Penerapan Algoritma A* Sebagai Algoritma Pencari Jalan Dalam Game

Ecky Putrady – NIM : 13508004
Jurusan Teknik Informatika, Institut Teknologi Bandung
Email : if18004@students.if.itb.ac.id

ABSTRAK

Makalah ini membahas tentang bagaimana suatu entitas di dalam game mampu mencari jalan terpendek dari titik koordinatnya sekarang menuju titik koordinat lain di dalam game. Kecerdasan yang dimiliki oleh entitas ini dapat dibangun menggunakan berbagai macam algoritma pencari jalan seperti Djikstra atau BFS (Best-First-Search). Dalam makalah ini, algoritma yang akan dibahas adalah algoritma A* (dibaca A-Star). A* adalah salah satu alternatif algoritma pencari jalan yang paling banyak dipakai dalam game. Algoritma-algoritma di atas, termasuk A* menggunakan graf sebagai konsep dasar pencarian jejak.

Kata kunci: Pencari jalan, *pathfinding*, A*, Graf.

1. PENDAHULUAN

Dewasa ini video game semakin terlihat canggih dan menawan. Tidak hanya dari segi tampilannya yang semakin detail dan memanjakan mata, namun juga dari segi kecerdasan dari entitas-entitas di dalam game. Seringkali kita menemukan adanya entitas-entitas di dalam game yang tidak berada di bawah kontrol pemain namun entitas-entitas tersebut mampu berlaku cerdas seakan-akan ada manusia yang mengontrolnya. Kecerdasan yang dimiliki oleh entitas tersebut merupakan kecerdasan buatan yang diprogram oleh programmer. Dengan memberikan kecerdasan buatan kepada entitas dalam game, entitas tersebut dapat berperilaku cerdas layaknya makhluk hidup. Hal tersebut akan menambah tingkat realitas dari suatu game.

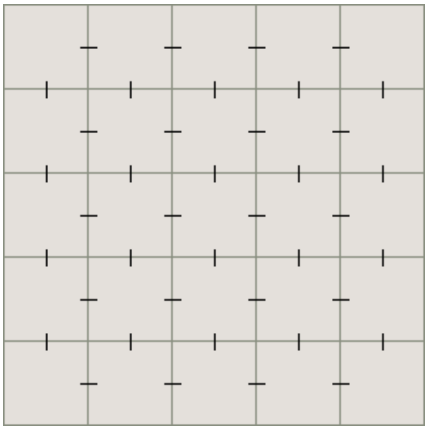
Salah satu kecerdasan buatan yang banyak dipakai dalam game adalah kecerdasan dalam mencari jalan. Seringkali pembuat game ingin agar suatu entitas dalam game yang tidak dikendalikan oleh pemain berpindah dari suatu lokasi ke lokasi lain layaknya manusia. Entitas dalam game yang cerdas seharusnya dapat mencari jalan terpendek dari posisinya ke posisi yang dituju. Entitas tersebut juga harus dapat menemukan jalan memutar terpendek apabila ada penghalang antara dia dan posisi yang dituju. Untuk itu, algoritma pencari jalan diperlukan agar suatu entitas yang tidak dikendalikan oleh pemain mampu menemukan jalan ke suatu lokasi di dalam game.

Gambar 1. Bagaimana memprogram kecerdasan mumi agar ia dapat mencari jalan menuju pemain dalam game Mummy Maze?

Algoritma pencari jalan yang akan dibahas di sini adalah algoritma A*. Algoritma A* merupakan algoritma pencari jalan terbaik dan merupakan gabungan dari algoritma Djikstra dan BFS. Ketiga algoritma ini menggunakan graf berbobot tidak berarah sebagai konsep dasar pencarian jejak.

Graf didefinisikan sebagai himpunan pasangan dari simpul (*node*) dan sisi (*edge*)[1]. Graf yang tidak berarah diartikan sebagai graf yang sisinya tidak memiliki orientasi arah. Graf berbobot memiliki arti bahwa sisi dari graf memiliki bobot nilai tertentu.

Peta (*map*) dalam suatu game dapat direpresentasikan dengan graf berbobot dan tidak berarah.



Gambar 2. Peta dalam game direpresentasikan dengan graf

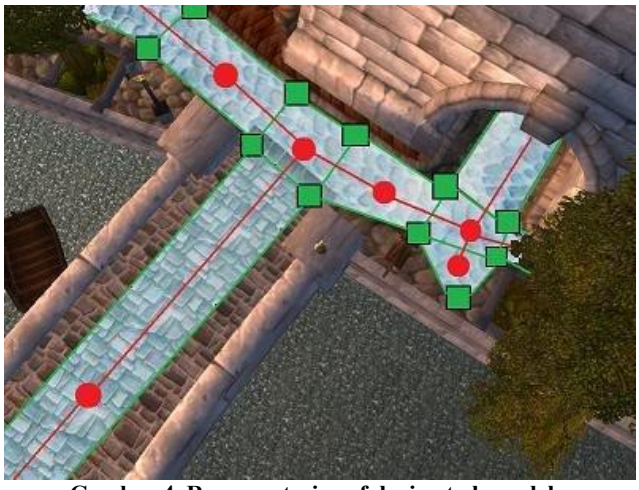
Setiap kotak dari graf pada gambar 2 adalah simpul. Simpul-simpul ini keterhubungan dengan simpul lainnya ditunjukkan dengan garis. Sehingga garis dapat diartikan sebagai sisi.

Apabila suatu simpul A terhubung melalui sisi dengan simpul B, maka entitas yang berada di A dapat berpindah ke B. Karena graf yang digunakan adalah graf tidak berarah maka entitas di B dapat berpindah ke A sebagaimana entitas di A dapat berpindah ke B. Sisi penghubung antara A dan B dapat kita beri bobot sebagai harga untuk berpindah dari satu simpul ke simpul lain di sisi tersebut.

Pembaca mungkin sadar bahwa tidak semua peta dalam game dapat direpresentasikan secara diskrit dan sederhana seperti itu. Sebagai contoh, bagaimana kita memetakan graf dari peta yang kontinyu seperti gambar di bawah ini :

Gambar 3. Peta yang kompleks

Kenyataannya, peta seperti di atas dapat dibuat grafnya sebagai berikut :



Gambar 4. Representasi graf dari peta kompleks

Peta di atas disebut kontinyu karena entitas-entitas di dalam game dapat berada tidak hanya di titik merah, tapi juga di area yang dibatasi *polygon* biru transparan.

Pencarian jalan pada peta kontinyu memerlukan algoritma A* yang dimodifikasi[3] dan tidak akan kita bahas dalam makalah ini.

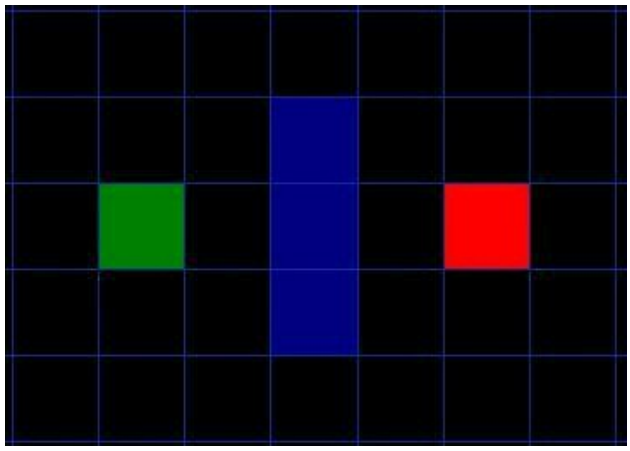
Algoritma pencari jalan lain, Djikstra mampu memastikan bahwa jalan yang dipilih adalah jalan terpendek. Namun Djikstra merupakan algoritma yang yang lamban, karena cara kerjanya *brute-force*[2]. Algoritma lain, yaitu BFS mampu mencari jalan lebih cepat, namun dalam beberapa kasus, bukan merupakan jalan terpendek[2]. A* merupakan algoritma yang menggabungkan kedua unsur dari BFS dan Djikstra. A* mampu mencari jalan terpendek seperti Djikstra dan hampir secepat BFS[2].

Rahasia dari A* adalah pemilihan simpul untuk pencarian jejak didasarkan dari dua hal, yaitu memilih simpul yang lebih dekat ke posisi awal (prinsip Djikstra) dan lebih dekat ke posisi tujuan (prinsip BFS).

Dalam terminologi standar yang digunakan ketika membahas A*, g(n) merepresentasikan harga jalan dari simpul awal ke simpul n, dan h(n) merepresentasikan perkiraan harga dari simpul n ke simpul tujuan. Selama iterasi, algoritma A* mencari simpul n yang memiliki harga f(n) = g(n) + h(n) yang paling kecil.

2. METODE

Algoritma A* akan dicermati melalui contoh kasus berikut ini :



Gambar 5. Contoh kasus pencarian jalan

Gambar diatas menunjukkan suatu peta di dalam game. Setiap kotak merepresentasikan simpul. Setiap kotak terhubung ke 8 kotak yang paling dekat. Artinya setiap simpul terhubung ke simpul lain yang berada di sebelah kanan, kiri, atas, bawah, atas-kanan, bawah-kanan, bawah-kiri, dan atas-kiri dari simpul tersebut. Kotak biru merupakan dinding, yaitu kotak yang tidak dapat dilalui.

Setiap simpul di sini tidak terhubung ke kotak berwarna biru.

Sekarang, kita akan mencari jalur terpendek dari posisi awal (kotak hijau) ke posisi tujuan (kotak merah). Karena kotak hijau tidak terhubung langsung ke kotak merah, maka kita perlu melewati simpul-simpul tertentu yang pada akhirnya akan mengantarkan kita ke kotak merah dengan jarak sependek mungkin.

2.1. Persiapan

Ada beberapa hal yang perlu kita definisikan terlebih dahulu dalam kasus ini.

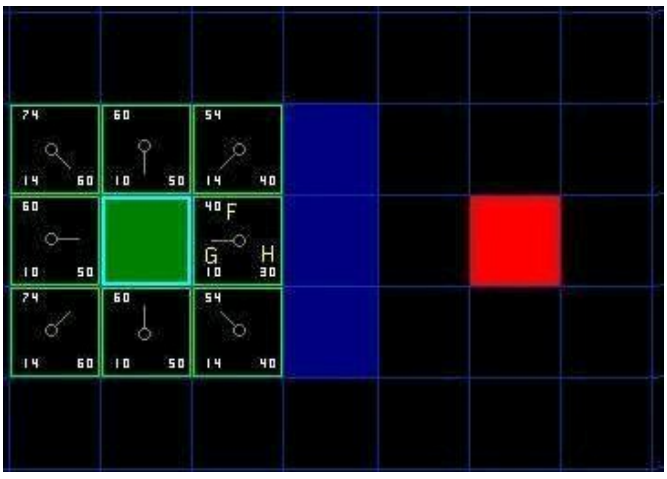
Setiap simpul harus memiliki informasi nilai h(n), yaitu estimasi harga simpul tersebut dihitung dari simpul tujuan, nilai g(n), yaitu harga simpul tersebut dihitung dari simpul awal, serta informasi *parent* yang berisi simpul yang merupakan *parent* dari simpul ini.

Kita memerlukan dua buah list dalam pencarian jalan terpendek, list terbuka dan list tertutup. List terbuka berisi simpul-simpul yang perlu kita selidiki. List tertutup berisi simpul-simpul yang sudah kita selidiki dan tidak perlu kita selidiki lagi.

Untuk mempermudah demonstrasi, harga untuk berpindah antar simpul secara vertikal ataupun horizontal adalah sebesar 10, dan untuk perpindahan secara diagonal sebesar 14.

2.2. Memulai Pencarian

Kita akan mulai melakukan pencarian dengan memasukkan posisi awal ke list terbuka serta mengosongkan list tertutup. Pertama kita akan mengeluarkan simpul dengan nilai f(n) terkecil dari list terbuka. Karena saat ini list terbuka hanya berisi simpul awal, maka simpul awal lah yang dikeluarkan. Kemudian, seluruh simpul yang terhubung langsung ke simpul awal dicek. Untuk setiap simpul yang dicek, informasi parentnya diisi dengan simpul awal dan dimasukkan ke dalam list terbuka. Simpul awal kemudian dimasukkan ke list tertutup karena sudah kita selidiki.



Gambar 6. Pencarian dimulai

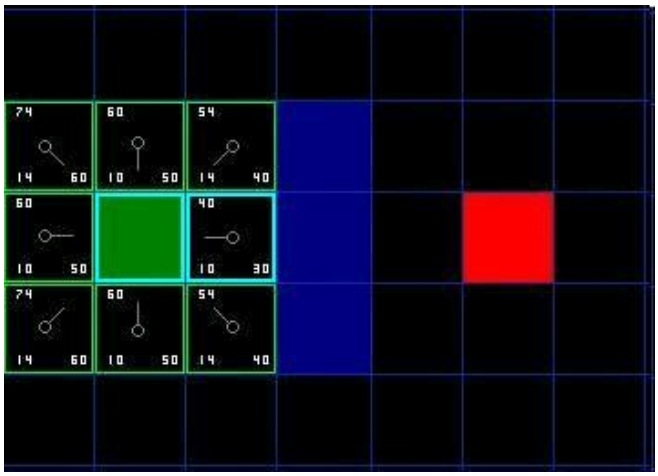
Gambar diatas menunjukkan simpul-simpul yang akan dijelajahi. Angka yang berada pada bagian kiri bawah dari kotak adalah nilai g(n). Perhatikan bahwa untuk setiap simpul yang vertikal dan horizontal dari simpul awal, g(n) bernilai 10. Sedangkan untuk simpul yang diagonal, g(n) bernilai 14. Hal ini sesuai dengan kesepakatan yang dituliskan pada sub bab sebelumnya. Angka pada kanan bawah adalah h(n) yaitu harga estimasi dihitung dari posisi akhir. Simpul di sebelah kanan simpul awal memiliki h(n) = 30. Perhatikan pula bahwa jarak antara simpul tersebut dengan simpul tujuan adalah 3 kotak horizontal. Sehingga harganya adalah 3 kali harga pergerakan horizontal, yaitu 3*10 = 30. Sedangkan h(n) di simpul sebelah kanan bawah adalah 40. Perhatikan bahwa jarak antara simpul ini dengan simpul tujuan adalah 3 kotak horizontal dan 1 kotak vertikal. Sehingga harga h(n)-nya adalah 3*10 + 1*10 = 40. Nilai yang tertera di bagian kiri-atas adalah f(n) = g(n) + h(n). Panah pada bagian tengah dari kotak menunjuk pada simpul yang menjadi *parent* dari simpul tersebut.

Kedelapan simpul di atas kemudian dimasukkan ke list terbuka.

2.3. Melanjutkan Pencarian

Selanjutnya kita akan memilih simpul dengan nilai f(n) terkecil dari list terbuka. Terhadap simpul yang dipilih kita lakukan :

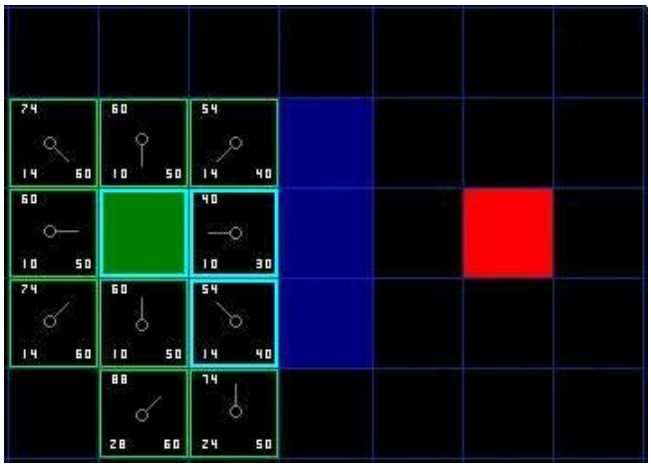
1) Mengeluarkan simpul tersebut dari list terbuka dan memasukkan ke list tertutup
2) Cek semua simpul yang terhubung langsung dengan simpul yang dipilih. Masukkan ke dalam list terbuka apabila simpul yang baru belum ada pada list terbuka serta men-*set parent* dari simpul baru tersebut ke simpul yang dipilih
3) Apabila simpul yang dicek sudah terdapat pada list terbuka, cek nilai g(n) nya. Apabila nilai g(simpul dipilih) + harga untuk bergerak ke simpul dipilih < g(simpul dicek), maka ubah parent dari simpul yang dicek ke simpul yang dipilih. Jika tidak, jangan lakukan apa-apa.



Gambar 7. Pencarian mendapatkan simpul di sebelah kanan simpul awal adalah simpul dengan f(n) terendah

Saat ini, simpul yang terpilih adalah simpul di sebelah kanan simpul awal. Kemudian seluruh simpul yang terhubung langsung dicek. Pada kasus simpul di bawah simpul terpilih dicek, kita menemukan bahwa simpul tersebut sudah berada di dalam list terbuka. Karena itu, kita akan membandingkan g(n) simpul terpilih ditambah harga untuk pergerakan secara vertikal (10) dengan g(n) dari simpul yang dicek. Karena g(simpul terpilih) = 10, harga untuk bergerak vertikal = 10, dan g(simpul dicek) = 14, maka hubungan yang didapatkan adalah 10+10 > 14. Karena nilai g(simpul dicek) lebih kecil, maka kita tidak men-*set parent* simpul dicek menjadi simpul terpilih.

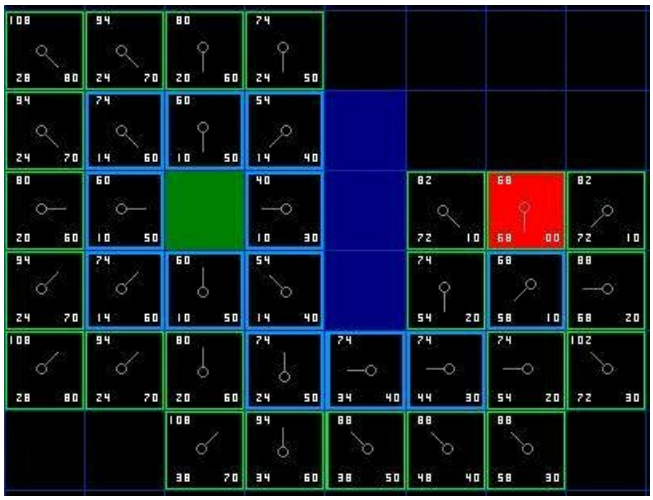
Dengan menggunakan cara yang sama, kita mendapatkan bahwa simpul terpilih yang baru adalah simpul di bawah dari simpul terpilih sebelumnya.



Gambar 8. Simpul baru dipilih

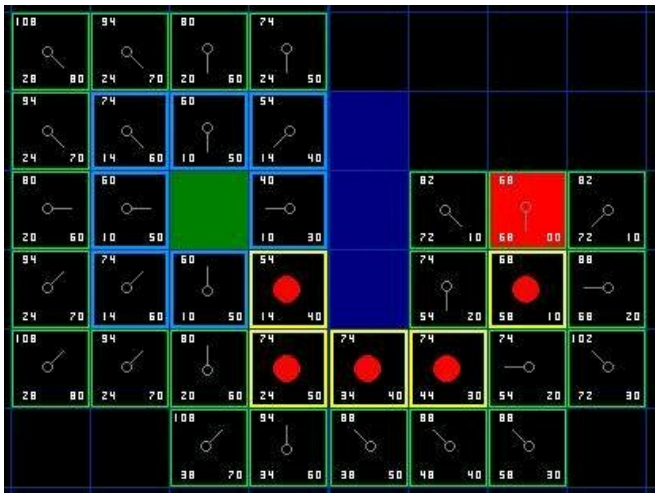
Simpul yang terpilih kembali melakukan pengecekan terhadap semua simpul yang terhubung. Di sini kita definisikan bahwa simpul tidak dapat terhubung secara diagonal apabila berada di dekat tembok.

Apabila kita mengulangi cara ini terus menerus, kita akan mendapatkan hasil berikut :



Gambar 9. Hasil akhir pencarian jalan

Selanjutnya kita hanya perlu menelusuri *parent* dari setiap simpul dimulai dari simpul tujuan, Seperti yang ditunjukkan gambar berikut :



Gambar 10. Jalan terpendek sudah ditemukan

Dari gambar di atas, jalan sudah ditemukan, yaitu simpul-simpul yang memiliki titik merah. Dapat kita perhatikan pula bahwa jalan yang ditemukan merupakan jalan yang terpendek.

2.4. Formalisasi Langkah Pengerjaan

Dengan merangkum langkah-langkah yang sudah kita lakukan, kita mendapatkan algoritmanya dan dituliskan sebagai berikut :

1) Inisialisi list terbuka dan list tertutup.
2) Masukkan simpul awal ke dalam list terbuka.
3) Ambil simpul dengan f(n) terendah dari list terbuka. Kita sebut ini sebagai simpul terpilih.
4) Masukkan simpul terpilih ke list tertutup.

5) Untuk seluruh simpul yang terhubung langsung dengan simpul terpilih, lakukan :
   1. Jika simpul yang dicek sudah termasuk dalam list tertutup, abaikan.
   2. Jika kondisi 1 tidak terpenuhi dan simpul belum termasuk di dalam list terbuka, masukkan simpul tersebut di list terbuka
   3. Jika kondisi 2 tidak terpenuhi (berarti simpul dicek berada pada list terbuka), dan g(simpul terpilih) ditambah harga dari simpul terpilih ke simpul dicek kurang dari g(simpul dicek), ubah parent dari simpul dicek menjadi simpul terpilih.
6) Ulangi nomor 3 sampai 6 jika kondisi di bawah ini tidak terpenuhi :
   1. Simpul tujuan sudah berada di dalam list tertutup, atau
   2. List terbuka tidak berisi apa-apa (tidak ditemukan jalan menuju simpul tujuan).
7) Telusuri parent dari setiap simpul dimulai dari simpul tujuan apabila jalan ditemukan.

Adapun *pseudocode* dari A* adalah sebagai berikut :

```
OPEN = priority queue containing START
CLOSED = empty set
while lowest rank in OPEN is not the GOAL:
    current = remove lowest rank item from OPEN
    add current to CLOSED
    for neighbors of current:
        cost = g(current) + movementcost(current, neighbor)
        if neighbor in OPEN and cost less than g(neighbor):
            remove neighbor from OPEN, because new path is better
        if neighbor in CLOSED and cost less than g(neighbor):
            remove neighbor from CLOSED
        if neighbor not in OPEN and neighbor not in CLOSED:
            set g(neighbor) to cost
            add neighbor to OPEN
            set priority queue rank to g(neighbor) + h(neighbor)
            set neighbor's parent to current

reconstruct reverse path from goal to start
by following parent pointers
```

IV. KESIMPULAN

Graf merupakan konsep dasar dari algoritma kecerdasan buatan yang berkaitan dengan pencarian jalan. Ada beberapa algoritma pencarian jalan seperti Djikstra, BFS, dan A*. Dalam game, algoritma pencarian jalan yang populer adalah A* karena mampu mencari jalur terpendek dengan cepat.

Hampir semua peta dalam game dapat direpresentasikan dalam graf. Karena itulah, algoritma A* merupakan algoritma utama sebagai pencarian jalan dalam kecerdasan buatan yang diaplikasikan di game.

REFERENSI

[1] http://en.wikipedia.org/wiki/Graph_%28data_structure%29 – Diakses pada 16 Desember 2009, pukul 13:22

[2]http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html – Diakses pada 16 Desember 2009, pukul 14:10

[3] http://www.ai-blog.net/archives/000152.html – Diakses pada 16 Desember 2009, pukul 16:45

[4]http://www.policyalmanac.org/games/aStarTutorial.htm – Diakses pada 16 Desember 2009, pukul 14:34