Algoritma Greedy pada Permainan Nine Square

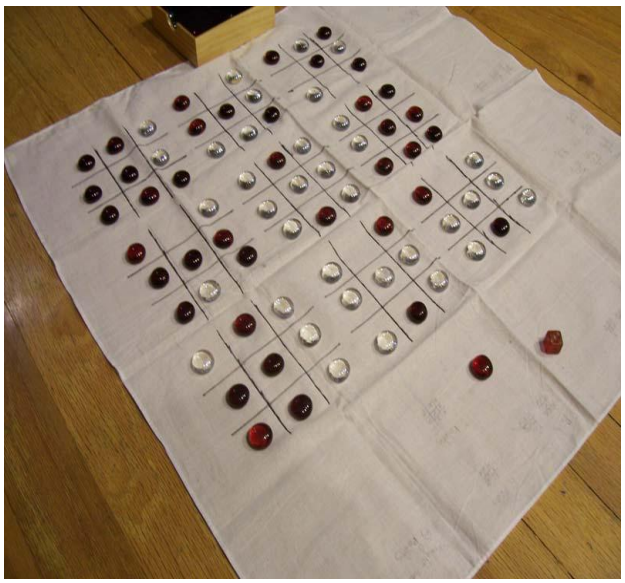
Hans Agastyra - 13509062
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
solvethistrick@gmail.com

***Abstrak*—Makalah ini dibuat untuk membahas dan memperlihatkan bagaimana algoritma greedy dapat digunakan pada permainan nine square. Walaupun algoritma greedy tidak selalu menjamin kemenangan pada penggunanya, namun cara berpikir yang digunakan orang banyak biasanya mirip dengan algoritma greedy sehingga melihat lebih lanjut bagaimana algoritma greedy ini dapat digunakan di beberapa jenis permainan menjadi menarik untuk dibahas.**

***Kata Kunci*—nine square,algoritma,greedy.**

I. PENDAHULUAN

Permainan Nine Square merupakan sebuah permainan yang dimodifikasi dari permainan Tic-Tac-Toe, permainan ini diciptakan oleh seorang *game designer* bernama Devin Monnens. Permainan ini sangat sederhana dan dapat dimainkan dengan media apapun mulai hanya dengan menggunakan kertas dan pensil ataupun menggunakan benda-benda lain seperti bidak agar membuat permainan lebih menarik namun dadu dengan 6 mata dadu merupakan syarat wajib untuk dapat memainkan nine square.



Gambar 1. Permainan Nine Square

Permainan ini dimainkan oleh dua orang secara bergiliran. Selayaknya *board game* lainnya, permainan nine square mempunyai beberapa aturan yang menjelaskan bagaimana cara bermain serta kondisi menang dari permainan ini. Seperti yang terlihat pada gambar, desain meja permainan yang digunakan merupakan 3x3 dari meja permainan Tic-Tac-Toe (dalam makalah ini, satu meja Tic-Tac-Toe akan disebut sebagai sebuah kotak).

Sasaran utama dari permainan ini adalah membuat sebanyak mungkin *line* yang dimana *line* sendiri didefinisikan sebagai susunan dari bidak pemain (pemain yang sama) yang membentuk sebuah garis lurus (horizontal, vertikal, diagonal) pada satu kotak.Tiap *line* mempunyai satu poin, Jika pemain membuat satu kotak penuh dengan bidaknya maka pemain tersebut dihitung mendapatkan 8 poin (3 poin horizontal, 3 poin vertikal, 2 poin diagonal). Pemain akan mendapatkan bonus 3 poin jika berhasil membentuk garis lurus pada 3 buah kotak.

Pemain yang mendapatkan giliran akan memulai permainan dengan cara memutar dadu dan mendapatkan kesempatan untuk meletakkan bidaknya sebanyak angka yang dimunculkan oleh dadu tersebut dan hal itu dilakukan terus menerus secara bergantian dengan pemain lain hingga permainan selesai. Permainan akan dinyatakan selesai setelah semua tempat terisi oleh bidak dari kedua pemain dan pemenang adalah pemain yang mempunyai poin lebih banyak.

II. ALGORITMA GREEDY

Algoritma greedy adalah sebuah algoritma yang biasa digunakan dalam permasalahan memilih solusi dengan perlu mempertimbangkan faktor-faktor tertentu yang dapat dipilih. Dengan memilih satu faktor sebagai pertimbangan utama, maka penentuan keputusan dapat dilakukan dengan harapan solusi terbaik dapat diperoleh. Namun pada kenyataannya, solusi yang didapat tidak akan selalu merupakan solusi terbaik. Salah satu keuntungan dari algoritma greedy adalah algoritma ini bekerja selayaknya bagaimana manusia memilih dalam kehidupan sehari-hari dan tidak rumit sehingga akan cepat dieksekusi.

Skema umum dari algoritma greedy adalah sebagai berikut:

1. Himpunan kandidat
   Sekumpulan nilai yang dapat dipilih sebagai solusi.

2.Himpunan solusi

Nilai yang akan sudah memenuhi segala persyaratan untuk menjadi solusi.

3.Fungsi seleksi

Fungsi yang digunakan untuk memilih kandidat mana yang akan dimasukkan ke dalam solusi.

4.Fungsi kelayakan

Fungsi yang mengecek apakah kandidat tidak melanggar aturan serta batasan dari permasalahan yang ada.

5.Fungsi obyektif

Fungsi untuk mendapatkan solusi yang diharapkan.

Ciri khas dari algoritma ini adalah algoritma greedy akan mengambil keputusan secara bertahap dan tidak memikirkan apa yang akan terjadi selanjutnya. Makalah ini akan menggunakan skema umum dari algoritma greedy ini untuk membentuk algoritma greedy yang sesuai untuk menentukan langkah pada permainan nine square.

III. ALGORITMA PENYELESAIAN

Terdapat banyak algoritma yang dapat digunakan dalam permainan ini. Namun yang akan dibahas lebih lanjut adalah algoritma greedy saja. Ada beberapa algoritma greedy yang akan menjadi alternatif pemilihan solusi. Pada algoritma yang disajikan pada makalah ini, arena permainan dilambangkan dengan *array* dua dimensi berukuran 9x9 yang disebut tipe Field. Kandidat merupakan sebuah *list* yang beranggotakan elemen yang memiliki atribut posX, posY dan disebut tipe Himpunan yang posX dan posY –nya telah diinisialisasi dengan posisi tiap petak yang ada di Field. Tidak semua algoritma dari fungsi ditampilkan dalam pseudocode makalah ini, dengan harapan kegunaan fungsi kecil tersebut sudah cukup jelas hanya dengan melihat deskripsi fungsinya.

Pseudocode Greedy by line

```
function LineGreedy(integer Turn,
Himpunan C, Field F) → Element
{fungsi yang mengembalikan Element
solusi dengan menggunakan algoritma
greedy by line}
DEKLARASI
Element S,x
ALGORITMA
S.init() {diinisialisasi}
while (not LAYAK(S,F) and not
C.empty()) do
  x ← SELEKSI(Turn,C,F)
  C ← C.remove(x)
  If LAYAK(x,F) then
    S ← x
  endif
endwhile
if LAYAK(S,F) then
  →S
else
  Output('Tidak ada solusi,
Permainan sudah selesai!')
endif
```

Pseudocode fungsi seleksi

```
function SELEKSI(integer Turn,
Himpunan C, Field F) → Element
{fungsi yang menyeleksi kandidat C
dengan mencari nilai terbesar dari
Value element pada kandidat dengan
memanfaatkan fungsi optimasi}

DEKLARASI
integer i,Value
Element Temp {asumsi sudah
diinisialisasi}

ALGORITMA
Value = -999
for i←0 to C.lastindex() do
  if(Optimation(Turn,C[i],F) >
Value)
    Value = Optimation(Turn,C[i],F)
    Temp = C[i]
  endif
endfor
→ Temp


function Optimation(integer Turn,
Element El, Field F) → integer
{Fungsi yang digunakan untuk
menentukan nilai optimasi dari
sebuah elemen El}

DEKLARASI
integer Value

ALGORITMA
Value ← 0
Value ← Horizontal(Turn,El,F) +
Vertical(Turn,El,F) +
Diagonal1(Turn,El,F) +
Diagonal2(Turn,El,F) +
isLine(Turn,El,F)
→ Value


function Horizontal(integer Turn,
Element El, Field F) → integer
{fungsi yang digunakan untuk
menentukan nilai pertimbangan
dengan melihat posisi horizontal
dari El pada Field}

DEKLARASI
```

```
function Horizon(integer Turn,
Element El, Field F)→ integer
{fungsi ini akan mengembalikan
nilai 1 jika posisi El masih
memungkinkan pemain untuk dapat
menciptakan line horizontal pada
satu kotak dan 0 jika tidak}

function Horizon3(integer Turn,
Element El, Field F)→ integer
{fungsi ini akan mengembalikan
nilai 1 jika posisi El masih
memungkinkan pemain untuk dapat
menciptakan line pada 3 kotak
horizontal dan 0 jika tidak}

ALGORITMA
→Horizon(Turn,El,P) +
Horizon3(Turn,El,P)


function Vertical(integer Turn,
Element El, Field F) → integer
{fungsi yang digunakan untuk
menentukan nilai pertimbangan
dengan melihat posisi vertikal dari
El pada Field}

DEKLARASI
function Vertic(integer Turn,
Element El, Field F)→ integer
{fungsi ini akan mengembalikan
nilai 1 jika posisi El masih
memungkinkan pemain untuk dapat
menciptakan line vertikal pada satu
kotak dan 0 jika tidak}

function Vertic3(integer Turn,
Element El, Field F)→ integer
{fungsi ini akan mengembalikan
nilai 1 jika posisi El masih
memungkinkan pemain untuk dapat
menciptakan line pada 3 kotak
vertikal dan 0 jika tidak}

ALGORITMA
→Vertic(Turn,El,F) +
Vertic3(Turn,El,F)


function Diagonal1(integer Turn,
Element El, Field F) → integer
{fungsi yang digunakan untuk
menentukan nilai pertimbangan
dengan melihat posisi diagonal
(dengan arah \) dari El pada Field}

DEKLARASI
```

```
function Diag(integer Turn, Element
El, Field F)→ integer
{fungsi ini akan mengembalikan
nilai 1 jika posisi El masih
memungkinkan pemain untuk dapat
menciptakan line diagonal '\' pada
satu kotak dan 0 jika tidak}

function Diag3(integer Turn,
Element El, Field F)→ integer
{fungsi ini akan mengembalikan
nilai 1 jika posisi El masih
memungkinkan pemain untuk dapat
menciptakan line pada 3 kotak
diagonal '\' dan 0 jika tidak}

ALGORITMA
→Diag(Turn,El,F) +
Diag3(Turn,El,F)


function Diagonal2(integer Turn,
Element El, Field F) → integer
{fungsi yang digunakan untuk
menentukan nilai pertimbangan
dengan melihat posisi diagonal
(dengan arah /) dari El pada Field}

DEKLARASI
function Diag(integer Turn, Element
El, Field F)→ integer
{fungsi ini akan mengembalikan
nilai 1 jika posisi El masih
memungkinkan pemain untuk dapat
menciptakan line diagonal '/' pada
satu kotak dan 0 jika tidak}

function Diag3(integer Turn,
Element El, Field F)→ integer
{fungsi ini akan mengembalikan
nilai 1 jika posisi El masih
memungkinkan pemain untuk dapat
menciptakan line pada 3 kotak
diagonal '/' dan 0 jika tidak}

ALGORITMA
→ Diag(Turn,El,F) +
Diag3(Turn,El,F)


function isLine(integer Turn,
Element El, Field F) → integer
{fungsi yang digunakan untuk
menentukan nilai pertimbangan
dengan melihat apakah posisi dari
El akan menghasilkan line pada F}

DEKLARASI
```

```
function checkLine(integer Turn,
Element El, Field F) → integer
{fungsi yang mengecek apakah El
menghasilkan line pada F,
mengembalikan true jika ya dan
false jika tidak}

ALGORITMA
if (checkLine(Turn,El,F) = true)
  → 5
else
  → 0
endif
```

Pseudocode fungsi kelayakan

```
function LAYAK(Element El, Field F)
→ boolean
{fungsi yang mengembalikan true
jika posisi El pada F belum
ditempati}

DEKLARASI

ALGORITMA
if(F[El.posX][El.posY] != 0) then
  → true
else
  → false
endif
```

IV. Analisis Algoritma

Algoritma yang ditampilkan pada bab 3 makalah ini belum melalui proses pengujian dengan program jadi memungkinkan adanya *bug*. Selain itu, algoritma *greedy* ini diciptakan hanya untuk per langkah yang diterima oleh pemain (dengan kata lain algoritma ini tidak memikirkan berapa jumlah langkah yang diperoleh dari pemutaran dadu).

Pada skema utama dari pseudocode algoritma *greedy* tersebut fungsi obyektif tidak ditampilkan karena solusi bukanlah susunan langkah melainkan hanya satu langkah saja (bukan dalam bentuk himpunan) oleh karena itu fungsi kelayakan sudah cukup menjamin bahwa elemen kandidat tersebut sudah bisa disebut solusi.

Selain itu, pada pseudocode fungsi seleksi, pemberian nilai pada fungsi isLine (di dalam fungsi Optimation) sangatlah besar karena saat itu terjadi berarti dengan pemain mengambil langkah tersebut, pemain akan mendapatkan poin secara langsung.

Pendekatan utama dari algoritma yang ditampilkan adalah bagaimana pemain dapat memilih lokasi petak yang paling strategis, yaitu memiliki kesempatan untuk menciptakan *line* baik horizontal, vertikal, maupun diagonal dengan tidak melupakan petak yang akan menjadikan *line* pada sebuah kotak. Dengan pendekatan ini, asumsi dadu akan memberikan nilai rata-rata kemunculan angka 3 maka lebih baik menciptakan sebuah *line* terlebih dahulu dibanding melakukan blok pada lawan.

V. Kesimpulan

Algoritma *greedy* dapat digunakan pada berbagai macam masalah dan tidak terkecuali pada permainan Nine Square. Algoritma yang diberikan pada makalah ini baru memikirkan aspek kesempatan menciptakan *line*. Masih banyak faktor yang belum diperhatikan seperti bagaimana jika pemain lain memiliki kesempatan menciptakan *line*, memotong *line* lawan, dan jumlah kesempatan yang diperoleh melalui dadu yang diputar. Namun, algoritma ini sudah cukup memberikan pemotongan waktu yang cukup besar dibanding algoritma *Brute Force* dengan hanya memikirkan *line* yang mungkin didapatkan.

Referensi

[1] Munir, Rinaldi. "Diktat Kuliah IF3051 Strategi Algoritma", Program Studi Teknik Informatika ITB, 2009.

[2] http://www.deserthat.com/html/game_design/paper/nine_squares.html diakses pada tanggal 8 Desember 2011 pukul 12.00.

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

Hans Agastyra - 13509062