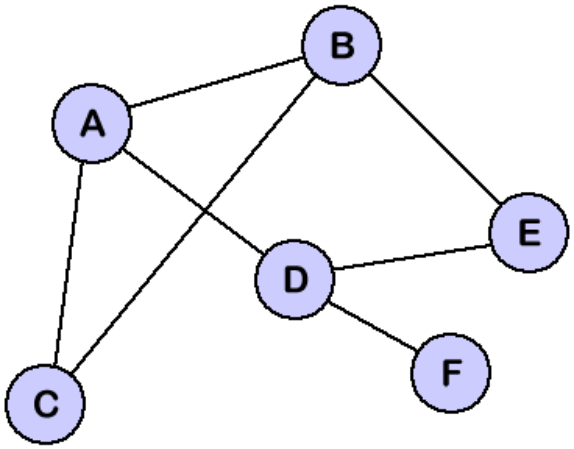
Implementasi Graf dalam Penentuan Rute Terpendek pada Moving Object

Firdaus Ibnu Romadhon/13510079
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
Fibronacci_vistar@yahoo.co.id

Makalah ini berjudul "Implementasi Graf dalam Penentuan Rute Terpendek pada Moving Object". Ini berbeda dengan menentukan pada objek yang diam. Untuk mecari rute terpendek pada objek diam, tinggal mencari dengan algoritma greedy saja, namun untuk menentukan objek bergerak diperlukan penanganan khusus untuk hal ini. Dalam makalah ini akan dijelaskan salah satu metode untuk mencari rute terpendek pada moving objek, yaitu dengan metode Count and Move. Metode ini mencari semua rute yang dapat dilewati oleh subjek menuju objek kemudian menentukan rute terpendeknya dengan menghitung jumlah simpul pada graf yang dilewati. Semakin sedikit simpul pada graf tersebut, semakin pendek rute yang dihasilkan. Dari makalah ini dapat disimpulkan bahwa untuk mencari rute terpendek perlu didasarkan pada kebutuhan, apakah graf yang digunakan kompleks atau tidak.

I. PENDAHULUAN

Dalam kehidupan sehari-hari, mungkin tidak akan terlepas dari pemilihan jalan mana yang paling efektif untuk dikerjakan dibandingkan jalan lain. Boleh jadi tujuan yang ingin dicapai adalah sama, tetapi akan lebih efisien kalau jalan yang dilalui lebih mudah untuk ditempuh. Sebagai contoh adalah ketika hendak pergi dari kota A menuju kota B, secara naluri orang akan memilih rute yang terpendek. Itu tidak salah, namun juga perlu diperhitungkan aspek-aspek lainnya yang akan menghambat perjalanan kita seperti factor macet, jalanan yang tidak rata, dan lain-lain. Permasalahan timbul ketika tujuan yang akan dicapai tersebut bergerak (*moving object*). Ketika sudah ditemukan jalur terpendek dan ternyata tujuan berubah, maka harus dilakukan pencarian ulang jalan terbendek tersebut. Metode semacam ini sering dipakai pada game-game seperti Pac Man, Bomberman, atau game-game RPG di mana musuh akan berusaha mengejar tokoh utama.



Gambar 1. Contoh graf

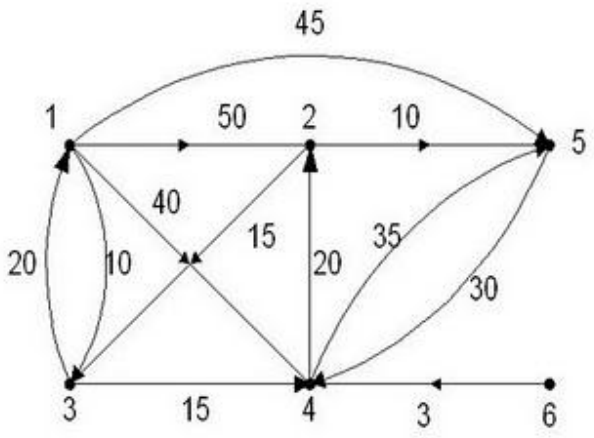
Graf merupakan himpunan dari objek-objek seperti titik, simpul, atau sudut dihubungkan oleh penghubung yang dinamakan garis atau sisi. Graf G didefinisikan sebagai pasangan himpunan (V,E), ditulis dengan notasi G = (V,E), yang dalam hal ini V adalah himpunan tidak kosong dai simpul-simpul dan E adalah himpunan sisi yang menghubungkan sepasang simpul. Graf digunakan untuk merepresentasikan objek-objek diskrit dan nada hubungannya dengan objek-objek tersebut.

Beberapa istilah umum dalam graf yaitu:

1. Bertetangga, yaitu dua buah simpul yang terhubung langsung dengan sebuah sisi.
2. Bersisian, yaitu sisi-sisi yang membentuk simpul tersebut.
3. Simpul terpencil, yaitu simpul yang tidak memiliki sisi yang bersisian dengannya.
4. Graf kosong, yaitu graf yang himpunan sisinya merupakan himpunan kosong.
5. Derajat, yaitu jumlah sisi yang bersisian dengan simpul tersebut.
6. Sirkuit atau siklus, yaitu lintasan yang berawal dan berakhir pada simpul yang sama.
7. Upagraf, yaitu graf yang masih merupakan bagian dari graf lain.
8. Graf berbobot, yaitu graf yang setiap sisinya diberi sebuah harga.
9. Graf planar, yaitu graf yang bla digambarkan

dalam bidang dua dimensi, sisi-sisinya tidak saling memotong.

10. Graf isomorfik, yaitu dua graf yang sama namun secara geometri berbeda.



Gambar 2. Graf berbobot

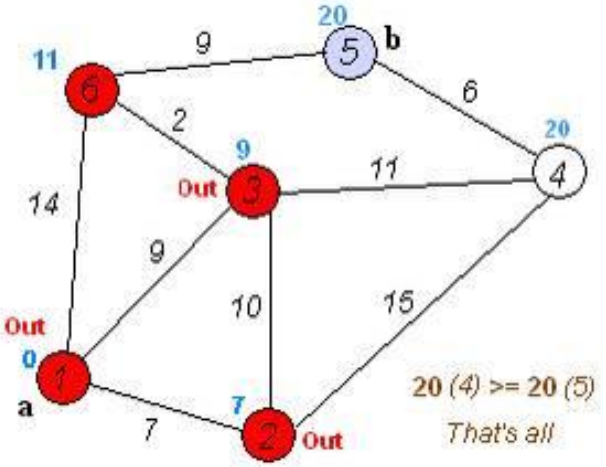
II. Pencarian Rute Terpendek

Persoalan lintasan terpendek yaitu menemukan lintasan terpendek antara dua atau beberapa simpul lebih yang berhubungan. Persoalan mencari lintasan terpendek di dalam graf merupakan salah satu persoalan optimasi. Persoalan ini biasanya direpresentasikan dalam bentuk graf. Graf yang digunakan dalam pencarian lintasan terpendek atau shortest path adalah graf berbobot, 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 digunakan di sini adalah bahwa semua bobot bernilai positif. Kata "terpendek" jangan selalu diartikan secara fisik sebagai panjang minimum, sebab kata "terpendek" berbeda-beda makanya bergantung pada tipikal perasoalan yang akan diselesaikan. Namun, secara umum "terpendek" berarti meminimalisasi bobot pada suatu lintasan di dalam graf. Dalam sejarahnya, terdapat beberapa metode pencarian rute terpendek dengan menggunakan graf, diantaranya adalah algoritma Dijkstra dan algoritma Bellman-Ford.

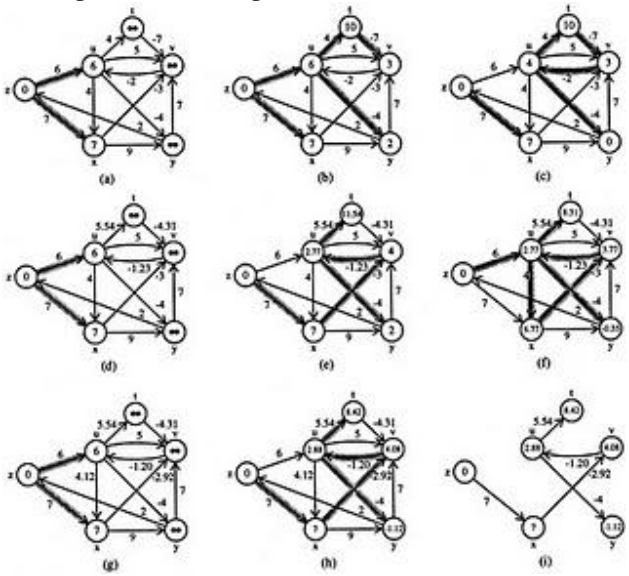
Algoritma Dijkstra ditemukan oleh seorang ilmuwan computer, Edsger Dijkstra, adalah sebuah algoritma yang dipakai dalam memeahkan permasalahan jalan terpendek dengan menggunakan sebuah graf berarah dengan bobot sisi yang bernilai positif. Cara kerja algorritma ini memakai strategi *greedy* yaitu setiap langkah yang dipilih sisi dengan bobot terkecil yang menghubungkan sebuah simpul simpul terpilih dengan simpul lain yang belum terpilih. Algoritma dijkstra adalah sebagai berikut:

```
function Dijkstra(G, w, s)
    for each vertex v in V[G]                    // Initializations
        d[v] := infinity
        previous[v] := undefined
    d[s] := 0                                    // Distance from s to s
    S := empty set
    Q := V[G]                                    // Set of all vertices
    while Q is not an empty set                  // The algorithm itself
        u := Extract_Min(Q)
        S := S union {u}
        for each edge (u,v) outgoing from u
            if d[u] + w(u,v) < d[v]              // Relax (u,v)
                d[v] := d[u] + w(u,v)
                previous[v] := u
```



Gambar 3. Algoritma Dijkstra

Algoritma Bellman-Ford sama seperti pada algoritma Dijkstra, digunakan untuk mencari jalan terpendek dari sebuah graf namun algoritma ini hanya bisa digunakan dalam graf berbobot negative.



Gambar 4. Algoritma Bellman-Ford

Notasi algoritma Bellman-Ford adalah sebagai berikut:

```
record vertex {
   list edges
   real distance
   vertex predecessor
}
record edge {
   node source
   node destination
   real weight
}

function BellmanFord(list vertices, list
edges, vertex source)
/* pengimplementasian terhadap graf yang
direpresentasikan oleh list of simpul dan
sisi, dan mengubah simpul sehingga jarak
dan predesesornya menyimpan jarak
terpendek */

// inisialisasi graf
for each vertex v in vertices:
   if v is source then
      v.distance = 0
   else
      v.distance := infinity
      v.predecessor := null

// periksa setiap simpul
for i from 1 to size(vertices):
   for each edge uv in edges:
      u := uv.source
      v := uv.destination
// uv is the edge from u to v
      if v.distance>u.distance+uv.weight
         v.distance:=u.distance+uv.weight
         v.predecessor := u

// mencari loop yang berbobot negatif
for each edge uv in edges:
   u := uv.source
   v := uv.destination
   if v.distance > u.distance + uv.weight
      error "Graf mengandung loop negatif"
```

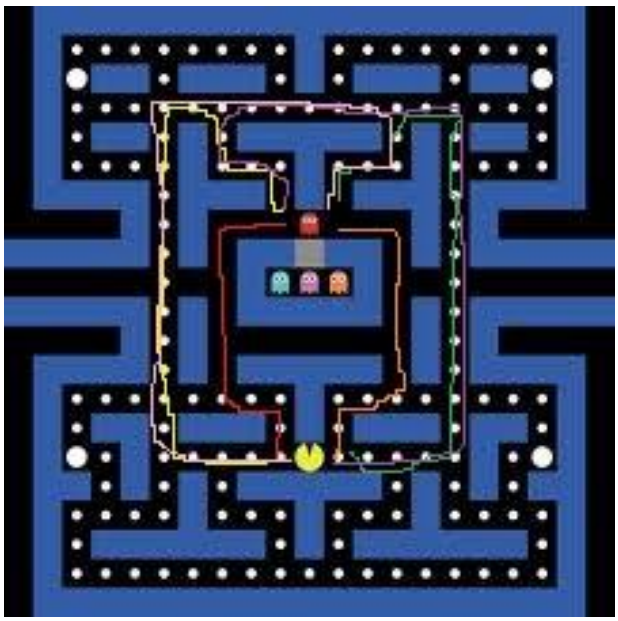
III. TEORI COUNT AND MOVE

Baik algoritma Dijkstra maupun algoritma Bellman Ford digunakan untuk mencari rute terpendek dari sebuah simpul ke simpul lainnya. Artinya, simpul tujuan tersebut akan diam di situ selamanya dan tidak akan bergerak. Jika tiba-tiba objek tujuannya berpindah tempat, perhitungan akan diulang dan ini akan sangat merepotkan jika perpindahan tempatnya lebih cepat dibandingkan proses program tersebut berjalan. Hal ini mungkin tidak akan terjadi jika grafnya masih berbentuk sederhana. Akan tetapi jika grafnya sudah kompleks dan hampir menyerupai maze (labirin) program akan terlambat berjalan dan hasilnya tidak akan sesuai dengan harapan. Salah satu cara yang dapat digunakan untuk mencari rute tercepat dari objek yang bergerak adalah metode *Count and Move*.

Metode ini menggunakan algoritma untuk menghitung jarak subjek dengan objek dahulu dari beberapa rute. Jika sudah menemukan rute terpendek, subjek akan bergerak melalui rute tersebut. Untuk lebih mudahnya, penulis akan menjelaskannya melalui ilustrasi sebuah game yang menggunakan sistem *moving object*, yaitu Pac Man.

Gambar 5. Peta game Pac Man

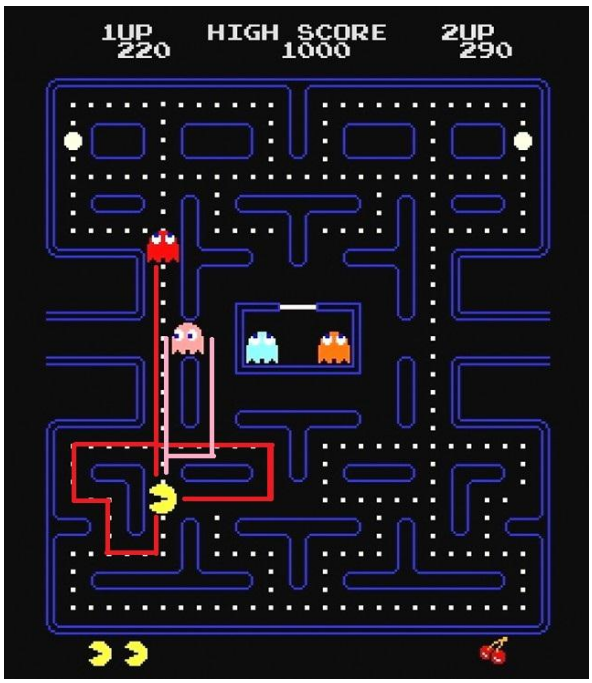
Dalam hal ini, yang akan menjadi subjeknya adalah hantu yang akan mengejar sebuah *moving object* yaitu Pac Man. Pertama-tama, hantu harus menentukan terlebih dahulu semua rute yang dapat digunakan untuk mengejar Pac man. Rute tersebut harus merupakan lintasan Hamilton, yaitu tidak boleh melewati simpul yang sama untuk kedua kalinya. Simpul direpresentasikan dengan titik-titik putih pada peta Pac Man di atas. Perlu diingat bahwa walaupun titik tersebut sudah tidak ada karena dimakan, namun simpul tersebut masih ada secara dalam algoritma.



Gambar 6. Peta yang telah dicari rutenya

Setelah pencarian rute selesai, dimulailah perhitungan rute mana yang paling pendek, dalam hal ini adalah rute dengan warna merah. Pertama yang harus dilakukan adalah mencari apahak di antara rute-rute tersebut terdapat simpul di mana dilewati oleh dua atau lebih rute yang sama. Jika simpul tersebut sudah ditentukan, maka rute yang dieliminasi adalah rute yang paling banyak melewat simpul dari titik awal menuju simpul tersebut. Jika sudah ditentukan jarak terpendeknya, hantu akan bergerak melalui rute tersebut. Setiap pergerakan dari satu simpul ke simpul lainnya harus tetap dicari rute yang baru dan dihitung jarak masing-masing rute agar jika Pac Man bergerak menghindar, hantu akan langsung meresponnya dengan menentuka rute baru atau akan melalui rute yang sama. Metode ini efektif untuk berbagai macam tipe peta, termasuk peta yang berbentuk maze yang sangat rumit.

Metode di atas memang efektif untuk mematikan Pac Man, akan tetapi kompleksitas algoritmanya akan membuat lebih lambat sehingga diperlukan metode merhitungan jarak yang baru. Perhitungan jarak ini akan membuat program berjalan lebih cepat, namun tidak efektif dalam menentukan jarak terpendeknya terutama ketika harus dihadapkan pada permasalahan maze. Cara lain tersebut yaitu dengan menentukan arah objek tersebut. Jika melihat Gambar 5, dapat ditentukan bahwa posisi Pac Man terdapat di bawah. Karena itu, rute yang ditetapkan adalah hanya warna merah dan jingga. Sama seperti sebelumnya, pada setiap langkah hantu, dicari semua rute dan panjang lintasannya.



Gambar 7. Metode pencarian rute dengan arah

IV. KESALAHAN UMUM

Terlepas karena metode ini merupakan hasil pemikiran penulis sendiri, metode ini masih memiliki banyak kekurangan. Yang pertama, penulis belum bisa menulis algoritma konkrit dalam pencarian rute terdekat ini. Penulis hanya bisa memberikan konsep atau ide pencariannya saja. Yang kedua, algoritma ini memerlukan pencarian berulang ketika subjek berpindah menuju simpul berikutnya. Yang ketiga, untuk peta yang kompleks seperti maze, pencarian rute untuk metode pertama memerlukan waktu yang lama sedangkan untuk metode kedua dapat terjebak ke jalur yang lebih panjang karena pada dasarnya rute dalam maze tidak bisa ditebak.

V. KESIMPULAN

Pada dasarnya, mencari rute terdekat dari sebuah graf bergantung pada kebutuhan dan kompleksitasnya. Jika kebutuhannya adalah untuk mencari rute tercepat dalam maze, maka metode mencari semua rute adalah pilihan tepat. Akan tetapi jika graf yang digunakan tidak terlalu kompleks, maka setiap simpul diberi info koordinat sehingga bisa langsung mengetahui koordinat target dan menentukan rutenya.

REFERENSI

[1] Rosen, Kenneth H, *Discrete mathematics and Its pplications 5th Ed.* New York: McGraw-Hill, 2003.

[2] Munir, Rinaldi. *Matematika Diskrit 3th Ed.* Bandung: Palasari

[3] Wikipedia, http://id.wikipedia.org/wiki/Teori_graf.12 Desember 2011. Pukul 08.00

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, 12 Desember 2011

ttd

Firdaus Ibnu Romadhon/13510079