Algoritma Greedy untuk AI dalam Permainan DotA

Kevin Leonardo Handoyo/13509019
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
xmc_z@live.com

***Abstrak* — DotA merupakan sebuah permainan komputer yang sangat populer hingga saat ini. Permainan ini merupakan permainan yang dapat dimainkan oleh maksimal sepuluh pemain. Dalam permainan ini terdapat dua kelompok, yaitu *sentinel* dan *scourge*, dimana setiap kelompok dapat berisi maksimal lima pemain. Tujuan permainan ini adalah menghancurkan gedung utama dari kelompok lawan. Karena DotA merupakan permainan *multiplayer*, terkadang orang merasa bila sedang sendirian dan ingin bermain ataupun berlatih bermain DotA, karena itulah dibutuhkan AI yang dapat menggantikan peran pemain manusia lainnya. Agar sebuah AI dapat memberikan suatu "tantangan" dalam bermain, dibutuhkan pengambilan-pengambilan keputusan yang tepat dalam bermain, salah satunya adalah memilih unit mana yang harus diserang terlebih dahulu. Algoritma greedy yang merupakan salah satu algoritma optimasi dapat digunakan sebagai dasar dari penentuan sebuah AI ketika hendak melakukan penyerangan terhadap unit lawan. Dalam permainan ini, algoritma greedy yang digunakan merupakan algoritma greedy yang tergantung pada keadaan unit-unit lawan, seperti jenis pasukan, peran dari Hero lawan, serta HP yang tersisa dari suatu unit lawan.**

***Kata Kunci*— DotA, algoritma greedy, AI**

I. PENDAHULUAN

Defense of the Ancients, atau yang lebih dikenal dengan sebutan DotA, adalah sebuah permainan yang merupakan *custom scenario* dari permainan strategi berjudul Warcraft III. *Custom scenario* ini pertama dibuat sejak Warcraft III masih belum memiliki ekspansi, atau disebut juga Warcraft III: Reign of Chaos, kemudian pengembangan dari *scenario* ini terus berlanjut sampai sekarang dimainkan dalam ekspansi dari Warcraft III yaitu The Frozen Throne. Versi pertama dari DotA dikembangkan oleh seorang *developer* yang memiliki alias "Eul" pada tahun 2003. Kemudian ketika mulai pindah *platform* ke Frozen Throne, *map* ini dikembangkan oleh "Guinsoo". Pengembangan *map* ini sejak tahun 2005 hingga sekarang dilakukan oleh seorang *developer* dengan alias "IceFrog".

Permainan ini hingga sekarang masih merupakan salah satu permainan yang paling banyak digemari orang di dunia. Pemainnya mencakup mulai dari anak-anak sampai orang dewasa. Permainan ini, yang sebenarnya hanya merupakan sebuah *custom scenario*, bahkan lebih sering dimainkan dan lebih terkenal daripada permainan utamanya sendiri yaitu Warcraft III. Permainan ini juga sering dijadikan lomba, bahkan dijadikan sebagai salah satu permainan yang setiap tahunnya diadakan pertandingan skala internasional.

Gambar 1. Tampilan *loading screen* DotA

DotA merupakan sebuah permainan *multiplayer*, hal ini menyebabkan para pemainnya semakin senang memainkannya karena dapat dilakukan bersama dengan teman-temannya. Namun akibatnya, ketika kita sedang sendirian, entah saat kita ingin bermain sendiri ataupun berlatih ketika baru memulai bermain DotA, akan terjadi kesulitan dan akan membosankan apabila permainan ini tidak memiliki AI yang dapat dijadikan teman dan lawan. Karena itulah, beberapa *developer* juga mengembangkan AI untuk *map* DotA.

Gambar 2. Tampilan "room" DotA

Sebuah AI harus memiliki suatu kemampuan untuk dapat mengambil keputusan-keputusan yang tepat dalam permainan, salah satunya adalah untuk menentukan unit mana yang harus diserang terlebih dahulu ketika bertemu beberapa unit lawan. Pemilihan unit ini didasarkan pada beberapa keputusan seperti keberadaan dan peran Hero lawan, HP yang tersisa dari unit lawan, serta jenis unit lawan.

II. DASAR TEORI

A. Gameplay DotA

Dalam permainan DotA, para pemain yang berjumlah maksimum 10 orang dibagi menjadi 2 kelompok, yaitu *Scourge* dan *Sentinel*. Kelompok *Sentinel* memiliki markas yang terletak di pojok kiri bawah peta, sedangkan *Scourge* memiliki markas yang terletak di pojok kanan atas peta.

Tujuan dari permainan ini adalah untuk menghancurkan gedung utama, yang disebut Ancient, milik pihak lawan. Kedua Ancient tersebut dilindungi oleh para pemain ataupun AI yang memainkan sebuah unit yang disebut Hero, serta unit-unit yang digerakkan AI yang disebut *creep*.

Permainan ini memiliki fokus untuk memperkuat Hero yang telah kita pilih agar kita mudah untuk menyerang musuh dan menghancurkan Ancientnya. Kita tidak dapat mengendalikan unit-unit seperti bangunan dan *creep*. Kita juga tidak dapat membuat unit-unit baru.

Gambar 3. Bangunan *Sentinel* dan *Scourge*

Hero dapat diperkuat dengan dua cara, yaitu dengan menaikkan level dan membeli *equipment*. Setiap kali Hero kita berada di dekat unit musuh yang terbunuh, Hero kita akan mendapat *experience point*. *Experience point* dibutuhkan untuk menaikkan level suatu Hero. Level maksimal yang dapat dicapai sebuah hero adalah 25. Dengan menaikkan level, atribut Hero tersebut bertambah, serta kita dapat mempelajari atau menaikkan level suatu skill. Bila Hero kita membunuh unit lawan, maka Hero kita akan mendapat gold yang dapat dipergunakan untuk membeli *equipment*.

Gambar 4. Tampilan permainan DotA

B. Heroes

Pada awal permainan, pemain manusia maupun AI memilih sebuah Hero yang akan dikendalikan olehnya sepanjang permainan. Hero adalah unit yang kuat dan memiliki komposisi atribut serta skill yang berbeda satu dengan lainnya. Setiap pemain dapat memilih salah satu dari Hero yang tersedia untuk setiap kelompok. Bila mode permainan adalah Allpick, maka pemain dapat memilih Hero yang merupakan Hero dari kelompok lawan, sedangkan bila mode permainan Allrandom, pemain akan dipilihkan Hero secara acak.

Suatu Hero memiliki 3 atribut yaitu Strength, Agility, dan Intelligence. Setiap Hero memiliki atribut utama yang merupakan salah satu dari ketiga atribut tersebut. Atribut Strength mempengaruhi jumlah HP (*Hit Point*) yang dimiliki oleh suatu Hero. Agility mempengaruhi kecepatan serangan dan besar Armor dari suatu Hero. Intelligence mempengaruhi jumlah MP (*Mana Point*) yang dimiliki suatu Hero.

Hero yang memiliki atribut utama tertentu biasanya memiliki beberapa karakteristik yang hamper sama. Hero Strength biasanya merupakan petarung jarak dekat, serta dapat berperan sebagai *tanker* dan *stunner/disabler*. Hero Agility biasanya merupakan Hero yang sangat bergantung pada *equipment* dan biasanya berperan sebagai *hitter*. Hero Intelligence biasanya merupakan petarung jarak jauh dan dapat berperan sebagai *healer*, *nuker*, dan *disabler/stunner*.

Pembagian Hero berdasarkan peran sebenarnya tidak memiliki aturan khusus, namun para pemain DotA biasanya mengelompokkan Hero berdasarkan peran-peran tertentu. Dalam makalah kali ini, saya akan membagi peran menjadi tiga jenis, yaitu *tank*, *carry*, dan *support*. *Tank* adalah Hero yang memiliki HP dan/atau Armor yang

sangat tinggi. Ia juga bertugas untuk "menahan" serangan terhadap timnya dan mengalihkan perhatian lawan. *Carry* adalah hero yang memiliki tingkat serangan yang sangat tinggi. Ia bertugas untuk menyerang dan membunuh lawan-lawannya. *Support* adalah Hero yang dapat menghentikan pergerakan musuh atau memperkuat Hero teman.

Dalam kenyataan bermain DotA, para pemain biasanya menerapkan strategi untuk membunuh Hero *support* terlebih dahulu karena sangat "mengganggu", kemudian baru membunuh *carry*, dan terakhir baru membunuh *tank*. Namun apabila ada Hero yang sekarat, biasanya akan langsung segera dibunuh dengan sedikit pertimbangan akan peran dari Hero tersebut.

Gambar 5. Icon Hero di DotA

C. Algoritma Greedy

Dalam kehidupan sehari-hari, banyak terdapat persoalan yang menuntut pencarian solusi optimum. Persoalan tersebut dinamakan persoalan optimasi (*optimization problems*). Persoalan optimasi adalah persoalan yangtidak hanya mencari sekedar solusi, tetapi mencari solusi terbaik (*best*). Solusi terbaik adalah solusi yang bernilai minimum atau maksimum dari sekumpulan alternatif solusi yang mungkin. Contohnya menentukan lintasan terpendek dalam sebuah graf, menentukan total keuntungan maksimum dari pemilihan beberapa objek, dan sebagainya. Pada persoalan optimasi, kita diberikan sekumpulan kendala (*constraint*) dan fungsi optimasi. Solusi yang memenuhi semua kendala disebut solusi layak (*feasible solution*). Solusi layak yang mengoptimumkan fungsi optimasi disebut solusi optimum.

Algoritma *greedy* mungkin merupakan metode yang paling popular untuk memecahkan persoalan optimasi. Algoritma ini sederhana dan lempang (*straightforward*). Secara harafiah *greedy* artinya rakus atau tamak, yaitu sifat yang berkonotasi negatif. Orang yang tamak biasanya akan mengambil sebanyak mungkin makanan (atau harta lainnya) yang tersedia tanpa memikirkan konsekuensi ke depan. Prinsip *greedy* adalah: "*take what you can get now!*". Ambil apa yang dapat Anda peroleh sekarang! Prinsip ini juga diadopsi dalam pemecahan masalah optimasi, tetapi tentu dalam konteks positif. Dalam kehidupan sehari-hari pun kita seringkali menggunakan prinsip *greedy*, misalnya:

1. Memilih beberapa jenis investasi (penanaman modal)
2. Mencari jalur tersingkat dari Bandung ke Surabaya
3. Memilih jurusan di Perguruan Tinggi
4. Bermain kartu remi

Algoritma *greedy* membentuk solusi langkah per langkah (*step by step*). Terdapat banyak pilihan yang perlu dieksplorasi pada setiap langkah solusi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya. Sebagai contoh, jika kita menggunakan algoritma *greedy* untuk menempatkan komponen di atas papan sirkuit (*circuit board*), sekali lagi sebuah komponen telah ditetapkan posisinya, komponen tersebut tidak dapat dipindahkan lagi.

Pendekatan yang digunakan di dalam algoritma *greedy* adalah membuat pilihan yang "tampaknya" memberikan perolehan terbaik, yaitu dengan membuat pilihan optimum lokal (*local optimum*) pada setiap langkah dengan harapan bahwa sisanya mengarah ke solusi optimum global (*global optimum*).

Algoritma *greedy* adalah algoritma yang memecahkan masalah langkah per langkah, pada setiap langkah:

1. mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsip "*take what you can get now!*")
2. berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global. Pada setiap langkah di dalam algoritma *greedy* kita baru memperoleh optimum lokal. Bila algoritma berakhir, kita berharap optimum lokal menjadi optimum global. Algoritma *greedy* mengasumsikan bahwa optimum local merupakan bagian dari optimum global.

Algoritma greedy disusun oleh elemen-elemen berikut:

1. Himpunan kandidat, C
   Berisi elemen-elemen pembentuk solusi.
2. Himpunan solusi, S
   Berisi kandidat-kandidat yang terpilih sebagai solusi persoalan.
3. Fungsi seleksi, predikat SELEKSI
   Memilih kandidat yang paling memungkinkan mencapai solusi optimal. Kandidat yang sudah dipilih

ada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.

4. Fungsi kelayakan (*feasible*), predikat LAYAK
   Memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala (*constraints*) yang ada. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan kandidat yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi.
5. Fungsi obyektif
   Fungsi yang memaksimumkan atau meminimumkan nilai solusi (misalnya panjang lintasan, keuntungan, dan lain-lain).

III. METODE

Ada dua jenis pendekatan yang dapat dilakukan dengan algoritma greedy dalam menyelesaikan masalah untuk menentukan unit mana yang harus diserang. Yang pertama adalah menyerang musuh yang memiliki darah paling rendah. Yang kedua adalah apabila ada Hero musuh dalam jangkauan AI, maka AI akan menyerang Hero berdasarkan prioritas role, urutannya AI akan berusaha membunuh *support* dahulu, kemudian *carry*, dan terakhir *tank*.

A. Greedy by Lowest HP

Yang dilakukan pada algoritma greedy by lowest HP ini adalah pertama mendaftarkan semua unit yang berada dalam jangkauan, kemudian mengecek unit manakah yang memiliki HP terendah, kemudian memutuskan untuk menyerang unit tersebut.

Elemen greedy:

1. Himpunan kandidat: unit musuh yang berada dalam jangkauan.
2. Himpunan solusi: unit musuh yang memiliki HP terendah.
3. Fungsi seleksi: pilih unit yang memiliki HP terendah.
4. Fungsi layak: semua unit yang dipilih layak.
5. Fungsi obyektif: menyerang unit yang memiliki HP terendah.

Dari pseudocode yang telah dibuat, fungsi greedy ini mengembalikan nilai index dari list unit musuh yang memiliki nilai HP terendah.

```
int lowestHP (ListMusuh L)
{
    HPmin = 99999;
    indexMin = 999;
    for (i=0;i < JmlhMusuh; i++)
    {
        if (L[i].HP < HPmin)
        {
            HPmin=L[i].HP;
            indexMin=i;
        }
    }
    return indexMin;
}
```

Gambar 6. Pseudocode greedy by lowest HP

B. Greedy by Priority

Algoritma ini pertama mendaftarkan unit musuh yang berada dalam jangkauan, kemudian menyerang unit dengan prioritas tertinggi. Dalam kasus ini, atribut prioritas dalam tipe bentukan unit akan ditambah nilai 1,2,dan 3 dimana nilai 1 berarti Hero *tank*, 2 berarti *carry*, dan 3 berarti *support*, dengan nilai semakin besar berarti prioritasnya untuk diserang semakin besar. Selain itu, ditambahkan juga 2 nilai prioritas untuk unit yang HPnya <50%, dan 2 nilai prioritas tambahan lagi untuk yang HPnya dibawah 20%.

Elemen greedy:

1. Himpunan kandidat: unit musuh yang berada dalam jangkauan.
2. Himpunan solusi: unit musuh yang memiliki nilai prioritas tertinggi.
3. Fungsi seleksi: pilih unit yang memiliki nilai prioritas tertinggi.
4. Fungsi layak: semua unit yang dipilih layak.
5. Fungsi obyektif: menyerang unit yang memiliki nilai prioritas tertinggi.

```
int priority (ListMusuh L)
{
    maxPrio = -1;
    indexMaxPri = 999;
    for (i=0;i < JmlhMusuh; i++)
    {
        if (L[i].Priority > maxPrio)
        {
            maxPrio=L[i].Priority;
            indexMaxPri=i;
        }
    }
    return indexMaxPri;
}
```

Gambar 7. Pseudocode greedy by priority

Dari pseudocode yang telah dibuat, fungsi greedy ini mengembalikan nilai index dari list unit musuh yang memiliki nilai prioritas tertinggi.

C. Fungsi Penentuan Penyerangan

Dengan menggunakan kedua algoritma greedy yang sudah dibuat sebelumnya, dapat ditentukan unit mana yang harus diserang berdasarkan HP ataupun peran. Dalam penerapannya di dalam AI permainan DotA, kedua fungsi tersebut harus digabungkan. Kedua algoritma tersebut digabungkan dengan alasan apabila ada dua unit atau lebih memiliki nilai prioritas yang sama, maka unit yang memiliki HP terendah akan diserang terlebih dahulu karena dengan unit musuh lebih sedikit maka serangan yang akan kita terima juga akan lebih sedikit.

Pada implementasinya, fungsi ini harus terus dipanggil setiap beberapa waktu sekali. Hal ini dikarenakan kondisi unit-unit musuh selalu berubah, baik itu keluar dari jangkauan, ataupun jumlah HP unit musuh yang berubah-ubah(terpukul atau di*heal*).

```
int pilihUnit (ListMusuh L)
{
    int index,count;
    count=0;
    for (i=0;i < JmlhMusuh; i++)
    {
        generatePriority(i); // inisialisasi nilai priority
    }
    maxPrio = -1;
    indexMaxPri = 999;
    for (i=0;i < JmlhMusuh; i++)
    {
        if (L[i].Priority > maxPrio)
        {
            maxPrio=L[i].Priority;
            indexMaxPri=i;
            count=1;
        }
        else if (L[i].Priority = maxPrio)
        {
            count++;
        }
    }
    if (count>1)
    {
        HPmin = 99999;
        indexMin = 999;
        for (i=0;i < JmlhMusuh; i++)
        {
            if ((L[i].HP < HPmin) && (L[i].Priority = maxPrio))
            {
                HPmin=L[i].HP;
                indexMin=i;
            }
        }
        index=indexMin;
        return index;
    }
    else {index=indexMaxPri;return index;}
}
```

Gambar 8. Pseudocode fungsi pilih unit diserang

IV. KESIMPULAN

Dalam menentukan unit mana yang harus diserang oleh AI dalam permainan DotA, dapat digunakan algoritma greedy gabungan dari by lowest HP dan by priority. Algoritma greedy gabungan ini ketika dicoba di dalam permainan akan memberikan hasil yang hampir optimal dan cukup mempersulit musuh.

REFERENSI

[1] Munir, Rinaldi. *Diktat Kuliah IF3051 Strategi Algoritma*. 2009.
[2] http://en.wikipedia.org/wiki/Defense_of_the_Ancients
[3] http://kikay-online-world.blogspot.com/

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, 9 Desember 2011

ttd

Kevin Leonardo Handoyo/13509019