Pencarian Jalur Terpendek dengan Menggunakan Graf dan *Greedy* dalam Kehidupan Sehari-hari

Andika Mediputra - NIM : 13509057
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
andika.mediputra@students.itb.ac.id

***Abstrak* — Makalah ini membahas tentang penggunaan teori dalam strategi algoritma yaitu mencari *shortest path* (lintasan terpendek) dengan memakai graf serta menggunakan algoritma *greedy* dan penerapannya dalam kehidupan sehari-hari. Lintasan terpendek adalah lintasan yang paling minimal jaraknya dengan tujuan tempat yang sama. Ada berbagai cara dan langkah untuk menentukan suatu lintasan terpendek, hasilnya mungkin bervariasi namun bertujuan sama yaitu mencari lintasan yang seefektif dan seoptimal mungkin. Jika menggunakan kendaraan maka aplikasi *shortest path* ini sangat berguna untuk mengefisiensikan bahan bakar kendaraan tersebut dan lebih hemat waktu untuk mencapai tempat yang dituju.**

***Kata Kunci* — Jarak, Graf, *Greedy*, Jalur, Waktu.**

I. PENDAHULUAN

Dalam kehidupan sehari-hari, masyarakat selalu bergerak ke suatu tempat untuk mencapai tujuannya masing-masing. Di jaman globalisasi ini, seringkali waktu menjadi sangat berharga bagi setiap manusia, sehingga ada perumpamaan "Waktu adalah uang". Oleh karena itu dalam bepergian ke suatu tempat, tidak sedikit masyarakat mencari dan menggunakan jalur terpendek dan secepat mungkin untuk mencapai tujuan tempat tersebut.

Dalam bepergian terkadang kita membutuhkan peta atau untuk jaman sekarang yang lebih modern dengan menggunakan GPS (*Global Positioning System*). Dari gambar peta tersebut bisa dilihat banyak variansi jalur yang bisa digunakan untuk mencapai tempat yang dituju. Bisa dilihat dan dicari manakah jalur terpendek yang bisa digunakan untuk bepergian.

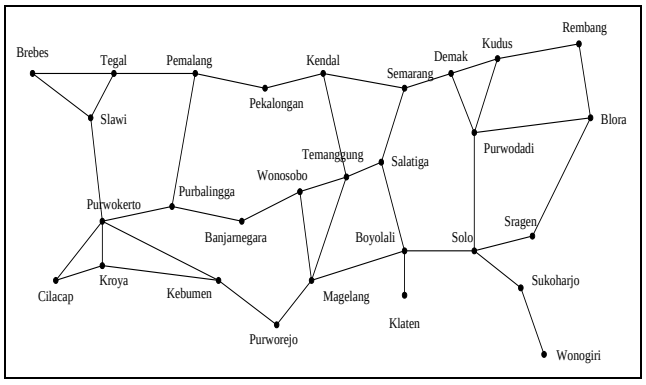
Shortest path ini sangat berguna bagi pemilik kendaraan pribadi, contohnya ialah dengan memakai jalur terpendek dibandingkan dengan jalur yang lebih panjang, konsumsi BBM kendaraan tersebut dengan asumsi kecepatan kendaraan tersebut sama kecepatannya antara jalur yang pendek dengan jalur yang panjang, akan terlihat bahwa dengan memakai jalur terpendek konsumsi BBM tersebut akan lebih irit. Waktu yang digunakan selama perjalanan pun menjadi lebih singkat, dengan asumsi tidak ada hambatan atau kemacetan di jalur pendek maupun jalur yang lebih panjang. *Shortest path* itu memiliki banyak keuntungan.

II. METODE

Dalam makalah ini penulis akan membahas teori strategi algoritma yang berhubungan dengan judul makalah ini yaitu tentang graf dan penerapan algoritma *greedy*. Bagaimana cara melihat dan menggunakan graf dalam kehidupan sehari-hari serta memakai cara *greedy* untuk mencari dan memakai *shortest path* ini. Berbagai algoritma juga telah tersedia dan bisa dipakai untuk mencari jalur terpendek, namun yang penulis bahas di makalah ini hanya sebatas penerapan dengan algoritma *greedy*.

2.1 Graf

Teori graf merupakan pokok bahasan yang sudah tua usianya namun memiliki banyak terapan sampai saat ini. Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antar objek-objek tersebut. Representasi visual dari graf adalah dengan menguyatakan objek dinyatakan sebagai noktah, bulatan, atau titik, sedangkan hubungan antara objek dinyatakan dengan garis. Sesungguhnya peta adalah sebuah graf, yang dalam hal ini kota dinyatakan sebagai bulatan sedangkan jalan dinyatakan sebagai garis.



Gambar 1. Peta jaringan jalan raya di Provinsi Jawa Tengah yang merupakan suatu graf

Secara matematis, graf didefinisikan sebagai berikut:
Graf G didefinisikan sebagai pasangan himpunan (*V*,*E*) yang dalam hal ini:

V = himpunan tidak kosong dari simpul-simpul (*verticles* atau *node*) = $\{v_1, v_2, ..., v_n\}$
E = himpunan sisi (*edges* atau *arcs*) yang menghubungkan sepasang simpul = $\{e_1, e_2, ..., e_n\}$
atau dapat ditulis singkat notasi G = (V,E)

2.1.1 Jenis-jenis Graf

- Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf:
  a. Graf sederhana
     Graf yang tidak mengandung gelang maupun sisi-ganda dinamakan graf sederhana.
  b. Graf tak-sederhana
     Graf yang mengandung sisi ganda atau gelang dinamakan graf tak-sederhana (*unsimple graph*).

- Berdasarkan orientasi arah pada sisi:
  a. Graf tak-berarah
     Graf yang sisinya tidak mempunyai orientasi arah disebut graf tak-berarah.
  b. Graf berarah
     Graf yang setiap sisinya diberikan orientasi arah disebut graf berarah.

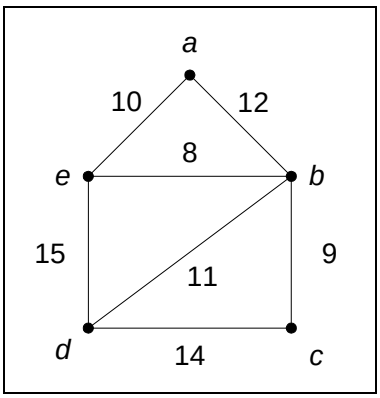
2.1.2 Terminologi Dasar

- Lintasan (*Path*)
  Lintasan yang panjangnya n dari simpul awal v_0 ke simpul tujuan v_n di dalam graf G ialah barisan berselang-seling simpul-simpul dan sisi-sisi yang berbentuk v_0, e_1, v_1, e_2, v_2,... , v_{n-1}, e_n, v_n sedemikian sehingga $e_1 = (v_0, v_1)$, $e_2 = (v_1, v_2)$, ... , $e_n = (v_{n-1}, v_n)$ adalah sisi-sisi dari graf G.

- Siklus (*Cycle*) atau Sirkuit (*Circuit*)
  Lintasan yang berawal dan berakhir pada simpul yang sama disebut sirkuit atau siklus.

- Terhubung (*Connected*)
  Dua buah simpul v_1 dan simpul v_2 disebut terhubung jika terdapat lintasan dari v_1 ke v_2. G disebut graf terhubung (*connected graph*) jika untuk setiap pasang simpul v_i dan v_j dalam himpunan V terdapat lintasan dari v_i ke v_j. Jika tidak, maka G disebut graf tak-terhubung (*disconnected graph*).

- Graf Berbobot
  Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot). Bobot pada tiap sisi dapat menyatakan jarak antara dua buah kota, biaya perjalanan antara dua buah kota, waktu tempuh pesan (*message*) dari sebuah simpul komunikasi ke simpul komunikasi lain (dalam jaringan komputer), ongkos produksi, dan sebagainya. Terminologi graf berbobot inilah yang menjadi pembahasan dalam makalah ini.



Gambar 2. Graf berbobot

2.1.3 Beberapa Aplikasi Graf

Terdapat banyak aplikasi yang berkaitan dengan graf. Di dalam aplikasi itu, graf digunakan sebagai alat untuk merepresentasikan atau memodelkan persoalan. Berdasarkan graf yang dibentuk, barulah persoalan tersebut diselesaikan. Ada beberapa aplikasi yang berkaitan dengan lintasan/sirkuit di dalam graf, diantaranya adalah:

- Lintasan terpendek
- Persoalan pedagang keliling
- Persoalan tukang pos Cina
- Pewarnaan graf

2.2 Algoritma Greedy

Algoritma *greedy* mungkin merupakan metode yang paling populer untuk memecahkan persoalan optimasi. Algoritma ini sederhana dan lempang (*straightforward*). Prinsip *greedy* adalah "*Take what you can get now!*".

Algoritma *greedy* membentuk solusi langkah per langkah (*step by step*). 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 dirubah lagi pada langkah selanjutnya. Pendekatan yang digunakan di dalam algoritma *greedy* adalah membuat pilihan yang "tampaknya" memberikan perolehan terbaik, yaitu dengan membuat pilihan optimum lokal (*local optimum*) pada setiap langkah dengan harapan bahwa sisanya mengarah ke solusi optimum global (*global optimum*).

2.2.1 Skema Umum Algoritma Greedy

Persoalan optimasi dalam konteks algoritma *greedy* disusun oleh elemen-elemen sebagai berikut:

1. Himpunan kandidat, C
   Himpunan ini berisi elemen-elemen pembentuk solusi.
2. Himpunan solusi, S
   Berisi kandidat-kandidat yang terpilih sebagai solusi persoalan.
3. Fungsi seleksi
   Fungsi yang pada seiap langkah memilih kandidat yang paling memungkinkan mencapai

solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.

4. Fungsi kelayakan (*feasible*)
   Memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala (*constraints*) yang ada.
5. Fungsi obyektif
   Fungsi yang memaksimumkan atau meminimumkan nilai solusi; misalnya panjang lintasan, keuntungan, dan lain-lain.

Pseudo-code algoritma *greedy* adalah sebagai berikut:

```
function greedy (input C: himpunan_kandidat)→
himpunan_kandidat
  { Mengembalikan solusi dari persoalan optimasi
    dengan algoritma greedy
    Masukan: himpunan kandidat C
    Keluaran: himpunan solusi yang bertipe
              himpunan_kandidat
  }

Deklarasi
 x : kandidat
 S : himpunan_kandidat

Algoritma:
 S ← {} {inisialisasi S dengan kosong}
 while (not SOLUSI(S)) and (C ≠ {} ) do
   x ← SELEKSI(C) {pilih sebuah kandidat dari C}
   C ← C - {x} {elemen himpunan kandidat
                berkurang satu}
   if LAYAK(S ∪ {x}) then
     S ← S ∪ {x}
   endif
 endwhile
 { SOLUSI(S) or C = {} }

 if SOLUSI(S) then
   return S
 else
   write('tidak ada solusi')
 endif
```

2.2.2 Contoh-contoh Algoritma Greedy

Beberapa contoh optimasi yang bisa diselesaikan dengan algoritma *greedy* antara lain:

1. Masalah penukaran uang
2. Minimisasi waktu di dalam sistem
3. Persoalan *knapsack*
4. TSP
5. Penjadwalan *job* dengan batas waktu
6. Pohon merentang minimum
7. Lintasan terpendek dengan satu simpul asal
8. Pemampatan data dengan algoritma Huffman

III. ANALISIS PERMASALAHAN

Persoalan mencari lintasan terpendek di dalam graf merupakan persoalan salah satu optimasi. Graf yang digunakan dalam pencarian lintasan terpendek ialah graf berbobot (*weighted graph*), yaitu graf yang setiap sisinya diberikan suatu nilai atau bobot. Bobot pada sisi graf dapat menyatakan jarak antar kota, waktu pengiriman pesan, ongkos pembangunan, dan sebagainya. Asumsi yang kita gunakan di sini adalah bahwa semua bobot bernilai positif. Kata "terpendek" jangan selalu diartikan secara fisik sebagai jarak minimum, sebab kata "terpendek" berbeda-beda maknanya bergantung pada tipikal persoalan yang diselesaikan. Namun, secara umum "terpendek" berarti meminimasi bobot pada suatu lintasan di dalam graf.

Contoh terapan pencarian lintasan terpendek misalnya simpul pada graf dapat merupakan kota, sedangkan sisi menyatakan jalan yang menghubungkan dua buah kota. Bobot sisi graf dapat menyatakan jarak antara dua buah kota atau rata-rata waktu tempuh antara dua buah kota. Apabila terdapat lebih dari satu lintasan dari kota A ke kota B, maka persoalan lintasan terpendek disini adalah menentukan jarak terpendek atau waktu tersingkat dari kota A ke kota B.

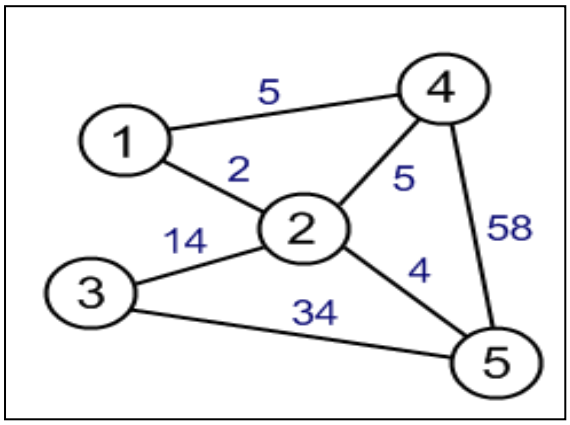
Ada beberapa macam persoalan lintasan terpendek, antara lain:

1. Lintasan terpendek antara dua buah simpul tertentu.
2. Lintasan terpendek antara semua pasangan simpul.
3. Lintasan terpendek dari simpul tertentu ke semua simpul yang lain.
4. Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu.

Sampai saat ini, sudah banyak algoritma mencari lintasan terpendek yang pernah ditulis orang. Algoritma lintasan terpendek yang paling terkenal adalah algoritma Dijkstra. Aslinya, algoritma Dijkstra diterapkan untuk mencari lintasan pada graf berarah. Namun, algoritma ini juga benar untuk graf tak-berarah. Algoritma Dijkstra mencari lintasan terpendek dalam sejumlah langkah. Algoritma ini menggunakan prinsip *greedy*. Prinsip *greedy* pada algoritma Dijkstra menyatakan bahwa pada setiap langkah kita memilih sisi yang berbobot minimum dan memasukannya ke dalam himpunan solusi.

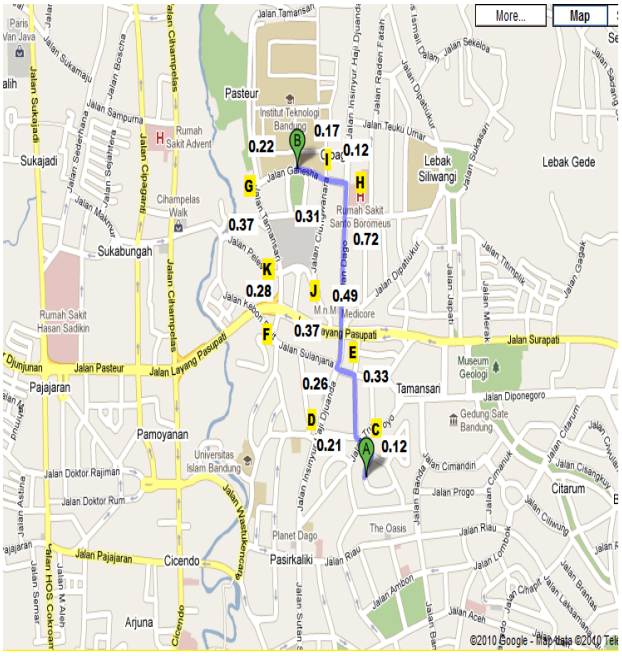
Dalam mencari lintasan terpendek, bisa juga dengan menggunakan teori pohon. Dalam hal pencarian lintasan terpendek salah satu yang relevan adalah dengan algoritma Prim dan algoritma Kruskal. Algoritma Prim membentuk pohon merentang minimum langkah per langkah.

Pada setiap langkah kita mengambil sisi dari graf *G* yang mempunyai bobot minimum namun terhubung dengan pohon merentang *T* yang telah terbentuk. Pada algoritma Kruskal, sisi-sisi graf diurut terlebih dahulu berdasarkan bobotnya dari kecil ke besar. Sisi yang dimasukkan ke dalam himpunan T adalah sisi graf *G* sedemikian sehingga *T* adalah pohon. Ada perbedaan diantara kedua algoritma ini namun hasil akhir yang didapat ialah sama, yaitu jarak yang dilalui ialah minimum.



Gambar 3. Contoh lintasan dengan graf berbobot

Saatnya menerapkan teori graf tersebut untuk mencari lintasan terpendek dalam kehidupan sehari-hari. Gambar di atas menunjukkan peta sebagian kecil kota Bandung, yang penulis teliti di sini adalah peta dari rumah penulis menuju kampus Institut Teknologi Bandung.



Gambar 4. Peta dari rumah penulis ke ITB

Dari peta tersebut terlihat bahwa banyak jalan yang bisa ditempuh, dalam hal ini posisi rumah penulis ada di A, menuju ke ITB, dalam peta ini dilambangkan dengan kode B. Tentunya dalam menuju ke tempat tujuan selalu ingin dicari manakah jalur yang paling cepat dan pendek untuk ditempuh. Asumsi bahwa tidak ada kemacetan yang terjadi dan dengan menggunakan kecepatan yang sama pada tiap jalur yang dilalui.

Peta di atas bisa dilihat sebagai graf berbobot dimana simpul-simpul dilambangkan dengan kode A, B, C, D, E, F, G, H, I, J, K, dan bobot yang tertulis di peta adalah jarak yang penulis ukur dengan menggunakan bantuan *Google Earth* dan *Google Maps* dalam satuan kilometer (km). Setiap simpul ke simpul yang lainnya sudah diukur jaraknya oleh penulis, tinggal dipergunakan bobot tersebut sesuai keperluan.

Dengan menggunakan algoritma *greedy*, masalah dapat diselesaikan dengan memilih jalur satu per satu. Sekali jalur tersebut dipilih, maka jalur tersebut tidak dapat dibatalkan atau dirubah.

Elemen-elemen yang digunakan adalah sebagai berikut:

1. Himpunan kandidat:
   Himpunan banyaknya jalur yang bisa dipakai beserta rincian jarak serta waktu untuk mencapai tujuan.
2. Himpunan solusi:
   Daftar jalur yang bisa dipilih untuk mencapai tujuan.
3. Fungsi seleksi
   Pilihlah jalur yang paling pendek dan dengan waktu tempuh yang paling sebentar untuk mencapai tujuan.
4. Fungsi kelayakan
   Memeriksa apakah nilai total dari durasi waktu perjalanan tidak melebihi dari durasi yang ditentukan oleh pengguna
5. Fungsi obyektif
   Pengambilan jalur yang dilakukan dengan jarak yang terpendek serta waktu yang relatif cepat.

Adapun strategi *greedy* yang digunakan adalah sebagai berikut:

1. *Greedy by distance*
   Pada setiap langkah, pilihlah jalur (simpul) yang jaraknya paling sedikit. Strategi ini mencoba mencari jarak yang terpendek untuk mencapai tempat tujuan.
2. *Greedy by duration*
   Pada setiap langkah, pilihlah jalur (simpul) yang memakan waktu paling sedikit dalam penempuhannya. Strategi ini mencoba mencari waktu tercepat untuk mencapai tempat tujuan.

Ada banyak variasi jalan yang bisa dipakai untuk menuju tempat tujuan, tentunya setiap jalur mempunyai kelebihan dan kekurangan masing-masing. Semakin pendek jalur yang digunakan maka akan semakin cepat sampai di tempat tujuan dan lebih menghemat waktu, dan juga jika menggunakan kendaraan pribadi akan semakin menghemat konsumsi bahan bakar yang digunakan. Penjelasan lebih detail akan dipaparkan lewat tabel berikut ini.

Simpul	Jarak dalam km
A, C	0,12
C, D	0,21
C, E	0,33
E, F	0,37
E, H	0,72
E, J	0,49
F, K	0,28
K, G	0,37

J, I	0,31
H, I	0,12
I, B	0,17
G, B	0,22
D, E	0,26

Tabel 1. Jarak diantara 2 simpul

Jalur yang digunakan	Jarak total dalam km
A-C-D-E-H-I-B	1,60
A-C-E-H-I-B	1,46
A-C-E-F-K-G-B	1,69
A-C-E-J-I-B	1,56
A-C-D-E-F-K-G-B	1,83

Tabel 2. Perbandingan jarak dari masing-masing rute yang digunakan

Berdasarkan hasil penelitian dan dengan menggunakan algoritma *greedy by distance* terhadap rute yang digunakan penulis dengan total jaraknya, terlihat bahwa rute yang menunjukkan shortest path ialah rute A-C-E-H-I-B dengan total jarak sebesar 1,46 km, tergambar di peta dengan jalur yang berwarna biru. Dengan mengambil rute antar simpul yang terpendek maka didapatlah jalur Rute yang lain juga bisa digunakan, namun terlihat bahwa rute lainnya itu lebih panjang dibanding rute terpendek ini.

Walaupun memiliki jalur yang terpendek, jalur yang digunakan ini tidak memiliki waktu tercepat untuk mencapai tujuan. Ini dikarenakan terdapat lampu merah diantara simpul yang dilalui, kondisi jalan utama yang ramai dilewati kendaraan, dan sebagainya. Untuk lebih jelasnya bisa dilihat di gambar ini, lampu merah ditandai dengan kotak berwarna merah:

Gambar 5. Jalur yang bertemu dengan lampu merah

Dengan menggunakan asumsi penulis berkendara ke tempat tujuan dengan menggunakan motor berkecepatan rata-rata 30 km/jam dan selalu berhenti di lampu merah dengan waktu rata-rata 1 menit setiap berhenti di lampu merah tersebut dan dengan kondisi jalan utama yang ramai, maka durasi perjalanan rata-rata dalam satuan menit akan diperlihatkan di tabel berikut ini:

Jalur yang digunakan	Waktu tempuh (menit)
A-C-D-E-H-I-B	6
A-C-E-H-I-B	5
A-C-E-F-K-G-B	6
A-C-E-J-I-B	4
A-C-D-E-F-K-G-B	7

Tabel 3. Perbandingan waktu tempuh dari masing-masing rute yang digunakan

Berdasarkan penelitian dan dengan menggunakan algoritma *greedy by duration* terhadap rute yang digunakan penulis, terlihat bahwa jalur yang memiliki waktu tempuh paling sedikit adalah jalur A-C-E-J-I-B dengan waktu tempuh selama 4 menit. Ternyata dengan menggunakan *greedy* yang berbeda, waktu tempuh jalur ini lebih cepat jika dibandingkan dengan jalur *greedy by distance* dikarenakan jalur ini bukan merupakan jalur utama dan lebih sedikit bertemu dengan lampu merah. Walaupun jarak tempuhnya berbeda sedikit dengan jalur *greedy by distance*. Jalur *greedy by duration* ini jalur dengan waktu tersingkat untuk mencapai tujuan, cocok untuk kondisi dimana seseorang yang sangat menghargai dan menggunakan waktu yang ada.

Terbukti dengan menggunakan algoritma *greedy*, bahwa shortest path itu bisa dicari. Shortest path ini sangat berguna untuk mencapai tujuan dengan waktu yang relatif cepat karena menggunakan jarak yang lebih pendek dibanding lainnya dan menghemat waktu juga pengeluaran bensin bagi yang memakai kendaraan pribadi, karena waktu sangatlah berarti dan waktu tidak bisa kembali.

IV. KESIMPULAN

Persoalan mencari lintasan terpendek di dalam graf dan dengan menggunakan algoritma *greedy* merupakan persoalan salah satu optimasi. Mencari *shortest path* (jalur terpendek) untuk mencapai suatu tempat merupakan cara efektif bagi siapa saja agar lebih menghemat waktu, serta biaya dalam mencapai suatu tujuan. Ada berbagai cara untuk mencari jalur terpendek melalui berbagai algoritma, tiap algoritma tersebut memiliki kelebihan dan kekurangannya masing-masing, tinggal bagaimana menerapkannya dalam kehidupan sehari-hari sesuai dengan kebutuhannya.

REFERENSI

[1] http://maps.google.com/ (tanggal akses 8 Desember 2011 pukul 21.00)

[2] http://www.algolist.net/Data_structures/Graph (tanggal akses 8 Desember 2011 pukul 19.00)

[3] Munir, Rinaldi. 2009. *Strategi Algoritma*. Bandung: Penerbit Institut Teknologi Bandung.

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, 9 Desember 2011

ttd

Andika Mediputra
13509057