Penerapan Algoritma Greedy dan Algoritma BFS dalam Pembuatan Artificial Intelligence pada Permainan Halma

Kevin Leonardo Limitius - 13516049
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
kevin_limit@yahoo.com

Abstract*—Sekarang, sudah banyak game yang dibuat dalam berbagai platform seperti PC, console, maupun mobile. Namun sebelum game dengan berbagai platform ini berkembang, terdapat permainan-permainan yang dimainkan di atas papan yang biasa disebut sebagai board game. Game jenis ini biasa dimainkan dengan 2-4 orang. Salah satu dari game yang termasuk ke dalam jenis ini adalah Halma. Halma adalah permainan papan yang dapat dimainkan oleh 2 ataupun 4 orang dengan tujuan untuk memindahkan bidak yang dimiliki ke seberang ujung papan permainan. Pada makalah ini akan dibahas mengenai penerapan algoritma Greedy dan juga algoritma Breadth First Search (BFS) untuk membuat AI dalam permainan halma.

Keywords—board game, Halma, algoritma Greedy, algoritma BFS, AI

I. PENDAHULUAN

Sebelum video game dikembangkan di berbagai platform seperti *PC*, *console*, ataupun *mobile*, terdapat *board game* atau permainan papan yang dipakai untuk mengisi waktu luang dan bercengkerama dengan orang lain. Permainan papan sendiri memiliki banyak jenis, mulai dari game pemosisian seperti tic tac toe, game strategi abstrak seperti catur, game balapan seperti backgammon, game simulasi ekonomi seperti monopoly dan masih banyak lagi.

Permainan papan yang akan dibahas pada makalah ini adalah halma. Halma adalah permainan papan yang dibuat pada tahun 1880-an dengan tujuan permainan untuk memindahkan bidak yang dimiliki ke ujung seberang papan permainan.

Gambar 1 Permainan Halma (diambil dari : [1])

Walaupun halma termasuk ke dalam permainan papan, sekarang halma sudah dijadikan permainan di dalam *PC* dan juga versi *mobile*-nya. Karena dalam permainan ini dibutuhkan pemain minimal 2 orang, maka dibutuhkan lawan main untuk memainkan permainan ini. Oleh karena itu, dibutuhkan *Artificial Intelligence* (AI) untuk menjadi lawan main jika tidak terdapat orang lain untuk dijadikan lawan main.

AI sendiri adalah program untuk membuat mesin memiliki kepintaran mendekati kemampuan manusia dalam mejalankan tugas yang biasa manusia kerjakan dalam kehidupan sehari-hari. Dalam kali ini, AI diharapkan dapat menggantikan peran manusia sebagai lawan main dalam bermain halma. Agar AI dapat menggantikan peran manusia dibutuhkan algoritma untuk mengatur pergerakan dari AI untuk melawan pemain dan semakin baik algoritma yang digunakan, semakin baik pula AI dapat melawan manusia dalam permainan halma.

Algoritma yang akan dipakai dalam makalah ini adalah algoritma Greedy dan algoritma BFS. Pemakaian algoritma greedy bertujuan agar AI dapat mengambil langka yang optimal berdasarkan kondisi yang ada dalam permainan karena pergerakan dari lawan hampir tidak ddapat diperkirakan dan algoritma BFS bertujuan agar AI dapat mencari jalur yang paling sesuai dan cepat untuk mencapai tujuan akhir.

Dalam makalah ini, pembaca diharapkan dapat mengembangkan lagi lebih lanjut algortima yang dapat dipakai agar AI dapat menjadi lawan main yang lebih kompetitif lagi. Yang terakhir, penulis juga berharap pembaca dapat menemukan penerapan lain dari algoritma Greedy dan juga algoritma BFS.

II. DASAR TEORI

A. *Algoritma Greedy*

Algoritma greedy adalah metode yang sederhana dan *straightforward* untuk menyelesaikan persoalan yang berkaitan dengan optimasi. Algoritma yang paling populer untuk memecahkan persoalan optimasi ini memiliki prinsip *"take what you can get now!"* yang menunjukkan bahwa algoritma ini pada dasarnya mengambil apa yang bisa diambil sekarang. Persoalan optimasi yang biasa diselesaikan adalah maksimasi dan minimasi, di mana maksimasi adalah mencari solusi yang bernilai maksimal dari kandidat solusi yang ada dan minimasi

berarti mencari solusi yang bernilai minimal dari kandidat solusi yang ada.

Algoritma ini menyelesaikan masalah yang ada langkah demi langkah dengan mengambil pilihan terbaik yang ada pada tiap langkahnya dan pilihan tersebut tidak dapat diubah kembali sehingga algoritma ini tidak mempedulikan efek dari pilihan tersebut ke depannya dengan harapan bahwa langkah yang sudah diambil dapat menghasilkan hasil dengan optimum global dengan mengambil nilai optimum lokal pada setiap langkahnya.

Dalam mencapai solusi menggunakan algoritma greedy terdapat elemen-elemen yang menjadi dasarnya, yaitu :

1. Himpunan kandidat, C
   Himpunan kandidat adalah himpunan yang menyimpan elemen-elemen yang dapat menjadi solusi dari persoalan yang ada.
2. Himpunan solusi, S
   Himpunan solusi adalah himpunan yang menyimpan elemen-elemen yang telah dipilih dari himpunan kandidat yang dinilai paling cocok untuk dijadikan solusi dari persoalan yang ada.
3. Fungsi seleksi
   Fungsi seleksi adalah fungsi yang digunakan untuk memilih kandidat yang ada dan memiliki peluang paling tinggi untuk menghasilkan solusi yang diinginkan. Hasil yang sudah dipilih dari fungsi yang dinyatakan dengan predikat SELEKSI ini tidak lagi diperhitungkan pada pengambilan keputusan pada langkah berikutnya. Hasil yang dipilih memperhatikan persoalan optimasi yang ada. Jika persoalan tersebut berkaitan dengan maksimasi maka nilai yang diambil adalah nilai yang terbesar, jika persoalan berkaitan dengan minimasi maka nilai yang diambil adalah yang terkecil.
4. Fungsi kelayakan
   Fungsi kelayakan adalah fungsi yang digunakan untuk memeriksa apakah kandidat yang dipilih untuk dijadikan solusi tidak melanggar *constraint* yang ada. Jika kandidat yang dipilih melanggar fungsi yang dinyatakan dengan predikan LAYAK ini, maka kandidat tersebut akan dibuang dan tidak dipertimbangkan lagi dalam pengambilan keputusan selanjutnya. Jika kandidat yang dipilih tidak melanggar fungsi kelayakan maka kandidat tersebut akan dimasukkan ke dalam himpunan solusi.
5. Fungsi objektif
   Fungsi objektif adalah fungsi yang dipakai sebagai faktor penentu dari suatu solusi (maksimum atau minimum). Contoh dari fungsi objektif adalah keuntungan dari data penjualan barang, panjang lintasan, dll.

Algoritma greedy memiliki skema umum dan predikat yang dapat digambarkan dalam contoh sebagai berikut :

```
function SELEKSI ( C : himpunan_kandidat) → kandidat
{me – return sebuah kandidat yang dipilih dari C berdasarkan kriteria tertentu}

function SOLUSI ( S : himpunan_kandidat) → boolean
{true jika S adalah solusi dari persoalan; sebaliknya false jika S belum menjadi solusi}

function LAYAK ( S : himpunan_kandidat) → boolean
{bernilai true jika S merupakan solusi yang tidak melanggar kendala; sebaliknya ebrnilai
false jika melanggat kendala}
```

Gambar 2 Contoh *pseudocode* predikat algoritma greedy (diambil dari : [2])

```
function greedy(input C: himpunan_kandidat)
→ himpunan_kandidat
{Mengembalikan solusi dari persoalan
optimasi dengan algoritma greedy
Masukan: himpunan kandidat C
Keluaran: himpunan solusi yang bertipe
himpunan_kandidat}

Deklarasi
  x : kandidat
  S : himpunan_kandidat

Algoritma:
S ← {} {inisialisasi S dengan kosong}
while (not SOLUSI(S)) and (C ≠ {} ) do
  x ← SELEKSI(C) {pilih sebuah kandidat
dari C}
  C ← C - {x} {elemen himpunan kandidat
berkurang satu}
  if LAYAK(S ∪ {x}) then
    S ← S ∪ {x}
  endif
endwhile
{SOLUSI(S) or C = {}}

if SOLUSI(S) then
  return S
else
  write('tidak ada solusi')
endif
```

Gambar 3 Contoh *pseudocode* algoritma greedy (diambil dari : [3])

Solusi optimum global yang dihasilkan oleh algoritma greedy tidak menjamin solusi optimum yang sebenarnya, namun hasil yang dihasilkan dapat merupakan solusi *sub-optimum* ataupun *pseudo-optimum*. Hal ini diakibatkan oleh cara kerja algoritma greedy yang tidak beroperasi secara menyeluruh dari semua alternatif solusi yang ada. Selain itu pemilihan fungsi seleksi yang berbeda juga dapat mempengaruhi hasil solusi dari algoritma greedy. Oleh karena itu dibutuhka pemilihan fungsi seleksi yang tepat agar solusi yang dihasilkan oleh algoritma greedy benar-benar optimum global. Oleh karena dua alasan di atas, algoritma greedy sering kali dipakai untuk mencari solusi optimum di mana solusi mutlak tidak benar-benar diperlukan sehingga hasil dari algoritma greedy dipakai sebagai aproksimasi dari solusi optimum yang sebenarnya.

B. Algoritma Breadth First Search (BFS)

Algoritma BFS merupakan salah satu dari dua algoritma pencarian solusi dengan melakukan traversal pohon berakar

yang ada. Pencarian solusi dengan jenis ini dilakukan dengan cara mengunjungi simpul-simpul yang ada pada pohon berakar secara sistematik. Solusi akan didapatkan jika semua simpul sudah diperiksa, jika solusi belum didapatkan maka simpul yang lain akan diperiksa hingga tercapai solusi yang diinginkan. Berbeda dengan pohon statik yang sudah tersedia sebelum pencarian secara traversal dilakukan, pada pohon dinamis yang dipakai oleh algoritma BFS dan juga algoritma DFS, pohon justru dibuat bersamaan dengan pencarian solusi. Pembentukan pohon dibuat berdasarkan solusinya, jika simpul yang terbentuk tidak mengarah kepada solusi yang diinginkan maka akan dibentuk simpul yang baru sampai solusi ditemukan. Pohon dinamis yang dihasilkan pun berbeda, pohon yang dihasilkan oleh algoritma BFS dinamai pohon BFS dan pohon yang dihasilkan oleh algoritma DFS dinamai pohon DFS.

Kedua pohon dinamis (disebut juga pohon ruang status) yang dihasilkan memiliki beberapa elemen yang dapat dijabarkan sebagai berikut :

1. Simpul merupakan *problem state* (simpul di dalam pohon yang memenuhi *constraint* dari persoalan yang ada). Simpul sendiri memiliki dua jenis, yaitu :
    a. Akar adalah *initial state* (Kondisi awal dari persoalan yang ada)
    b. Daun adalah *solution/goal state* (Hasil atau solusi yang diinginkan dari persoalan yang ada)
2. Cabang adalah operator yang mentransformasikan persoalan yang ada dari satu status ke status yang lain
3. Ruang status adalah himpunan dari semua simpul yang ada di dalam pohon
4. Ruang solusi adalah himpunan dari status solusi

Algoritma BFS dimulai dengan membangkitkan simpul akar terlebih dahulu lalu dilanjutkan ke seluruh simpul anak-anaknya kemudian dilanjutkan kepada semua suksesor anaknya hingga diperoleh solusi yang diinginkan. Secara umum, algoritma dari BFS adalah sebagai berikut :

1. Mengunjungi simpul v
2. Mengunjungi semua simpul yang bersebelahan dengan simpul v
3. Mengunjungi semua simpul yang belum dikunjungi dan simpul-simpul yang bertetangga dengan simpul-simpul yang sudah dikunjungi hingga solusi tercapai.

Berikut adalah contoh dari *pseudocode* algoritma BFS beserta dengan hal-hal yang dibutuhkan :

1. Matriks ketetangaan A = [a_{ij}] yang berukuran n x n, nilai dari a_{ij} = 1 jika simpul i dan j bertetanggaan dan nilai dari a_{ij} = 0 jika simpul i dan j tidak bertetangga.
2. Antrian Q untuk menyimpan simpul-simpul yang sudah dikunjungi
3. Tabel V yang memiliki tipe data *boolean* untuk menampung data dari simpul yang sudah dikunjungi
   Nilai dari tabel V[i] = true jika simpul i sudah dikunjungi
   Nilai dari tabel V[i] = false jika simpul i belum dikunjungi

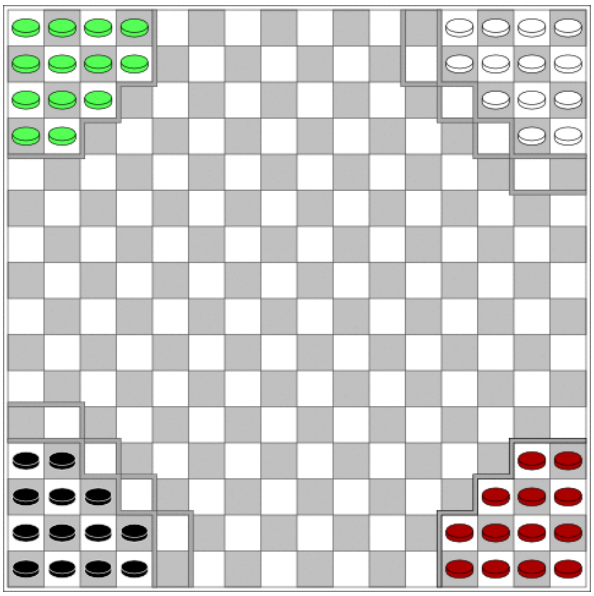
```
procedure BFS(input v:integer)
{ Traversal graf dengan algoritma pencarian
BFS.
Masukan: v adalah simpul awal kunjungan
Keluaran: semua simpul yang dikunjungi
dicetak ke layar }

Deklarasi
  w : integer
  q : antrian

  procedure BuatAntrian(input/output q :
antrian)
{membuat antrian kosong, kepala(q) diisi 0}

  procedure MasukAntrian(input/output q :
antrian, input v : integer)
{memasukkan v ke dalam antrian q pada
posisi belakang}

  procedure HapusAntrian(input/output q :
antrian, output v : integer)
{ menghapus v dari kepala antrian q }

  function AntrianKosong(input q : antrian)
→ boolean
{ true jika antrian q kosong, false jika
sebaliknya }

Algoritma:
  BuatAntrian(q) { buat antrian kosong }
  write(v) { cetak simpul awal yang
dikunjungi }
  dikunjungi[v]←true { simpul v telah
dikunjungi, tandai dengan true}
  MasukAntrian(q,v) { masukkan simpul awal
kunjungan ke dalam antrian }

{ kunjungi semua simpul graf selama antrian
belum kosong }
  while not AntrianKosong(q) do
    HapusAntrian(q,v) { simpul v telah
dikunjungi, hapus dari antrian }
    for w←1 to n do
      if A[v,w] = 1 then { v dan w
bertetangga }
        if not dikunjungi[w] then
          write(w) {cetak simpul yang
dikunjungi}
          MasukAntrian(q,w)
          dikunjungi[w]←true
        endif
      endif
    endfor
  endwhile
{ AntrianKosong(q) }
```

Gambar 4 Contoh *pseudocode* algoritma BFS (diambil dari : [4])

C. Pengenalan Terhadap Permainan Halma

Permainan halma yang berasal dari kata Yunani yang memiliki arti melompat ini adalah permainan yang menggunakan papan dengan kotak sebanyak 16x16 dan dimainkan oleh 2-4 orang. Di setiap pojok papan halma, terdapat garis tebal yang melingkupi 13 kotak yang menandakan tempat untuk menaruh bidak tiap pemain untuk permainan dengan 4 orang. Lalu terdapat 2 garis tambahan di 2 ujung papan untuk menandakan posisi awal permainan

untuk 2 orang yang melingkupi 19 kotak. Bidak dari halma sendiri memiliki berbagai bentuk namun bidak yang ada pasti memiliki 4 warna yang berbeda untuk membedakan bidak antar pemain.

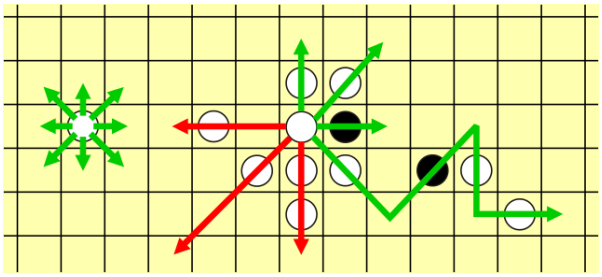


Gambar 5 Contoh pengaturan awal permainan halma (diambil dari : [5])

Tujuan dari permainan ini sendiri adalah untuk memindahkan seluruh bidak milik pemain ke ujung seberang papan halma dan pemain yang berhasil memindahkan seluruh bidaknya ke seberang adalah pemenangnya.

Aturan permainan dari halma adalah sebagai berikut :

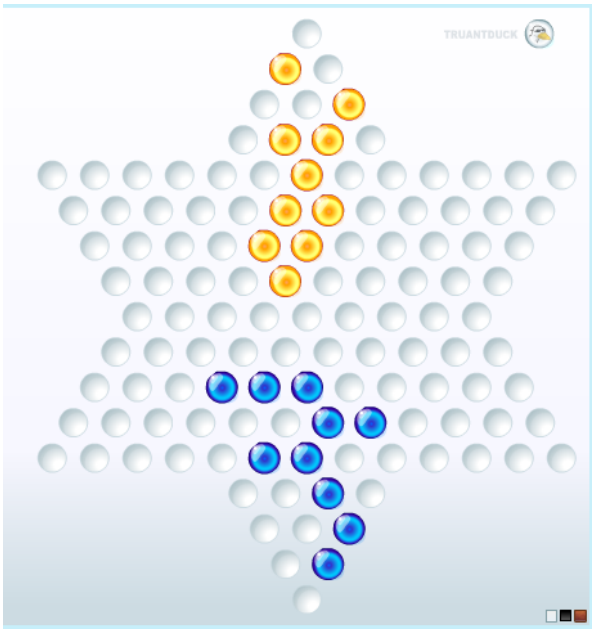
1. Permainan dimulai dengan posisi 13 bidak dari masing-masing pada papan seperti pada gambar 5 untuk permainan dengan 4 pemain, sedangkan untuk 2 pemain, penyusunan bidak sedikit berubah dengan jumlah bidak menjadi 19 dan disusun mengikuti garis tebal yang kedua.
2. Untuk menentukan pemain yang pertama kali memulai permainan dapat ditentukan secara bebas. Lalu permainan akan dilanjutkan secara bergantian sesuai dengan arah jarum jam.
3. Setiap putaran pemain hanya dapat menggerakan 1 bidak saja.
4. Bidak yang sendiri dapat bergerak secara horizontal, vertikal, maupun diagonal sebanyak 1 kotak.
5. Bidak dapat melompati bidak yang sendiri dari warna apapun secara horizontal, vertikal, maupun diagonal.
6. Bidak tidak dapat melompati 2 bidak yang berdempetan satu dengan yang lain.
7. Bidak dapat melompat berkali-kali dalam satu putaran.
8. Bidak musuh yang dilompati tidak akan dikeluarkan dari permainan atau ditangkap. (Semua bidak tidak ada yang keluar dari papan halma).
9. Permainan dinyatakan selesai saat salah satu pemain sudah berhasil memindahkan bidaknya ke ujung papan halma yang lain.



Gambar 6 Ilustrasi pergerakan bidak dalam halma. Gerakan berwarna hijau adalah gerakan yang diperbolehkan sedangkan gerakan berwarna merah adalah gerakan yang tidak diperbolehkan(diambil dari : [6])

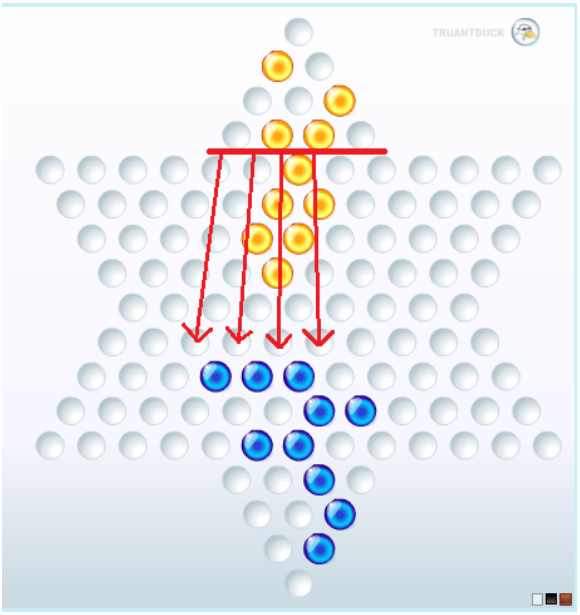
III. IMPLEMENTASI ALGORITMA GREEDY DAN BFS

Penerapan algoritma greedy dan BFS pada AI yang ingin dikembangkan dalam permainan Halma bertujuan untuk membuat lawan main yang optimum. Konteks optimum dalam hal ini adalah AI dapat menyelesaikan permainan dengan menggunakan jalur yang paling maksimal dan pilihan yang paling optimum pada setiap langkahnya. Untuk mempermudah penjelasan mengenai pengimplementasian algoritma dalam AI permainan halma dibuatlah studi kasus mengenai suatu kondisi dalam permainan halma yang akan ditunjukkan pada gambar di bawah ini.

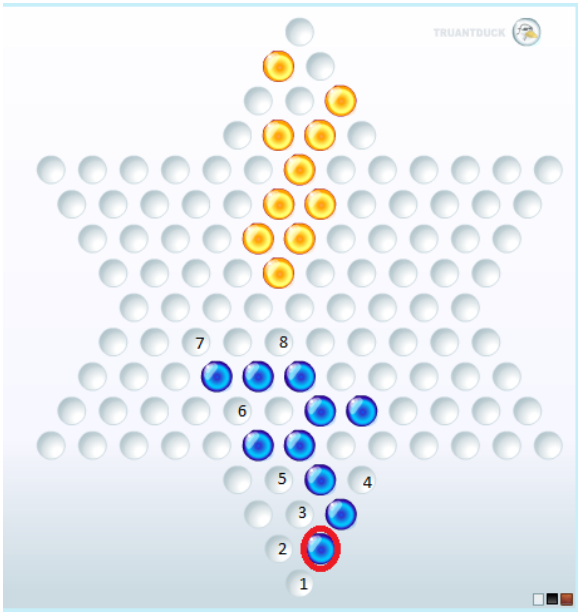


Gambar 7 Contoh studi kasus permainan halma (diambil dari : [7])

Pertama-tama akan digunakan algoritma greedy untuk mencari solusi akhir dari pergerakan bidak. Untuk menentukan solusinya, akan dicari titik yang paling dekat dari bidak dengan ujung seberang dari papan halma yang dapat dilihat pada gambar berikut :



Gambar 8 Kemungkinan solusi dari hasil peninjauan terhadap tujuan akhir dari bidak (diambil dari : [8])



Gambar 9 Titik-titik yang dapat diambil oleh bidak yang dilingkari (diambil dari : [9])

Algoritma greedy dari gambar di atas dapat diimplementasikan sebagai berikut :

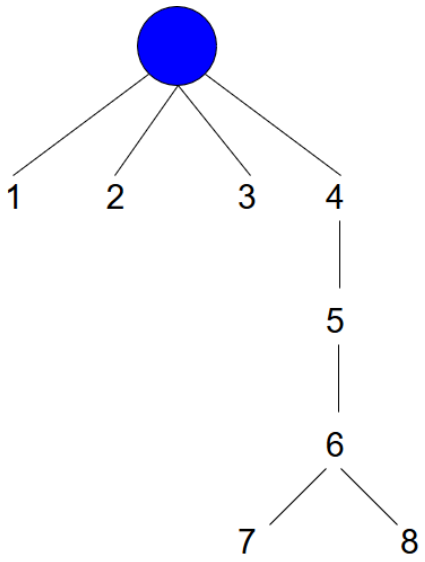
1. Himpunan kandidat
   Seluruh titik yang mungkin dicapai oleh bidak-bidak yang dimiliki pemain
2. Himpunan solusi
   Titik yang dapat dicapai oleh bidak dengan satu putaran dengan jarak paling dekat dengan ujung papan permainan halma
3. Fungsi seleksi
   Titik dengan jarak terdekat dari tujuan akhir dari bidak pada papan halma
4. Fungsi kelayakan
   Titik dapat dicapai oleh salah satu bidak dalam satu putaran

Kemungkinan terburuk yang dapat terjadi adalah terdapat banyak titik yang memiliki jarak yang sama dengan tujuan akhir dari bidak. Apabila hal ini terjadi, akan dipilih titik yang berada pada posisi paling tengah dari kandidat yang ada.

Setelah titik dengan jarak terpendek dari tujuan akhir bidak ditemukan, dipakailah algoritma BFS untuk menentukan jalur yang akan digunakan untuk mencapai titik tersebut. Bidak yang akan dilakukan pengecekan dengan algoritma BFS dimulai dari posisi bidak yang paling jauh dari tujuan akhir (paling belakang) dan akan diiterasi sampai bidak yang paling dekat dengan tujuan akhir apabila tidak ditemukan bidak yang dapat mencapai titik yang telah ditemukan oleh algoritma greedy.

Algoritma BFS akan menghasilkan peta jalur yang mungkin dari bidak yang dilakukan pengecekan. Kedalaman dari pohon akan menjadi indikator dari langkah yang diambil dari bidak tersebut dan langkah yang terpendek akan menjadi prioritas pengambilan jalur jika terdapat jalur yang menuju kepada titik yang telah ditentukan. Berikut adalah contoh pemrosesan BFS untuk salah satu bidak.

Dari contoh di atas akan terbentuk pohon BFS sebagai berikut :



Gambar 10 Pohon status yang dihasilkan

Titik yang ditentukan adalah titik dengan nomor 8 dan titik ini berada pada kedalaman ke 4. Penelusuran yang dilakukan oleh BFS untuk menemukan jalur dari bidak (Start) hingga titik 8 akan melalui titik 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8. Sehingga ditemukan jalur terpendek dari titik Start ke titik 8, yaitu S → 4 → 5 → 6 → 7 → 8.

Dalam makalah ini, pencarian jalur dilakukan dengan BFS karena mempertimbangkan hasil dari algoritma BFS yang *complete* karena jumlah cabang dari suatu simpul dapat dipastikan terbatas dan juga dapat menghasilkan jalur yang paling optimum karena semua kemungkinan jalur telah ditemukan.

IV. KESIMPULAN

Algoritma greedy dan algoritma BFS dapat diterapkan untuk membuat AI sebagai lawan main dalam permainan halma. Persoalan greedy yang dipakai adalah minimasi di mana faktor jarak dari titik yang paling dekat dengan tujuan akhir dari bidak (ujung seberang papan) sebagai acuan dari pemilihan solusi. Sedangkan algoritma BFS digunakan untuk menentukan jalur yang akan diambil oleh suatu bidak untuk mencapai titik tersebut.

V.UCAPAN TERIMA KASIH

Penulis mengucapkan syukur kepada Tuhan Yang Maha Esa karena atas anugerah-Nya, pembuatan makalah ini berjalan dengan lancar dan dapat terselesaikan. Ucapan terima kasih juga penulis sampaikan kepada I Kadek Yuda yang telah memberikan bantuan moril dan dukungan untuk mengambil tema makalah ini. Penulis juga menyampaikan terima kasih kepada Daniel Yudianto yang telah membantu saya dengan memberikan masukan mengenai algoritma yang saya pilih dalam makalah ini. Dan yang terakhir kepada Maha William Chandra yang memberikan hiburan dan juga motivasi selama pengerjaan makalah ini hingga makalah ini dapat diselesaikan dengan baik.

REFERENCES

The Editors of Encyclopaedia Britannica. “Halma”. Diakses dari https://www.britannica.com/topic/Halma-game pada 11 Mei 2018 pukul 20.00

“Halma” Diakses dari http://www.cyningstan.com/game/70/halma pada 11 Mei 2018 pukul 20.15

Professor John McCarthy. “What is AI? / Basic Questions”. Diakses dari http://jmc.stanford.edu/artificial-intelligence/what-is-ai/index.html pada 11 Mei 2018 pukul 21.00

Munir, Rinaldi. 2018. Diktat Kuliah IF2211 Strategi Algoritma. Institut Teknologi Bandung : Bandung.

[1] http://www.cyningstan.com/data-image/927/halma-in-a-modern-spears-edition

[2], [3], dan [4] Munir, Rinaldi. 2018. Diktat Kuliah IF2211 Strategi Algoritma. Institut Teknologi Bandung : Bandung.

[5] http://www.cyningstan.com/data-image/923

[6] http://www.wikiwand.com/en/Halma

[7], [8], dan [9] http://www.online-games-zone.com/pages/classic/halma.php

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, 13 Mei 2018

ttd

Kevin Leonardo Limitius - 13516049