Pembahasan Pencarian Lintasan Terpendek Menggunakan Algoritma Dijkstra dan A*

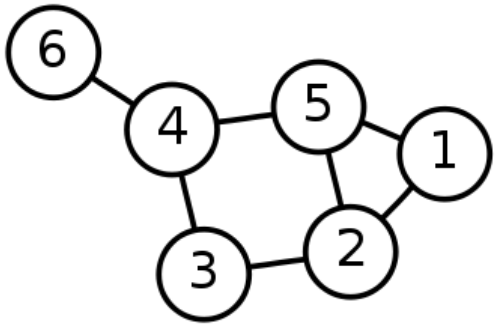
Willy Setiawan - 13508043
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
If18043@students.if.itb.ac.id

***Abstrak*—Dalam makalah ini, akan dianalisa mengnai algoritma Dijkstra dan algoritma A star (A*). Penggunaan kedua algoritma ini digunakan secara luas dalam pencarian jalur terbaik dalam berbagai hal. Kesimpulan yang ada dalam makalah ini yaitu perbandingan antara kedua algoritma tersebut, baik dari segi kelebihan dan kekurangan algoritma, serta pengaplikasian algoritma tersebut.**

***Kata kunci*—A star, Dijkstra, greedy, pencarian lintasan terpendek,**

I. PENDAHULUAN

Pencarian jalur terpendek atau yang disebut juga *shortest path problem* adalah salah satu permasalahan yang menarik untuk dianalisa. *Shortest path problem* adalah sebuah masalah dalam mencari jalan diantara 2 titik (atau simpul) dengan menggunakan bobot yang minimal.



Gambar 1. Sebuah graf dengan 6 titik dan 7 simpul

Aplikasi dari penyelesaian pencarian jalur terpendek bisa untuk berbagai hal. Bisa digunakan untuk memberitahukan jalan-jalan mana yang harus dilalui untuk memperoleh jarak terpendek, atau bisa saja untuk menyelesaikan permasalahan yang dapat digambarkan dengan sebuah graf, dalam permasalah robotic,transportasi , desain VLSI , dan lain-lain.

Untuk memecahkan permasalahan pencarian jalur terpendek ini, dapat digunakan beberapa metode, seperti algoritma pencarian A*, algoritma pencarian Bellman-Ford, dan masih banyak lagi. Algoritma yang akan dibahas adalah algoritma Dijkstra yang merupakan turunan dari algoritma *greedy* , dan algoritma A star (A*). Kedua algoritma tersebut adalah algoritma yang terbilang cukup mangkus dalam menyelesaikan permasalahan *shortest path problem*.

II. PENGENALAN TIAP ALGORITMA

Pada bagian ini, akan dijelaskan karakteristik dari tiap algoritma yang ada, yaitu algoritma greedy, algoritma Dijkstra, dan algoritma Johnson. Algoritma greedy juga akan dibahas karena algoritma ini berhubungan dengan algoritma Dijkstra,

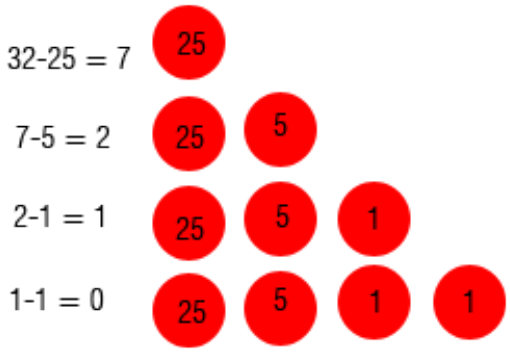
A. Mengenai Algoritma Greedy

Algoritma greedy adalah algoritma yang mengikuti pemecahan masalah secara metaheuristik dalam membuat pilihan optimal dalam setiap tahap pemilihan langkah dengan harapan memperoleh optimum global. Algoritma greedy membentuk solusi langkah per langkah (step by step). Terdapat banyak pilihan yang perlu dieksplorasi pada setiap langkah solusi. Sebagai gambaran.

Pendekatan yang digunakan di dalam algoritma greedy adalah membuat pilihan yang “tampaknya” akan memberikan perolehan terbaik, yaitu dengna membuat pilihan optimum local pada setiap langkah dengan harapan hasilnya akan mengarah ke optimum global.

Masalah yang cocok dalam menggambarkan aplikasi algoritma greedy adalah pada masalah penukaran uang koin. Seseorang memiliki uang koin yang bernilai 32, dan dia ingin menukarkannya. Nilai uang yang tersedia adalah 1,5,10, dan 25. Terdapat berbagai cara untuk mendapatkan koin dengan nilai 32. Dengan menggunakan algoritma greedy, kombinasi yang diperoleh adalah :

1. Sebuah koin 25,
2. Sebuah koin 5,
3. Dan 2 buah koin 1.



Gambar 2. Ilustrasi Algoritma Greedy

Dari gambar diatas, terlihat bahwa untuk mendapatkan koin yang bernilai 32 hanya dibutuhkan 4 koin.

Secara umum, algoritma greedy memiliki 5 elemen :

1. Himpunan kandidat, C.

   Himpunan ini berisi elemen pembentuk solusi. Contohnya adalah himpunan koin, himpunan job yang akan dikerjakan, himpunan simpul di dalam graf, dan lain-lain. Pada setiap langkah, satu buah kandidat diambil dari himpunannya.

2. Himpunan solusi, S.

   Berisi kandidat yang terpilih sebagai solusi persoalan.Himpunan solusi dapat juga dikatakan sebagai himpunan bagian dari himpunan kandidat.

3. Fungsi seleksi.

   Fungsi yang memilih kandidat terbaik untuk ditambahkan ke dalam solusi.

4. Fungsi kelayakan

   Fungsi yang memeriksa apakah suatu kandidat dapat digunakan untuk memberikan solusi. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan kandidat yang tidak layak dibuang dan tidak dipertimbangkan lagi.

5. Fungsi obyektif

   Fungsi yang memaksimumkan atau meminimumkan solusi.

Sebagai contoh, dalam permasalahan penukaran koin ini, elemen-elemen greedy yang ada adalah :

1. Himpunan kandidat : koin yang merepresentasikan nilai 1,5,10,25, paling sedikit mengandung satu koin untuk setiap nilai

2. Himpunan solusi : koin yang terpilih sehingga total nilai koin seluruhnya tepat sama jumlahnya dengan nilai uang yang ditukarkan.

3. Fungsi seleksi : pilih koin yang nilai nya paling tinggi dari himpunan kandidat yang tersisa.

4. Fungsi layak : periksa apakah nilai total dari himpunan koin yang dipilih tidak melebihi jumlah uang yang harus dibayar.

5. Fungsi obyektif : jumlah koin yang digunakan minimum

Secara umum, skema algoritma greedy dapat dirumuskan sebagai berikut :

1. Inisialisasi S dengan kosong

2. Pilih sebuah kandidat (dengan fungsi seleksi) dari C

3. Kurangi C dengan kandidat yang sudah dipilih dari langkah 2

4. Periksa apakah kandidat yang dipilih tersebut bersama-sama dengan himpunan solusi membentuk solusi yang layak atau feasible. Jika ya, masukkan kandidat tersebut ke dalam himpunan solusi ; jika tidak, buang kandidat tersebut dan tidak perlu dipertimbangkan lagi.

5. Periksa apakah himpunan solusi sudah memberikan solusi yang lengkap(dengan menggunakan fungsi solusi). Jika ya, berhenti; jika tidak, ulangi lagi dari langkah 2.

B. Mengenai Algoritma Dijkstra

Algoritma Dijkstra adalah sebuah algoritma yang dikembangkan oleh seorang ilmuwan komputer dari Belanda , Edsger Dijkstra. Algoritma ini adalah sebuah algoritma yang menyelesaikan pencarian jalur terpendek pada graf dengan nilai non negatif untuk bobot setiap simpul,menghasilkan pohon jalur terpendek.

Penjelasan mengenai algoritma Dijkstra adalah :

1. Tetapkan nilai jarak pada setiap simpul. Tetapkan 0 untuk simpul awal dan tak terbatas pada semua simpul yang lain
2. Tandai semua simpul sebagai belum dikunjungi. Tetapkan simpul sekarang sebagai simpul awal.

3. Untuk simpul sekarang, anggap semua tetangga yang belum dikunjungi dan hitung jarak terhadap simpul sekarang. Jika jarak sekarang lebih kecil dari jarak yang sebelumnya direkam, timpa nilainya.
4. Ketika kita selesai menghitung tetangga dari simpul sekarang,tandai sebagai telah dikunjungi. Jaraknya disimpan dan dinyatakan minimal.
5. Jika semua simpul telah dikunjungi, nyatakan sebagai selesai. Jika tidak, nyatakan simpul yang belum dikunjungi dengan jarak terkecil sebagai simpul sekarang dan ulangi langkah 3.

Algoritma Dijkstra adalah algoritma yang dikhususkan untuk pencarian jalan terbaik dalam sebuah graf.

C. Mengenai Algoritma A star (A*)

Algoritma A star atau yang ditulis juga A* adalah algoritma yang seringkali digunakan dalam pencarian jalan dan traversal graf. Algoritma ini diciptakan oleh Peter Hart, Nils Nilsson, dan Bertram Raphael pada tahun 1968. A* menggunakan best first search dan bobot minimal yang diberikan dari simpul awal ke simpul tujuan. A* menggunakan jarak ditambah biaya(biasa dinyatakan f(x)) dari fungsi heuristik untuk menentukan urutan pencarian mencapai simpul mana pada sebuah pohon. Biaya jarak ditambah heuristik terdiri dari 2 buah fungsi. Yang pertama adalah fungsi biaya jarak, yang merupakan biaya dari simpul awal sampai simpul sekarang (biasa dinyatakan dengan g(x)). Yang kedua adalah, sebuah perkiraan heuristik dari jarak ke tujuan (biasa dinyatakan h(x)).

Seperti semua algoritma pencarian yang lain, pertama-tama algoritma ini mencari rute yang tampaknya akan menuju ke simpul tujuan.Yang menjadi pembeda dengan algoritma yang lain adalah bahwa jarak yang telah dilewati juga ikut menjadi pertimbangan. Berawal dari simpul awal, dia mengatur sebuah priority queue dari simpul untuk ditraversal kan , yang dikenal sebagai open set. Semakin rendah f(x) untuk simpul yang diberikan,semakin tinggi prioritasnya. Pada setiap tahap dari algoritma, simpul dengan nilai f(x) terendah akan dihapus dari antrian, nilai f dan h dari tetangga nya akan diupdate,dan tetangga nya akan dimasukkan ke dalam antrian. Algoritma akan berlanjut sampai sebuah simpul tujuan memiliki nilai f yang lebih rendah dari semua simpul pada antrian. Nilai f dari tujuan adalah nilai jalur terpendek.

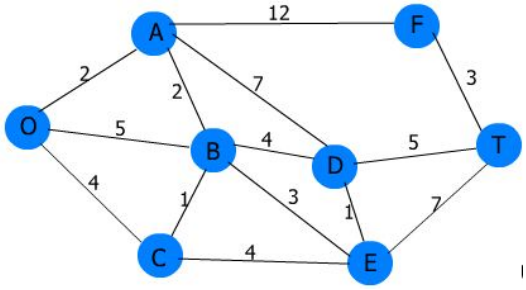
Pseudocode untuk algoritma A* adalah :

1. Buat graf pencari G,yang hanya berisi simpul awal,n0. Taruh n0 pada list bernama OPEN.
2. Buat sebuah list dengan nama CLOSED yang nilai awalnya kosong.
3. Jika OPEN tidak berisi,maka keluar dari program.
4. Ambil simpul pertama dari OPEN,hapus dari OPEN dan masukkan ke CLOSED.Anggap saja simpul n.
5. Jika n adalah simpul tujuan,keluar dari program dengan solusi yang diperoleh sepanjang pointer n ke n0 pada G .
6. Perlebar simpul n, buat sebuah set M,dari suksesor yang bukan pendahulu di G.
7. Buat pointer kepada n dari tiap member M yang tidak ada pada G. Tambahkan anggota G pada OPEN. Untuk tiap anggota m,dari M yang sudah berada pada OPEN atau CLOSED, mengarahkan ulang pointer ke n jika jalur terbaik ke m yang ditemukan adalah melalui n.Untuk tiap anggota M yang telah berada pada CLOSED, arahkan ulang pointer dari tiap pendahulu di G sehingga mereka mengarahkan mundur sepanjang jalan terbaik yang telah ditemukan.
8. Susun ulang daftar OPEN dengan urutan nilai f yang menaik.
9. Kembali ke langkah nomor 3 .

III. STUDI KASUS

A. Penyelesaian dengan Algoritma Dijkstra

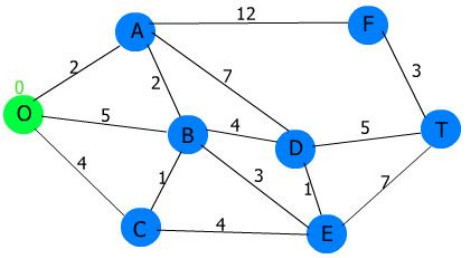
Terdapat sebuah kumpulan simpul dimana jarak diantara simpul dinyatakan pada nilai pada sisi yang berhubungan diantara mereka berdua.



Gambar 3. Sebuah graf yang memiliki keterkaitan antara yang satu dengan yang lain

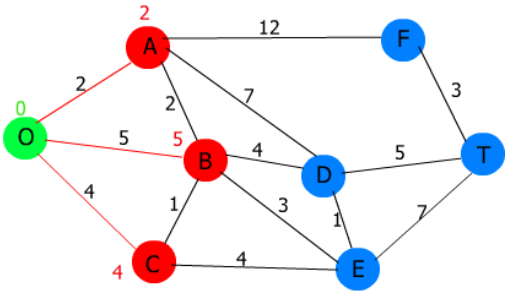
Permasalahannya adalah : bagaimana mencari rute untuk menghasilkan jalur terpendek dari titik awal O ke titik akhir T ?

Pertama-tama, labelkan nilai simpul O dengan angka 0, seperti gambar berikut :



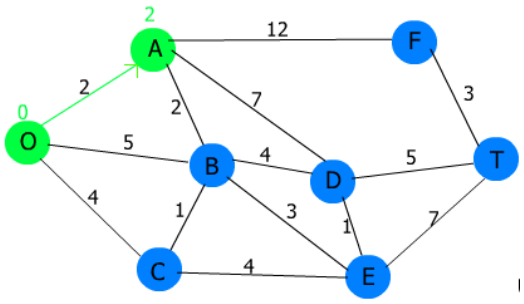
Gambar 4. Langkah pertama dalam algoritma Dijkstra

Lalu, identifikasi simpul-simpul mana yang belum dikunjungi, tapi terhubung dengan simpul awal, yaitu simpul O. Pada gambar, terlihat bahwa tetangga dari O adalah simpul A , B , dan C. Untuk setiap simpul yang memenuhi kriteria tersebut (simpul A,B, dan C), hitung jarak kandidat tersebut. Jarak kandidat = jarak menuju simpul + panjang sisi. Pilih simpul dengan bobot paling kecil.



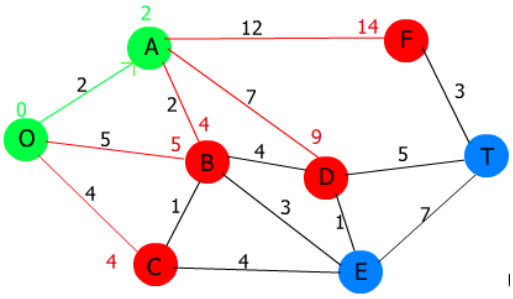
Gambar 5. Pemilihan simpul dengan bobot terkecil

Dari gambar 5, dapat diambil kesimpulan bahwa simpul A memiliki bobot minimum. Karena itu , tandai simpul A menjadi sudah dikunjungi dan labelkan dengan jarak kandidat.Tambahkan sudut ke kumpulan sudut.



Gambar 6. Pemilihan simpul terpendek

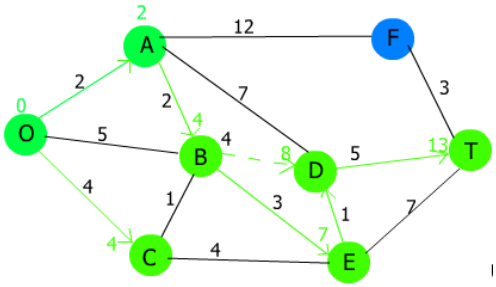
Identifikasi semua simpul yang belum teridentifikasi yang terhubung dengan sebuah simpul yang telah dikunjungi.Hitung semua jarak kandidat dari setiap sudut yang berhubungan.



Gambar 7. Pemilihan nilai minimum dari simpul-simpul yang belum dikunjungi

Terdapat 2 nilai yang sama, yaitu dari simpul O ke C dan simpul A ke B yang nilainya 4.Dalam kasus ini, kita pilih sembarang saja.Dalam kasus ini, yang diambil sebagai simpul yang telah dikunjungi adalah B. Lakukan hal yang sama seperti sebelumnya,yaitu masukkan sudut AB ke dalam kumpulan sudut.

Lakukan sampai simpul T telah dikunjungi.



Gambar 8. Simpul T telah masuk ke simpul yang telah dikunjungi

Terdapat simpul yang belum dikunjungi , da nada rute yang belum terselesaikan O-C. Terdapat 2 rute terpendek, yaitu : O-A-B-D-T, dan O-A-B-E-D-T. Keduanya memiliki bobot 13

B. Penyelesaian dengan Algoritma A*

Diberikan sebuah permasalahan, yaitu bagaimana caranya bergerak dari titik A menuju titik B tanpa tertabrak oleh tembok.

Gambar 9. Permasalahan

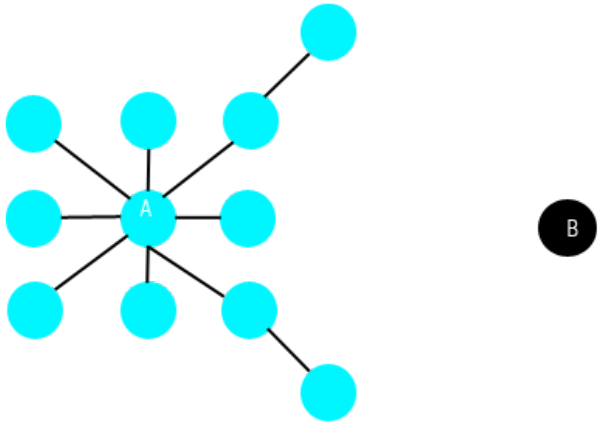
Dari permasalahan ini, dapat digambarkan sebagai graf dimana simpul menyatakan tiap petak nya, sedangkan sisi

menyatakan bobot dari tiap langkah nya. Anggap jalan lurus memiliki bobot 10 dan jalan yang miring memiliki bobot 14. Fungsi yang digunakan dalam pencarian jalur terpendek dengan algoritma ini adalah :

F = G + H , dimana

G= biaya pergerakan dari A ke sebuah petak

H= perkiraan pergerakan untuk bergerak dari kotak yang kita tuju ke kotak B. Biasa disebut sebagai heuristik.



Gambar 10. Gambaran singkat mengenai permasalahan

Kita akan mencari langkah dengan nilai F yang paling kecil.Dari gambar, terlihat bahwa jalur yang memiliki nilai terkecil adalah jalan ke kanan, karena memiliki nilai total 40 (10 dari langkah bergerak ke kanan dan 30 dari langkah heuristik dari petak tersebut ke kotak B tanpa memerhatikan tembok yang ada). Hapus petak tersebut dari OPEN dan masukkan petak tersebut ke CLOSED.

Lalu, dari petak tersebut, kita pilih lagi langkah yang memiliki nilai F minimum. Dari gambar diatas, terlihat bahwa jalan ke kanan,kanan atas,dan kanan bawah tidaklah mungkin karena terhalangi oleh adanya tembok.

Dari gambar, terlihat bahwa langkah ke atas atau ke bawah memiliki bobot yang minimum, oleh karena itu, pilih salah satu dari langkah tersebut, lalu hapus dari set OPEN , dan masukkan ke dalam CLOSED.

Ketika sudah sampai ke titik tujuan,untuk mengetahui langkah yang memiliki bobot maksimal, lakukan analisa dari list CLOSED.List CLOSED mengandung simpul-simpul yang mengarah ke nilai optimum.

IV. Analisa

Dari kedua permasalahan diatas, terlihat bahwa algoritma dijkstra dan A* memiliki kesamaan, yaitu penyelesaian yang mereka lakukan berbasiskan pada pencarian nilai optimum pada tiap tahap. Yang membedakan keduanya adalah terdapat nilai heuristik yang digunakan pada algoritma A*, sedangkan dijkstra tidak.

V. Kesimpulan

Yang dapat disimpulkan setelah menganalisis kedua algoritma ini adalah tiap algoritma memiliki kelebihan dan kekurangan masing-masing. Algoritma Dijkstra dapat memberikan hasil yang lebih cepat, tapi bisa saja memberikan hasil yang kurang akurat walau kemungkinan tersebut kecil. Algoritma A* lebih lambat dalam mencari jalur optimum,tapi hasil yang diberikan lebih akurat dalam algoritma Dijkstra.

Penggunaan algoritma A* cocok diaplikasikan ke dalam aplikasi permainan yang membutuhkan pencarian jalan oleh AI, karena bersifat heuristik, yang memiliki sifat menebak-nebak jalan yang harus dilalui. Sedangkan, algoritma Dijkstra cocok dalam mencari jalur terbaik untuk jalan yang telah diketahui bobot-bobotnya dan bobot tersebut beragam. Contohnya, dalam pencarian rute terbaik dalam jaringan.

Referensi

[1] http://en.wikipedia.org/wiki/Shortest_path_problem
Diakses pada tanggal 1 Desember 2010
[2] http://en.wikipedia.org/wiki/Dijkstra's_algorithm
Diakses pada tanggal 1 Desember 2010
[3] http://en.wikipedia.org/wiki/A*_search_algorithm
Diakses pada tanggal 1 Desember 2010
[4] http://www.edenwaith.com/products/pige/tutorials/a-star.php
Diakses pada tanggal 1 Desember 2010
[5] http://optlab-server.sce.carleton.ca/POAnimations2007/DijkstrasAlgo.html
Diakses pada tanggal 1 Desember 2010
[6] http://www.policyalmanac.org/games/aStarTutorial.htm
Diakses pada tanggal 2 Desember 2010
[7] Munir Rinaldi. 2005. Diktat Kuliah IF2251 Strategi Algoritmik, 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, 29 April 2010

ttd

Willy Setiawan - 13508043