Aplikasi Graf dalam Pengaturan Jadwal Penerbangan di Airport

Gilbran Imami (13509072)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
e-mail: 13509072@std.stei.itb.ac.id

***Abstrak*—Makalah ini membahas mengenai apa itu graf dan aplikasinya dalam mengatur jadwal (*scheduling*) di bandara (*airport*) yang biasanya meliputi kedatangan kedatangan (*arrival*) dan keberangkatan (*departure*). Pengaturan jadwal di bandara menggunakan sistem graf untuk memetakan proses-proses yang berupa antrian yang datang dan mengaturnya sedemikian hingga agar prosesnya tidak tak beraturan (*crowded).* Dalam kuliah telah mempelajari graf, dan dalam makalah ini akan mencoba menjelaskan algoritma-algoritma apa saja yang dibutuhkan untuk membangun suatu graf. Algoritma tersebut antara lain Algoritma Greedy, Algoritma Prim, dan Algoritma Dijkstra. Selain graf, diperlukan juga teori antrian (*queue*) yang menjadi input sekaligus output dalam proses penjadwalan. Ada beberapa proses dalam teori antrian. Proses tersebut antara lain *discrete-state*, proses Markov, proses *birth-death*, dan proses Poisson. Graf sangat tepat untuk digunakan sebagai pemodelan persoalan yang berkaitan antara suatu objek dengan objek lain dengan melakukan analisis antar objek tersebut. Selain itu, banyak algoritma dari graf yang berguna untuk mencari jarak minimum sehingga proses pengaturan jadwal dapat lebih simpel dan efisien. Karena itulah persoalan *scheduling* pada airport dapat dimodelkan dengan graf. Pada proses scheduling, beberapa pesawat datang dalam waktu yang berdekatan dan hampir bersamaan. Pesawat yang akan berangkat pun perlu diatur jadwalnya dengan baik agar tidak *delay* terlalu lama. Delay yang terlalu lama akan menimbulkan menurunnya kepercayaan konsumen. Pemanfaatan algoritma-algoritma dalam graf akan membuat masalah waktu penjadwalan lebih efisien karena semakin pendek jarak minimum dalam graf, maka akan semakin cepat proses penjadwalan yang terwakili dalam sebuah graf tersebut.**

***Kata kunci*—Graf, Teori Antrian, Algoritma Greedy, Algoritma Prim, Algoritma Dijkstra, Proses *discrete-state*, Proses Markov, Proses *birth-death*, Proses Poisson**

I. PENDAHULUAN

Graf adalah himpunan benda-benda yang disebut simpul (*vertex* atau *node*) yang terhubung oleh sisi (*edge*) atau busur (*arc*). Biasanya graf digambarkan sebagai kumpulan titik-titik (melambangkan simpul) yang dihubungkan oleh garis-garis (melambangkan sisi) atau garis berpanah (melambangkan busur). Suatu sisi dapat menghubungkan suatu simpul dengan simpul yang sama. Sisi yang demikian dinamakan gelang (*loop*).

Graf seringkali digunakan untuk merepresentasikan objek-objek yang bersifat diskrit dan hubungan antara objek-objek tersebut. Pengaplikasian graf yang sering digunakan dalam kehidupan adalah untuk memodelkan jalur transportasi, pengaturan jadwal, membangun sistem jaringan komputer, dan lain-lain.

Sebuah struktur graf bisa dikembangkan dengan memberi bobot pada tiap sisi. Graf berbobot dapat digunakan untuk melambangkan banyak konsep berbeda. Sebagai contoh jika suatu graf melambangkan jaringan jalan maka bobotnya bisa berarti panjang jalan maupun batas kecepatan tertinggi pada jalan tertentu. Ekstensi lain pada graf adalah dengan membuat sisinya berarah, yang secara teknis disebut graf berarah atau digraf (*directed graph*). Digraf dengan sisi berbobot disebut jaringan.

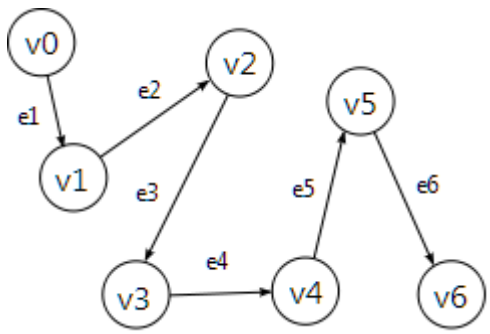
Selain graf, kita juga memerlukan teori antrian dalam menjelaskan proses manajemen jadwal dalam bandara. Antrian merupakan proses *input*, dilanjutkan dengan menunggu secara teratur berdasarkan urutan masuk, kemudian diproses secara bergilir. Ada beberapa tipe proses dalam menjalankan antrian, seperti proses *discrete-state*, proses Markov, proses *birth-death*, dan proses Poisson.

II. DASAR TEORI

2.1. Graf

Suatu graf G kita nyatakan sebagai G = (V, E) adalah suatu graf yang terdiri dari V sebagai himpunan tidak kosong dari simpul-simpul dan E sebagai himpunan pasangan dari simpul yang ada dalam V. Sebagai contoh definisi graf tersebut, yaitu:

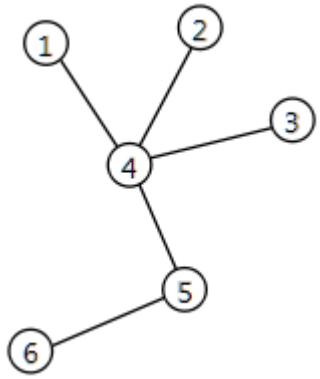
V= {1,2,3,4,5,6} dan *E* = {(1,2),(1,5),(2,3),(3,4),(4,5),(5,2),(4,6)}.



Gambar 1. Contoh Graf Sederhana

Graf T merupakan sebuah pohon jika dan hanya jika ada satu jalur sederhana pada setiap pasang simpul. Sisi (busur) dan semuanya terhubung, tanpa ada siklus (*loop*). Setiap simpul dapat menjadi akar pada pohon tersebut. Sebuah pohon merupakan graf yang tidak memiliki arah. Beberapa definisi pada pohon adalah sebagai berikut:

a. Setiap simpul terhubung dan tidak membentuk siklus.
b. Setiap simpul (kecuali akar) hanya memiliki satu simpul orang tua.
c. Setiap dua simpul dapat terhubung oleh *path* simpel dan *unique.*
d. Setiap simpul memiliki 0 atau lebih anak.
e. Sebuah sisi dengan 0 anak adalah merupakan daun.



Gambar 2. Pohon dengan 6 simpul dan 5 sisi

Berikut ini adalah beberapa terminologi yang berhubungan dengan teori graf :

1. Derajat (*degree*) dari suatu node, jumlah edge yang dimulai atau berakhir pada node tersebut. Contohnya, graf dengan 5 node berderajat 3 dan graf dengan node 2 berderajat 1.
2. Jalur (*path*) dua simpul adalah sebuah urutan simpul dan sisi yang dimulai dari simpul pertama dan berakhir di simpul yang kedua, dengan setiap sisi bersisian dengan simpul sebelum dan sesudahnya. Sebuah jalur disebut sederhana jika setiap sisi yang dilalui pada jalur tersebut hanya sekali saja. Contoh antara simpul 1 dan simpul 6 ada path
   $a \rightarrow c \rightarrow e$.
3. Jarak antara dua simpul adalah jumlah sisi minimum yang berada pada jalur dari simpul 1 ke simpul yang ke 2.
4. Jalur sederhana disebut sebagai siklus (*cycle* atau *loop*) apabila suatu path kembali ke titik asalnya atau simpul awal merupakan simpul akhir. Contoh: $e \rightarrow f \rightarrow d \rightarrow c \rightarrow e$.
5. Sebuah graf disebut terhubung jika terdapat jalur pada setiap pasang simpul 1 dan simpul ke 2.
6. Sebuah graf disebut berarah (digraf) jika sisi pada graf tersebut memiliki arah. Dengan demikian simpul (i, j) pada sisi e tidak secara langsung mengakibatkan bahwa simpul (j, i) juga ada pada e. Pada kondisi ini, sisi pada graf sering disebut dengan busur.
7. Graf disebut memiliki bobot (*weight*) jika pada setiap sisi (atau busur) diberi label dengan suatu bilangan.
8. Pohon (tree) merupakan salah satu jenis graf yang tidak mengandung cycle. Jika edge f dan a dalam digraf diatas dihilangkan, digraf tersebut menjadi sebuah tree. Jumlah edge dalam suatu tree adalah nV - 1. Dimana nV adalah jumlah vertex.

2.2. Antrian

Dalam proses pengaturan jadwal, yang digunakan adalah konsep antrian, dimana beberapa *task* datang dan menunggu untuk diatur dan dikerjakan.

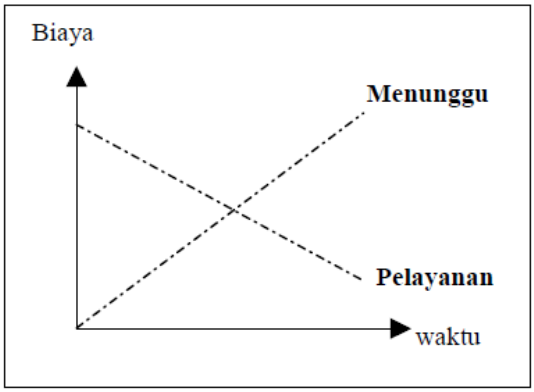
Antrian terdapat pada kondisi apabila obyek-obyek menuju suatu area untuk dilayani, namun kemudian menghadapi keterlambatan disebabkan oleh mekanisme pelayanan mengalami kesibukan.

Antrian timbul karena adanya ketidakseimbangan antara yang dilayani dengan pelayanannya.

Beberapa contoh antrian :

- Antrian pada pelayanan kasir supermarket
- Antrian kendaraan membeli bahan bakar
- Antrian pada lampu merah (orang menyebrang maupun kendaraan)
- Antrian pesawat akan mendarat di suatu bandara
- Antrian pelayanan dokter, dan lain-lain.

Hubungan yang menjadi problema suatu antrian, mencakup antara waktu menunggu dan waktu pelayanan (*service*), dijelaskan pada grafik di bawah ini:



Gambar 3. Grafik hubungan antara antrian dan service

Komponen yang mempengaruhi sistem antrian :

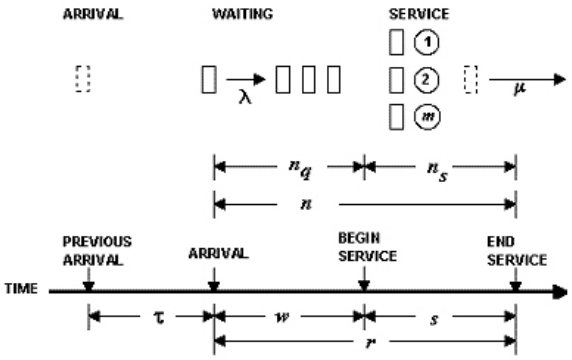
1. *Input* : distribusi jumlah kedatangan per satuan waktu, jumlah antrian yang dimungkinkan, maksimal panjang antrian, maksimal jumlah pelanggan.
2. *Proses Layanan* : distribusi jumlah kedatangan per satuan waktu, jumlah antrian

yang dimungkinkan, maksimal panjang antrian, maksimal jumlah pelanggan.

3. *Disiplin Antrian* : FIFO, LIFO, random, seleksi prioritas.

Beberapa variabel-variabel kunci yang digunakan dalam analisis antrian antara lain :

- *t*, yaitu waktu interarrival, adalah waktu antara dua kedatangan yang berturut-turut.
- λ adalah *mean rate* kedatangan = 1/E[*t*]. Dalam beberapa sistem, ini dapat berfungsi pada state sistem. Mean ini dapat tergantung pada jumlah job yang sudah berada dalam suatu sistem.
- *n* adalah jumlah job pada suatu sistem. Seringkali disebut juga sebagai panjang antrian, meliputi job yang sedang diterima untuk dilayani dan menunggu dalam antrian.
- *nq* adalah jumlah job yang sedang menunggu untuk menerima layanan. Ini selalu lebih kecil dari *n*, karena ini tidak termasuk job yang sedang menerima layanan.
- *s* adalah waktu layanan (service) per job.
- μ adalah mean rate layanan per server = 1/E[*s*]. Rate total layanan untuk *m* server adalah *m*μ.
- *ns* adalah jumlah job yang menerima layanan.
- *r* adalah waktu respon atau waktu dalam sistem. Waktu ini termasuk waktu menunggu layanan dan waktu ketika menerima layanan.
- *w* adalah waktu tunggu, yaitu interval waktu antara waktu kedatangan dan dimulainya layanan instan.



Gambar 4. Variabel yang digunakan dalam analisis antrian

Dalam pengaturan jadwal di bandara, graf dan antrian merupakan dua proses kunci untuk mengatur berjalanannya kegiatan di bandara. Ada beberapa algoritma-algoritma yang dapat menerjemahkan proses tersebut secara matematis, yang dibahas pada bagian 3.

III. Algoritma Untuk Membangun Graf dan Tipe Proses Antrian

3.1. Algoritma Greedy

Algoritma Greedy merupakan metode yang paling populer untuk memecahkan persoalan optimasi. Algoritma greedy membentuk solusi langkah per langkah yaitu :

- Terdapat banyak pilihan yang perlu dieksplorasi pada setiap langkah solusi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya.
- Pendekatan yang digunakan di dalam algoritmagreedyadalah membuat pilihan.
- Yang terlihat memberikan perolehan terbaik, yaitu dengan membuat pilihan optimum lokal (local optimum) pada setiap langkah dan diharapkan akan mendapatkan solusi optimum global (global optimum)

Berikut ini adalah contoh algoritma greedy:

```
procedure greedy (input C :
himpunan_kandidat; output S :
himpunan_solusi)
{ Menentukan solusi optimum dari persoalan optimasi
dengan algoritma greedy }

{ Deklarasi }
x : kandidat

{ Algoritma }
S ← {} { inisialisasi S dengan dengan kosong }
while (belum SOLUSI(S)) and (C ≠{}) do
x ← SELEKSI(C); { pilih kandidat dari C }
C ← C – {x}
      if LAYAK(S ∪ {x}) then
            S ← S ∪{x]
      endif
endwhile
{ SOLUSI(S) sudah diperoleh or C = {} }
```

3.2. Algoritma Prim

Algoritma Prim merupakan salah satu algoritma yang bekerja secara greedy. Algoritma Prim membentuk MST langkah per langkah. Pada setiap langkah dipilih sisi graf G yang mempunyai bobot minimum dan terhubung dengan MST yang telah terbentuk. Ada tiga langkah yang dilakukan pada algoritma Prim, yaitu :

- Pilih sisi graf G yang berbobot paling minimum dan masukan ke dalam T
- Pilih sisi (u,v) yang mempunyai bobot minimum dan bersisian dengan simpul di T, tetapi tidak membentuk sirkuit di T, lalu tambahkan ke dalam T.
- Ulangi langkah ke-dua sebanyak n-2 kali

Berikut ini adalah contoh dari algoritma Prim :

```
procedure prim (input G : graf, output
T : pohon)
{ Membentuk pohon merentang minimum T dari graf
terhubung G }

{ Deklarasi }
```

```
i, p, q, u, v : integer

{ Algoritma }
{ Cari sisi (p,q) dari E yang berbobot terkecil }
T ← {(p,q)}
for i ← 1 to n-2 do
{ Pilih sisi (u,v) dari E yang bobotnya terkecil
namun bersisian dengan suatu simpul di dalam T }
      T ← T ∪ {(u,v)}
endfor
```

3.3. Algoritma Dijkstra

Algoritma Dijkstra adalah sebuah algoritma *greedy* yang dipakai dalam memecahkan permasalahan jarak terpendek (*shortest path problem*) untuk sebuah graf berarah (*directed graph*) dengan bobot-bobot sisi (*edge weights*) yang bernilai tak-negatif.

Misalnya, bila *vertices* dari sebuah graf melambangkan kota-kota dan bobot sisi (*edge weights*) melambangkan jarak antara kota-kota tersebut, maka algoritma Dijkstra dapat digunakan untuk menemukan jarak terpendek antara dua kota.

Input algoritma ini adalah sebuah graf berarah yang berbobot (*weighted directed graph*) *G* dan sebuah sumber *vertex s* dalam *G* dan *V* adalah himpunan semua vertices dalam graph *G*.

Setiap sisi dari graf ini adalah pasangan vertices (*u*,*v*) yang melambangkan hubungan dari *vertex u* ke *vertex v*. Himpunan semua tepi disebut *E*.

```
function Dijkstra(G, w, s)
{ untuk setiap vertex v di V[G] }
        d[v] ← infinity
        previous[v] ← undefined
d[s] ← 0
S ← empty set
Q ← V[G]
while Q is not an empty set
        u ← Extract_Min(Q)
        S ← S union {u}
        for each edge (u,v) outgoing
from u
                if d[u] + w(u,v) < d[v]
                d[v] ← d[u] + w(u,v)
                previous[v] ← u
```

3.4. Proses stochastic atau Proses Discrete-State atau Continuous State

Proses discrete state memiliki bilangan nilai yang terbatas atau dapat dihitung. Sebagai contoh jumlah job dalam sistem $n(t)$ hanya dapat menggunakan nilai 0, 1,..*n*. Waktu tunggu di lain pihak dapat mengambil semua nilai pada garis hitung nyata. Maka proses ini merupakan proses yang berkelanjutan. Proses discrete-state stochastic sering pula disebut rantai stochastic.

3.5. Proses Markov

Jika *state* pada masa yang akan datang dari proses itu tidak tergantung pada masa yang telah lalu dan hanya tergantung pada masa sekarang saja, proses ini disebut Proses Markov. Pengetahuan *state* proses pada masa sekarang ini harus memadai. Proses discrete state Markov disebut rantai Markov.

Untuk memprediksi proses Markov selanjutnya yang ada di masa datang diperlukan pengetahuan *state* yang sedang berlangsung saat ini. Tidak dibutuhkan pengetahuan berapa lama proses terjadi di masa sekarang ini. Hal ini memungkinkan jika waktu *state* menggunakan distrtibusi eksponensial (memoryless). Ini akan membatasi aplikabilitas proses Markov.

3.6. Proses Birth-death

Area diskrit proses Markov dimana transisi jadi terlarang bagi *state* lain di sekelilingnya, disebut proses birth death. Untuk proses ini memungkinkan untuk merepresentasikan *state* dengan suatu integer dimana proses pada *state n* dapat berubah hanya ke *state n*+1 atau *n*-1.

Sebagai contoh adalah jumlah job dalam antrian. Kedatangan job dalam antrian (*birth*) menyebabkan *state* berubah menjadi +1 (plus satu), dan keberangkatan dari antrian karena telah sampai waktunya mendapatkan layanan (*death*) menyebabkan *state* berubah menjadi -1 (minus satu).

3.7. Proses Poisson

Jika waktu interarrival IID dan distribusi eksponensial tercapai, jumlah kedatangan dari *n* berlangsung dalam interval (*t*, *t*+*x*) berarti memiliki distribusi Poisson, dan oleh karena itu proses kedatangan diarahkan pada proses Poisson atau aliran Poisson. Aliran Poisson sangat populer dalam teori antrian karena kedatangan biasanya *memoryless* sebagai waktu interarrival terdistribusi secara eksponensial. Sebagai tambahan aliran Poisson memiliki properti :

a. Menggabungkan *k* aliran Poisson dengan mean rate λi hasil dalam aliran Poisson dengan mean rate λ diberikan dengan :

$$\lambda = \sum_{i=1}^{k} \lambda_i$$

b. Jika aliran Poisson di-split ke dalam *k* sub-aliran maka probabilitas job yang bergabung pada *i* sub-aliran adalah *pi*, Setiap sub-aliran juga Poisson dengan mean rate $pi\lambda$.

c. Jika kedatangan pada suatu server tunggal dengan waktu layanan yang eksponensial adalah Poisson dengan mean rate λ, Keberangkatan yang terjadi juga Poisson dengan rate yang sama λ. Menyediakan rate kedatangan λ lebih kecil dibandingkan rate pelayanan μ.

d. Jika kedatangan pada fasilitas layanan dengan *m* pusat layanan adalah Possion dengan mean rate λ, Keberangkatan juga merupakan aliran Poisson dengan rate yang sama λ, Menyediakan rate kedatangan λ lebih kecil dari rate total layanan . Ini adalah asumsi pada server, untuk memiliki distribusi eksponensial waktu layanan.

IV. PROSES SCHEDULING DALAM AIRPORT

Proses penjadwalan penerbangan di dalam airport diilustrasikan dengan graf yang memiliki interval tergantung dengan jumlah antrian yang masuk. Selayaknya sebuah graf, proses ini memiliki simpul-simpul dan sisi-sisi.

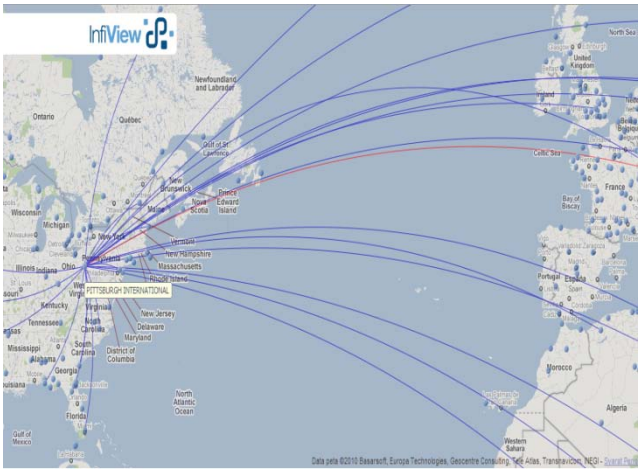
Sisi-sisi pada graf menunjukkan topologi antrian pada suatu jadwal penerbangan yang menghubungkan pesawat satu pada pesawat yang lainnya. Semakin pendek dan semakin sedikit suatu sisi, maka akan semakin cepat antrian diproses, yang berarti waktu delay antrian pesawat menjadi cepat dan terorganisir.

Simpul pada graf merepresentasikan pesawat yang menunggu untuk keberangkatan. Jumlah simpul-simpul ini berfluktuatif, dikarenakan pesawat hilir-mudik datang dan pergi. Semakin banyak pesawat, simpul yang ada akan semakin banyak dan terhubung ke simpul berikutnya dan mengakibatkan jumlah sisi semakin banyak.

Suatu simpul yang letaknya berdekatan atau sejajar memiliki arti bahwa kedua pesawat tersebut akan tinggal landas dengan waktu yang berdekatan atau bersamaan. Setelah suatu simpul sampai diujung simpul, maka antrian akan diproses yang berarti pesawat akan tinggal landas.

Permasalahannya adalah proses tersebut berlangsung terus menerus, dibarengi dengan datangnya pesawat baru yang berarti jumlah simpul dan jumlah sisi bertambah, sehingga graf yang dihasilkan akan menjadi sangat rumit.

Disinilah diperlukan algoritma-algoritma untuk mencari jalur terpendek dari simpul-simpul tersebut untuk mengurangi kerumitan graf, dan mengefisienkan penjadwalan. Jumlah sisi akan berkurang, sehingga tidak ada sisi (antrian) yang tidak diperlukan.



Gambar 5. Topologi jalur penerbangan bandara Pittsburgh International, Pennsylvania.

V. KESIMPULAN

Berdasarkan pemaparan yang diungkap di atas, graf dapat merepresentasikan proses penjadwalan penerbangan di suatu bandara. Graf tersebut dapat dibangun dan disederhanakan dengan menggunakan algoritma greedy, algoritma prim, algoritma dijkstra, maupun algoritma yang lain. Algoritma ini dapat dimanfaatkan untuk menyelesaikan persoalan antrian pesawat yang masuk yang terlalu rumit dengan mencari jalur yang paling efektif sehingga waktu yang digunakan untuk proses pengaturan jadwal menjadi lebih cepat.

VI. ACKNOWLEDGMENT

Ucapan terima kasih Saya ucapkan kepada Pak Rinaldi Munir atas dukungan, bimbingan dan berbagai referensi-referensi yang dapat digunakanuntuk penyelesaian makalah ini. Terima kasih juga tidak lupa Saya ucapkan kepada teman-teman prodi IF maupun STI yang telah bersedia untuk berdiskusi terhadap berbagai permasalahan dalam makalah ini.

REFERENSI

[1] Munir, Rinaldi, “Matematika Diskrit”, Informatika, 2003.
[2] http://en.wikipedia.org/wiki/Tree_%28graph_theory%29
[3] http://id.wikipedia.org/wiki/Teori_graf
Tanggal akses 15 Desember 17.00
[4] http://www.airport-technology.com/contractors/consult/quintiq/
Tanggal akses 15 Desember 19.00
[5] http://www.iata.org/whatwedo/passenger/scheduling/Pages/index.aspx
Tanggal akses 15 Desember 19.30

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 29 April 2010

Gilbran Imami (13509072)