Penggunaan Algoritma *Greedy* Dalam Penentuan Rute Wisata

Renusa Andra Prayogo (13511063)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13511063@std.stei.itb.ac.id

***Abstrak*—Penggunaan algoritma pencarian jarak terdekat antara dua titik (*shortest path problem*) sudah sangat banyak diterapkan pada kehidupan sehari-hari. Hal ini didorong dengan semakin berkembangnya perangkat mobile saat ini, yang memungkinkan penggunaan algoritma tersebut untuk menentukan rute untuk mencapai suatu tujuan dengan memanfaatkan teknologi GPS. Namun, pada saat ini belum banyak aplikasi dari algoritma greedy untuk menentukan rute perjalanan yang bukan berdasarkan jaraknya, tetapi berdasarkan rute wisata. Sehingga dapat digunakan oleh penggunanya saat melakukan perjalanan jauh antar kota.**

Indeks— Algoritma Greedy, shortest path, rute wisata, kota wisata

I. PENDAHULUAN

Latar belakang

Pada saat ini, sudah banyak aplikasi-aplikasi yang menerapkan fitur pencarian rute dari suatu titik ke posisi tujuan. Contohnya pada komputer adalah Google Maps. Kemudian karena semakin berkembangnya platform mobile, maka semakin banyak pula aplikasi yang dapat berjalan pada device mobile tersebut, seperti Sygic, nDrive, Waze dan sebagainya. Karena aplikasi tersebut berjalan pada mobile device, maka sebagian besar aplikasi tersebut sudah menggunakan modul GPS untuk menentukan posisi tepat dari device tersebut. Karena keakuratan yang diberikan oleh GPS, maka banyak aplikasi tersebut digunakan untuk mencari lokasi tujuan-tujuan seperti restoran, mall, rumah sakit, pom bensin dan sebagainya. Sedangkan pada umumnya untuk menentukan rute tersebut, pada umumnya aplikasi tersebut menggunakan algoritma dengan memperhitungkan jaraknya, kemacetan jalur, dan lebar jalurnya.

Dari contoh-contoh aplikasi diatas kebanyakan penerapannya digunakan pada daerah perkotaan, dan tidak banyak dari aplikasi tersebut yang ditujukan untuk menentukan rute jarak jauh antar kota ke kota lainnya sehingga hanya mampu menentukan rute terdekat antar kota tersebut.

Untuk perjalanan jauh, umumnya aplikasi tersebut hanya menginformasikan informasi penting seperti pom bensin, masjid, restoran terdekat dari posisi saat ini. Namun, untuk informasi rute tentang objek wisata terdekat, belum banyak aplikasi yang menerapkan hal tersebut. Dengan adanya informasi objek wisata saat perjalanan, maka pengguna dapat menentukan jalan yang efektif untuk mencapai tujuan sekaligus mengunjungi objek wisata terdekat.

Tujuan

Pada paper ini, dengan menggunakan algoritma greedy sebagai dasar, penulis bertujuan untuk dapat menemukan solusi algoritma untuk permasalahan penentuan rute wisata, dan menentukan apakah dengan algoritma tersebut dapat menghasilkan rute yang tidak terlalu jauh dari solusi optimal.

Untuk permasalahan ini, akan dibuat juga implementasinya dalam bentuk aplikasi bahasa Java, yang dapat mensimulasikan setidaknya lima kota dan tiga objek wisata.

II. DASAR TEORI

Algoritma Greedy

Algoritma greedy merupakan jenis algoritma yang menggunakan pendekatan penyelesaian masalah dengan mencari nilai maksimum sementara pada setiap langkahnya. Nilai maksimum sementara ini dikenal dengan istilah local maximum. Pada kebanyakan kasus, algoritma greedy tidak akan menghasilkan solusi paling optimal, begitupun algoritma greedy biasanya memberikan solusi yang mendekati nilai optimum dalam waktu yang cukup cepat.

Karena sifat greedy tersebut, maka algoritma greedy seringkali dianggap sebagai algoritma berkarakteristik "short sight" dan "non-recoverable". Oleh karena itu pula, algoritma greedy baiknya digunakan pada permasalahan yang tidak mementingkan solusi optimum, dan cocok untuk masalah-masalah sederhana.

Secara umum, algoritma greedy memiliki tiga jenis variasi, yaitu :

- *Pure greedy algorithm*
- *Orthogonal greedy algorithm*
- *Relaxed greedy algorithm*

Namun, pada paper ini yang akan dibahas hanyalah algoritma greedy yang paling dasar, yaitu *Pure greedy algorithm*.

Pada Algoritma Greedy, terdapat lima elemen yang harus ditentukan, yaitu

1. Himpunan kandidat, C,
2. Himpunan solusi, S,
3. Fungsi seleksi,
4. Fungsi kelayakan, dan
5. Fungsi obyektif.

Himpunan kandidat adalah himpunan semua kondisi yang dapat terjadi pada ruang persoalan, sedangkan himpunan solusi adalah himpunan bagian dari himpunan kandidat yang memenuhi kondisi yang ditentukan. Kemudian fungsi seleksi adalah fungsi yang menentukan kondisi mana yang terbaik pada himpunan solusi, dan fungsi kelayakan adalah fungsi yang memeriksa apakah solusi yang didapat masih terdapat pada konstrain yang diberikan.

Dari kelima elemen tersebut, dapat dikatakan bahwa algoritma greedy adalah sebuah algoritma yang melibatkan pencarian sebuah himpunan bagian, S, dari himpunan kandidat, C; yang dalam hal ini, S harus memenuhi beberapa kriteria yang ditentukan, yaitu menyatakan suatu solusi dan S dioptimisasi oleh fungsi obyektif.

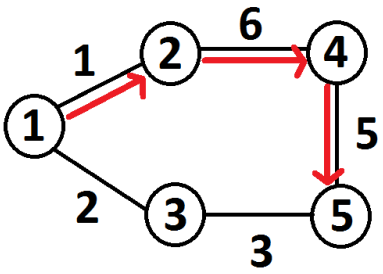
Berikut ini adalah skema umum dari algoritma greedy :

```
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
```

Contoh permasalahan yang dapat diselesaikan dengan algoritma greedy antara lain :

- *Coin change problem*
- *Knapsack problem*
- *Graph coloring problem*
- *Job assignment problem*
- *Prim Kruskal*
- *Dijkstra algorithm*
- *Huffmann algorithm*
- *Shortest path problem*

Penerapan algoritma greedy yang dibahas pada paper ini adalah adalah penentuan jarak terdekat dari suatu titik ke titik lain (*Shortest path problem)*, yang merupakan salah satu contoh penerapan algoritma greedy seperti yang dituliskan diatas. Misalkan pada permasalahan ini terdapat lima kota seperti dibawah ini :

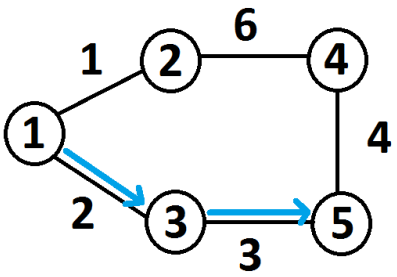


Gambar 2.1 Rute dengan algoritma greedy

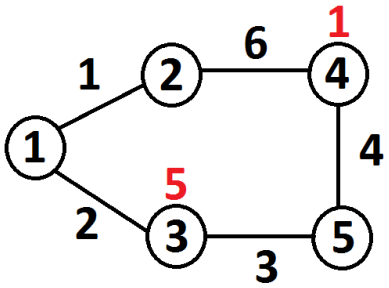
Pada permasalahan diatas, algoritma greedy diminta untuk menentukan rute dari titik 1 ke titik 5. Dari permasalahan tersebut, ditentukan lima elemen algoritma greedy sebagai berikut :

1. Himpunan kandidat, C
   Himpunan semua rute yang dapat dilalui.
2. Himpunan solusi, S
   Himpunan rute yang dapat mengarah pada titik 5.
3. Fungsi seleksi
   Memilih rute terdekat yang dapat dilalui dari posisi saat ini.
4. Fungsi kelayakan
   Memeriksa apakah rute yang dituju sudah pernah dikunjungi.
5. Fungsi obyektif
   Mencapai tujuan (titik 5) dengan jarak terdekat.

Dari algoritma tersebut, didapatkan rute yang dilalui berupa 1-2-4-5 dengan total jarak 12. Karena pada saat pemilihan jalurnya greedy hanya melakukan perhitungan secara lokal, maka mungkin saja solusinya tidak optimal. Pada permasalahan ini, algoritma greedy tidak dapat memberikan solusi yang tepat, karena solusi yang optimal pada permasalahan ini adalah dengan menggunakan rute 1-3-5 dengan jarak 5, seperti pada gambar 2.2 dibawah ini.



Gambar 2.2 Solusi optimum untuk contoh 2.1



Gambar 3.1 Ilustrasi rute

Secara umum, algoritma yang akan digunakan pada paper ini mirip dengan yang dijelaskan pada contoh diatas, namun dengan sedikit perubahan untuk menyesuaikan penggunaan algoritma ini pada topik paper ini.

III. Analisis Pemecahan Masalah

Tujuan dari algoritma greedy pada permasalahan ini adalah untuk menentukan rute yang terbaik dari suatu titik ke titik lain dengan mempertimbangkan jarak tempuh dan tempat wisata yang dikunjungi.

Pada algoritma ini, untuk mendapatkan solusi akan dilakukan dua kali perhitungan greedy, yaitu greedy berdasarkan jarak terdekat, dan greedy berdasarkan tempat wisata. Pada perhitungan greedy berdasarkan tempat wisata, selain diperhitungkan jaraknya juga diperhitungkan juga "ranking" dari tempat wisata tersebut, sehingga setiap tempat wisata memiliki dapat "bobot" masing-masing yang akan digunakan pada fungsi seleksi pada algoritma greedy. Solusi umum dari greedy ini akan ditentukan dari hasil kedua algoritma tersebut. Oleh karena itu, elemen dari algoritma ini adalah sebagai berikut :

1. Himpunan kandidat, C
   Himpunan semua rute yang dapat dilalui.
2. Himpunan solusi, S
   Himpunan rute yang dapat mengarah pada titik tujuan..
3. Fungsi seleksi
   Memilih rute terdekat yang dapat dilalui dari posisi saat ini atau titik yang memiliki tempat wisata. (detil lebih lanjut dijelaskan pada bagian implementasi).
4. Fungsi kelayakan
   Memeriksa apakah rute yang dituju sudah pernah dikunjungi.
5. Fungsi obyektif
   Mencapai tujuan dengan jarak terdekat dan dengan melalui tempat wisata sebanyak mungkin.

Sebagai ilustrasi, seperti pada gambar 3.1 diatas terdapat lima titik yang berperan sebagai kota, yang dua diantaranya terdapat bilangan berwarna merah diatasnya. Dua kota tersebut pada ilustrasi diatas adalah kota yang memiliki tempat wisata, sedangkan kota lainnya yang tidak terdapat bilangan merah tersebut diasumsikan tidak memiliki tempat wisata. Nilai bilangan tersebut berkisar dari satu(1) hingga lima (5) dengan 5 berarti tempat wisata tersebut paling menarik("ranking" paling tinggi) dan 1 berarti paling tidak menarik.

Berdasarkan elemen-elemen yang telah ditentukan diatas, maka dari algoritma yang digunakan akan mneghasilkan rute 1-3-5. Perbedaan antara algoritma standar (pada gambar 2.1) dengan algoitma ini adalah pada saat berada pada titik 1, algoritma juga mempertimbangkan titik 3 yang selain terhubung dengan titik 1, titik tersebut juga memiliki tempat wisata. Kemudian algoritma juga membandingkan perbedaan jarak antara titik 2 ke titik 5 dengan titik 3 ke titik 5. Untuk perhitungan ini, tempat wisata tidak diperhitungkan, sehingga hanya perhitungan jarak saja. Dari data yang didapat tersebut (titik mana yang terhubung, jarak titik tersebut dengan titik akhir, dan ada/tidaknya tempat wisata pada titik-titik tersebut) maka ditentukanlah rute yang dilalui. Pada contoh ini, titik yang dipilih adalah titik 3.

IV. Implementasi dan Pengujian

Implementasi

Sebagai implementasi dari algoritma yang dijelaskan pada bab III, maka dibuatlah implementasinya dalam bahasa Java. Pada implementasi ini, terdapat tiga bagian utama pada implementasi ini, yaitu bagian objects, algorithm, dan main. Pada objects terdapat dua kelas, yaitu kelas City dan Site. Pada algorithm terdapat kelas Greedy yang didalamnya implementasi dari algortima greedy Berikut ini adalah struktur data dari kelas City :

```
public class City {
// Atribut
	private ArrayList<City> neighbors;
	private ArrayList<int> distance;
	private ArrayList<Site> sites;
	private int rate;
	private String name;
	private boolean hasSite;
```

```
// Konstruktor
      public City();

// Method
      private int calculateRate();
}
```

Pada kelas tersebut terdapat atribut sebagai berikut :

- Neighbor : atribut yang menyimpan array City yang merepresentasikan kota-kota yang berhubungan langsung dengan kota tersebut.
- Distance : atribut yang menyimpan array integer berupa jarak dari kota tersebut ke kota-kota yang bersebelahan.
- Sites : Atribut menyimpan daftar objek wisata yang terdapat pada kota tersebut.
- Rate : Atribut yang menyimpan total rate dari seluruh objek wisata yang ada di kota.
- Name : nama objek kota tersebut
- hasSite : attibut boolean yang menyimpan status apakah kota tersebut memiliki objek wisata.

```
public class Site {
// Atribut
      private String name;
      private int rate;

// Konstruktor
      public Site();
}
```

Pada kelas Site hanya terdapat dua atribut, yaitu name dan rate. Atribut name menyimpan nama dari objek wisata dan rate menentukan tingkat rating dari objek wisata tersebut.

Berikutnya adalah kelas greedy. Pada kelas ini diimplementasikan seluruh algoritma yang diperlukan pada algoritma greedy yang digunakan berikut ini adalah pseudocode dari algoritma yang digunakan :

function run(input start:string, dest:string) → integer
{ *Mengembalikan nilai 0 jika greedy berhasil, dan mengembalikan nilai -1 jika gagal.*
Masukan : nama kota awal, nama kota tujuan
Keluaran : integer untuk menunjukkan apakah operasi berhasil atau gagal
}

Deklarasi
visited : array of City
finished : boolean
startNode : City
selectedCity : City
tempRate : integer

Algoritma
startNode ← getFirstCity(start)
visited[0] ← startNode
while(!finished)
for (setiap kota yang dapat dikunjungi)
if (temperate > calculateRank(startNode, i)
selectedCity ← getNeighbors(startNode,i)
temperate ← calculateRank(startNode, i)
endif
endfor
startNode ← selectedCity
visited[n] ← selectedCity

if (reachedDestination(getName(selectedCity)))
→ 0
endif
endwhile
→ -1

Pada implementasi diatas, pemilihan kota yang dipilih ditentukan berdasarkan fungsi seleksi berupa calculateRank dan fungsi pembatas berupa variabel visited. Fungsi calculateRank mengembalikan nilai berupa jarak yang dikombinasikan dengan perhitungan dengan objek wisata yang dimiliki kota tersebut. Kemudian fungsi pembatasnya berupa variabel visited, bila sudah dikunjungi maka kota tersebut tidak boleh dikunjungi kembali. Berikut ini adalah fungsi seleksi yang digunakan pada algoritma diatas :

function calculateRank(input now:City, dest:integer) → integer
{ *Mengembalikan nilai integer berupa jarak yang dikalikan dengan faktor objek wisata dengan rumus :*
Jarak/total rank
Masukan : Kota saat ini, id kota tujuan
Keluaran : integer untuk menunjukkan jarak dengan dikalikan faktor objek wisata
}

Deklarasi
totalRate : integer

Algoritma
totalRate ← rate(now, dest)
→ jarak(now, dest) / totalRate

Pada fungsi seleksi diatas, setiap pilihan kota ditentukan berdasarkan jaraknya dari kota saat ini. Namun jarak tersebut juga diperhitungkan dengan faktor objek wisata dengan rumus sebagai berikut :

Final Distance = Distance / total rate sites

Untuk fungsi kelayakan yang digunakan adalah fungsi contains, yang mengembalikan nilai true apabila kota yang ingin dikunjungi sudah pernah dikunjungi sebelumnya. Fungsi ini diperlukan untuk memastikan bahwa algoritma tidak akan mengarahkan ke tempat yang sama berulang kali (berputar-berputar terus).

function contains(input visited:array of City, newCity:City) → boolean
{ *Berfungsi sebagai fungsi pembatas dengan memastikan*

bahwa kota yang akan dikunjungi belum pernah dikunjungi sebelumnya.

Masukan : Array kota yang sudah dikunjungi, kota yang ingin diperiksa

Keluaran : mengembalikan nilai boolean true apabila objek newCity terdapat pada array visited.

}

Deklarasi

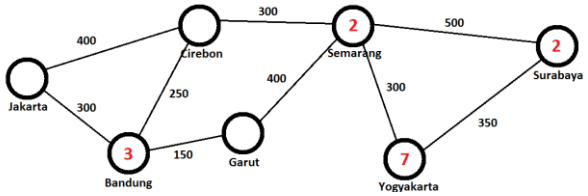
i : integer

Algoritma

```
for (i = 0 to visited array size)
    if (visited[i] = newCity)
        → true
    endif
endfor
```

Pengujian

Pada pengujian ini, digunakan contoh rute sebagai berikut ini :



Gambar 4.1 Contoh rute untuk pengujian

Pada contoh diatas, total terdapat tujuh buah kota, dan empat diantaranya merupakan kota dengan objek wisata. Untuk pengujian ini, dilakukan pencarian rute dari kota "Jakarta" ke kota "Surabaya", dengan perintah sebagai berikut :

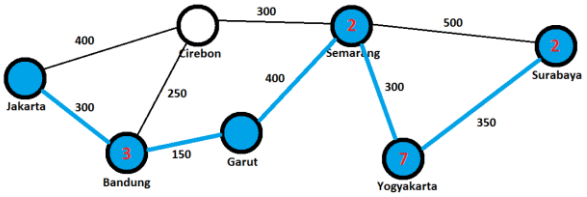
```
greedy.Run("Jakarta", "Surabaya");
```

Dari contoh diatas, didapatkan rute yang didapat dari algoritma greedy diatas :

Output console

```
Go to Bandung
Visit Tangkuban parahu
Go to Garut
Go to Yogyakarta
Visit Keraton
Visit Pantai parangtritis
Go to Semarang
Visit Ambarawa
Go to Surabaya
Visit Tugu pahlawan
Reached destination!
Total distance : 1400 km
Total visits : 5
```

Rute yang dihasilkan :

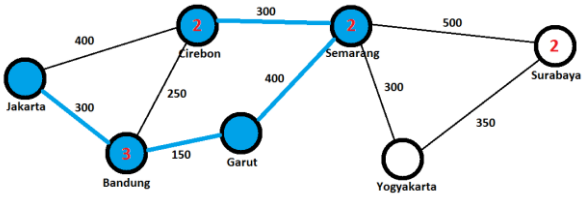


Gambar 4.2 Ilustrasi rute yang dihasilkan

Dari rute yang dihasilkan diatas, ditunjukkan bahwa algoritma yang diimplementasi dapat menentukan rute dari posisi awal ("Jakarta") ke posisi tujuan ("Surabaya"). Dari algoritma greedy tersebut didapatkan rute dari Jakarta-Surabaya dengan jarak total sebesar 1400 km dengan melalui objek wisata. Jika dibandingkan dengan algoritma brute force (jarak optimal) :

Algoritma greedy : 1400 km, 5 objek wisata
Algoritma brute force : 1200 km, 2 objek wisata

Namun, dari hasil pengujian dengan contoh sampel yang berbeda (seperti gambar 4.3), ditemukan bahwa algoritma dapat mengalami kesalahan sehingga tidak dapat menentukan rute ke tujuan. Hal ini diakibatkan oleh kondisi dimana algoritma bergerak ke kota yang seluruh kota yang bisa didatanginya sudah dikunjungi sebelumnya, maka pada kondisi tersebut program akan berhenti dan gagal mencapai tujuan. Oleh karena itu, untuk mengatasi kondisi seperti diatas, maka pada algoritma harus diberikan validasi tambahan agar algoritma tersebut tidak mengarahkan dirinya pada kota "buntu" dimana bila sampai pada kota tersebut algoritma tidak dapat bergerak ke kota lainnya.



Gambar 4.3 Contoh rute yang gagal diselesaikan

Seperti gambar diatas, seharusnya algoritma dapat menentukan rute dari "Jakarta" ke "Surabaya", namun pada contoh diatas algoritma berhenti bergerak pada kota "Cirebon". Hal ini diakibatkan karena pada saat algoritma sampai pada kota "Semarang", algoritma ini akan memilih jalur ke kota "Cirebon" walaupun jika sampai pada kota tersebut algoritma akan berhenti berjalan.

V. KESIMPULAN

Dari analisis dan pengujian yang telah dilakukan, maka didapati bahwa algoritma greedy yang diterapkan dapat menentukan rute antara suatu titik ke titik yang lainnya dengan berhasil.

Jika dibandingkan dengan algoritma brute force (untuk mendapat solusi jarak optimal), rute yang dihasilkan tidak

terlalu jauh (pada pengujian terdapat perbedaan 200 km) dan dapat menentukan rute yang melalui lebih banyak objek wisata (pada pengujian greedy mendapat 5 objek wisata jika dibandingkan dengan brute force yang mendapat 2 saja). Oleh karena itu, dapat disimpulkan bahwa dengan menggunakan algoritma ini dapat ditentukan rute perjalanan dari suatu kota ke kota lainnya dengan mempertimbangkan objek wisata yang dikunjungi namun juga menjaga agar jarak yang ditempuh juga tidak terlalu jauh.

VI. DAFTAR REFERENSI

[1] Munir, Rinaldi, "Diktat Kuliah IF2211 Strategi Algoritma", Bandung ITB, 2013.

[2] Munir, Rinaldi, "Algoritma Greedy". Program Studi Teknik Informatika ITB, 2013

[3] http://bertzzie.com/knowledge/analisis-algoritma/5-Greedy.html, 18 Desember 2013, 19:34.

[4] www.princeton.edu/~achaney/tmve/wiki100k/docs/Greedy_algorithm.html, 19 Desember 2013, 23.13

[5] www.cs.ucsb.edu/~suri/cs130b/greedy3.pdf , 20 Desember 2013, 14.10

[6] homepages.ius.edu/RWISMAN/C455/html/notes/Chapter16/Greedy.html, 20 Desember 2013 14.20

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, 20 Desember 2013

Renusa Andra Prayogo (13511063)