.2.1. Struct menyimpan koefisien variabel ft dan sc dengan tipe data Integer. Selain itu, struct juga memuat prosedur operator < untuk membandingkan properti variabel ft antar variabel bertipe data struct line.

```
1 struct line{
2   int ft, sc;
3   line(){}
4   line(int ft, int sc):ft(ft),sc(sc){}
5   bool operator < (line rhs) const{ return ft <
        rhs.ft; }
6 };
```

Kode Sumber 4.4 Struct Line

4.4.2 Implementasi Fungsi valid

Fungsi valid digunakan untuk menyeleksi garis yang akan dimasukkan dalam *convex hull*. Fungsi ini menyeleksi perpotongan antara tiga garis. Apabila memenuhi kondisi maka garis yang terbaru akan dimasukkan ke dalam array yang merepresentasikan garis pembentuk *hull*.

```
1 bool valid(line a, line b, line c){
2   return (ll)(a.sc-c.sc)*(b.ft-a.ft) <
3          (ll)(a.sc-b.sc)*(c.ft-a.ft);
4 }
```

Kode Sumber 4.5 Fungsi valid

4.4.3 Implementasi Fungsi value

Fungsi value digunakan untuk menghitung biaya minimum total terbaru. Nilai akan tetap bernilai INF apabila $curr.sc$ (to-

tal biaya minimum sementara bernilai INF. Sedangkan apabila $curr.sc$ sebelumnya tidak bernilai INF akan diperbarui.

```
1 int value(line curr, int x){
2     return curr.sc==INF?INF:curr.ft*x+curr.sc;
3 }
```

Kode Sumber 4.6 Fungsi value

4.4.4 Implementasi Fungsi InsertLine

Fungsi InsertLine digunakan untuk menambahkan garis baru ke dalam array. Persamaan garis akan diperiksa apakah akan menghasilkan nilai biaya yang minimum valid. Apabila menghasilkan nilai minimum, garis tersebut akan disimpan.

```
1 int InsertLine(int l, int r, line cur){
2   while(r-l>1 &&
        !valid(lines[r-2],lines[r-1],cur)) r--;
3   lines[r++] = cur;
4   return r;
5 }
```

Kode Sumber 4.7 Fungsi InsertLine

4.4.5 Implementasi Fungsi SlideLine

Fungsi SlideLine digunakan untuk memilih persamaan garis yang tersimpan dalam array. Nantinya, persamaan garis akan diperiksa berdasarkan nilai l dan r. Apabila menghasilkan nilai minimum, garis tersebut akan dipertahankan dalam array.

```
1 int SlideLine(int l, int r, int x){
2   while(l<r-1 && value(lines[l],x) >
        value(lines[l+1],x)) l++;
3   return l;
4 }
```

Kode Sumber 4.8 Fungsi SlideLine

4.4.6 Implementasi Fungsi Main

Fungsi main merupakan implementasi algoritma yang dirancang pada *pseudocode* 3.7. Implementasi fungsi main dapat dilihat pada kode sumber 4.9.

```
1  int main(){
2    while(1){
3      char _c;
4      int n, k, a = 0;
5      getInt(n) getInt(k)
6      if(n==0&&k==0) return 0;
7      for(int i = 0; i < n; i++ ){
8        getInt(arr[i].ft) getInt(arr[i].sc)
9      }
10     std::sort(arr,arr+n);
11
12     DP[n][0] = 0;
13     for(int i = 0; i < n; i++ ) DP[i][0] = INF;
14
15     for(int rem = 1; rem <= k; rem++ ){
16       a ^= 1;
17       DP[n][a] = INF;
18       int l = 0, r = 0;
19       int cur_cs;
20
21       for(int x = n-1; x >= 0; x-- ){
22         (DP[x+1][a^1]==INF)? cur_cs = INF: cur_cs
               = DP[x+1][a^1]+arr[x].sc*(x+1);
23         line cur = line(-arr[x].sc,cur_cs);
24         r = insert_line(l, r, cur);
25         l = slide_line(l,r,x);
26         DP[x][a] = value(lines[l],x);
27       }
28     }
29     printf("%d\n",DP[0][a]);
30   }
31
32   return 0;
33 }
```

Kode Sumber 4.9 Fungsi main

[*Halaman ini sengaja dikosongkan*]

BAB V
UJI COBA DAN ANALISIS

Pada bab ini dijelaskan tentang uji coba dan analisis dari implementasi yang telah dilakukan pada Tugas Akhir ini. Uji coba dilakukan dengan skenario yang tertentu dan dilanjutkan dengan tahap evaluasi dari metode penyelesaian yang digunakan.

5.1 Lingkungan Uji Coba

Lingkungan uji coba yang digunakan untuk uji kebenaran pada situs penilaian daring SPOJ, yaitu kluster *Cube* dengan spesifikasi sebagai berikut.

1. Perangkat Keras
   (a) Processor Intel Xeon E3-1220 v5 (5 CPUs)
   (b) *Random Access Memory* 1536MB
2. Perangkat Lunak
   (a) Compiler GCC 6.3.0

Lingkungan uji coba yang digunakan untuk melakukan uji coba kinerja menggunakan komputer pribadi penulis yang memiliki spesifikasi sebagai berikut

1. Perangkat Keras
   (a) Processor Intel® Core™ i7-6700HQ CPU @ 2.60GHz (8 CPUs), 2.6GHz
   (b) *Random Access Memory* 8192MB
2. Perangkat Lunak
   (a) Sistem Operasi Windows 10 Education 64-bit
   (b) Visual Studio Code
   (c) Bahasa Pemrograman C++

(d) Compiler GCC 7.4.0 (Ubuntu 7.4.01ubuntu1 18.04.1) untuk Windows Subsystem Linux

5.2 Skenario Uji Coba

Subbab ini akan menjelaskan hasil pengujian program untuk menyelesaikan permasalahan CLOUDMG-*Cloud Computing*. Metode pengujian yang dilakukan adalah sebagai berikut.

1. Pengujian luar. Pengujian ini menggunakan *Online Judge* untuk menguji kebenaran dan kinerja program.
2. Pengujian lokal. Pengujian ini menggunakan mesin yang digunakan dalam pengembangan untuk mengukur kinerja program.

Dalam pengujian lokal, dibuat beberapa kasus uji berdasarkan batasan pada subbab 1.3 dengan menggunakan nilai random.

Untuk pengujian luar, uji coba dilakukan dengan mengirimkan kode program ke situs penilaian daring SPOJ masing-masing sebanyak 10 kali.

5.3 Uji Coba Kebenaran

Pada subbab ini akan dibahas mengenai uji coba kebenaran yang dilakukan dengan mengirim kode sumber terkait ke dalam situs penilaian daring SPOJ.

Berikut bukti hasil pengujian sumber kode menggunakan algoritma Pemrograman Dinamis *Convex Hull*.

1. Hasil uji coba kebenaran kode sumber (Gambar 5.1)

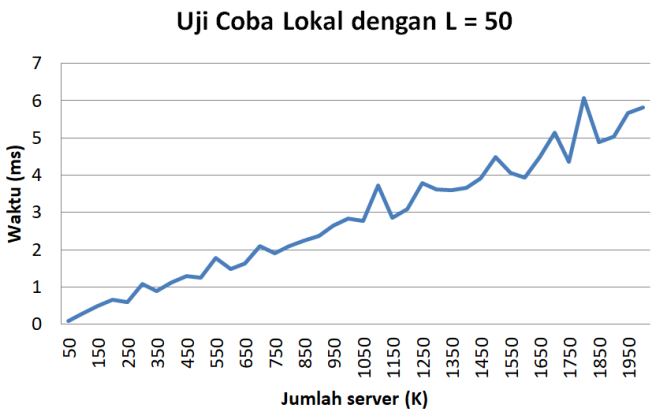
27308264	■	2021-01-20 14:01:03	Cloud Computing	**accepted** edit ideone it	0.04	4.5M	CPP14

Gambar 5.1 Hasil Uji Coba Kebeneran Situs SPOJ dengan strategi algoritma Pemrograman Dinamis *Convex Hull*

5.4 Uji Coba Kinerja Lokal

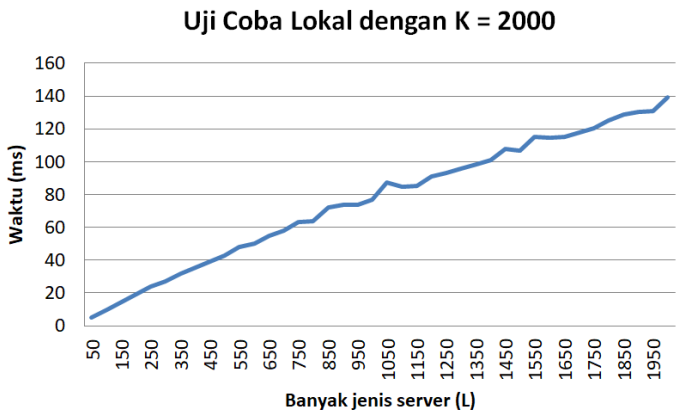
Pada subbab ini akan ditampilkan hasil uji coba kinerja dari kode program untuk menyelesaikan permasalahan CLOUDMG: *Cloud Computing*. Pengujian dilakukan terhadap kelompok masukan yang telah dijelaskan pada subbab 5.2. Detil masukan dapat dilihat pada Lampiran A. Langkah pengujian kinerja dilakukan dengan langkah sebagai berikut.

1. Rekam waktu tepat sebelum komputasi penyelesaian masalah dilakukan.
2. Melakukan komputasi penyelesaian masalah untuk masukan kasus uji.
3. Rekam waktu tepat setelah komputasi penyelesaian masalah dilakukan.
4. Menghitung durasi waktu komputasi dengan mengurangi waktu selesai komputasi dengan waktu sebelum komputasi.
5. Ulangi untuk seluruh kasus uji.



Gambar 5.2 Rata-rata *Running Time* untuk Setiap Kasus Uji dengan nilai L tetap

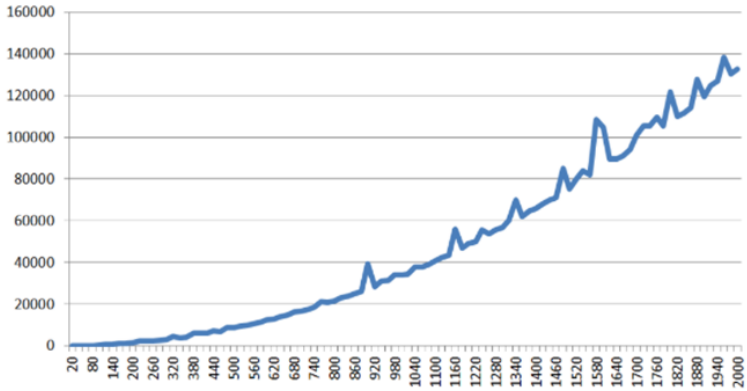
Grafik 5.2 menampilkan rata-rata kinerja dengan melakukan implementasi pemrograman dinamis *convex hull* dengan banyaknya jenis server yang konstan yaitu $l = 50$. Dapat diambil kesimpulan bahwa komputasi yang dihasilkan meningkat secara linear.



Gambar 5.3 Rata-rata *Running Time* untuk Setiap Kasus Uji dengan nilai K tetap

Grafik 5.3 menampilkan rata-rata kinerja dengan melakukan implementasi pemrograman dinamis *convex hull* dengan banyaknya jenis server yang konstan yaitu $k = 2000$. Dapat diambil kesimpulan bahwa komputasi yang dihasilkan meningkat secara linear.

Sedangkan grafik 5.4 menampilkan rata-rata kinerja dengan melakukan implementasi pemrograman dinamis *convex hull* dengan banyaknya jenis server dan jumlah server yang diperlukan berbanding lurus. Dapat diperoleh kesimpulan bahwa nilai K dan L berbanding lurus menghasilkan komputasi yang meningkat secara linearithmik.

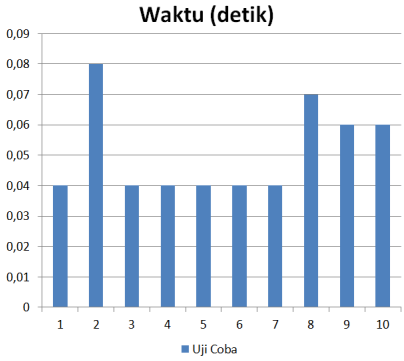


Gambar 5.4 Rata-rata *Running Time* untuk Setiap Kasus Uji dengan nilai K dan L berbanding lurus

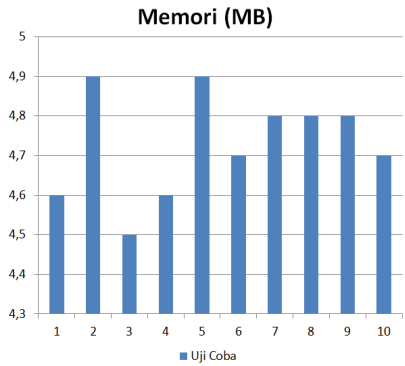
5.5 Uji Coba Kinerja Luar

Pada subbab ini akan ditampilkan hasil uji coba kinerja dari strategi algoritma Pemrograman Dinamis *Convex Hull* untuk menyelesaikan permasalahan CLOUDMG: *Cloud Computing*. Pengujian dilakukan dengan cara mengirimkan kode program ke situs penilaian daring SPOJ.

Rata-rata hasil pengumpulan kode berkas dengan algoritma pemrograman dinamis *convex hull* adalah $0,051$ detik dengan memori $4,73$ MB. Sedangkan nilai waktu minimal yang bisa dicapai adalah 0.04 detik dengan memori 4.4 MB Hasil uji coba pada situs SPOJ dapat dilihat pada gambar 5.5 dan 5.6



Gambar 5.5 Grafik Waktu Hasil Pengumpulan Kode Berkas Sebanyak 10 Kali



Gambar 5.6 Grafik Memori Hasil Pengumpulan Kode Berkas Sebanyak 10 Kali

5.6 Analisis dan Kesimpulan Umum

Dari hasil uji coba kinerja lokal dan uji coba kinerja luar, dapat disimpulkan bahwa strategi algoritma pemrograman dinamis *convex hull* efektif dan relatif efisien. Dari hasil uji kinerja lokal dengan beberapa skenario uji dan *submission* pada *grader* SPOJ di-

peroleh bahwa hasil output yang dihasilkan sudah sesuai dan benar. Dari segi tingkat efisien, solusi dengan mengimplementasikan strategi algoritma pemrograman dinamis untuk studi kasus *Cloud Computing* mampu menghasilkan rata-rata waktu 0.51 detik dan rata-rata memori sebesar 4.73MB hasil dari *submission* pada *grader* SPOJ. Selain itu juga, hasil grafik dengan uji coba kinerja lokal menunjukkan bahwa komputasi meningkat secara linearithmik.

[*Halaman ini sengaja dikosongkan*]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini dijelaskan mengenai kesimpulan dari hasil uji coba yang telah dilakukan serta saran-saran tentang pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang

6.1 Kesimpulan

Berdasarkan penjabaran di subbab-subbab sebelumnya dan dari hasil uji coba yang dilakukan, dapat disimpulkan beberapa poin terkait penyelesaian permasalahan CLOUDMG: *Cloud Computing*.

1. Permasalahan CLOUDMG: *Cloud Computing* dapat dimodelkan dengan menggunakan strategi algoritma pemrograman dinamis *convex hull* dan terbukti cukup efektif dan efisien.
2. Permasalahan CLOUDMG: *Cloud Computing* dengan batasan pada soal dapat diselesaikan menggunakan strategi algoritma pemrograman dinamis *convex hull* dengan waktu minimal $0,04$ detik, waktu maksimal $0,08$ detik, memori minimal $4,4$ MB, dan memori maksimal $4,7$ MB.

6.2 Saran

Pada tugas akhir ini, tentunya terdapat banyak kekurangan disamping beberapa nilai yang dapat diambil. Di bawah ini adalah saran-saran yang didapat setelah pengerjaan tugas akhir ini:

1. Optimasi yang mungkin dilakukan yaitu dengan cara melakukan penyederhanaan persamaan dan optimasi komputasi proses operasi untuk mempercepat waktu uji.
2. Melakukan optimasi dalam segi penggunaan memori.

[*Halaman ini sengaja dikosongkan*]

DAFTAR PUSTAKA

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest **and** C. Stein, *Introduction to Algorithms*, 3 **edition**. London: MIT Press, 2009.

[2] Z. Galil **and** K. Park, *Dynamic Programming with convexity, concavity, and sparsity**. London: Columbia University, 1992.

[3] G. Saketh. (2016). "Dynamic Programming Optimisation with Convex Hull Trick." Accessed on: 2021-02-18, **url**: https://pclub.in/tutorial/algos/2016/08/22/dpconvex.html.

[4] SPOJ, "Cloud Computing, "2012, Accessed on: 2020-06-18. **url**: https://www.spoj.com/problems/CLOUDMG/.

[*Halaman ini sengaja dikosongkan*]

LAMPIRAN A: DATA UJI

Tabel A.1 Data Uji dengan Nilai n = 40(1)

No	d	p
1	140	140
2	1580	1580
3	772	772
4	1199	1199
5	1332	1332
6	819	819
7	1771	1771
8	686	686
9	1108	1108
10	890	890
11	220	220
12	523	523
13	1167	1167
14	1153	1153
15	575	575
16	1991	1991
17	739	739
18	707	707
19	1471	1471
20	814	814

Tabel A.2 Data Uji dengan Nilai n = 40(2)

No	d	p
21	1858	1858
22	1868	1868
23	291	291
24	989	989
25	271	271
26	956	956
27	1709	1709
28	1766	1766
29	970	970
30	129	129
31	344	344
32	1461	1461
33	60	60
34	1467	1467
35	659	659
36	1743	1743
37	638	638
38	781	781
39	780	780
40	1745	1745

Tabel A.3 Data Uji dengan Nilai n = 50(1)

No	d	p
1	202	207
2	555	557
3	584	589
4	1718	1723
5	1134	1139
6	1012	1017
7	561	566
8	832	837
9	1579	1584
10	1484	1494
11	1447	1457
12	1341	1351
13	1073	1083
14	1250	1260
15	1461	1471
16	753	763
17	1337	1347
18	413	423
19	1946	1956
20	1301	1401
21	1815	1815
22	351	351
23	809	809
24	219	219
25	1152	1152

Tabel A.4 Data Uji dengan Nilai n = 50(2)

No	d	p
26	40	40
27	528	528
28	1169	1169
29	687	687
30	715	715
31	1057	1062
32	1240	1245
33	1618	1623
34	2	2
35	1323	1328
36	751	761
37	1014	1019
38	1888	1893
39	1592	1597
40	949	954
41	1728	1738
42	1043	1053
43	646	656
44	1163	1173
45	654	664
46	116	126
47	277	287
48	10	10
49	538	548
50	232	242

Tabel A.5 Data Uji dengan Nilai n = 60(1)

No	d	p
1	1365	1365
2	661	661
3	532	532
4	310	310
5	844	844
6	1147	1147
7	141	141
8	216	216
9	1722	1722
10	1966	1966
11	290	290
12	1147	1147
13	1836	1836
14	279	279
15	33	33
16	526	526
17	1239	1239
18	805	805
19	410	410
20	301	301

Tabel A.6 Data Uji dengan Nilai n = 60(2)

No	d	p
21	138	138
22	1333	1333
23	1689	1689
24	296	296
25	1029	1029
26	1740	1740
27	886	886
28	1455	1455
29	548	548
30	885	885
31	1319	1319
32	264	264
33	1545	1545
34	1850	1850
35	573	573
36	741	741
37	996	996
38	714	714
39	956	956
40	1070	1070

Tabel A.7 Data Uji dengan Nilai n = 60(3)

No	d	p
41	1031	1031
42	1598	1598
43	216	216
44	1219	1219
45	1876	1876
46	249	249
47	1744	1744
48	1114	1114
49	1405	1405
50	505	505
51	1766	1766
52	1894	1894
53	189	189
54	1454	1454
55	541	541
56	1218	1218
57	1194	1194
58	1426	1426
59	672	672
60	93	93

[*Halaman ini sengaja dikosongkan*]

LAMPIRAN B: Hasil Uji Coba pada Situs SPOJ Sebanyak 10 Kali

Berikut merupakan lampiran hasil uji coba kode program sebanyak 10 kali.

ID		DATE	PROBLEM	RESULT	TIME	MEM	LANG
27389426	☐	2021-02-04 12:55:06	Cloud Computing	accepted edit ideone it	0.06	4.9M	CPP14
27389422	☐	2021-02-04 12:54:34	Cloud Computing	accepted edit ideone it	0.06	4.7M	CPP14
27389417	☐	2021-02-04 12:53:33	Cloud Computing	accepted edit ideone it	0.06	4.8M	CPP14
27308287	☐	2021-01-20 14:04:39	Cloud Computing	accepted edit ideone it	0.07	4.8M	CPP14
27308279	☐	2021-01-20 14:02:56	Cloud Computing	accepted edit ideone it	0.04	4.8M	CPP14
27308270	☐	2021-01-20 14:01:37	Cloud Computing	accepted edit ideone it	0.04	4.7M	CPP14
27308268	☐	2021-01-20 14:01:27	Cloud Computing	accepted edit ideone it	0.04	4.9M	CPP14
27308265	☐	2021-01-20 14:01:13	Cloud Computing	accepted edit ideone it	0.04	4.6M	CPP14
27308264	☐	2021-01-20 14:01:03	Cloud Computing	accepted edit ideone it	0.04	4.5M	CPP14
27308261	☐	2021-01-20 14:00:24	Cloud Computing	accepted edit ideone it	0.08	4.9M	CPP14

Gambar B.1 Hasil Pengumpulan Kode Program Utama

[*Halaman ini sengaja dikosongkan*]

BIODATA PENULIS

Penulis bernama Bella Septina Ika Hartanti, putri terakhir dari empat bersaudara yang lahir pada tanggal 9 September 2000 di Kediri, Jawa Timur. Penulis telah menuntut ilmu di Sekolah Dasar Ngronggo 8 pada tahun 2006 hingga 2012, Sekolah Menengah Pertama Negeri 3 Kediri pada tahun 2012 hingga 2015, dan Sekolah Menengah Atas Negeri 1 Kediri pada tahun 2015 hingga 2017. Pada masa penulisan, penulis sedang menempuh masa studi S1 di Institut Teknologi Sepuluh Nopember, Surabaya di Departemen Teknik Informatika. Selama masa studi, penulis memiliki ketertarikan di bidang *data mining* dan rancang bangun aplikasi. Keinginan penulis dalam membantu sesama juga mendorong penulis menjadi asisten dosen pada beberapa mata kuliah. Di luar kesibukan akademik, penulis juga berkontribusi dalam berbagai kepanitiaan, baik dalam skala kecil (yaitu dalam kampus) maupun skala nasional. Organisasi yang penulis ikuti adalah SRE ITS (2020-2021). Selain itu, penulis juga aktif dalam Schematics sebagai staf pada tahun 2018 dan staf ahli NLC pada tahun 2019. Penulis dapat dihubungi melalui surel di sptnbella09@gmail.com.