Penggunaan Algoritma Greedy Dalam Permainan

Warcraft III Tower Defense

Taufik Prasetiya 13509029
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509029@std.stei.itb.ac.id

***Abstract*— Dalam makalah ini akan dibahas tentang pemanfaatan algoritma greedy untuk mendapatkan nilai maksimal dalam permainan Warcraft III tower defense. Dalam makalah ini juga akan dijelaskan bagaimana algoritma greedy digunakan untuk menyelesaikan permainan tersebut, dengan disertai gambar-gambar yang mendukung penjelasan penyelesaian tersebut. Dijelaskan pula bagaimana contoh kasus yang menyebabkan hasil algoritma tidak optimal.**

Kata kunci: greedy,Warcraft III, tower defense.

I. PENDAHULUAN

Algoritma greedy merupakan salah satu algoritma yang sangat terkenal, tanpa disadari algoritma ini sering kita gunakan untuk menyelesaikan masalah yang sangat penting dan riskan maupun untuk penyelesaian masalah kecil yang sering kita hadapai, misalkan dalam permainan Warcraft III *tower defense*. Tujuan dari permaian ini adalah mencoba untuk menghentikan musuh yang melintasi sebuah peta permainan menggunakan *tower*(bangunan yang dapat menyerang) yang terus menyerang musuh yang lewat..

1.1. Warcraft III: The Frozen Throne

Warcraft III: The Frozen Throne adalah permainan *real-time strategy* yang dikembangkan untuk Microsoft Windows, Mac OS dan Mac OS X oleh Blizzard Entertainment. Ini adalah ekspansi resmi untuk Warcraft III: Reign of Chaos,. Dirilis di toko-toko di seluruh dunia dalam berbagai bahasa mulai pada tanggal 1 Juli 2003, itu termasuk unit baru untuk setiap *race*, ras netral baru, empat *campaign*, lima Hero netral, kemampuan untuk membangun sebuah toko dan berbagai perbaikan lainnya seperti kemampuan untuk upgrade. Unit Laut yang diperkenalkan kembali, mereka telah hadir di Warcraft II, namun tidak hadir dalam Reign Of Chaos. Blizzard Entertainment telah merilis patch untuk permainan untuk memperbaiki bug, menambahkan fitur baru, dan keseimbangan multiplayer.

Gameplay Real-time Strategy

Dalam permainan real-time strategi, layar dibagi menjadi area peta dunia yang menampilkan permainan dan daerah, unit, dan bangunan, dan overlay antarmuka yang berisi perintah dan kontrol produksi. sering kali terdapat "radar" atau "Minimap" gambaran dari seluruh peta. Pemain biasanya diberikan perspektif isometrik, atau *free-roaming camera* dari sudut pandang udara untuk game 3D modern.Pemain dapat berpindah antar layar dengan menggerakkan mouse dan dapat memberikan perintah dengan *mouse* atau menggunakan cara pintas dengan *keyboard*.

Gameplay umumnya terdiri dari pemain yang diposisikan di suatu tempat di peta dengan beberapa unit atau bangunan yang mampu membangun unit lain / bangunan. Sering kali, tetapi tidak selalu, pemain harus membangun struktur yang spesifik untuk membuka unit dengan teknologi yang lebih maju. Sering kali pemain harus membuat tentara untuk membela diri dari bentuk serangan musuh dan untuk menghancurkan basis musuh. Terkadang game RTS juga memiliki batas unit maksimal yang dapat dibuat. Umumnya fokus utama dari game RTS adalah mengumpulkan sumber daya,

1.2. Tower Defense

Tower defense adalah sebuah subgenre dari permainan real-time strategy. Tujuan *tower defense* adalah untuk mencoba menghentikan musuh dengan membangun menara yang menembak mereka saat mereka lewat. Musuh dan menara biasanya memiliki kemampuan, dan biaya beragam. Ketika musuh dikalahkan, pemain mendapatkan uang atau poin, yang digunakan untuk membeli menara.

Penempatan posisi menara adalah strategi penting dari permainan. Banyak game, seperti Flash Element *Tower Defense*, fitur musuh yang dijalankan melalui "labirin",

yang memungkinkan pemain untuk menempatkan menara di tempat strategis untuk efektivitas.

Gameplay
Permainan *tower defense* dicirikan oleh posisi unit statis oleh pemain untuk bertahan melawan unit musuh yang bergerak yang mencoba untuk berpindah dari titik awal ke titik akhir. Ada batas maksimal musuh yang sampai titik akhir. Biasanya terdapat rute perjalanan musuh yang statis dan terdapat tempat bagi pemain untuk meletakkan tower nya, sementara yang lain menggunakan rute bebas dimana pemain harus menentukan rute perjalanan dari musuh. Beberapa game menggunakan campuran keduanya.

Ini adalah tema umum permainan *tower defense* untuk memiliki unit udara yang tidak melewati labirin, melainkan terbang di atas menara

II. DASAR TEORI *GREEDY*

Algoritma *greedy* adalah sebuah algoritma yang mengikuti pemecahan masalah heuristik untuk membuat pilihan yang optimal secara lokal pada setiap tahap dengan harapan menemukan optimum global. Pada beberapa masalah, strategi *greedy* tidak menghasilkan solusi yang optimal, tapi tetap sebuah heuristik *greedy* lokal mungkin solusi optimal yang mendekati solusi optimal global.

Sebagai contoh, strategi *greedy* untuk *Travelling salesman problem* (yang mana kompleksitas komputasi tinggi) adalah heuristik berikut: "Pada setiap tahap mengunjungi sebuah kota yang belum dikunjungi terdekat ke kota saat ini". Heuristik ini tidak menemukan solusi terbaik, tapi berakhir dalam sejumlah langkah yang wajar; mencari solusi yang optimal biasanya memerlukan banyak langkah-langkah masuk akal. Dalam optimasi matematika, algoritma *greedy* memecahkan masalah kombinatorial yang memiliki sifat-sifat matroids.

Spesifik
Secara umum, algoritma *greedy* memiliki lima pilar:

- Himpunan kandidat
  Himpunan berisi elemen-elemen pembentuk solusi
- Fungsi seleksi
  Fungsi ini akan memilih kandidat yang paling memungkinkan untuk mencapai solusi optimal. Sesuai dengan prinsip algoritma *Greedy*, kandidat yang sudah dipilih pada suatu langkah tidak bisa diubah di langkah selanjutnya.
- Fungsi kelayakan
  Fungsi ini akan memeriksa kelayakan suatu kandidat yang telah dipilih. Dalam arti, kandidat tersebut dan himpunan solusi yang terbentuk tidak melanggar constraints yang ada. Bila kandidat layak, maka kandidat tersebut akan dimasukkan ke dalam himpunan solusi, dan jika kandidat tersebut tidak layak, maka kandidat akan dibuang dan tidak akan dipertimbangkan lagi dalam pencarian solusi optimum.
- Fungsi obyektif
  Fungsi ini akan membuat nilai solusi maksimum atau minimum, sesuai dengan jenis optimasi apa yang dibutuhkan.
- Himpunan solusi
  Himpunan ini berisi kandidat-kandidat yang terpilih sebagai solusi dari permasalahan optimasi yang akan diselesaikan

Algoritma *greedy* menghasilkan solusi yang baik pada beberapa persoalan matematika, tapi tidak untuk beberapa persoalan lainnya.biasanya masalah yang dapat diselesaikan algoritma ini dengan baik, akan mempunyai 2 properties :

- *Greedy* choice property
  Kita bisa membuat apa pun yang tampaknya pilihan terbaik saat ini dan kemudian memecahkan submasalah yang timbul kemudian. Pilihan yang dibuat oleh algoritma *greedy* mungkin tergantung pada pilihan yang dibuat sejauh ini tetapi tidak pada pilihan masa depan atau semua solusi untuk subproblem tersebut. Secara iterative membuat salah satu pilihan *greedy* setelah yang lain, mengurangi masing-masing persoalan yang diberikanmenjadi persoalan yang lebih kecil. Dengan kata lain, algoritma *greedy* tidak pernah mempertimbangkan kembali pilihan mereka. Ini adalah perbedaan utama dari pemrograman dinamis, yang lengkap dan dijamin untuk menemukan solusi. Setelah setiap tahap, pemrograman dinamis membuat keputusan berdasarkan semua keputusan yang dibuat pada tahap sebelumnya, dan dapat kembali ke tahap sebelumnya untuk solusi.

- *Optimal structure*
  Persoalan mempunya *Optimal structure* jika solusi optimal untuk masalah ini memuat solusi optimal untuk sub-masalah

Kasus Kegagalan *Greedy*

Untuk beberapa masalah, algoritma *greedy* gagal menghasilkan solusi optimal, dan bahkan dapat menghasilkan solusi yang mungkin terburuk. Salah satu contoh adalah masalah *Travelling salesman problem* yang disebutkan di atas.

Algoritma *greedy* dapat dicirikan dengan “penglihatan pendek” dan “non-recoverable”. Algoritma ini idelan hanya untuk masalah yang mempunyai “optimal substructure”. . Meskipun demikian, algoritma *greedy* yang paling cocok untuk masalah yang sederhana (misalnya memberikan perubahan).Namun hal penting untuk dicatat bahwa algoritma *greedy* dapat digunakan sebagai algoritma seleksi untuk memprioritaskan pilihan

dalam pencarian, atau cabang dan algoritma terikat. Ada beberapa variasi untuk algoritma *greedy*:

- Algoritma *Greedy* Murni
- Algoritma *Greedy* Orthogonal
- *Relaxed Greedy Algorithm*

III. PERSOALAN DAN BATASAN

3.1. Persoalan

Dalam permainan warcraft III tower defense kita harus mampu membunuh musuh yang lewat semaksimal mungkin. Kita dapat melakukan hal itu dengan cara membangun tower statis ditempat-tempat yang ditentukan. Untuk mendapatkan hasil yang maksimal salah satu strateginya adalah kita harus mampu menempatkan tower yang statis tersebut di tempat yang strategis. Untuk menempatkan tower ditempat yang strategis itulah digunakan algoritma *greedy*. Contoh tampilan peta yang akan digunakan dalam makalah ini adalah sebagai berikut

Gambar 1

3.2. Batasan

Tower

Asumsi yang digunakan adalah

- Menggunakan 1 level tower saja
- Menggunakan 1 jenis tower
- Jangkauan masing-masing tetap dan tertentu
- Tower ditempatkan di satu kotak atau blok yang tetap

Creep

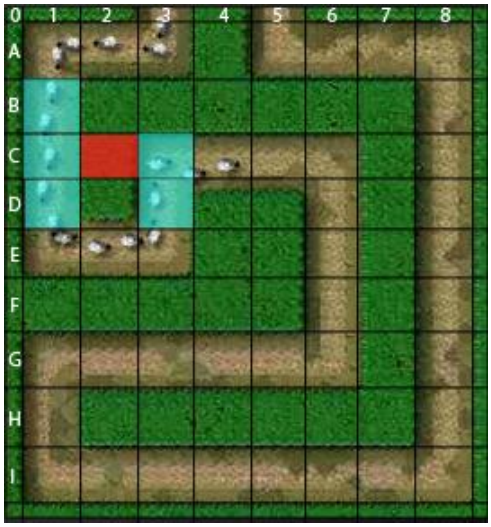
Creep / musuh dalam penjelasan ini adalah :

- Creep berjalan pada jalur yang tetap
- Kecepatan berjalan dari creep tetap
- Kemampuan bertahan dari serangan tiap gelombangnya tetap

Arena

Arena yang digunakan dibagi menjadi kotak-kotak yang lebih kecil untuk menentukan posisi.

Gambar peta yang telah ditambah dengan batasan adalah sebagai berikut :



Gambar 2

IV. ANALISIS

Dari persoalan dan batasan yang telah dijelaskan seperti diatas bahwa kita harus membunuh sebanyak mungkin musuh yang berjalan dengan batasan radius serangan yang terbatas. Oleh karena penempatan tower harus sesuai.
Dalam persoalan ini algoritma dapat dipakai untuk menyelesaikan permasalahan.

4.1. Ide Dasar

Dengan menggunakan algoritma *greedy* kita mencoba untuk menyelesaikan masalah penempatan tower yang efektif, dimana tujuannya adalah untuk mendapatkan nilai yang maksimal. Dengan algoritma *greedy* kita mencoba untuk mencari tempat-tempat untuk meletakkan tower dimana jarak serang ke musuh paling banyak. Dengan cara ini diharapkan kita mampu membunuh musuh sebanyak mungkin karena jumlah serangan yang kita lakukan terhadap musuh juga lebih banyak.

4.2. Elemen Greedy

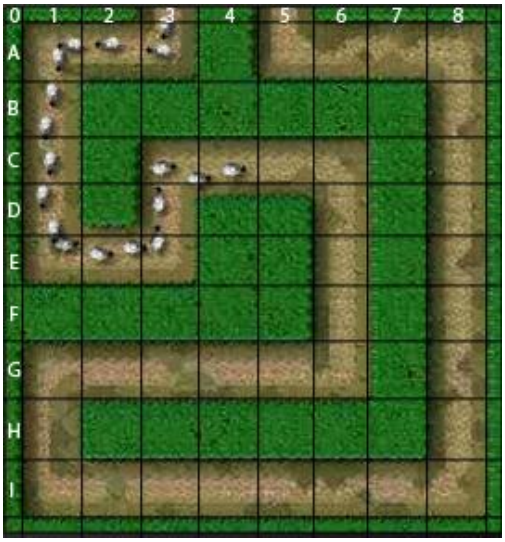
Untuk menyelesaikan masalah penempatan tower ini menggunakan elemen-elemen *greedy* antara lain

- Himpunan kandidat
  Dalam permasalahan ini himpunan kandidat adalah titik-titik yang ada di peta.
- Fungsi seleksi
  Fungsi untuk memilih titik di peta yang sesuai. Titik yang dipilih adalah titik yang mempunyai radius serangan yang besar.
- Fungsi kelayakan
  Fungsi untuk mengecek apakah titik tersebut dapat dibangun tower. Suatu titik dikatakan dapat dibangun tower adalah jika pada titik tersebut tidak ada tower lain dan titik tersebut bukan jalan

yang dilalui oleh creep serta uang untuk membangun tower cukup.

- Fungsi obyektif
  Fungsi obyektif dari adalah untuk menggunakan tower sesedikit mungkin.
- Himpunan solusi
  Himpunan solusi dapi permasalahan ini adalah himpunan titik-titik yang merupakan solusi tempat tower dibangun.

4.3. Penjelasan Algoritma

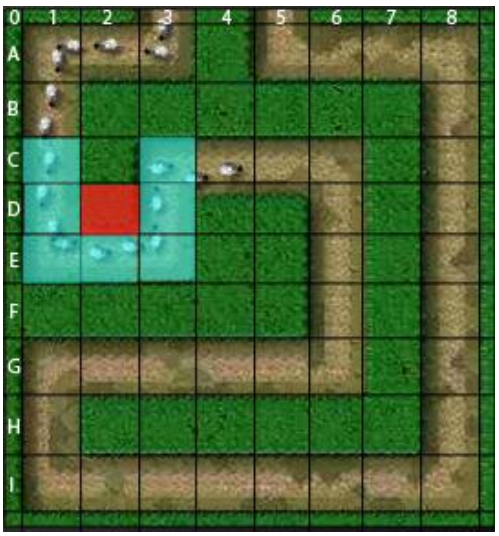


Gambar 3

Algoritma dan urutan pemrosesan untuk mendapatkan titik-titik himpunan solusi adalah sebagai berikut :

1. Proses yang pertama adalah mengambil titik-titik di peta yang masih kosong yang dapat dibangun tower. Sesuai dengan gambar 3 maka didapatkan himpunan titik (misal A) sebagai berikut
   A = { (1,F) , (2,B), (2,C), (2,D), (2,F), (3,B) , (3,F), (3,H), (4,A), (4,B), (4,D), (4,E), (4,F), (4,H), (5,B), (5,D), (5,E), (5,F), (5,H), (6,B), (6,H), (7,B), (7,C), (7,D), (7,E), (7,F), (7,G) , (7,H) }

2. Dari array yang telah didapat pada langkah 1
   Maka kita tinggal mencari tower yang ada didalam array A (langkah 1) dengan menggunakan fungsi findMax, untuk mendapatkan tower dengan radius serangan yang paling besar. Jika telah diketemukan maka titik tersebut akan dikeluarkan dari array A
   Misalkan pada langkah ini didapatkan titik pertama adalah (2,D). maka tower akan diletakkan di titik tersebut sesuai gambar berikut



Gambar 4

3. Langkah nomer 2 akan terus dilakukan hingga uang yang didapatkan habis.

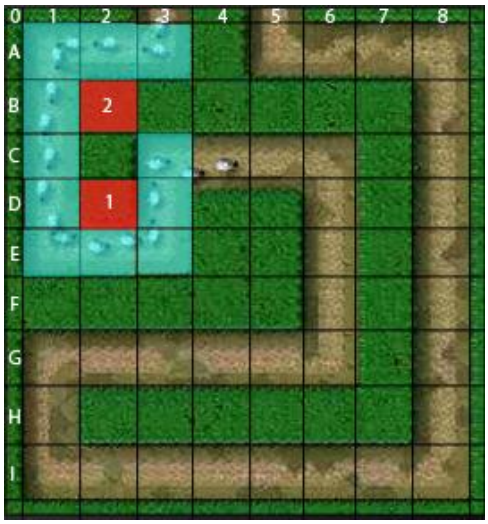
Pseudocode dari algoritma diatas adalah sebagai berikut:

```
function isAble(input P : peta, point
:titik) => Boolean
{fungsi ini akan mengembalikan Boolean
apakah titik tersebut dapat dibangun
tower}

function makeEnable(input P : peta) =>
array of point
{fungsi ini akan mengembalikan titik
di peta yang dapat dibangun tower,
menggunakan fungsi isAble}

function findMax(input listP : array
of point, P : peta} => point
{fungsi ini mengembalikan titik yang
mempunyai radius serang terbesar dan
mengeluarkan titik tersebut dari
listP}

Procedure putTower(input P : peta , T
: point)
{ I.S : peta yang masih kosong
  F.S : titik T di peta di bangun
tower}

Procedure placeTower(input P:peta)
{I.S : peta yang masih kosong
 F.S : peta telah dibangun tower}

Peta : P
list : array of point

list <- makeEnable(P)
while(not isEmpty(list) and uang >
harga_tower)
   putTower(P,findMax(list,P))
```

4.4 Kasus tidak optimal

Pada sub bab ini akan dijelaskan bahwa tidak selamanya algoritma *greedy* tersebut mampu memberikan solusi optimal. Salah satu kasus yang terjadi adalah jika tower telah ditempatkan di suatu titik yang dianggap telah sesuai dengan fungsi kelayakan, fungsi seleksi , dan fungsi obyektif . maka titik itu akan dimasukkan ke himpunan solusi , begitu pula seterusnya. Namun jika titik-titik solusi berdekatan dan creep(musuh) tidak mati semua. Maka titik tersebut bukan merupakan titik-titik yang optimal.

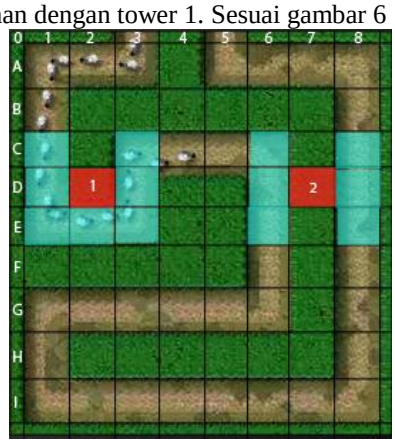
Untuk lebih jelasnya akan digambarkan dalam gambar berikut :



Gambar 5

Sesuai gambar diatas tower 1 diletakkan di tempat sesuai hasil fungsi seleksi, lalu algoritma akan meletakkan tower ditempat 2. Jika creep berhasil lolos (tidak mati) dari kedua tower diatas maka pemain akan kehilangan nilai dan titik-titik tersebut bukan titik-titik yang optimal

Akan lebih optimal jika tower yang kedua tidak diletakkan bersebelahan dengan tower 1. Sesuai gambar 6



Gambar 6

V. KESIMPULAN

Dari pemparan yang telah dijelaskan sebelumnya, kesimpulan yang didapatkan adalah algoritma *greedy* dapat memberikan solusi untuk persoalan penempatan tower dalam permainan tower defense. Akan tetapi dengan batasan-batasan yang jelas.

Walaupun algoritma ini masih ada kelemahan dimana tidak selalu mendapatkan solusi yang optimal, tetapi algoritma ini mampu untuk memberikan referensi bagaimana memilih titik untuk penempatan tower.

REFERENSI

Munir, Rinaldi. *Diktat Kuliah IF3051 Strategi Algoritma.* Bandung : Program Studi Teknik Informatika, Institut Teknologi Bandung, 2009.

http://en.wikipedia.org/wiki/Tower_defense
tanggal akses 7 desember 2011

http://en.wikipedia.org/wiki/Warcraft_III:_The_Frozen_Throne
tanggal akses 7 desember 2011

http://www.emanueleferonato.com/2007/10/06/make-a-flash-game-like-flash-element-tower-defense-part-1/
tanggal akses 8 desember 2011

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, 8 Desember 2011

Taufik Prasetiya / 13509029