Algoritma *Branch & Bound* dalam permainan "*the Treasure Hunter*"

Satria Ardhe Kautsar - 13505004

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jln. Ganesha 10, Bandung
e-mail: sdf_88@yahoo.com

ABSTRAK

***The Treasure Hunter* adalah sebuah permainan komputer dimana kita sebagai *player* diharuskan menjelajah *map* yang berupa *grid* yang terdiri dari *space* kosong dan dinding, untuk mencapai harta karun. Pada *game* tersebut terdapat fitur *auto* yang menggunakan algoritma *Branch and Bound* untuk menentukan rute optimal. Algoritma *Branch and Bound* yang digunakan merepresentasikan simpul pada pohon ruang status sebagai *node* yang memiliki beberapa parameter : posisi(x,y), *parent*(x,y), dan *cost*(*c*). Nilai taksiran (*cost*) ditentukan berdasarkan jarak yang sudah ditempuh dari *start*, dan jarak dari *node* saat ini terhadap *goal*. Terdapat juga beberapa array : *Open List* sebagai kumpulan *node* yang hidup, *Closed List* sebagai kumpulan *node* yang telah dimatikan, dan *Solution List* sebagai daftar langkah-langkah yang merupakan solusi.**

Kata kunci: *the Treasure Hunter*, *A*, Branch and Bound*.

1. PENDAHULUAN

The Treasure Hunter adalah sebuah permainan komputer dimana kita sebagai *player* diharuskan menjelajah *map* berupa *grid* yang terdiri dari *space* kosong dan dinding, untuk mencapai harta karun (*goal point*).

Game ini memiliki beberapa *mode*, yaitu *map editor, manual play,* dan *auto play*. Pada *map editor*, kita dapat merancang *map* yang selanjutnya akan kita gunakan dalam *mode manual play* maupun *auto play.* Pada *manual play*, kita dapat menjalankan sendiri karakter yang ada di dalam *game* untuk mencapai posisi harta karun. Sedangkan pada *auto play*, karakter akan dikendalikan oleh *AI* yang menggunakan algoritma *Branch and Bound* dalam menentukan *path*, yang selanjutnya akan kita bahas lebih dalam pada makalah ini.

2. *BnB* dalam kasus *pathfinding*

Dalam kasus pencarian rute (*pathfinding*), terdapat satu versi dari algoritma *BnB* yang sering digunakan, yaitu algoritma A*.

Sama seperti algoritma *BnB* pada umumnya, A* juga merupakan algoritma *BFS* yang dioptimasi dengan menggunakan fungsi *heuristik* sebagai acuan pembangkitan anak simpul pada pohon ruang status.

Yang membedakan A* dari algoritma *BnB* umum adalah adanya *open list* dan *closed list*, serta perbandingan yang dilakukan sebelum menghasilkan simpul-simpul anak. Berikut ini adalah skema umum pada algoritma A* :

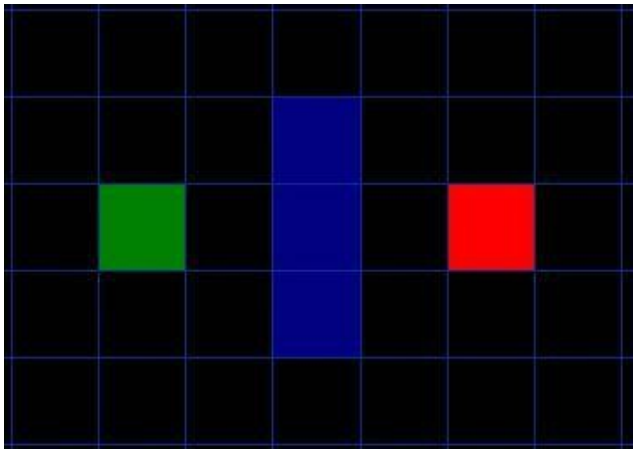
1. Mulai dari *Start Point (S)*, masukkan *start point node* ke dalam *Open List*.
2. Jika *Open List* kosong, maka pencarian gagal. Stop.
3. Ambil *node* dengan nilai *cost* terkecil yang ada di dalam *Open List*.
4. Jika *node tersebut (n)* adalah *Goal Point (G)*, bentuk *Solution List*. Pencarian sukses. Stop.
5. Untuk setiap *child node (n')* yang mungkin dari *n* :
    a. Jika *n'* telah ada pada *Open/Closed List*, dan jarak yang telah ditempuh dari *S* lebih besar daripada yang ada pada *Open/Closed List* tersebut, batalkan pembangkitan *n'*.
    b. Jika *n'* telah ada pada *Closed List*, hilangkan *n'* dari *Closed List*.
    c. Jika *n'* belum ada pada *Open List*, masukkan *n'* ke dalam *Open List*.
6. Masukkan *n* ke dalam *Closed List*.
7. Kembali ke 2.

3. Penerapan *BnB* pada *game "the Treasure Hunter"*

Dalam metode ini, kita menganggap masing-masing *tile* (kotak-kotak pada *map*) sebagai satu buah *node* yang masing-masing memiliki beberapa parameter :

1. *Position* : (x,y) yang menandakan lokasi *tile* tersebut.
2. *Parent* : (x,y) yang menandakan lokasi dari *parent node*. (digunakan dalam pembentukan solusi)
3. *fromStart* : (n) menandakan jarak yang telah ditempuh dari *start point* hingga ke *node* tersebut.
4. *toGoal* : (n) menandakan jarak dari posisi *node* tersebut terhadap *goal point*.
5. *Cost* : (n) nilai taksiran yang merupakan gabungan dari *fromStart* dan *toGoal*.

Mari kita lihat contoh pencarian yang dilakukan pada kasus sederhana berikut :



Gambar 1. Kasus sederhana pencarian dengan *BnB*.

Pada gambar 1, *start point* ditandai dengan kotak berwarna hijau, *goal point* merah, dan dinding ditandai dengan kotak berwarna biru.

Pencarian dimulai dengan *start point* sebagai *current node*, masukkan *start point* ke dalam *Open List.*

Tabel 1. Kondisi *Open List* dan *Closed List* saat ini

Open List	*Closed List*
(2 , 3) c=4 *start point	-

Current node = -

Lalu ambil *node* dengan *cost* terkecil dari dalam *Open List* (dalam hal ini adalah *start point*). *Current node* = (2,3).

Tabel 2. Kondisi *Open List* dan *Closed List* saat ini

Open List	*Closed List*
-	-

Current node = (2,3)

Bangkitkan anak *node* dari *current node.* Ternyata tidak ada yang sudah ada di *Open/Closed List.* Masukkan masing-masing ke dalam *Open List*, lalu masukkan *current node* ke dalam *Open List*.

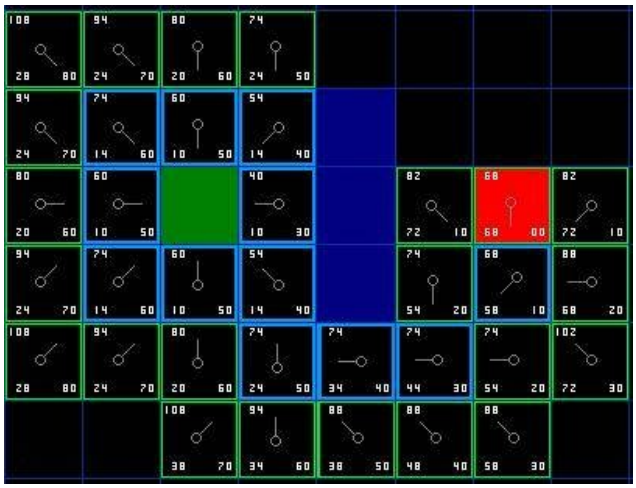
Set *current node* baru berdasarkan *cost* paling kecil yang ada pada *Open List* (ternyata semua memiliki nilai *c* yang sama, ambil berdasarkan aturan *FIFO*), lalu ambil *node* tersebut dari *Open List.*

Tabel 3. Kondisi *Open List* dan *Closed List* saat ini

Open List	*Closed List*
(3,3) c=1	(2,3)
(2,4) c=1	-
(1,3) c=1	-

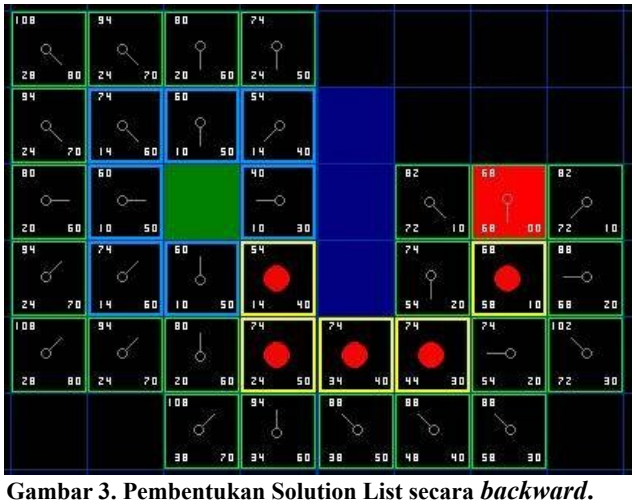
Current node = (2,2)

Ulangi terus langkah-langkah berdasarkan skema yang ada pada Bab 2 sehingga akhirnya akan terbentuk *Open List* dengan *goal point* di dalamnya dan *goal point* tersebut memiliki nilai *cost* paling kecil saat itu.



Gambar 2. Posisi akhir saat *goal point* ditemukan.

Karena *goal point* sudah ditemukan, bentuk *Solution List* secara *backward* dimulai dari *goal point* dengan mengacu pada nilai *Parent*, hingga sampai pada *start point*.



Gambar 3. Pembentukan Solution List secara *backward*.

4. KESIMPULAN

Pencarian rute terpendek oleh *AI* yang ada pada *game "the Treasure Hunter"* menggunakan algoritma A*, yaitu pengembangan dari algoritma *Branch and Bound* yang memiliki ciri khas berupa adanya *Open* dan *Closed list.*

REFERENSI

[1] Munir, Rinaldi . "Strategi Algoritmik", Program Studi Teknik Informatika STEI ITB, 2007 .

[2] http://gamedev . net/ diakses tanggal 19 mei 2008, pukul 19:00 WIB .