Penggunaan Algoritma DFS dan BFS Dalam Pencarian Jarak Tedekat di *Game* Civilization V

Rifkiansyah Meidian Cahyaatmaja - 13511084
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13511084@stei.itb.ac.id

Abstrak - **Algoritma BFS dan DFS adalah dua algoritma yang sangat sering digunakan dalam pencarian jalan menuju sebuah tujuan tertentu dimana diketahui terdapat jalan menuju tujuan. Jalan tersebut dapat memiliki cost ataupun tidak dan algoritma BFS dan DFS dapat digunakan untuk mencari jalan baik dengan menghitung cost maupun tidak. Civilization V adalah sebuah permainan strategi berbasis-giliran dimana setiap *tile* didalam permainan direpresentasikan oleh sebuah bidang heksagonal yang terhubung ke enam tempat lainnya yang dapat dilalui oleh unit yang memiliki poin pergerakan. Dalam makalah ini akan dibahas mengenai penggunaan algoritma BFS dan DFS untuk mencari jalan menuju tempat tujuan didalam permainan Civilization.**

Kata Kunci - cost, jalan, poin pergerakan,,unit, tile.

I. Pendahuluan

Algoritma Breadth-First Search adalah sebuah algoritma untuk mencari solusi dari sebuah graf yang terhubung satu sama lain di setiap nodenya dengan metode melewati semua daerah terdekat dengan daerah asal untuk mencari jalan menuju tujuan. Breadth-First Search memiliki sifat berupa: komplit jika nilai b terbatas; optimal jika langkah = biaya; memiliki kompleksitas waktu $O(b^d)$; dan memiliki kompleksitas ruang $O(b^d)$ dengan b adalah jumlah graf terdekat dari *node* sebelumnya dan d adalah kedalaman yang harus dicari.

Algoritma Depth-First Search adalah sebuah algoritma untuk mencari solusi dari sebuah graf yang terhubung satu sama lain di setiap nodenya dengan metode melewati daerah yang lebih jauh lebih dahulu untuk mencari jalan menuju tujuan. DFS memiliki sifat berupa: komplit jika nilai b terbatas dan memiliki penanganan '*redundant paths'*, dan '*repeated states'*; tidak optimal; memiliki kompleksitas waktu $O(b^m)$; dan memiliki kompleksitas ruang O(bm) dengan b merupakan jumlah *node* di sekitar *node* asal dan m merupakan kedalaman tujuan dalam graf.

Civilization V adalah sebuah permainan strategi berbasis-giliran yang dapat dimainkan dalam *platform* Windows. Permainan ini memiliki banyak sekali elemen yang berbentuk jaringan ataupun pohon didalamnya. Didalam jaringan ini sendiri ada macam-macam sifat yang harus diperhatikan, diantaranya harga pergerakan *unit*, *combat modifier*, sumber daya yang dimiliki dan sumber daya di sekitar sebuah bidang, serta kemampuan *unit* untuk bergerak melewati tempat tempat itu. Sebenarnya masih banyak perhitungan yang digunakan untuk menjalankan sebuah *unit* kedalam suatu bidang, tapi dalam makalah ini yang akan dibahas adalah pergerakan *unit* dari sebuah bidang ke bidang lainnya.

II. Civilization V

Civilization V adalah permainan strategi dimana setiap pemain memiliki tujuan untuk menguasai dunia menggunakan salah satu dari beberapa tujuan yang disediakan oleh *game*. Setiap pemain dapat menciptakan dan menggerakkan *unit* miliknya sendiri. Pergerakan ini adalah salah satu inti dari permainan, dimana tanpa pergerakan pemain tidak dapat melakukan pencarian tempat-

tempat baru yang memiliki potensi untuk mendirikan sebuah kota ataupun bertemu musuh atau sekutu potensial. Pergerakan dalam permainan Civilization V memiliki beberapa hal yang harus diperhitungkan, diantaranya adalah *cost* yang dibutuhkan saat melewati jenis tanah tertentu, rintangan yang tidak dapat dilewati baik berupa rintangan alami maupun batas kota milik pemain lain, dan musuh yang berdiam di sebuah bidang. Setiap unit memiliki *movement point* atau poin pergerakannya sendiri-sendiri dan beberapa unit tidak dapat melalui *terrain* tertentu, sehingga faktor-faktor tersebut juga harus diperhitungkan. Unit hanya menggunakan poin pergerakan berdasarkan tempat yang ia tuju, dengan beberapa pengecualian.

A. Pergerakan Biasa

Sebuah unit dapat bergerak secara normal melewati *terrain* yang berbentuk *Plains, Tundra, Snow, Desert,* ataupun *Grassland*. Terrain-terrain tersebut tidak memiliki bonus maupun penalti apapun dalam pergerakannya.

Gambar 1. Pergerakan menuju sebuah gurun.

B. Pergerakan 2 Movement Point.

Disebut pergerakan -1 karena *terrain* yang dimiliki memiliki sifat menggunakan satu tambahan poin pergerakan yang dimiliki sebuah unit. Misalnya jika sebuah *unit* memiliki 2 poin pergerakan ia dapat berjalan melewati dua bidang gurun, namun jika ia berjalan melewati sebuah bukit, maka ia hanya dapat berjalan satu kali. *Terrain* yang memiliki sifat seperti ini diantaranya adalah *Hill, Forest, Jungle,* dan *Marsh*. Dalam Gambar 1, *Hill* adalah bidang yang berada di sebelah kiri tempat tujuan dan juga tempat asal pergerakan.

C. Pergerakan Melewati Air

Ada dua jenis pergerakan melewati air untuk *unit* yang berbasis di darat. Yang pertama adalah pergerakan melewati sungai, dan yang kedua adalah pergerakan menuju laut.

Gambar 2. Pergerakan Melewati Sungai

Dalam gambar 2 terlihat jika unit tersebut melewati sungai yang ada di sebelah kirinya, maka *unit* itu hanya dapat bergerak satu petak saja. Sifat sebuah sungai adalah menghabiskan semua poin pergerakan *unit* yang melewatinya. Hal ini bagaimanapun, dapat diatasi dengan memiliki sebuah teknologi bernama *Engineering* yang dapat diriset ditengah permainan dan membuat sebuah jalan/jembatan melintasi sungai tersebut.

Gambar 3. Unit Darat yang bergerak di laut.

Setiap *unit* darat dapat bergerak diatas laut setelah pemain memiliki akses terhadap sebuah teknologi yang bernama *Optics*. Bagaimanapun, *unit* darat ini memiliki kekuatan tempur yang jauh dibawah kekuatannya ketika berada di darat dan juga lemah terhadap *unit* yang dibuat khusus

untuk bertempur diatas laut. Pergerakan unit ini juga terbatas di tempat-tempat tertentu saja, dan sampai pemain memiliki akses terhadap teknologi bernama *Astronomy*, *unit* darat ini tidak dapat melintasi bidang laut lepas/samudra.

D. *Terrain* yang tidak dapat dilintasi.

Beberapa *terrain* tidak dapat dilintasi sama sekali, kecuali oleh beberapa *unit* ataupun negara yang memiliki kemampuan khusus untuk melewatinya.

Gambar 4. Pergerakan menuju gunung.

Mountain atau pegunungan adalah salah satu *terrain* yang tidak dapat dilewati. Jika mencoba untuk melewatinya, maka kursor akan berwarna merah seperti pada gambar 4. Beberapa *terrain* lain yang tidak dapat dilewati adalah *natural wonder* yang merupakan salah satu fitur dari permainan ini. Dalam melewati rintangan seperti ini, tidak ada jalan lain selain memutar atau mencari jalan yang dapat ditembus.

Di sisi lain, *terrain* yang dikuasai pemain lain juga tidak dapat dilewati tanpa resiko terjadinya perang. Walau begitu, pemain dapat melakukan perjanjian dengan pemain lain penguasa daerah tersebut untuk melewati daerah tersebut sebagaimana melewati daerah-daerah lainnya.

E. Pergerakan melewati jalan/rel kereta api

Road dan *Railroad* didalam permainan ini adalah sebuah fitur yang dibuat oleh *unit* khusus yang bernama *Worker*. Fitur ini memungkinkan unit darat untuk bergerak dengan cepat melewati *terrain* yang ada dan mengurangi poin pergerakan yang digunakan untuk melintasi *terrain* tersebut.

Gambar 5. Pergerakan melewati jalan.

Dalam gambar 5 dapat dilihat bahwa unit dapat bergerak lebih jauh jika melewati *terrain* yang memiliki bidang *road* diatasnya. Bidang *railroad* membuat poin pergerakan yang digunakan lebih sedikit lagi jika dibanding dengan bidang *road* dan juga memberikan bonus yang berbeda, dimana *road* memberikan bonus harta terhadap kota yang dihubungkan,, dan *railroad* memberikan bonus produksi.

F. Pergerakan melewati musuh atau daerah sekitar musuh.

Pemain dapat menggerakan *unit* yang dimilikinya melewati daerah disekitar sebuah *unit* musuh yang sedang berjaga, namun melakukan hal ini akan berakibat hilangnya semua poin pergerakan *unit* tersebut. Hal ini disebut dengan *Zone of Control* dari *unit* lawan tersebut.

Gambar 6. Zone of Control

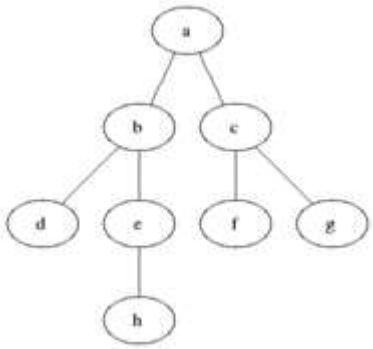
Jika pemain memutuskan untuk melakukan penyerangan terhadap *unit* yang dimiliki oleh musuh, maka penggunaan poin pergerakan yang diperlukan adalah 1(untuk menyerang) + poin pergerakan yang dibutuhkan untuk bergerak menuju bidang yang ditempati oleh musuh.

III. Algoritma BFS dan DFS

A. Algoritma Breadth-First Search

Algoritma Breadth-First Search adalah algoritma pencarian graf dimana graf ditelusuri mulai dari akar hingga tujuan yang dicari secara horizontal dari satu node ke node lain yang berada pada tingkat yang sama terlebih dahulu. Pencarian solusi menggunakan BFS dapat digambarkan melalui algoritma dibawah ini:

1. Masukkan simpul akar ke antrian Q. Jika akar merupakan solusi, maka berhenti.
2. Jika Q kosong, maka berhenti.
3. Ambil *node* s dari kepala antrian, bangkitkan semua anaknya dan masukkan kedalam antrian Q (enqueue).
4. Jika s merupakan solusi maka berhenti. Jika tidak maka lakukan kembali dari langkah 2.



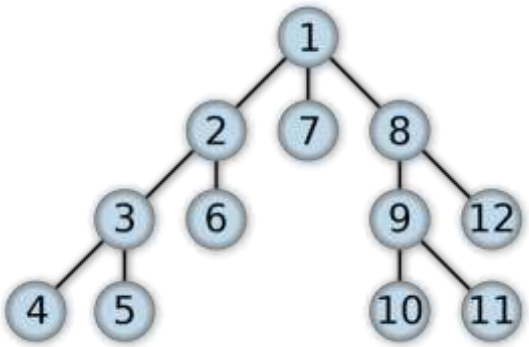
Gambar 7. Urutan pencarian algoritma BFS.

B. Algoritma Depth-First Search

Algoritma Depth-First Search adalah algoritma pencarian graf dimana graf ditelusuri mulai dari akar hingga *node* tujuan melewati anak dari setiap node terlebih dahulu. Saat mencapai simpul ujung dan belum mendapatkan solusi, maka DFS akan melakukan backtrack ke simpul sebelumnya dan melanjutkan ke simpul anak lainnya yang belum dikunjungi. Dimana BFS menggunakan struktur data Queue, DFS menggunakan struktur data Stack.

Pencarian solusi menggunakan DFS dapat digambarkan melalui algoritma dibawah ini:

1. Masukkan simpul akar kedalam stack S. Jika merupakan solusi, maka berhenti.
2. Jika S kosong, maka berhenti
3. Ambil simpul v dari top stack S, bangkitkan semua anaknya dan masukkan kedalam stack.
4. Jika v solusi maka berhenti. Jika tidak maka lakukan kembali dari langkah 2.



Gambar 8. Urutan pencarian algoritma DFS.

IV. Implementasi

Pada bagian ini akan dibahas bagaimana implementasi strategi BFS dan DFS pada permainan Civilization V untuk menggerakan sebuah unit darat bertipe infanteri dari tempat asal menuju tempat tujuan tanpa memperhitungkan cost.

Dalam kasus ini, akan dilakukan perjalanan sederhana oleh sebuah *unit Gatling Gun* dari sebuah *terrain Grassland* menuju sebuah terrain *Tundra* dengan sumber daya *Deer* didalamnya. Dalam kasus ini, setiap *node* direpresentasikan oleh sebuah bidang heksagonal. Di sekitar sebuah *node*,terdapat enam node lain yang berdekatan. *Node* baru yang dibangkitkan adalah hasil dari perubahan state ke state baru.

Gambar 9, Unit Gatling Gun dan *tile* tujuan.

Perjalanan ke salah satu *node* yang ada di sekitar *node* asal akan mengubah *state*. Algoritma akan berhenti mencari ketika sudah menemukan sebuah *tile tundra* dengan sumber daya *Deer* didalamnya.
Sebuah gerakan valid jika mendekati *tile tundra* (berupa *tile tundra* atau memiliki *tile tundra* di sekitarnya) dan tidak valid jika menjauhi *tile tundra*

A. Implementasi BFS

Pseudocode dari algoritma BFS yang digunakan sebagai berikut:

```
Procedure BFS(input/output array
of node S)
--------------------------------
state currentState, newState
queue of state states
initiateQueue(states)

 while states not empty do
  currentState=states.pop()
   foreach moveDirection do
    if move is valid do
  newState=state.move(newDirection)
  states.enqueue(newState)
 endif
 if state is solution do
  S.enqueue(state.tile)
  endif
 endfor
```

Proses pencarian dengan BFS untuk mencari jalan dalam Civilization V dapat dijabarkan lebih lanjut dalam algoritma berikut ini:

1. Inisialisasi antrian Q dengan memasukkan semua state awal dimana setiap state awal merupakan sebuah bidang heksagonal tempat unit berdiam.
2. Jika Q kosong maka berhenti.
3. Ambil *node* h dari kepala antrian dan bangkitkan semua anak yang valid. Selain berupa *tundra* atau memiliki *tundra* di sekitarnya, tile valid juga belum dilalui.
4. Untuk setiap anak yang dibangkitkan periksa setiap bidang heksagonal yang berada di dekatnya.
6. Masukkan semua anak yang dibangkitkan kedalam antrian Q dibelakang kepala antrian.
7. Jika h merupakan solusi, maka masukkan h kedalam himpunan solusi S. Sebuah *node* merupakan solusi ketika ia memiliki sifat *tundra* dan *Deer* di dalamnya.
8. Kembali ke langkah kedua untuk memeriksa solusi lain.

B. Implementasi DFS

Pseudocode dari algoritma DFS yang digunakan sebagai berikut:

```
Procedure DFS(input/output stack
of state states, input/output
array of node S)
--------------------------------
state currentState, newState

   foreach moveDirection do
    if move is valid do
  newState=state.move(newDirection)
  states.push(newState)
 endif
 if state is solution do
  S.push(state.tile)
  endif
   DFS(states)
 endfor
```

Pencarian dengan DFS untuk mencari jalan dalam Civilization V dijabarkan sebagai berikut:

1. Inisialisasi stack S dengan memasukkan semua state awal.
2. Jika S kosong maka berhenti.

3. Ambil *node* s dari teratas stack dan bangkitkan semua anak yang valid.
4. Update state dari setiap anak yang dibangkitkan.
5. Masukkan semua anak yang dibangkitkan ke atas stack Q.
6. Jika s merupakan solusi maka masukkan s kedalam himpunan solusi S.
7. Kembali ke langkah 2 untuk mencari solusi lain.

V. Hasil Implementasi

Menggunakan algoritma diatas, ditemukan bahwa pergerakan menggunakan BFS cenderung menemukan jalur menuju tempat tujuan jika dibandingkan dengan algoritma DFS. Hal ini dikarenakan pencarian DFS yang cenderung mencari ke kedalaman yang lebih besar terlebih dahulu dibandingkan dengan BFS. Menggunakan DFS dalam sebuah peta jenis *huge* dalam permainan Civilization V dapat membuat perhitungan yang amat lama dikarenakan banyaknya jumlah simpul anak dimana setiap simpul selalu memiliki setidaknya 5 anak.

Gambar 10, Urutan pencarian Algoritma DFS.

Di sisi lain, penggunaan algoritma BFS memiliki kemungkinan lebih besar untuk menemukan tujuan jika *tile* tujuan cukup dekat, akan tetapi algoritma BFS dapat memakan waktu amat lama jika tempat tujuan sangat jauh atau memiliki kedalaman yang sangat besar dalam bentuk graf. Bagaimanapun, penggunaan algoritma BFS menjamin adanya solusi yang dapat ditemukan jika dibandingkan dengan penggunaan algoritma DFS.

Gambar 11, Urutan pencarian Algoritma BFS.

VI. Kesimpulan

Algoritma BFS dan DFS dapat digunakan untuk mencari jalan menuju sebuah *tile* tertentu dalam permainan Civilization V. Bagaimanapun, banyak sekali faktor didalam permainan yang membutuhkan improvisasi terhadap algoritma BFS dan DFS agar dapat digunakan secara optimal didalam permainan Civilization V. Algoritma BFS dan DFS yang dijelaskan dalam makalah ini hanya membahas dasar pergerakan tanpa memperhitungkan *cost* dari poin pergerakan. Strategi BFS dan DFS dapat memakan waktu sangat lama jika jalur yang akan ditempuh semakin jauh dan peta tempat permainan semakin besar.

DAFTAR PUSTAKA

[1] Civilopedia, Game Civilization V.

[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2014-2015/stima14-15.htm

[3] http://civilization.wikia.com/wiki/Civilization_Games_Wiki

[4] http://en.wikipedia.org/wiki/Depth-first_search

[5] http://en.wikipedia.org/wiki/Breadth-first_search

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, 4 Mei 2015

Rifkiansyah Meidian Cahyaatmaja - 13511084