Penentuan Langkah dengan Greedy dalam Permainan Ludo

Lio Franklyn Kemit/13509053
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
lio.f.kemit@gmail.com

Abstrak—Ludo adalah sebuah permainan papan yang dimainkan hampir seluruh belahan dunia. Permainan ini dimenagnkan dengan memindahkan pion-pion yang kita miliki menuju kotak tujuan. Berbagai taktik dapat digunakan untuk mencapai tujuan tersebut. Dengan algoritma *greedy* kita dapat membuat berbagai taktik untuk dapat memenangkan permainan ini. Pada makalah ini akan dibahas berbagai taktik *greedy* yang dapat digunakan untuk memainkan permainan.

Kata kunci—*Greedy*, Ludo, Papan, Pion, Taktik.

I. PENDAHULUAN

Siapa yang tidak kenal Ludo? Mungkin diantara kita sering memainkan permainan ini sewaktu kita kecil. Ludo adalah sebuah permainan papan yang dimainkan oleh dua sampai empat pemain yang menggerakkan pion-pionnya menuju kotak tujuan sesuai dengan angka yang tertera pada dadu yang diacak. Ludo pertama sekali muncul di India yang lebih dikenal dengan nama Pachisi pada abad keenam. Kemudian ludo berkembang di berbagai negara dan dimainkan hampir seluruh dunia. Ludo sering menjadi pilihan permainan anak kecil.

Gambar 1Papan Permainan Ludo

Permainan ludo bertujuan untuk menjalankan semua pion yang dimiliki pemain menuju kotak tujuan di tengah papan. Kita dapat memilih salah satu warna pion dari empat warna, biasanya merah, biru, kuning, dan hijau, yang kita ingin mainkan.

Cara bermain untuk mencapai tujuan dapat dilakukan dengan bermacam-macam taktik. Salah satu taktik yang sering digunakan antara lain berusaha secepat mungkin mencapai tujuan atau berusaha menghalangi pemain lain. Taktik-taktik tersebut biasanya muncul dari pengalaman selama bermain.

Mungkin dulu kita tidak sadar, tentang taktik-taktik tersebut. Tetapi sebenarnya cara-cara atau taktik-taktik yang kita mainkan tersebut sebenarnya adalah penerapan salah satu algoritma penyelesaian masalah yaitu algoritma *greedy*. Dengan strategi berdasarkan algoritma *greedy* yang berbeda-beda tersebut kita berusaha untuk mencapai tujuan dan menang dengan cara yang berbeda-beda pula.

II. GREEDY

Greedy adalah salah satu algoritma yang paling sering digunakan untuk menyelesaikan sebuah persoalan optimasi. Persoalan optimasi adalah persoalan dimana kita mencari suatu solusi yang dapat menyelesaikan persoalan tersebut secara optimum.

Persoalan optimasi secara garis besar terbagi dua, yaitu:

1. Persoalan optimasi maksimasi

Persoalan optimasi maksimasi adalah persoalan dimana kita mencari solusi untuk memperoleh hasil sebesar mungkin.

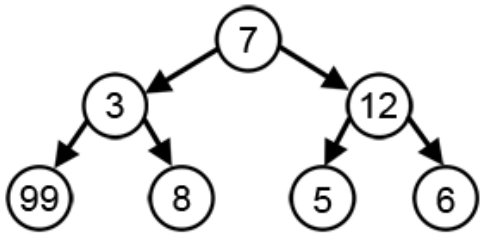
2. Persoalan optimasi minimasi.

Persoalan optimasi minimasi adalah persoalan dimana kita mencari solusi biasanya dengan membutuhkan biaya sesedikit mungkin.

Beberapa persoalan yang biasanya diselesaikan dengan *greedy* antara lain persoalan *knapsack*, yaitu persoalan penempatan barang ke dalam sebuah kantung dengan batasan berat namun dapat menghasilkan keuntungan sebesar mungkin atau persoalan penukaran uang, yaitu persoalan menukarkan uang dengan sejumlah koin namun dengan jumlah koin sesedikit mungkin.

Algoritma *greedy* memiliki konsep sesuai dengan namanya *greedy* yang berarti rakus atau tamak. Algoritma ini akan membentuk solusi langkah per langkah. Pada setiap langkah penyelesaian masalah terdapat banyak pilihan langkah yang dapat kita ambil. Pada algoritma ini kita berusaha mengambil keputusan terbaik yang dapat diambil pada saat ini. Pengambilan langkah terbaik pada saat ini disebut optimum lokal. Dengan mengambil

langkah yang memberikan optimum lokal pada setiap langkah diharapkan kita dapat mencapai optimum global.



Gambar 2 Kasus pemilihan yang tidak optimal

Namun pada kenyataanya, optimum global yang diperoleh dengan menggunakan algoritma *greedy* belum tentu merupakan solusi terbaik. Seperti pada kasus pada gambar 2. Kasus ini adalah kasus pencarian total nilai maksimum dari angka-angka. Dengan algoritma *greedy*, pada tahap awal yang dipilih adalah angka 12 dan kemudian angka 6. Jika ditotal menghasilkan 25. Padahal jika kita memilih 3, kemudian kita memilih 99, kita dapat memperoleh angka dengan total 109.

Hal ini disebabkan algoritma *greedy* tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada. Selain itu terdapat beberapa fungsi seleksi yang berbeda, sehingga kita harus memilih fungsi yang tepat jika kita ingin algoritma ini menghasilkan solusi yang optimal. Jadi dapat kita simpulkan algoritma *greedy* tidak selalu memberikan solusi yang optimal.

III. LUDO DAN ATURAN PERMAINAN

Ludo dimainkan oleh dua sampai empat pemain. Setiap pemain menggerakkan pion sesuai dengan pilihan warnanya masing-masing. Setiap pemain memiliki empat buah pion yang harus digerakkan menuju kotak tujuan di tengah papan. Pion-pion tersebut digerakkan di sebuah papan seperti pada gambar 1.

Karena dimainkan oleh banyak orang di seluruh dunia, terdapat juga banyak aturan-aturan yang berbeda-beda. Namun untuk makalah ini, langkah dan aturan permainan ludo adalah sebagai berikut:

1. Pada awal permainan, setiap pemain meletakkan pionnya di tempat setiap warna.
2. Pemain yang sedang bermain melemparkan dadu.
3. Pemain harus terlebih dahulu mendapatkan angka enam untuk memindahkan pion dari daerah siap ke kotak awal.
4. Pemain menggerakkan pion-pion miliknya sejumlah angka yang tertera pada dadu yang telah dilempar.
5. Jika angka yang didapat enam lagi, pemain dapat memilih antara mengeluarkan pion yang berada di daerah siap atau menggerakkan pion yang sudah berada di luar.
6. Jika pemain meletakkan pionnya di tempat yang sudah ditempati oleh pion pemain lain maka pemain memukul pion tersebut dan pion tersebut akan kembali ke daerah siap.
7. Jika pemain meletakkan pionnya di tempat yang ditempati oleh pion pemain tersebut, pion tersebut dapat ditumpuk dan tidak boleh dilewati oleh pemain lainnya.
8. Pemain yang sudah melakukan satu putaran harus masuk ke jalur tujuan dan melemparkan dadu sesuai angka yang dibutuhkan untuk masuk agar dapat masuk ke kotak tujuan.

IV. PENERAPAN ALGORITMA GREEDY

Terdapat beberapa taktik yang mengimplementasikan greedy untuk menyelesaikan permainan ludo ini. Setiap taktik berisi cara yang berbeda. Pada bagian ini akan dibahas beberapa cara penyelesaian tersebut.

Sebelumnya akan dibahas struktur data permainan ludo ini. Pada permainan ludo ini terdapat kelas pion yang memiliki atribut berupa posisi pion. Kelas pion ini akan dimiliki oleh kelas pemain yang memiliki *array* yang berisi empat buah pion. Kelas pemin juga memiliki prosedur melangkah yang akan mengubah posisi pion pada kelas papan. Terkahir terdapat kelas dadu yang akan mengacak angka dan dapat diambil oleh kelas lain.

A. Greedy by no one left behind

Pada *greedy* ini, algoritma akan berusaha untuk mengerakkan semua pion menuju titik tujuan bersama-sama. Setiap langkah yang dapat diambil digunakan untuk menggerakkan semua pion agar semakin dekat dengan tujuan. Sebelum melangkah dipilih pion dengan prioritas tertinggi. Pion dengan prioritas tertinggi adalah pion yang paling belakang dan dapat melangkah. Dengan memilih pion paling belakang kita melangkah dengan tujuan seluruh pion bergerak bersama menuju tujuan. Di bawah ini adalah *pseudocode greedy by no one left behind*.

```
Procedure GreedyByNoOneLeftBehind()
{Prosedur ini akan menentukan pengambilan
langkah pada saat ini}
Kamus
chosen: integer
Algoritma
if Win()=true then
     {Do Nothing}
else
     if getDadu()=6 then {prosedur untuk
mengambil putaran dadu saat ini}
          getPionOut() {mengeluarkan pion}
     else
     repeat getDadu() times
          chosen=farPion()  {pilih salah
satu pion dengan jarak paling jauh dari
tujuan}
          movePion(chosen) {prosedur untuk
menggerakkan sebuah pion satu langkah}
```

B. Greedy by hitter

Pada *greedy* ini, algoritma akan berusaha untuk menggerakkan pion agar dapat memukul pion pemain lain. Pemilihan pion yang digerakkan adalah berdasarkan

prioritas tergantung kondisi pion saat ini. Prioritas paling tinggi adalah pion yang paling dekat pion lain dan dapat memukul pion pemain lain. Prioritas kedua adalah jika tidak bisa memukul pion lain, pion tersebut akan maju mendekati pion lawan yang terdekat. Jika tidak ada pion yang memungkinkan, maka algoritma akan mencari pion lain yang dapat digerakkan. Di bawah ini adalah *pseudocode greedy by hitter*.

```
Procedure GreedyByHitter()
{Prosedur ini akan menentukan pengambilan
langkah pada saat ini}
Kamus
chosen: integer
move: integer
Algoritma
if Win()=true then
      {Do Nothing}
else
      if getDadu()=6 then
            getPionOut()
      else
            chosen=nearestEnemyPion()
{memilih pion yang paling dekat pion lain}
      if(ableEat(getDadu(),chosen)=true)   then
{menentukan apakan pion dapat memukul pion
lain}
                  repeat  moveToEat()  times
{menghitung jarak dari pion ke lawan}
                        movePion(chosen)
                        move= move-1
            else if (moveToEat()<getDadu())
                  repeat moveToEat()-1 times
                        movePion(chosen)
                        move= move-1
      if (move≠0) then
            chosen=moveablePion()    {mengacak
pion yang dapat dijalankan}
            repeat move times
                  movePion(chosen)
```

C. Greedy by quick move

Pada *greedy* ini, algoritma akan berusaha untuk menggerakkan pion secepat mungkin menuju tujuan. Pemilihan pion yang akan digerakkan berdasarkan prioritas. Prioritas tertinggi adalah pion yang dapat bergerak dan berada paling dekat dengan tujuan. Dengan menggerakkan pion yang paling dekat dengan kotak tujuan, kita akan lebih cepat mencapai tujuan. Selama pion yang paling dekat dengan kotak tujuan tidak terhalangi, maka seluruh langkah yang dapat diambil akan digunakan untuk memajukan pion tersebut. Jika tidak ada yang bisa bergerak maka algoritma akan menggerakkan pion lainnya sejumlah angka yang tertera di dau atau sisa langkah yang dapat diambil.

```
Procedure GreedyByQuickMove()
{Prosedur ini akan menentukan pengambilan
langkah pada saat ini}
Kamus
chosen: integer
move: integer
Algoritma
if Win()=true then
      {Do Nothing}
else
      move=getDadu()
      chosen=nearPion()   {memilih   pion   yang
paling dekat dengan tujuan}
      if chosen≠0 then
            while  (ableMove(chosen)=true  and
move>0) then
                        movePion(chosen)
                        move=move-1
      if move=6 then
            getPionOut()
      else
            chosen=moveablePion()
            repeat move times
                  movePion(chosen)
```

D. Greedy by blocking

Pada *greedy* ini, algoritma akan berusaha untuk menggerakkan pion dengan tujuan menghalangi pemain lain. Prioritas utama dari algoritma ini adalah menggabungkan lebih dari satu pion agar dapat menjadi penghalang. Jika ada pion yang dapat digabung, pion tersebut akan maju ke tempat pion yang ada. Kemudian penghalang tersebut akan tetap diam sampai tidak ada langkah lain yang dapat diambil. Jika tidak ada pion yang dapat digabung, maka algoritma akan memilih salah satu pion selain pion yang menjadi penghalang yang dapat dijalankan sejauh langkah yang tersisa atau angka yang keluar pada dadu.

```
Procedure GreedyByBlocking()
{Prosedur ini akan menentukan pengambilan
langkah pada saat ini}
Kamus
chosen: integer
move: integer
Algoritma
if Win()=true then
      {Do Nothing}
else
      move=getDadu()
      chosen=ableBlock()  {memilih  pion  yang
dapat membuat penghalang}
      if chosen≠0 then
            while (makeBlock(chosen)≠true and
move>0) then
                  movePion(chosen)
                  move=move-1
      if move=6 then
            getPionOut()
      else
            chosen=moveablePion()
            repeat move times
                  movePion(chosen)
```

V. Analisis Algoritma Greedy

Pada bagian sebelumnya, sudah dibahas empat algoritma *greedy* yang dapat digunakan untuk menyelesaikan persoalan permainan Ludo ini. Setiap algoritma tersebut memiliki kelebihan dan kekurangannya masing-masing. Selain itu terdapat juga kondisi yang dibutuhkan agar algoritma *greedy* tersebut dapat berjalan seoptimal mungkin. Pada bagian ini akan dibahas kekurangan dan kelebihan masing-masing algoritma. Lalu akan dibahas perbandingan keempat algoritma tersebut.

A. Greedy by no one left behind

Dengan menggunakan algoritma ini, kemenangan dapat

diraih dengan kecepatan yang menengah dan jika tidak menghadapai kendala, keempat pion dapat dengan cepat masuk bersamaan ke kotak tujuan karena pada algoritma ini semua pion diusahakan dapt bergerak bersamaan. Selain itu, algoritma ini memungkinkan munculnya penghalang-penghalang yang mungkin dapat menghalangi pemain lain ketika pion-pion tersebut melangkah dengan rapat. Namun, kekurangan dari *greedy* ini adalah dikarenakan semua pion bergerak bersamaan, posisi antara satu pion dan pion lainnya akan berdekatan. Dengan kondisi ini, ketika ada pion pemain lain yang mendekat dan akan melangkah, kemungkinan untuk salah satu pion dipukul oleh pion pemain lainnya lebih besar sehingga algoritma ini kurang cocok untuk dijalankan pada kondisi tersebut.

B. Greedy by hitter

Dengan menggunakan algoritma ini, tujuan utama pemain adalah menghalangi pemain lain untuk menang dengan mengejar dan melangkah pada kotak yang berisi pion pemain lain. Penggunaan algoritma ini akan mempersulit pemain lain untuk mencapai kemenangan dan membuat pemain lain mengulang-ulang lagi langkahnya. Karena algoritma ini bersifat menunggu kesempatan untuk memukul pion lainnya, pion-pionnya akan maju secara perlahan-lahan. Karena pion-pion tersebut maju secara perlahan kemenangan dapat diperoleh dalam waktu yang cukup lama. Namun algoritma ini dapat juga mencapai kemenangan dengan cepat karena algoritma ini juga bersifat mengejar pion pemain lainnya. Jika pion yang dikejar bergerak dengan cepat maka pion yang mengejarnya juga akan secara tidak langsung bergerak dengan cepat juga. Selain itu jika tidak ada pion pemain lain berada pada jalur, maka algoritma ini akan terkesan biasa saja.

C. Greedy by quick move

Algoritma greedy ini dapat memenangkan permainan paling cepat bila dibandingkan dengan algoritma lainnya. Karena algoritma ini akan berusaha untuk selalu maju dengan cepat menuju kotak tujuan selama tidak ada penghalang. Selain itu penggunaan algoritma ini akan membuat pion-pionnya lebih aman dari serangan lawan dikeranakan pion yang berada di luar daerah siap atau yang bisa diserang berjumlah 1-2 pion sehingga kemungkinan untuk diserangnya lebih kecil. Namun algoritma ini akan menghadapi kesulitan jika ada pemain lain yang menghalangi langkah algoritma ini. Selain itu jika satu-satunya pion yang keluar dipukul oleh pemain lain maka akan butuh usaha lebih besar untuk memulai lagi dikarenakan tidak ada cadangan pion di luar daerah siap. Untuk menyelesaikan persoalan dengan baik, pada algoritma ini dibutuhkan keberentungan yang lumayan besar karena semakin besar angka yang didapatkan maka akan semakin cepat pion menuju kotak tujuan.

D. Greedy by blocking

Algoritma ini akan berusaha untuk membuat penghalang yang akan menutupi langkah pemain lain. Penghalang ini akan tetap dibiarkan ada selama ada pion lain yang dapat digerakkan selain pion yang menjadi penghalang. Penghalang tersebut biasanya berada di daerah kotak awal permainan. Algoritma ini dapat meraih kemenangan dikarenakan menghalangi pemain lain. Menghalangi pemainan memungkinkan pion lain untu maju terus sedangkkan pion pemain lain tertahan di belakang penghalang, Namun kemenangannya akan diperoleh lebih lama karena pergerakan pion hanya dilakukan pion selain penghalang. Selain itu langkah menuju kotak tujuan akan berkutan jika ada langkah lain yang dapat menghasilkan penghalang. Algoritma ini baik digunakan jika permainan berada pada kondisi dimana semua pion pemain lainnya berada di belakang penghalang yang dibangun.

E. Analisis gabungan

Jika kita membandingkan keempat algoritma di atas, algoritma yang mungkin dapat menyelesaikan persoalan paling cepat adalah algoritma *greedy by quick move*. Namun untuk merealisasikannya pemain membutuhkan keberuntungan yang cukup besar. Algoritma yang paling aman adalah algoritma *greedy by blocking* karena algoritma ini akan berusaha menghalangi pion pemain lain agar pionnya sendiri dapat maju menuju kotak tujuan. Karena pion pemain terhalangi, maka pion yang bergerak sendiri tersebut akan lebih aman karena kemungkinan untuk dipukul pion pemain lain akan mengecil.

Dari keempat algoritma yang ada di atas dapat dilihat bahwa setiap algoritma *greedy* punya kelebihan masing-masing namun mempunyai kekurangan masing-masing juga. Kekurangan tersebut berasal dari kondisi-kondisi tertentu yang menyebabkan kekurangan tersebut menghalangi cara kerja algoritma *greedy* tersebut. Sehingga algoritma tersebut cocok digunakan hanya untuk kondisi tertentu saja. Namun, pada kenyataannya selama permainan berlangsung kondisi apapun dapat muncul dan mungkin tidak sesuai dengan prediksi kita, bahkan kondisi yang muncul mungkin adalah kondisi yang menyebabkan kekurangan suatu algoritma.

Cara untuk mengatasi ini adalah dengan mencoba analisa kemungkinan-kemungkinan kondisi yang ada. Dari hasil analisis tersebut coba buat suatu algoritma yang merupakan gabungan dari algoritma-algoritma yang ada dengan kondisi-kondisi terntentu. Dari hasil penggabungan tersebut diharapkan kita dapat mengatasi setiap kondisi dengan solusi terbaiknya dan dapat menyelesaikan persoalan ini dengan lebih sangkil dan mangkus.

VI. KESIMPULAN

Persoalan permainan ludo adalah persoalan yang dapat diselesaikan dengan mengimplementasikan algoritma *greedy*. Dalam implementasinya kita dapat membuat beberapa alternatif pilihan algoritma *greedy*, antara lain *greedy by no one left behind*, *greedy by hitter*, *greedy by quick move*, atau *greedy by blocking*. Setiap algoritma *greedy* tersebut mempunyai kelebihan dan kekurangan masing-masing tergantung kondisi yang ada pada saat tersebut. Untuk menghasilkan algoritma yang lebih sangkil dan mangkus, kita dapat menggabungkan keempat algoritma *greedy* tersebut dan membuat suatu algoritma greedy yang dapat menyelesaikan persoalan ludo ini dengan berbagai kondisi.

REFERENSI

[1] Rinaldi Munir, "Diktak Kuliah IF3051 Strategi Algoritma", Program Studi Teknik Informatika: Bandung, 2009, bab3.

[2] http://en.wikipedia.org/wiki/Ludo_(board_game), Ludo, diakses pada tanggal 5 Desember 2011.

[3] http://en.wikipedia.org/wiki/Greedy_algorithm, Greedy Algorithm, diakses pada tanggal 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

Lio Franklyn Kemit/13509053