Perangkat Lunak Pemahaman The Lift-To-Front Algorithm untuk Menyelesaikan Problema Maximum Flow

Marto Sihombing, Hansen Tanjaya
STMIK IBBI
Jl. Sei Deli No. 18 Medan, Telp. 061-4567111 Fax. 061-4527548
e-mail: martosihombing@yahoo.com

Abstrak

Permasalahan maximum flow network (aliran maksimum) merupakan permasalahan paling sederhana yang berhubungan dengan flow network. Setiap sisi berarah dianggap sebagai saluran yang memiliki kapasitas maksimum dimana material mengalir. Simpul merupakan titik pertemuan dari setiap saluran. Material mengalir melalui simpul tanpa berkumpul di dalamnya. Dengan perkataan lain, skala dari material yang memasuki sebuah simpul harus sama dengan skala dari material yang keluar dari simpul tersebut. Permasalahan ini menanyakan skala maksimum dari material yang dapat dikirimkan dari sumber ke tujuan tanpa melanggar setiap batasan kapasitas pada sistem. Permasalahan ini dapat diselesaikan dengan menggunakan algoritma lift-to-front. Algoritma lift-to-front bekerja dengan menggunakan list dengan memelihara sebuah daftar vertex pada jaringan. Dimulai dari depan, metode meninjau daftar, secara berulang memilih sebuah vertex u yang overflow dan kemudian membuangnya dari daftar, yaitu melakukan operasi 'dorong dan angkat' (push and lift) sampai u tidak memiliki kelebihan positif (positive excess). Saat sebuah vertex diangkat, vertex tersebut dipindahkan ke depan daftar (maka metode ini disebut 'lift-to-front') dan metode memulai peninjauannya sekali lagi. Tujuan penelitian ini adalah untuk mempelajari proses kerja dari algoritma Lift-to-Front dalam menyelesaikan permasalahan maximum flow pada flow network. Metode yang digunakan dalam penelitian ini adalah model Waterfall. Perangkat lunak akan menampilkan proses kerja dari algoritma Lift-to-Front secara terperinci tahap demi tahap dalam mencari maximum flow network. Perangkat lunak juga mencatat langkah-langkah kerja algoritma sehingga perangkat lunak dapat digunakan untuk membantu pemahaman mengenai proses kerja dari algoritma Lift-to-Front.

Kata kunci : *lift to front algorithm, maximum flow.*

Abstract

Problems of the maximum network flow (maximum flow) is the simplest problems associated with network flow. Each side considered trending channel that has a maximum capacity in which material flows. Node is the meeting point on each channel. Material flowing through the node without gathered in it. In other words, the scale of the material that enters a node must be equal to the scale of the material that came out of that node. This issue is looking for answers about the maximum scale of the material that may be sent from source to destination without violating any capacity constraints of the system. This problem can be solved by using algorithms lift-to-front. Algorithm lift-to-front work by using the list to maintain a list of vertices in the network. Starting from the front, the method of reviewing the list, repeatedly choose a vertex u that overflow and then remove it from the list, which is conducting operations "push and lift '(push and lift) until u do not have a positive surplus (positive excess). When a vertex is removed, is transferred to the front vertex list (this method is called 'lift-to-front') and the method will start again from the beginning. The purpose of this research is to study the working process of the algorithm-to-Front Lift in solving the maximum flow problem on a flow network. The method used in this study is the Waterfall model. The software will show the working process of the algorithm-to-Front Lift in detail step by step in finding the maximum flow network. The software also records the steps the algorithm so that the software can be used to aid understanding of the work process of the algorithm Lift-to-Front.

Keywords: *lift to front algorithm, maximum flow.*

1. Pendahuluan

Teori *graph* merupakan pokok bahasan yang sudah tua usianya namun memiliki banyak terapan sampai saat ini. *Graph* digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Representasi visual dari *graph* adalah dengan menyatakan objek sebagai noktah, bulatan atau titik, sedangkan hubungan antara objek dinyatakan dengan garis. *Graph* berarah dapat digunakan untuk memodelkan sebuah "*flow network*" dan menggunakannya untuk menjawab pertanyaan mengenai arus material. *Flow network* dapat digunakan untuk memodelkan cairan mengalir melalui pipa,

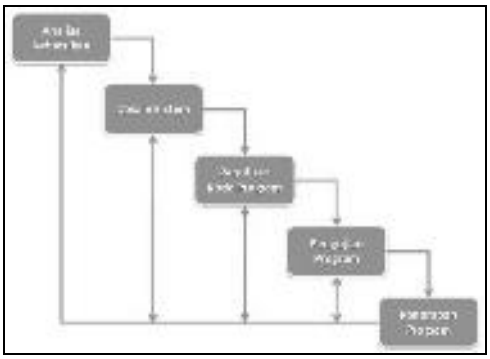
bagian dari suatu benda melalui jalur pemasangan, arus listrik melalui jaringan elektrik, informasi melalui jaringan komunikasi, dan sebagainya. Contoh ilustrasi dari problema *flow-network* ini adalah sebagai berikut, misalkan terdapat problema pengangkutan dari *Lucky Puck Company* dengan pabrik di Vancouver sebagai titik sumber dan gudang di Winnipeg sebagai titik tujuan. Barang akan dikirimkan melalui kota-kota penghubung, tetapi hanya c(u, v) peti per hari yang dapat dikirimkan dari kota u ke kota v. Setiap sisi diberi label dengan kapasitasnya.

Salah satu algoritma yang dapat digunakan untuk menyelesaikan permasalahan *maximum flow* pada *flow network* ini, yaitu algoritma Lift-to-Front. Metode *lift-to-front* merupakan sebuah variasi atau pengembangan dari metode *preflow-push* yang memiliki waktu eksekusi sebesar $O(V^3)$ dan secara asimtot memiliki waktu eksekusi yang sebagus $O(V^2E)$. Metode *lift-to-front* memelihara sebuah daftar *vertex* pada jaringan. Dimulai dari depan, metode meninjau daftar, secara berulang memilih sebuah *vertex* u yang *overflow* dan kemudian membuangnya dari daftar, yaitu melakukan operasi 'dorong dan angkat' (*push and lift*) sampai u tidak memiliki kelebihan positif (*positive excess*). Saat sebuah *vertex* diangkat, *vertex* tersebut dipindahkan ke depan daftar (maka metode ini disebut '*lift-to-front*') dan metode memulai peninjauannya sekali lagi. Alasan tersebut yang mendasari ketertarikan peneliti untuk melakukan sebuah penelitian yang diberi judul " Perangkat Lunak Pemahaman *The Lift-to-front Algorithm* untuk Menyelesaikan Problema *Maximum Flow*"

2. Metodologi Penelitian

Metodologi penelitian dilakukan dengan mengumpulkan data terlebih dahulu. Proses pengumpulan data dilakukan dengan dua cara yaitu : studi literatur studi lapangan. Studi literatur dilakukan dengan mengumpulkan, mengidentifikasi, dan mengolah data-data tertulis yang berasal dari buku-buku, surat kabar, majalah, tulisan ilmiah lainnya dan situs-situs penunjang yang berkaitan dengan penelitian penulis dan studi lapangan dilakukan mencari bahan penelitian dengan dengan cara wawancara langsung, atau dengan pengamatan terhadap objek dari percobaan ataupun pada objek survei. Data penelitian bersumber dari *Data Primer* yaitu data yang didapatkan dari pengukuran maupun pengamatan secara langsung di lapangan dan *data Sekunder* yaitu data yang didapatkan dari sumber lain misalnya instansi pemerintah, swasta maupun perorangan yang telah melakukan pengamatan secara langsung di lapangan.

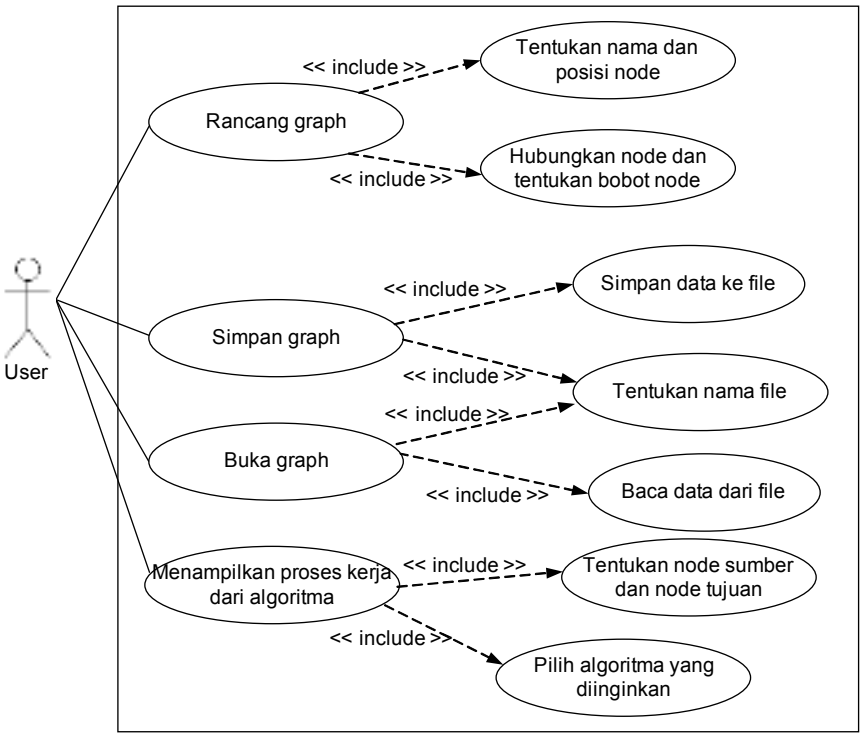
Setelah melakukan mengumpulkan data, penelitian dikembangkan dengan menerapkan model air terjun (Model *Waterfall*). Model *waterfall* mengusulkan sebuah pendekatan kepada perkembangan *software* yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan. Model ini melingkupi aktivitas–aktivitas berikut.



Gambar 1. Model W*aterfall*

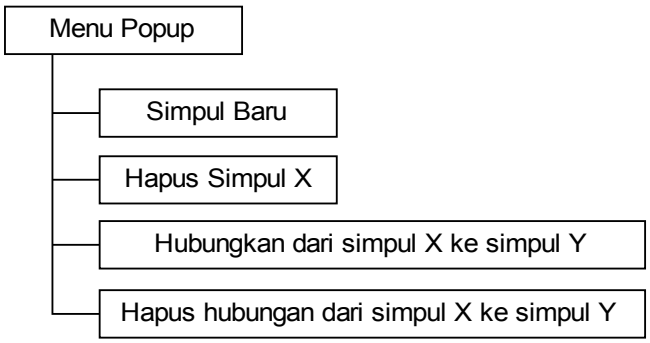
Keterkaitan dan pengaruh antar tahap ini dikarenakan *output* sebuah tahap dalam model *waterfall* merupakan *input* bagi tahap berikutnya, dengan demikian ketidaksempurnaan hasil pelaksanaan tahap sebelumnya adalah awal ketidaksempurnaan tahap berikutnya. Memperhatikan karakteristik ini, sangat penting bagi tim pengembang ataupun penulis untuk secara bersama-sama melakukan analisa kebutuhan dan desain sistem sesempurna mungkin sebelum masuk ke dalam tahap penulisan kode program.

Metode perancangan yang dilakukan terdiri dari perancangan use case dan user interface. Use case digunakan untuk menganalisis dan memodelkan sistem.



Gambar 3. Diagram *Use Case* dari Perangkat Lunak

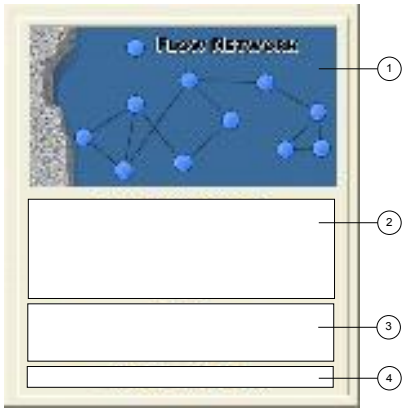
Pada gambar di atas, terlihat bahwa use case dari sistem hanya terdiri dari sebuah aktor saja yaitu *user* (pemakai). Semua data yang diperlukan untuk proses pemahaman akan dimasukkan oleh *user* ke dalam sistem. Setelah itu, sistem akan melakukan proses validasi terhadap semua data input dan menampilkan proses kerja dari algoritma yang diinginkan apabila semua data input valid. Narasi dari *use case* dijelaskan pada tabel 2.1.



Gambar 4. Rancangan Menu *Popup*

Rancangan user interface dari perangkat lunak pemahaman *Preflow-push algorithm* dan *Lift-to-front algorithm* ini terdiri dari *form* berikut : *splash screen*, *input graph*, pemahaman Algoritma *Preflow-Push*, pemahaman algoritma *Lift-to-Front* dan *About*.

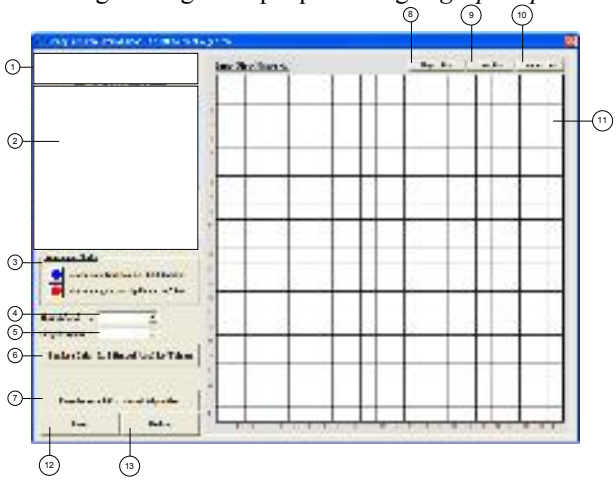
Form '*Splash Screen'* merupakan *form* pembuka (awal) dari perangkat lunak dan berisi nama dan identitas perangkat lunak.



Gambar 5. Rancangan *Form* 'Splash Screen'

Keterangan :
1 : *Icon perangkat lunak*.
2 : Nama perangkat lunak.
3 : Nama identitas perangkat lunak.
4 : Nama universitas.

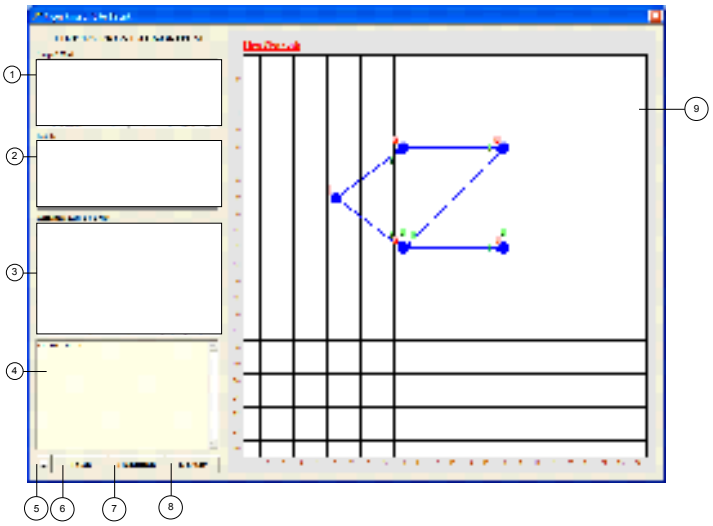
Form Input Graph berfungsi sebagai tempat perancangan *graph input*.



Gambar 6. Rancangan *Form* 'Input Graph'

Keterangan :
1 : Nama perangkat lunak.
2 : Keterangan mengenai metode yang dipakai.
3 : Keterangan grafik.
4 : *Combobox* 'Simpul Awal', berisi simpul yang menjadi simpul awal.
5 : *Combobox* 'Simpul Tujuan', berisi simpul yang menjadi simpul tujuan.
6 : Tombol 'Periksa Jalur', berfungsi untuk memeriksa jalur dari simpul awal ke simpul tujuan.
7 : Tombol 'Lift-to-Front Algorithm', berfungsi untuk membuka *form* pemahaman algoritma *Lift-to-Front*.
8 : Tombol 'Hapus Graf', berfungsi untuk mengosongkan daerah input graf.
9 : Tombol 'Buka Graf', berfungsi untuk membuka graf yang telah disimpan sebelumnya.
10 : Tombol 'Simpan Graf', berfungsi untuk menyimpan *input* graf.
11 : Daerah *input* graf.
12 : Tombol 'About', berfungsi untuk membuka *form About*.
13 : Tombol 'Keluar', berfungsi untuk menutup *form*.

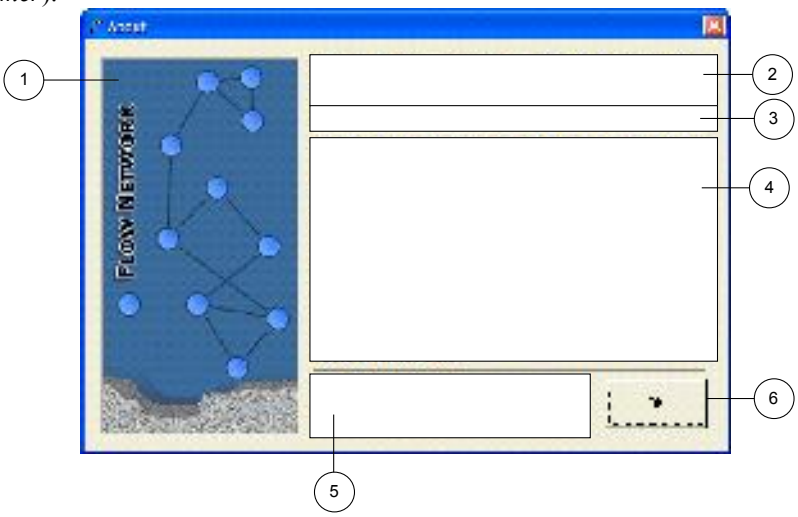
Form pemahaman Algoritma *Lift-to-Front* berfungsi untuk menampilkan prosedur kerja dari algoritma Lift-to-Front dimana proses kerjanya akan ditampilkan secara terperinci mulai dari tahapan awal hingga tahapan akhir.



Gambar 7. Rancangan *Form* 'Pemahaman Algoritma Lift-to-Front'

Keterangan :
1 : tabel yang berisi simpul graf.
2 : tabel yang berisi hubungan antar simpul graf.
3 : tabel yang berisi algoritma Lift-to-Front.
4 : keterangan eksekusi algoritma.
5 : Tombol 'Simpan', berfungsi untuk menyimpan keterangan eksekusi.
6 : Tombol 'Jalan', berfungsi untuk menjalankan algoritma Lift-to-Front.
7 : Tombol 'Hentikan/Lanjutkan', berfungsi untuk menghentikan untuk sementara atau melanjutkan kembali algoritma Lift-to-Front.
8 : Tombol 'Keluar', berfungsi untuk menutup *form* pemahaman.
9 : Daerah tampilan graf.

Form About berfungsi untuk menampilkan data-data pribadi mengenai perancang perangkat lunak (*programmer*).



Gambar 8. Rancangan *Form* 'About'

Keterangan :
1 : *icon* perangkat lunak.
2 : nama perangkat lunak.
3 : daerah tampilan data-data penyusun.
4 : tampilan mengenai permasalahan yang diangkat dalam perangkat lunak.
5 : daerah tampilan jurusan dan universitas.
4 : tombol 'OK', yang berfungsi untuk menutup *form*.

3 Analisis dan Hasil

Sebelum merancang perangkat lunak, maka terlebih dahulu perlu dilakukan analisis persyaratan terhadap perangkat lunak yang mencakup analisis fungsional dan non fungsional. Kemudian, baru dilakukan analisis struktur

3.1 Analisis Persyaratan

Analisis persyaratan terhadap sistem mencakup analisis fungsional yang mendeskripsikan fungsionalitas-fungsionalitas yang harus dipenuhi oleh perangkat lunak dan analisis non fungsional yang mendeskripsikan persyaratan non fungsional yang berhubungan dengan kualitas sistem.

3.2 Analisis Fungsional

Adapun beberapa persyaratan fungsional yang harus dipenuhi oleh perangkat lunak adalah sebagai berikut: *user* dapat melakukan perancangan terhadap *graph input* sesuai dengan keinginan, *Graph input* hasil rancangan dapat divalidasi untuk mengetahui apakah *graph* tersebut merupakan *graph* yang *valid* atau bukan, yaitu apakah semua *node* pada *graph* terhubung dan apakah *node* tujuan dapat diakses dari *node* sumber, *Graph input* hasil rancangan dapat disimpan ke dalam sebuah data *file* dan dapat dibuka kembali apabila diperlukan, Perangkat lunak mampu menunjukkan langkah-langkah kerja dari algoritma Preflow-push dan Lift-to-front dalam mencari solusi dari problema *flow* maksimum pada *flow network* dan Perangkat lunak mampu menampilkan hasil proses perhitungan dari algoritma Preflow-push dan Lift-to-front, yang dapat disimpan ke dalam sebuah *file* teks. *User* dapat menghentikan secara sementara (*pause*) dan melanjutkan kembali (*resume*) alur kerja algoritma di dalam perangkat lunak, supaya *user* dapat mengamati dan mengerti alur kerja algoritma. Perangkat lunak akan menampilkan animasi sederhana yang menunjukkan cara kerja dari *flow network*.

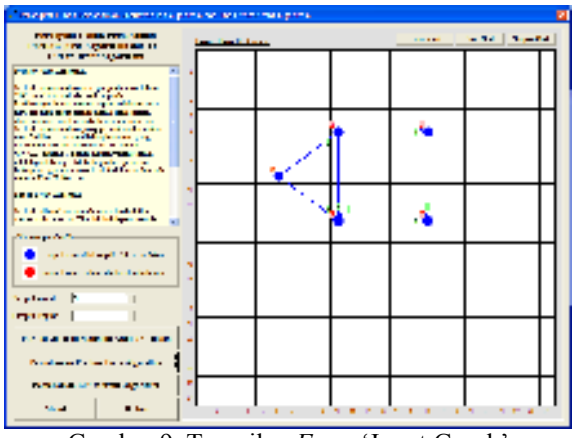
3.3 Analisis Non Fungsional

Untuk mengidentifikasikan berbagai persyaratan non-fungsional yang terkait dengan kualitas sistem digunakan Kerangka PIECES sebagai berikut : *Performance*, perangkat lunak harus dapat menampilkan proses kerja dari algoritma *Preflow-push* dan *Lift-to-front* secara terperinci tahapan demi tahapan. *Information*, perangkat lunak harus mampu menampilkan laporan hasil proses perhitungan dari algoritma *Preflow-push* dan *Lift-to-front*. *Economics*, perangkat lunak dapat dijalankan di semua versi sistem operasi Windows.Control, perangkat lunak dapat menampilkan proses pemahaman dari algoritma *Preflow-push* dan *Lift-to-front* apabila semua input yang dimasukkan *valid*. Ada atau tidaknya solusi tergantung pada graph input hasil rancangan.*Efficiency*, proses simulasi dari algoritma *Preflow-push* dan *Lift-to-front* dapat dihentikan dan dilanjutkan kembali sesuai keinginan. *Service*, perangkat lunak hanya dapat menampilkan proses pemahaman apabila telah dilakukan pengaturan data keadaan awal dari proses pemahaman yaitu graph input yang berupa graph berarah yang berbobot beserta properti-propertinya.

3.4 Hasil Implementasi

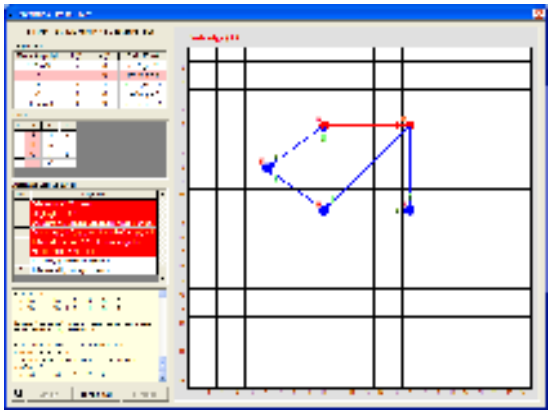
Berikut ini adalah tampilan dari hasil eksekusi perangkat lunak pemahaman algoritma Preflow-Push dan algoritma Lift-to-Front.

1. Tampilan *form* ‘Input Graph’.
   Form ini merupakan tempat perancangan *graph input*. Tampilan *form* ‘Input Graph’ dapat dilihat pada gambar 4.21.



Gambar 9. Tampilan *Form* ‘Input Graph’

2. Tampilan *form* 'Pemahaman Algoritma Lift-to-Front':
Form ini berfungsi untuk menampilkan proses penentuan *maximum flow network* dengan menggunakan algoritma Lift-to-Front. Tampilan *form* 'Pemahaman Algoritma Lift-to-Front' dapat dilihat pada gambar 4.22.



Gambar 10. Tampilan *Form* 'Pemahaman Algoritma Lift-to-Front'

Hasil eksekusi perangkat lunak adalah sebagai berikut:
Keterangan:
---

1) Untuk setiap node i, set h(i) = 0, e(i) = 0 dan h(awal) = jumlah vertex.
h(S) = 0, e(S) = 0
h(A) = 0, e(A) = 0
h(B) = 0, e(B) = 0
h(C) = 0, e(C) = 0
h(T) = 0, e(T) = 0

Set h(S) = jumlah simpul pada graf.
h(S) = 5

2) Untuk semua hubungan simpul awal, balikkan semua arah panah yang berasal dari simpul awal.

- Balikkan arah panah dari S ke A (bobot sisi = 5)
  Bobot sisi yang akan dibalikkan adalah 5
  e(A) = e(A) + 5 = 0 + 5 = 5

- Balikkan arah panah dari S ke B (bobot sisi = 4)
  Bobot sisi yang akan dibalikkan adalah 4
  e(B) = e(B) + 4 = 0 + 4 = 4

3) Bentuk list L.
  Set u ke awal list L.

4) u = 1, periksa simpul A

Tidak ada edge yang dapat di-push, lift simpul A,
h(A) = h(A) + 1 = 1

4) u = 1, periksa simpul A

Masih terdapat edge yang bisa di-push pada simpul A, yaitu (A,B)

- Balikkan arah panah dari A ke B (bobot sisi = 3)
  Bobot sisi yang akan dibalikkan adalah 3
  e(A) = e(A) - 3 = 5 - 3 = 2
  e(B) = e(B) + 3 = 4 + 3 = 7

Masih terdapat edge yang bisa di-push pada simpul A, yaitu (A,C)

- Balikkan arah panah dari A ke C (bobot sisi = 7)
  Bobot sisi yang akan dibalikkan adalah 2
  e(A) = e(A) - 2 = 2 - 2 = 0
  e(C) = e(C) + 2 = 0 + 2 = 2

Tidak terdapat edge lagi yang bisa di-push pada simpul A

5) u = next(u) , periksa simpul B

4) u = 2, periksa simpul B

Tidak ada edge yang dapat di-push, lift simpul B,
h(B) = h(B) + 1 = 1

4) u = 2, periksa simpul B

Masih terdapat edge yang bisa di-push pada simpul B, yaitu (B,T)

- Balikkan arah panah dari B ke T (bobot sisi = 6)
  Bobot sisi yang akan dibalikkan adalah 6
  e(B) = e(B) - 6 = 7 - 6 = 1
  e(T) = e(T) + 6 = 0 + 6 = 6

Tidak terdapat edge lagi yang bisa di-push pada simpul B

Tidak ada edge yang dapat di-push, lift simpul B,
h(B) = h(B) + 1 = 2

4) u = 1, periksa simpul B

Masih terdapat edge yang bisa di-push pada simpul B, yaitu (B,A)

- Balikkan arah panah dari B ke A (bobot sisi = 3)
  Bobot sisi yang akan dibalikkan adalah 1
  e(B) = e(B) - 1 = 1 - 1 = 0
  e(A) = e(A) + 1 = 0 + 1 = 1

Tidak terdapat edge lagi yang bisa di-push pada simpul B

5) u = next(u) , periksa simpul A

4) u = 2, periksa simpul A

Masih terdapat edge yang bisa di-push pada simpul A, yaitu (A,C)

- Balikkan arah panah dari A ke C (bobot sisi = 5)
  Bobot sisi yang akan dibalikkan adalah 1
  e(A) = e(A) - 1 = 1 - 1 = 0
  e(C) = e(C) + 1 = 2 + 1 = 3

Tidak terdapat edge lagi yang bisa di-push pada simpul A

5) u = next(u) , periksa simpul C

4) u = 3, periksa simpul C

Tidak ada edge yang dapat di-push, lift simpul C,

h(C) = h(C) + 1 = 1

4) u = 3, periksa simpul C

Masih terdapat edge yang bisa di-push pada simpul C, yaitu (C,T)

- Balikkan arah panah dari C ke T (bobot sisi = 3)
  Bobot sisi yang akan dibalikkan adalah 3
  e(C) = e(C) - 3 = 3 - 3 = 0
  e(T) = e(T) + 3 = 6 + 3 = 9

Tidak terdapat edge lagi yang bisa di-push pada simpul C

5) u = next(u) , periksa simpul

Semua simpul e sudah 0, maka graf maximum flow network sudah diperoleh.

4. Kesimpulan dan Saran

Setelah menyelesaikan penelitian ini, penulis menarik beberapa kesimpulan bahwa perangkat lunak mampu menampilkan proses kerja dari algoritma Lift-to-Front secara terperinci tahap demi tahap sehingga perangkat lunak dapat digunakan untuk membantu pemahaman atas proses kerja dari algoritma Lift-to-Front dalam mencari maximum flow network, perangkat lunak ini dapat diimplementasikan dalam menyelesaikan berbagai masalah yang berhubungan dengan maximum flow network, seperti aliran arus atau aliran air dalam pipa / saluran, pemilihan simpul pada algoritma Lift-to-Front lebih terstruktur dengan menggunakan List, sehingga algoritma tersebut lebih cepat dalam menghasilkan graf solusi. Untuk pengembangan perangkat lunak lebih lanjutpertimbangkan beberapa saran berikut : menambahkan algoritma maximum flow network yang lain sehingga dapat dilakukan perbandingan algoritma mana yang lebih baikdan perangkat lunak dapat dikembangkan lebih lanjut dengan menerapkan algoritma Lift-to-Front untuk menyelesaikan permasalahan aliran arus atau aliran air dalam pipa serta permasalahan maximum flow network lainnya.

Daftar Pustaka

[1] Cormen H. Leiserson E. Rivest L. Introduction to Algorithms. Mc Graw Hill Book Company. 1990.
[2] Hariyanto B. Struktur Data. Edisi-2. Bandung: Informatika. 2003.
[3] Munir R. Matematika Diskrit. Bandung: Informatika. 2005.
[4] Munir R. Lidia L. Algoritma dan Pemrograman. Edisi Ke-2. 2002.
[5] Putra R. The Best Source Code Visual Basic. Jakarta : PT. Elex Media Komputindo. 2005.
[6] Ramadhan A. 36 Jam Belajar Komputer Visual Basic 6.0. Jakarta: PT. Elex Media Komputindo. 2004.
[7] Sommerville I. Software Engineering. Fourth Edition. 1996.
[8] Flow network - Wikipedia, the free encyclop