Penerapan Algoritma A* dalam Penentuan Lintasan Terpendek

Johannes Ridho Tumpuan Parlindungan/13510103
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13510103@std.stei.itb.ac.id

***ABSTRACT*—Pada makalah ini akan dibahas tentang penentuan lintasan terpendek antara ITB dengan UNIKOM. Problem lintasan terpendek ini adalah salah satu dari contoh aplikasi graf. Lintasan terpendek dapat diperoleh dengan bermacam-macam algoritma, tetapi pada makalah ini hanya dibahas tentang penggunaan algoritma A* saja. Algoritma A* adalah sebuah algoritma yang bekerja dengan memroses satu per satu kemungkinan jarak setiap simpul yang ada lalu kemudian menentukan mana lintasan yang memiliki jarak paling pendek.**

***Index Terms*—graf, lintasan terpendek, algoritma A*.**

I. PENDAHULUAN

Setiap orang tentu sering bepergian ke suatu tempat baik untuk bekerja, rekreasi, atau melakukan berbagai kegiatan lainnya. Dalam perjalanan mungkin kita sering merasa kesal karena terlambat sampai ditujuan karena tidak memilih jalur yang tepat dan lebih efektif. Untuk menghemat waktu, tenaga, dan bahan bakar tentunya, saat bepergian ke suatu tempat dapat digunakan jalan dengan jarak terpendek. Jarak terpendek ini dapat ditentukan dengan menerapkan salah satu aplikasi graf yaitu penentuan lintasan terpendek (shortest path).

Problem lintasan terpendek dapat diselesaikan dengan banyak algoritma seperti algoritma Dijkstra, algoritma A*, Floyd-Warshall, Johnson's *algorithm*, dan Perturbation *theory*.

Dengan algoritma-algoritma ini, dapat diperoleh penyelesaian berbagai macam problem jarak terpendek, tidak hanya problem yang dibahas pada makalah ini saja.

II. PENJELASAN TENTANG GRAF, LINTASAN TERPENDEK, DAN ALGORITMA A*

2.1. Graf

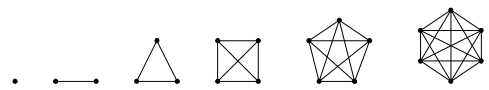
Graf digunakan untuk merepresentasikan hubungan antara objek-objek diskrit. Graf terdiri dari simpul dan sisi.

Graf dapat dibagi menjadi dua jenis berdasarkan ada atau tidaknya sisi ganda. Graf sederhana adalah graf yang tidak mengandung sisi ganda. Graf tidak sederhana adalah graf yang mengandung sisi ganda.

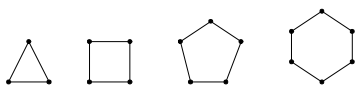
Graf juga dapat dibagi menjadi dua jenis berdasarkan orientasi arah pada sisi. Graf yang tidak memiliki orientasi arah adalah graf tak berarah. Sedangkan graf yang setiap sisinya memiliki orientasi arah adalah graf berarah.

Terminologi graf:

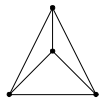
- Ketetanggaan : dua buah simpul bertetangga jika terhubung langsung melalui sisi
- Bersisian : sebuah sisi bersisian dengan simpul jika dibentuk oleh simpul tersebut
- Simpul terpencil : simpul yang tidak memiliki sisi yang bersisian dengannya
- Graf Kosong : graf yang himpunannya adalah himpunan kosong
- Derajat : jumlah sisi yang bersisian dengan suatu simpul
- Lintasan : suatu jalur yang dilalui pada sisi-sisi graf
- Siklus : lintasan yang posisi awalnya sama dengan posisi akhirnya
- Connected : suatu graf dinyatakan terhubung jika ada lintasan yang menghubungkan simpul dari kedua graf
- Upagraf merentang : upagraf yang mengandung semua simpul dari graf asalnya
- Cut-set : sisi yang jika dihilangkan akan membuat graf terhubung menjdai tidak terhubung
- Graf berbobot : graf yang memiliki nilai di tiap sisinya



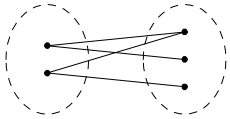
- Graf khusus:



- Graf lengkap : graf yang tiap simpulnya memiliki sisi dengan simpul lainnya
- Graf lingkaran : graf yang tiap simpulnya berderajat dua
- Graf teratur : graf yang tiap simpulnya berderajat sama



- Graf bipartit : graf yang memiliki dua himpunan dan tiap simpul di himpunan tersebut terhubung ke simpul himpunan lainnya



Representasi graf :

- Matriks ketetanggaan : simpul dengan simpul
- Matriks bersisian : simpul dengan sisi
- Senarai ketetanggaan : simpul dengan list simpul tetangganya

Graf yang sama tetapi secara geometrik berbeda disebut graf isomorfik. Dua buah graf yang isomorfik memiliki jumlah simpul yang sama berderajat tertentu dan memiliki jumlah sisi yang sama.

Graf planar adalah graf yang dapat digambarkan pada bidang datar dengan sisi-sisi tidak saling berpotongan. Graf bidang adalah graf planar yang digambarkan sebagai sisi-sisi yang tidak saling berpotongan. Untuk menentukan suatu graf planar atau tidak dapat digunakan ketidaksamaan euler yaitu e<=3n-6, atau untuk $K_{3,3}$ e<=2n-4. Selain itu bisa juga dengan menggunakan teorema Kuratowski.

Lintasan euler adalah lintasan yang melalui tiap sisi sekali. Sirkuit euler adalah sirkuit yang melalui tiap sisi sekali. Graf euler adalah graf yang mengandung sirkuit euler. Graf semi euler adalah graf yang memiliki lintasan euler.

Lintasan Hamilton adalah lintasan yang melalui tiap sisi sekali. Sirkuit Hamilton adalah sirkuit yang melalui tiap simpul sekali, kecuali simpul awal yang juga simpul akhir. Graf Hamilton adalah graf yang mengandung sirkuit Hamilton. Graf semi Hamilton adalah graf yang memiliki lintasan Hamilton.

Aplikasi graf dapat dilihat pada problem lintasan terpendek, persoalan pedagang keliling, persoalan tukang pos Cina, dan pewarnaan graf.

2.2. Lintasan Terpendek

Lintasan terpendek adalah jarak minimal antara simpul. Terdapat beberapa problem dalam lintasan terpendek. Pertama adalah single source shortest path problem, pada problem ini ditentukan lintasan terpendek dari satu sumber ke semua simpul lainnya pada suatu graf. Problem kedua adalah single destination shortest path problem, pada problem ini ditentukan lintasan terpendek antara banyak node sumber ke satu node tujuan. Problem ini adalah kebalikan dari problem pertama. Problem ketiga adalah all pairs shortest path problem, di sini ditentukan lintasan terpendek yang ada di antara semua simpul pada suatu graf.

Untuk menyelesaikan problem lintasan terpendek, dapat digunakan beberapa algoritma. Algoritma pertama adalah Dijkstra's *Algorithm* yang berfungsi untuk menyelesaikan single source shortest path problem. Kemudian Bellman-Ford *Algorithm* untuk menyelesaikan single source shortest path problem dengan sisi yang dapat bernilai negatif. Lalu A* Search *Algorithm* untuk single pair shortest path problem. Algoritma keempat adalah Floyd–Warshall *algorithm* yang berfungsi untuk menyelesaikan all pairs shortest paths problem. Lalu Johnson's *algorithm* untuk menyelesaikan all pairs shortest paths. Dan terakhir adalah Perturbation *theory*. Pada makalah ini, hanya akan digunakan algoritma A* untuk menentukan lintasan terpendek dari ITB ke UNIKOM.

*2.3. Algoritma A**

Algoritma A* ditemukan oleh Peter E. Hart pada 1968. Ia mengoptimalkan kerja dari A2 hasil temuan Bertram Raphael pada 1967 dengan menggunakan *consistent heuristic* dengan hanya sedikit perubahan. A2 adalah hasil dari pengembangan A1 yang ditemukan oleh nils Nilsson pada 1964.

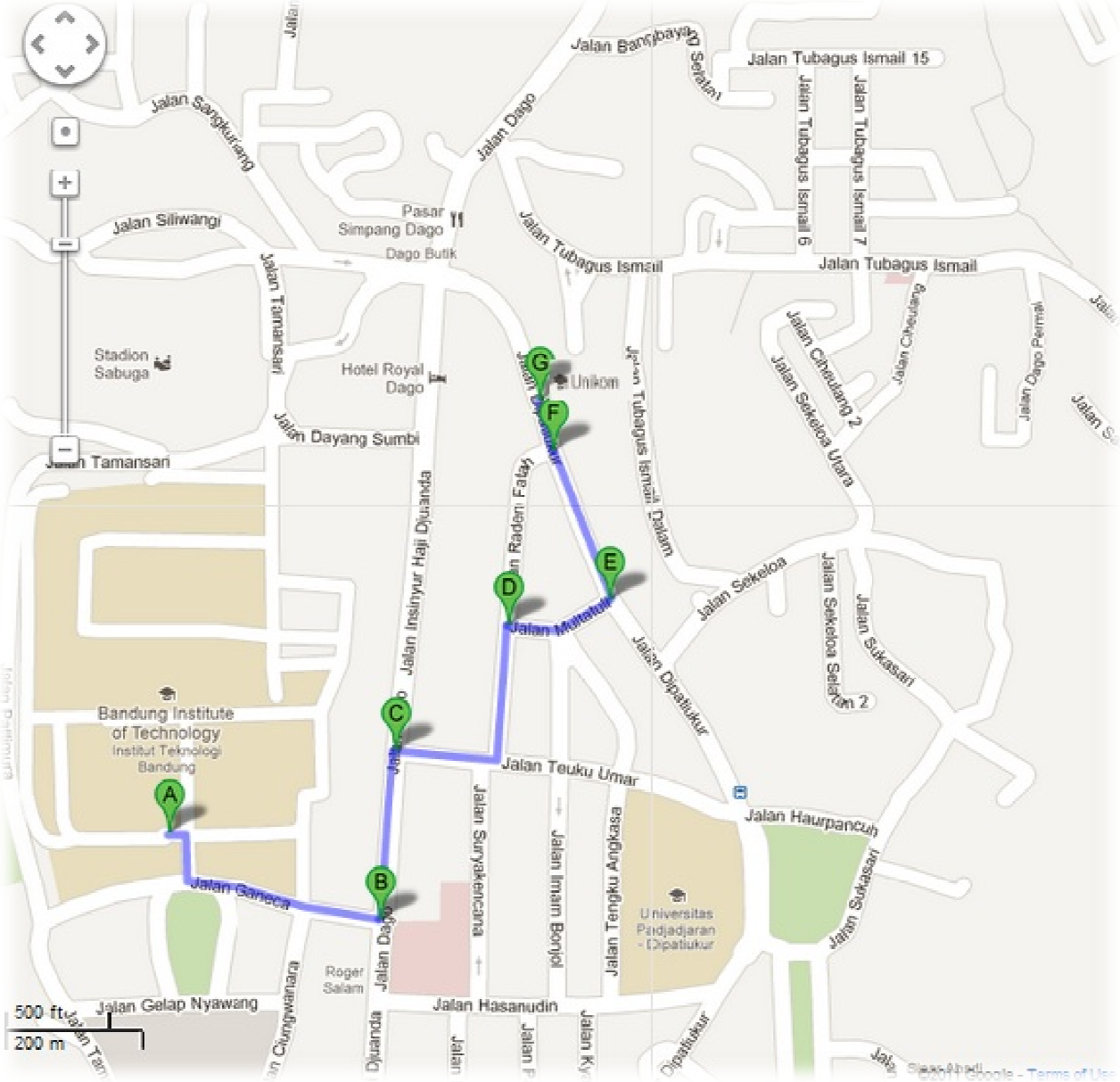
Prinsip dari algoritma ini adalah melakukan traversal satu per satu pada tiap simpul untuk memperoleh lintasan terpendek pada suatu graf. Algoritma A* akan menghitung jarak salah satu lintasan, lalu menyimpannya dan kemudian menghitung jarak lintasan lainnya. Ketika seluruh lintasan telah selesai dihitung, algoritma A* akan memilih lintasan yang paling pendek.

Berikut adalah pseudocode dari A*:

```
function A*(start,goal)
     closedset := the empty set    // The set of nodes already
evaluated.
     openset := {start}    // The set of tentative nodes to be
evaluated, initially containing the start node
     came_from := the empty map    // The map of navigated nodes.

     g_score[start] := 0    // Cost from start along best known path.
     h_score[start] := heuristic_cost_estimate(start, goal)
     f_score[start] := g_score[start] + h_score[start]    // Estimated
total cost from start to goal through y.

     while openset is not empty
         x := the node in openset having the lowest f_score[] value
         if x = goal
             return reconstruct_path(came_from, came_from[goal])

         remove x from openset
         add x to closedset
         foreach y in neighbor_nodes(x)
             if y in closedset
                 continue
             tentative_g_score := g_score[x] + dist_between(x,y)

             if y not in openset
                 add y to openset
                 tentative_is_better := true
             else if tentative_g_score < g_score[y]
                 tentative_is_better := true
             else
                 tentative_is_better := false

             if tentative_is_better = true
                 came_from[y] := x
                 g_score[y] := tentative_g_score
                 h_score[y] := heuristic_cost_estimate(y, goal)
                 f_score[y] := g_score[y] + h_score[y]

     return failure

 function reconstruct_path(came_from, current_node)
     if came_from[current_node] is set
         p := reconstruct_path(came_from, came_from[current_node])
         return (p + current_node)
     else
         return current_node
```

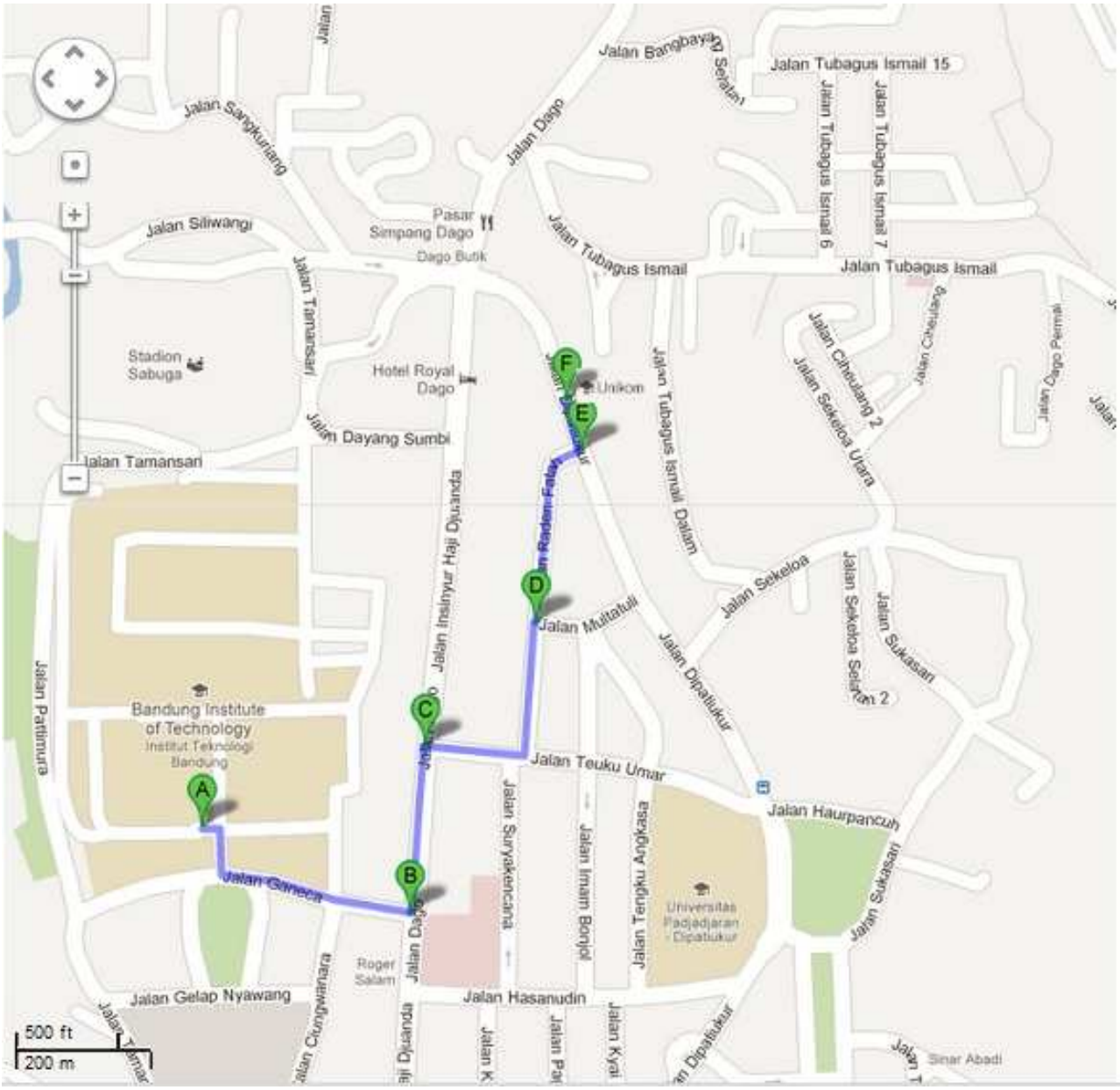
III. GAMBAR PETA DAN KETERANGAN

Pada gambar 1, 2, 3 diberikan peta posisi ITB dan UNIKOM, dengan kemungkinan lintasan yang berbeda.



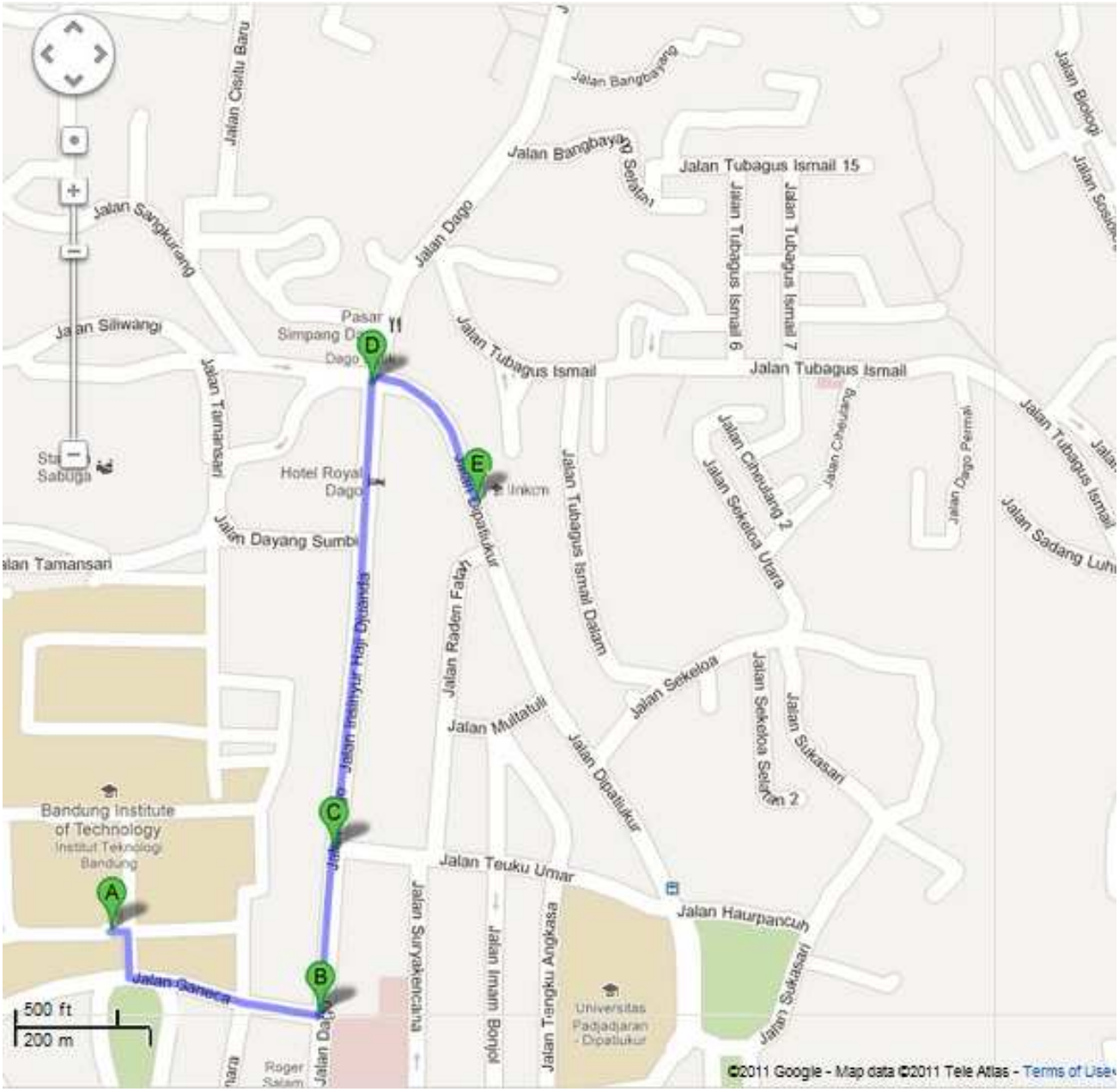
Gambar 1 Lintasan I yang dilalui dari ITB ke UNIKOM dengan jalur A-B-C-D-E-F-G

Pada lintasan I, jalur yang ditempuh adalah A-B-C-D-E-F-G. Jarak A ke B adalah 400 m, B ke C 250 m, C ke D 350 m, D ke E 160 m, E ke F 240 m, dan F ke G 75 m. Sehingga jarak total dari lintasan I adalah 1,5 km.



Gambar 2 Lintasan II yang dilalui dari ITB ke UNIKOM dengan jalur A-B-C-D-E-F

Pada lintasan II, dilalui lintasan dengan jalur A-B-C-D-E-F. Jarak A ke B adalah 400 m, B ke C 250 m, C ke D 350 m, D ke E 290 m, E ke F 75 m, dan F ke G 75 m. Sehingga jarak total dari lintasan II adalah 1,3 km.



Gambar 3 Lintasan III yang dilalui dari ITB ke UNIKOM dengan jalur A-B-C-D-E

Pada lintasan III, dilalui lintasan dengan jalur A-B-C-D-E. Jarak A ke B adalah 400 m, B ke C 250 m, C ke D 700 m, D ke E 270 m. Sehingga jarak total dari lintasan III adalah 1,6 km.

IV. PEMROSESAN DENGAN ALGORITMA A*

Dengan menggunakan algoritma A*, akan diperoleh lintasan terpendek yang dilalui. Pertama-tama simpul sumber adalah simpul A. Kemudian dari A, bergerak ke B, lalu ke C. Sampai di sini semua lintasan masih memiliki jalur yang sama. Kemudian pada pada pemilihan jalur selanjutnya yaitu keempat akan terjadi perbedaan. Algoritma A* kemudian memilih jalur terpendek diantara tiga kemungkinan yang ada yaitu simpul D pada lintasan I atau lintasan II dengan jarak 350 m. Sehingga lintasan III untuk sementara ditinggalkan. Kemudian algoritma A* akan memilih jarak terpendek selanjutnya sebagai simpul ke lima, yaitu simpul E pada lintasan I dengan jarak 160 m. Sehingga sekarang hanya akan dihitung lintasan I. Lalu algoritma A* akan menghitung jarak ke simpul selanjutnya yaitu F dan kemudian G.

Setelah selesai dengan kemungkinan pertama yaitu lintasan I, algoritma A* akan bekerja seperti tadi hanya saja kemungkinan lintasan I sudah tidak diperhitungkan lagi. Kemudian diperoleh panjang masing-masing lintasan yaitu lintasan I 1,5 km, lintasan II 1,3 km, dan lintasan III 1,6 km. Setelah semua kemungkinan lintasan selesai dihitung, algoritma A* akan memperoleh lintasan mana yang paling pendek dari ketiga kemungkinan lintasan tadi, yaitu lintasan II.

V. KESIMPULAN

Untuk memecahkan problem lintasan terpendek dapat digunakan algoritma A*. Algoritma ini memperhitungkan semua kemungkinan lintasan yang ada, lalu kemudian membandingkan kemungkinan tersebut satu demi satu. Pada problem yang telah dibahas diperoleh lintasan II sebagai lintasan terpendek.

REFERENSI

[1] Munir, Rinaldi. 2008. *Struktur Diskrit*. Bandung: Penerbit Institut Teknologi Bandung.

[2] http://maps.google.com/

[3] http://en.wikipedia.org/wiki/Shortest_path_problem

[4] http://en.wikipedia.org/wiki/A*_search_*algorithm*

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, 11 Desember 2011

ttd

Johannes Ridho T. P.
13510103