**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**USULAN TUGAS AKHIR**

# **IDENTITAS PENGUSUL**

**Nama : Anggakara Hendra Nandana**

**NRP : 5108100075**

**Dosen Wali : Umi Laili Yuhana, S.Kom., M.Sc.**

1. **JUDUL TUGAS AKHIR**

***“Implementasi Algoritma Pencarian k Jalur Sederhana Terpendek Dalam Graf”***

1. **LATAR BELAKANG**

Dalam teori graf, permasalahan jalur terpendek adalah permasalahan menentukan jalur antara dua vertex di dalam sebuah graf berarah dengan bobot terkecil. Jalur dalam graf didefinisikan sebagai sebuah rangkaian vertex dimana untuk masing – masing vertex terdapat sebuah edge yang menuju vertex selanjutnya pada rangkaian tersebut. Bobot dari sebuah jalur merupakan hasil penjumlahan bobot semua edge yang menyusun vertex – vertex pada jalur.

Dalam dunia nyata kadang terdapat sebuah permasalahan yang dapat direpresentasikan dalam graf, dimana harus mencari jalur terpendek, tetapi jalur yang didapat tidak bisa digunakan karena adanya suatu halangan pada jalur tersebut. Hal ini menyebabkan harus memilih alternatif lain berupa jalur terpendek berikutnya, agar biaya yang dikeluarkan tidak memiliki selisih yang terlalu besar dari jalur terpendek pertama. Biaya tersebut bisa berupa waktu, tenaga, atau materi.

Oleh karena itu, pada tugas akhir ini penulis mencoba melakukan studi mengenai algoritma – algoritma untuk menemukan sejumlah *k* jalur sederhana terpendek dalam sebuah graf, serta kemudian mengimplementasi algoritma tersebut menjadi sebuah aplikasi menggunakan bahasa pemrograman C++.

1. **RUMUSAN MASALAH**

Beberapa permasalahan yang dibahas dalam tugas akhir ini dapat diuraikan sebagai berikut:

1. Bagaimana menjelaskan dan membuktikan kebenaran algoritma pencarian *k* jalur sederhana terpendek dalam graf?
2. Bagiamana menggunakan struktur data yang sesuai untuk mengimplementasikan algoritma agar menghemat memori dan waktu saat pengeksekusian program?
3. **BATASAN MASALAH**

Permasalahan yang dibahas dalam tugas akhir ini memiliki batasan – batasan sebagai berikut:

1. Implementasi algoritma pencarian *k* jalur sederhana terpendek dalam graf menggunakan bahasa pemrograman C++ dengan bantuan perangkat lunak Microsoft Visual Studio 2010
2. Pustaka yang digunakan saat mengimplementasi algoritma menggunakan pustaka yang terdapat dalam *Standard Template Library (STL)*
3. **TUJUAN TUGAS AKHIR**

Tujuan dari pengerjaan tugas akhir ini adalah menganalisa algoritma pencarian k jalur sederhana terpendek dalam sebuah graf dan mengimplementasikan algoritma tersebut menjadi sebuah program dengan bahasa C++.

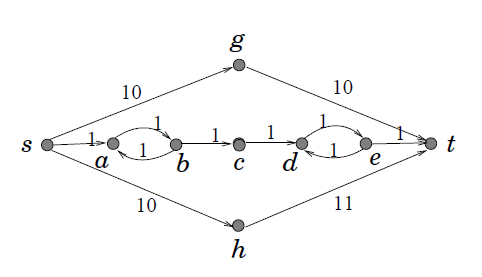
1. **MANFAAT TUGAS AKHIR**

Manfaat yang diharapkan dari hasil tugas akhir ini adalah memberikan referensi untuk mengimplementasi algoritma pencarian *k* jalur sederhana terpendek dalam graf dengan menggunakan struktur data yang sesuai, sehingga nantinya dapat digunakan untuk membuat aplikasi lain yang membutuhkan pencarian k jalur sederhana terpendek.

1. **RINGKASAN TUGAS AKHIR**

Permasalahan pencarian *k* jalur sederhana terpendek dalam sebuah graf merupakan generalisasi permasalahan pencarian jalur terpendek dalam graf, dimana dalam kasus ini tidak hanya diminta untuk menemukan satu jalur terpendek, melainkan beberapa jalur terpendek dalam urutan menanjak berdasarkan bobot jalur. Ketentuan pada permasalahan ini adalah jalur yang akan dicari harus sederhana, yaitu tidak memiliki loop. Pada tahun 1975 sudah ditemukan sebuah algoritma dengan kompleksitas waktu *O(m + kn log n)* untuk menyelesaikan permasalahan pencarian *k* jalur sederhana terpendek dalam graf, kemudian perkembangan terbaru adalah pengoptimasian dari Eppstein dengan kompleksitas waktu *O(m + n log n + k)*.

Pada graf berarah yang terdapat pada gambar 1, tiga jalur sederhana terpendek dari vertex *s* ke vertex *t* yang dapat ditemukan masing – masing memiliki panjang 6 (*s-a-b-c-d-e-t*), 20 (*s-g-h*), dan 21 (*s-h-t*). Jika tidak ada ketentuan bahwa tidak boleh ada loop dalam jalur, maka jalur terpendek kedua yang dapat diperoleh akan memiliki panjang 8, yaitu jalur (*s-a-b-a-b-c-d-e-t*), namun jalur tersebut tidak boleh diambil karena ada loop (*a-b-a*). Dua jalur *A* dan *B* dianggap berbeda apabila ada vertex pada urutan yang sama, yang berbeda pada kedua jalur.



Gambar 1 - Contoh graf berarah

Langkah – langkah yang dilakukan untuk menemukan *k* jalur sederhana terpendek dari vertex sumber *s* ke vertex tujuan *t* adalah sebagai berikut:

* Inisialisasi sebuah struktur jalur bercabang *T* yang berisi vertex tunggal *s*, dan letakkan *jalur*(*s, t)* ke dalam heap. Pada awalnya ada sebuah kelas ekuivalensi *C*(*s*), yang berhubungan dengan semua jalur *s-t*.
* Lakukan perulangan pada langkah – langkah berikut sebanyak *k* kali:
  1. Ekstraksi nilai minimum dari heap. Nilai tersebut adalah milik dari sebuah jalur *P*.
  2. Jika *P* termasuk dalam kelas ekuivalensi *C*(*u*) untuk beberapa vertex *u*, maka
     1. Tambah sebuah cabang (*u*, *tp*) ke *T* yang merepresentasikan sufiks dari *P* setelah *u*
     2. Hapus jalur – jalur dari *C*(*u*) yang memiliki setidaknya satu edge yang sama dengan *P* setelah *u* dan masukkan semua selain *P* ke kelas ekuivalensi baru *C*(*u*, *tp*)
  3. Selain itu (maka *P* termasuk dalam kelas ekuivalensi *C*(*u*, *v*) untuk beberapa cabang (*u*, *v*))
     1. Definisikan *w* adalah vertex dimana *P* terpisah dari cabang pada *branchPath*(*u*, *v*)
     2. Sisipkan sebuah vertex baru yang diberi label *w*, dan pisah cabang (*u*, *v*) menjadi dua cabang (*u, v*) dan (*w*, *v*). Tambahkan cabang kedua (*w*, *tp*) yang merepresentasikan sufiks dari *P* setelah *w*.
     3. Distribusikan kembali jalur pada *C*(*u*, *v*) \ *P* di antara empat kelas baru *C*(*u*, *w*), *C*(*w*, *v*), C(*w*, *tp*), dan *C*(*w*), tergantung pada letak cabang mereka dari branchPath(*u*, *v*) dan / atau *P*.
        1. Jalur yang terpisah pada *branchPath*(*u*, *v*) sebelum node *w* termasuk ke dalam C(*u*, *w*).
        2. Jalur yang terpisah pada *branchPath*(*w*, *v*) setelah node *w* termasuk ke dalam C(*w*, *v*).
        3. Jalur yang terpisah pada *P* sebelum node *w* termasuk ke dalam C(*w*, *tp*).
        4. Jalur yang terpisah pada *P* setelah node *w* termasuk ke dalam C(*w*).
  4. Untuk setiap ekuivalensi kelas baru atau yang telah diganti (maksimal ada empat), hitung jalur terpendek dari *s* ke *t* pada kelas tersebut. Masukkan jalur – jalur tersebut ke dalam heap.

1. **METODOLOGI**

Ada beberapa tahap dalam proses pengerjaan tugas akhir ini, yaitu sebagai berikut:

1. **Studi Literatur**

Pada tahap ini merupakan tahap mempelajari dan menganalisa berbagai macam algoritma pencarian *k* jalur sederhana terpendek dalam graf yang terdapat pada berbagai literatur. Kemudian melakukan analisa mengenai struktur data yang sesuai untuk mengimplementasi algoritma yang terdapat dalam literatur.

1. **Implementasi**

Pada tahap ini dilakukan implementasi dari algoritma yang telah dipelajari menjadi aplikasi. Bahasa pemrograman yang digunakan adalah C++, dengan bantuan perangkat lunak Microsoft Visual Studio 2010 sebagai *Integrated Development Enviroment (IDE).*

1. **Uji coba dan evaluasi**

Pada tahap ini aplikasi akan diuji dengan berbagai macam data masukan untuk menguji kebenaran hasil implementasi algoritma serta menguji waktu eksekusi aplikasi untuk berbagai macam data masukan. Pada tahap ini juga dilakukan optimasi dari hasil implementasi apabila aplikasi masih kurang efisien.

1. **Penyusunan buku tugas akhir**

Tahap ini merupakan penyusunan laporan berupa buku tugas akhir sebagai dokumentasi pelaksanaan tugas akhir, yang mencakup seluruh teori, implementasi, serta hasil pengujian yang telah dikerjakan.

1. **JADWAL PEMBUATAN TUGAS AKHIR**

Tugas akhir ini diharapkan bisa dikerjakan sesuai jadwal, sebagai berikut.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Tahapan | 2012 | | | |
| Maret | April | Mei | Juni |
| 1 | Studi Literatur |  |  |  |  |
| 2 | Implementasi |  |  |  |  |
| 3 | Uji coba dan evaluasi |  |  |  |  |
| 4 | Penyusunan Buku Tugas Akhir |  |  |  |  |

1. **DAFTAR PUSTAKA**
2. John Hershberger , Matthew Maxel , Subhash Suri, Finding the k shortest simple paths: A new algorithm and its implementation, *ACM Transactions on Algorithms (TALG)*, v.3 n.4, p.45-es, November 2007
3. J. Y. Yen. Finding the K shortest loopless paths in a network. *Management Science*, 17:712–716, 1971.
4. D. Eppstein. Finding the k shortest paths. *SIAM J. Computing*, 28(2):652–673, 1998.

**LEMBAR PENGESAHAN**

###### Surabaya, 29 Pebruari 2012

Menyetujui,

Pembimbing I

Yudhi Purwananto, S.Kom, M. Kom

NIP: 19700714 199703 1002

Pembimbing II

Rully Soelaiman S.Kom, M. Kom

NIP: 19700213 199402 1 001