Penerapan Algoritma *Greedy* Pada Permainan *Killbots*

Muhammad Reza Mandala Putra - 13509003
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung 40132, Indonesia
13509003@std.stei.itb.ac.id

***Sari*—Algoritma *Greedy* adalah salah satu algoritma yang sering dipakai dalam menyelesaikan masalah yang ada dalam kehidupan sehari-hari, seperti memecahkan masalah *Knapsack*, penukaran uang, penjadwalan *Job*, lintasan terpendek, dan lain sebagainya. Algoritma *Greedy* juga diterapkan dalam pembuatan intelegensia buatan (*Artificial Intelligent*, atau disingkat AI) untuk CPU. Salah satu aplikasi permainan yang menerapkan algoritma *Greedy* adalah aplikasi permainan *Killbots*. Yang akan dibahas pada makalah ini adalah penerapan algoritma *Greedy* pada permainan *Killbots* untuk menentukan langkah CPU agar mendapatkan jarak terdekat ke posisi pemain.**

***Kata Kunci — Killbots, Greedy, permainan, solusi optimum*.**

I. PENDAHULUAN

Dewasa ini, aplikasi permainan banyak sekali dijumpai. Aplikasi permainan yang telah dibuat ada yang hanya dapat dimainkan pada konsol tertentu, dan ada juga yang *multi-platform.* Aplikasi permainan yang telah dibuat ada yang bersifat berbayar da nada pula yang dapat dimainkan secara cuma-cuma.

Salah satu aplikasi permainan yang dapat dimainkan secara cuma-Cuma adalah *Killbots*. *Killbots* merupakan salah satu permainan sederhana yang berjalan pada komputer, baik komputer Desktop maupun Notebook. Permainan ini biasanya disertakan dalam paket instalasi dari salah satu sistem operasi berbasis Linux, yaitu Fedora. Aplikasi permainan *Killbots* dimainkan oleh seorang pemain. Aplikasi permainan tersebut dimainkan dengan cara menghindarkan karakter pemain dari kejaran *tank-tank* robot. Permasalahan yang terdapat dalam pembuatan intelegensia buatan untuk *tank-tank* robot adalah algoritma untuk membuat *tank-tank* robot bergerak sehingga menghasilkan jarak terdekat antara *tank-tank* robot dengan karakter pemain. Salah satu algoritma yang dapat digunakan untuk menyelesaikan masalah tersebut adalah algoritma *Greedy*.

II. ALGORITMA *GREEDY*

A. Definisi Algoritma Greedy

Algoritma *Greedy* merupakan salah satu metode untuk memecahkan persoalan optimasi tersebut. Algoritma ini bersifat sederhana dan lempang (*straightforward*). Secara harfiah, *Greedy* berarti rakus atau tamak. Prinsip *Greedy* adalah: "*take what you can get now*". Ambil apa yang dapat diperoleh sekarang. Prinsip tersebut diadopsi dalam pemecahan masalah optimasi. Algoritma *Greedy* akan 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. Pendekatan yang dilakukan di dalam algoritma *Greedy* adalah membuat pilihan yang "tampaknya" memberikan perolehan terbaik, yaitu dengan membuat pilihan optimum lokal pada setiap langkah dengan harapan bahwa sisanya mengarah ke solusi optimum global. Secara ringkas, algoritma *Greedy* adalah algoritma yang memecahkan masalah langkah per langkah, di mana pada setiap langkah:

Mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsip "*take what you can get now!*"). Asumsi bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

B. Skema Umum Algoritma Greedy

Dalam pemecahan masalah optimasi, algoritma *Greedy* disusun oleh elemen-elemen sebagai berikut:

a. Himpunan kandidat, C
   Himpunan ini berisi elemen-elemen pembentuk solusi. Contohny adalah himpuan koin, himpunan simpul di dalam graf.
b. Himpunan solusi, S.
   Berisi kandidat-kandidat yang terpilih sebagai solusi persoalan. Himpunan solusi merupakan himpunan bagian dari himpunan kandidat.
c. Fungsi seleksi, dinyatakan dengan predikat SELEKSI

Fungsi yang pada setiap langkah memilih kandidat yang paling memungkinkan mencapai solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya. Biasanya setiap kandidat, x, di-*assign* sebagai sebuah nilai numerik, dan fungsi seleksi memilih x yang mempunyai nilai terbesar atau memilih x yang mempunyai nilai terkecil.

d. Fungsi kelayakan (*feasible*), dinyatakan dengan predikat LAYAK
Fungsi yang memeriksa apaakh suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala (*constraint*) yang ada. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan kandidat yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi.

e. Fungsi obyektif
Fungsi yang memaksimumkan atau meminimumkan nilai solusi (misalnya panjang lintasan, keuntungan, dan lain-lain).

Persoalan optimasi yang diselesaikan dengan algoritma *Greedy* melibatkan pencarian sebuah himpunan bagian, S, dari himpunan kandidat, C; di mana S harus memenuhi beberapa kriteria yang ditentukan, yaitu menyatakan suatu solusi dan S dioptimasi oleh fungsi obyektif. Optimasi tersebut dapat berarti minimisasi atau maksimasi, bergantung pada persoalan yang dipecahkan. Optimum global yang dihasilkan dari algoritma *Greedy* belum tentu merupakan solusi optimum (terbaik), tetapi dapat merupakan solusi sub-optimum atau pseudo-optimum. Hal ini dikarenakan oleh dua faktor berikut:

- Algoritma *Greedy* tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada (seperti pada metode *exhaustive search*)
- Pemilihan fungsi SELEKSI: fungsi SELEKSI biasanya didasarkan pada fungsi obyektif.

Dengan kata lain, algoritma *Greedy* tidak selalu menghasilkan solusi yang benar-benar optimum. Algoritma *Greedy* sering menjadi basis untuk pendekatan heuristik. Jika persoalan dapat dipecahkan secara eksak (memberikan solusi yang optimum) dengan algoritma *Greedy*, maka pembuktian kebenaran dari algoritma *Greedy* tersebut dapat dianggap sebagai sebuah proses yang non-trivial.

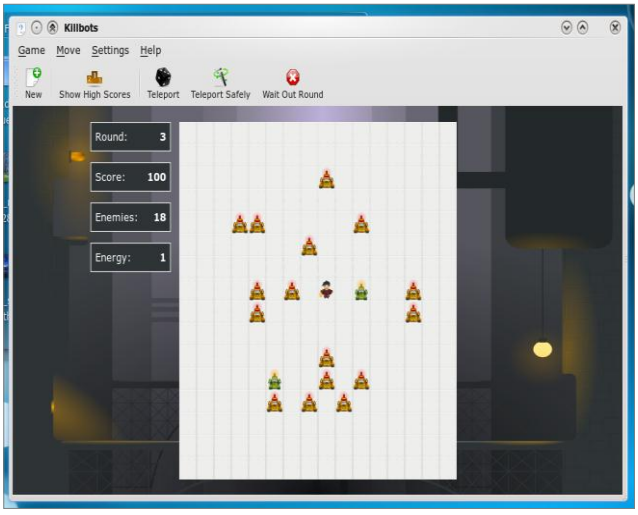
Skema umum dari algoritma *Greedy* dapat kita tuliskan sebagai berikut :

1. Inisialisasi S dengan kosong.
2. Pilih sebuah kandidat dari C (dengan SELEKSI).
3. Kurangi C dengan kandidat yang telah terpilih di atas.
4. Periksa apakah kandidat yang dipilih tersebut bersama – sama dengan himpunan solusi membentuk solusi yang layak (dengan LAYAK). Jika ya, masukkan kandidat ke himpunan solusi; jika tidak buang kandidat tersebut dan tidak perlu ditelaah lagi.
5. Periksa apakah himpunan solusi yang sudah terbentuk telah memberikan solusi yang lengkap (dengan SOLUSI). Jika ya, berhenti; jika tidak, ulangi dari langkah 2.

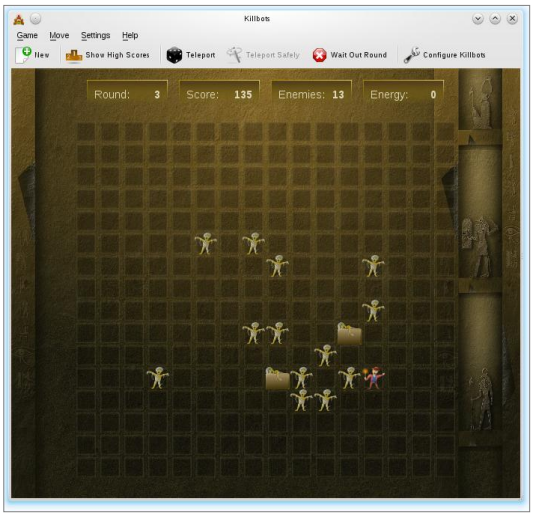
III. *KILLBOTS*

A. Sekilas tentang Killbots

Killbots – singkatan dari *Killer Robots* – adalah sebuat permainan sederhana yang dimainkan oleh satu orang pemain. Permainan ini dilakukan secara bergantian antara pemain dengan CPU (sering disebut komputer). Cara memainkannya adalah dengan cara menghindarkan pemain dari *tank-tank* robot yang mendekatinya. Pemain juga dapat menghancurkan *tank-tank* yang ada dengan cara membuat *tank-tank* tersebut saling menabrak. Caranya adalah dengan membuat dua *tank* bergerak pada satu titik. Apabila pada satu titik terdapat dua tank atau lebih, tank-tank tersebut akan hancur akibat saling bertabrakan. Pemain juga dapat menghancurkan *tank-tank* tersebut dengan cara membuat *tank* bergerak pada satu titik tempat *tank* yang telah hancur sebelumnya berada. Pemain dinyatakan menang apabila pemain berhasil menghancurkan semua *tank-tank* yang ada pada arena permainan. Akan tetapi, pemain juga dapat dinyatakan kalah apabila salah satu *tank* berhasil menabrak pemain. Berikut ini adalah beberapa *screenshot* dari permainan *Killbots.*



Gambar 1. Screenshot tampilan permainan Killbots dengan tema Robot Kill



Gambar 2. Screenshot tampilan permainan Killbots dengan tema Mummy Madness

B. Gameplay pada Killbots

Pada awal permainan, pemain dihadapkan pada sebilah papan 2D di layar monitor. Papan tersebut berukuran 16 x 16 kotak. Pada awal permainan, posisi pemain berada di tengah-tengah papan dan di sekelilingnya terdapat beberapa *tank* yang jaraknya berbeda-beda. Pemain digerakkan dengan menggunakan tetikus *(mouse).* Pemain bergerak ketika *mouse* diklik dan pergerakan pemain ditentukan oleh posisi *cursor* tetikusnya. Misalnya, apabila *cursor* berada di kiri pemain maka pemain akan bergerak ke kiri sebanyak satu kotak setelah dilakukan kilk kiri pada tetikus. Namun apabila pemain melakukan klik kanan pada tetikus, karakter pada papan akan melakukan teleportasi secara acak.

Setelah pemain bergerak, *tank-tank* robot secara otomatis akan bergerak juga secara serentak. Mereka akan bergerak mengikuti arah pergerakan pemain. Apabila dua *tank* atau lebih menuju satu kotak yang sama, *tank-tank* tersebut akan bertabrakan dan *tank-tank* tersebut akan hancur. Setelah *tank* hancur, pemain akan mendapatkan skor sesuai dengan tipe *tank* yang hancur. Apabila semua *tank* telah hancur, pemain dapat melanjutkan permainan ke ronde berikutnya. Namun apabila *tank* robot berhasil menabrak pemain, permainan dinyatakan berakhir.

IV. PENERAPAN *GREEDY* PADA *KILLBOTS*

Dalam konteks algoritma *Greedy*, elemen-elemen persoalan pergerakan robot adalah sebagai berikut:

1. Himpunan kandidat adalah himpunan arah.
2. Himpunan bagian dari himpunan arah, S, adalah solusi jika jarak pemain dengan posisi robot setelah melangkah lebih dekat daripada jarak pemain dengan posisi robot sebelumnya.
3. S disebut layak jika jumlah langkah yang ada di dalam S tidak melebihi jumlah langkah maksimum robot.
4. Fungsi obyektif adalah meminimumkan jarak antara robot dengan pemain berdasarkan langkah-langkah yang ada di dalam S.
5. Fungsi seleksi adalah memilih kandidat (yaitu arah) yang menghasilkan jarak terdekat antara robot dengan pemain.

Berikut adalah contoh pergerakan pemain dari posisi pada awal permainan di ronde 2 menuju ke arah bawah.

Gambar 3. Posisi awal pemain dan robot pada ronde 2

Gambar 4. Posisi pemain dan robot pada saat pemain bergerak ke bawah

Berikut ini adalah *pseudo-code* yang penulis coba buat untuk melakukan pergerakan robot.

```
Procedure move(input/output listOfRobots : List of Bot, input player : Player)
{ Mengerakkan semua robot berdasarkan posisi pergerakan pemain.
  I.S.  : posisi robot sembarang
  F.S. : posisi robot berpindah sejauh satu kotak berdasarkan posisi pemain
}

KAMUS LOKAL
   distance : integer { jarak antara robot dengan pemain }
   i : integer { variable untuk melakukan iterasi }
ALGORITMA
   for each (Bot bot in listOfRobots)
     i traversal [1..maxSteps]
       if (bot.getX() = player.getX()) then
          distance ← bot.getX()- player.getX()
          if (distance > 0) then
             bot.moveLeft()
          else
             bot.moveRight()
```

```
else if (bot.getY()  = player.getY() ) then
    distance ← bot.getY() - player.getY()
      if (distance > 0) then
        bot.moveUp()
      else
        bot.moveDown()
else
    bot.moveDiagonal(player.getX(), player.getY())
```

Prosedur move(listOfRobots, player) ini digunakan setelah pemain melakukan pergerakan. Pada prosedur ini, setiap robot yang ada di dalam listOfRobot (robot yang masih "hidup") akan dilakukan perbandingan antara posisinya saat itu dengan posisi pemain setelah bergerak. Apabila robot berada pada koordinat sumbu-x yang sama dengan koordinat sumbu-x pemain maka robot akan digerakkan secara horizontal mendekati pemain. Namun apabila robot berada pada koordinat sumbu-y yang sama dengan koordinat sumbu-y pemain, robot akan digerakkan secara vertikal mendekati pemain. Selain daripada kedua kasus tersebut, akan dilakukan pergerakkan robot secara diagonal mendekati pemain. Jumlah robot melangkah bergantung pada tipe robot tersebut.

Berikut ini adalah hasil enumerasi himpunan bagian arah pergerakan robot untuk tipe robot 1 dan 2 dengan studi kasus posisi awal pemain berada pada koordinat (1,2) dan posisi awal robot berada pada koordinat (4,7).

Tipe Robot	Jumlah Langkah Maksimum	Posisi Pemain	Posisi Robot Sebelum Bergerak	Arah Pergerakan	Posisi Robot Sesudah Bergerak	Keterangan
1	1	<1,2>	<4,7>	Kiri	<3,7>	Layak
1	1	<1,2>	<4,7>	Kanan	<5,7>	Tidak Layak
1	1	<1,2>	<4,7>	Atas	<4,6>	Layak
1	1	<1,2>	<4,7>	Bawah	<4,8>	Tidak Layak
1	1	<1,2>	<4,7>	Serong Kiri Atas	<3,6>	Layak
1	1	<1,2>	<4,7>	Serong Kanan Atas	<5,6>	Tidak Layak
1	1	<1,2>	<4,7>	Serong Kiri Bawah	<3,8>	Tidak Layak
1	1	<1,2>	<4,7>	Serong Kanan Bawah	<4,8>	Tidak Layak
1	1	<1,2>	<4,7>	Atas, Kiri	<3,6>	Tidak Layak
1	1	<1,2>	<4,7>	Atas, Kanan	<5,6>	Tidak Layak
1	1	<1,2>	<4,7>	Bawah, Kiri	<3,8>	Tidak Layak
1	1	<1,2>	<4,7>	Bawah, Kanan	<4,8>	Tidak Layak
2	2	<1,2>	<4,7>	Kiri	<3,7>	Layak
2	2	<1,2>	<4,7>	Kanan	<5,7>	Tidak Layak
2	2	<1,2>	<4,7>	Atas	<4,6>	Layak
2	2	<1,2>	<4,7>	Bawah	<4,8>	Tidak Layak
2	2	<1,2>	<4,7>	Serong Kiri Atas	<3,6>	Layak
2	2	<1,2>	<4,7>	Serong Kanan Atas	<5,6>	Tidak Layak
2	2	<1,2>	<4,7>	Serong Kiri Bawah	<3,8>	Tidak Layak
2	2	<1,2>	<4,7>	Serong Kanan Bawah	<5,8>	Tidak Layak
2	2	<1,2>	<4,7>	Atas, Kiri	<3,6>	Layak
2	2	<1,2>	<4,7>	Atas, Kanan	<5,6>	Tidak Layak
2	2	<1,2>	<4,7>	Bawah, Kiri	<3,8>	Tidak Layak
2	2	<1,2>	<4,7>	Bawah, Kanan	<5,8>	Tidak Layak
2	2	<1,2>	<4,7>	Serong Kiri Atas, Serong Kiri Atas	<2,5>	Layak
2	2	<1,2>	<4,7>	Serong Kanan Atas, Serong Kanan Atas	<6,5>	Tidak Layak
2	2	<1,2>	<4,7>	Serong Kiri Bawah, Serong Kiri Bawah	<2,9>	Tidak Layak
2	2	<1,2>	<4,7>	Serong Kanan Bawah, Serong Kanan Bawah	<6,9>	Tidak Layak

Tabel 1. Hasil Enumerasi Himpunan Bagian Arah Pergerakan Robot

Untuk tipe robot 1, arah-arah pergerakan robot yang bersifat layak dari himpunan arah tersebut antara lain arah pergerakan ke kiri, atas, dan serong kiri atas. dari ketiga arah tersebut, yang menghasilkan jarak terdekat dengan pemain pada tabel di atas adalah arah pergerakan ke serong kiri atas. Ini merupakan solusi optimum untuk kasus dimana posisi pemain berada di wilayah kiri atas robot. Sedangkan untuk tipe robot 2, arah-arah pergerakan robot yang bersifat layak dari himpunan arah tersebut antara lain arah pergerakan ke kiri, atas, serong kiri atas, dan kombinasi di antara tiga arah tersebut. Dari arah-arah tersebut, yang menghasilkan jarak terdekat dengan pemain pada tabel di atas adalah arah pergerakan ke serong kiri atas dan dilanjutkan ke arah serong kiri atas. Ini merupakan solusi optimum untuk tipe robot 2.

Untuk kompleksitas algoritma pada prosedur move(listOfRobots, player), penulis malakukan perhitung dengan cara sebagai berikut.

```
for each (Bot bot in listOfRobots)                                  m
  i traversal [1..maxSteps]                                         O(n)
    if (bot.getX() = player.getX()) then
      distance ← bot.getX()- player.getX()                          O(1)
      if (distance > 0) then
        bot.moveLeft()                                              O(1)
      else
        bot.moveRight()                                             O(1)
    else if (bot.getY()  = player.getY() ) then
      distance ← bot.getY() - player.getY()                         O(1)
      if (distance > 0) then
        bot.moveUp()                                                O(1)
      else
        bot.moveDown()                                              O(1)
    else
      bot.moveDiagonal(player.getX(), player.getY())                O(1)
```

Kompleksitas waktu asimptotik algoritma :

= m . (O(n) . (max(O(1) + max(O(1), O(1)), O(1) + max(O(1), O(1)), O(1))))

= m . (O(n) . max(O(1) + O(1), O(1) + O(1), O(1))))

= m . ((O(n) . max(max(1,1), max(1,1), O(1))))

= m . ((O(n) . max(1, 1, O(1))))

= m . O(n) = O(m . n) = O(mn)

Jadi didapat kompleksitas algoritma pada prosedur move(listOfRobots, player) adalah O(mn), dengan *m* adalah jumlah robot yang ada di dalam listOfRobots dan *n* adalah jumlah maksimum langkah pada robot.

V. KESIMPULAN

Dari hasil pembahasan di atas, dapat ditarik kesimpulan sebagai berikut:

1. Algortima *Greedy* dapat diterapkan pada permainan *Killbots.*
2. Solusi langkah optimum untuk melakukan pergerakan pada robot bergantung pada posisi pemain setelah bergerakdan posisi awal robot dan kemiringan garis yang ditarik dari kedua posisi tersebut.
3. Kompleksitas algoritma untuk melakukan pergerakan robot adalah O(mn), dengan *m* adalah jumlah robot yang masih tersisa dan *n* adalah jumlah maksimum langkah pada robot.

REFERENSI

[1] Munir, Rinaldi, *Diktat Kuliah IF3051 Strategi Algoritma*. Bandung : STEI ITB, 2011, halaman 26-69.

[2] Munir, Rinaldi, *Matematika Diskrit Revisi Keempat*, Bandung : INFORMATIKA, 2010, halaman 519.

[3] http://docs.kde.org/stable/en/kdegames/killbots/howto.html tanggal akses 8 Desember 2011

[4] http://en.wikipedia.org/wiki/Greedy_algorithm tanggal akses 9 Desember 2011

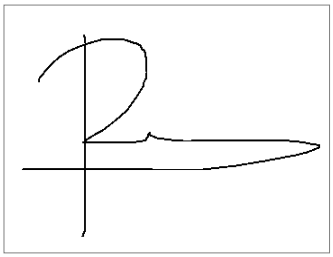
[5] http://www.informatika.org/~rinaldi/Stmik/2011-2012/Algoritma%20Greedy.ppt tanggal akses 8 Desember 2011.

[6] http://www.kde.org/announcements/4.2/edu_games.php 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, 9 Desember 2011



Muhammad Reza Mandala Putra - 13509003