Route Planning untuk Jungling Efisien dalam permainan DOTA 2

Makalah IF2211 Strategi Algoritma

Andhika Arta Aryanto - 13520081
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520081@std.stei.itb.ac.id

***Abstract*— DOTA2 adalah permainan online strategi bertipe *Multiplayer Online Battle Arena* (MOBA). Pada permainan ini, setiap pemain harus dapat memanfaatkan waktunya dengan seefektif mungkin, salah satunya dalam kasus *jungling*. Makalah ini dibuat dalam tujuan menemukan rute yang paling optimal untuk melakukan jungling menggunakan algoritma Branch and Bound.**

Keywords*— *DOTA2, Route Pathing, UCS, Greedy Best-First Search, A*

I. PENDAHULUAN

DOTA2 adalah permainan online strategi MOBA yang dibuat oleh Valve. Multiplayer Online Battle Arena (MOBA) adalah sebuah *sub-genre* dari *video games* dengan tipe strategi di mana terdapat 2 tim yang saling berhadapan. 1 tim terdiri dari 5 pemain dengan masing – masing pemain mengontrol seorang *hero* yang memiliki kemampuan serta atribut unik dari *hero* lainnya.

Kedua tim akan saling berhadapan di map DOTA2 dan dibagi menjadi dua "sisi", yaitu *radiant* dan *dire*. Map DOTA2 terbagi menjadi 3 lane dan tujuan utama dari pemain adalah menghancurkan bangunan utama tim lawan (Ancient).

Figure 1. Map DOTA2 dengan Label Sumber : https://dota2.fandom.com/wiki/Map

Saat mulai permainan, masing – masing pemain akan memilih 1 *hero* unik dari 123 *hero* yang ada dalam permainan DOTA2. Setiap pemain akan mengisi sebuah posisi dalam permainan. Posisi ini akan menentukan lane untuk pemain. Misal untuk *Radiant*. Posisi 1 dan 5 akan menempati lane bawah dan melawan Posisi 3 dan 4 dari *Dire*. Begitu juga sebaliknya. Lalu Posisi 2 akan melakukan 1v1 di lane tengah. Angka angka pada posisi juga memiliki makna tradisional dimana biasanya makin kecil angka berarti posisi tersebut memiliki prioritas yang lebih besar dalam mendapatkan gold, mendapatkan kill, dan lain – lain. Untuk namanya sebagai berikut : Posisi 1 (Safe Lane), Posisi 2 (Mid Lane), Posisi 3 (Off Lane), Posisi 4 (Soft Support), Posisi 5 (Hard Support).

Dalam Permainan ini masing masing *hero* yang dikontrol pemain dapat bertambah kuat dengan 2 cara, cara pertama adalah dengan mendapat *exp* untuk menaikkan level *hero* lalu cara kedua adalah mendapatkan gold untuk membeli *item* yang nantinya akan memberi *hero attribute* lebih atau bahkan sebuah *ability* baru. Untuk memenangkan permainan, banyak strategi yang perlu dipertimbangkan, namun hal ini dapat dibuat menjadi suatu hal kuantitatif karena tim dengan *hero* yang lebih kuat (Level lebih tinggi, Gold lebih banyak) akan lebih mudah memenangkan permainan.

Oleh karena itu, setiap pemain (khususnya posisi 1 dan 2) harus dapat mendapatkan *exp* dan *gold* secara seefesien mungkin. Beberapa cara yang bisa dilakukan untuk mendapat kedua hal ini adalah membunuh *creeps* atau pemain lainnya. Salah satu cara membunuh *creeps* adalah dengan melakukan jungling, yaitu membunuh para *neutral creeps* yang berada pada daerah jungle pada map. Agar bisa mendapatkan *advantage* dari *hero* musuh, pemain dapat mencoba mencari cara agar bisa melakukan *jungling* secepat mungkin. Pada makalah kali ini, akan digunakan algoritma route planning untuk menemukan rute manakah yang bisa pemain tempuh agar bisa melakukan jungling seefisien mungkin

Tentu biasanya dalam route planning, sudah ada bobot yang jelas untuk masing masing tempat. Misal, terdapat jarak apabila ingin mencari rute perjalanan dari satu kota ke kota lainnya. Dalam permainan DOTA2 terdapat faktor lain sehingga

perhitungan bobot akan dilakukan berbeda yang nantinya akan dijelaskan.

II. LANDASAN TEORI

A. *Uninformed Search Algorithms*

Uninformed serach adalah algoritma searching yang bekerja mirip dengan cara *brute force*, algoritma ini tidak memiliki informasi mengenai tree yang akan dilakukan pencarian selain informasi cara bergerak antar nodenya. Karena hal ini, algoritma ini juga biasanya disebut dengan *Blind Search*. Pencarian dalam algoritma ini akan berdasarkan suatu urutan yang berbeda tiap tipe pencariannya

Berikut beberapa tipe dari Uninformed Search Algorithm :

a) *Breadth-first Search*

Breadth-first Search atau BFS adalah salah satu algoritma yang paling sering digunakan dalam traversal sebuah tree atau graph. Bila dilihat dari namanya algoritma ini bergerak secara "*Breadthwise"* atau bisa dibilang seperti melebar terlebih dahulu. Algoritma BFS melakukan searching dari root node dan melakukan ekspansi kepada semua anak node pada level sekarang sebelum mencari ke node pada level berikutnya. Algoritma ini biasanya akan memakan waktu eksekusi yang tidak lama, namun karena node yang dibangkitkan banyak, memori yang digunakan cukup besar.

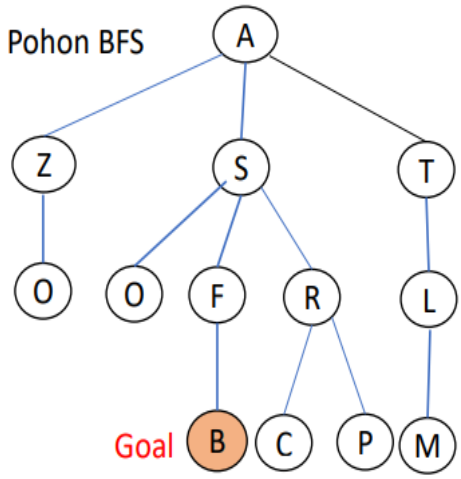

Figure 2. Contoh Pohon BFS

Misal pada gambar di atas dimulai dari A yang membangkitkan Z, S,T. Lalu Z membangikitkan O, S membangkitkan OFR, T membangkitkan L dan seterusnya sampai menemukan goal node. BFS diimpelementasi menggunakan struktur data queue yang bersifat FIFO

b) *Depth-first Search*

Depth-first Search atau DFS adalah algoritma yang berbentuk rekursif. Seperti namanya, algoritma ini akan melakukan traversal secara "Depthwise" di mana dia akan melakukan traversal suatu node sampai kedalaman/level paling bawah terlebih dahulu barulah membangkitkan node lain yang terdapat pada level node awal tersebut. Implementasi algoritma ini biasanya menggunakan struktur data Stack. Bila dibandingkan dengan BFS, DFS membutuhkan memori yang lebih kecil karena hanya perlu menyimpan kumpulan node dalam stack dari root ke node sekarang, sementara BFS harus menyimpan semua node yang akan dibangkitkan ke sebuah queue. Algoritma DFS ini bisa dibilang membutuhkan waktu yang lama karena hanya melakukan searching pada 1 akar sampai daun paling bawah, sehingga apabila goal node tidak ada di rute itu membutuhkan waktu yang lebih lama lagi untuk traversal rute – rute lain. Berikut ilustrasi gambar :

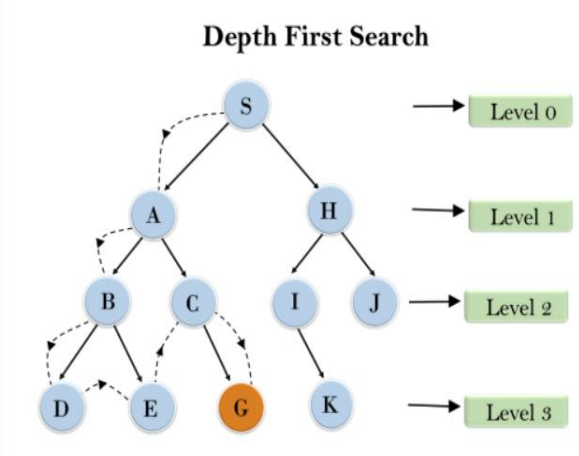

Figure 3. Contoh Pohon DFS Sumber : https://www.javatpoint.com/ai-uninformed-search-algorithms

c) *Depth-Limited Search Algorithm*

Depth-Limited Search pada intinya adalah DFS namun diberikan limit kedalaman sebelumnya. Hal ini dapat memberikan solusi untuk kemungkinan adanya infinte loop pada algoritma DFS karena rute yang sangat dalam. Namun, algoritma ini memungkinkan *incompleteness* di mana solusi tidak ditemukan karena misal goal node ternyata berada dibawah limit yang kita berikan , berikut ilustrasi pohon DLS dengan limit kedalaman 2 :

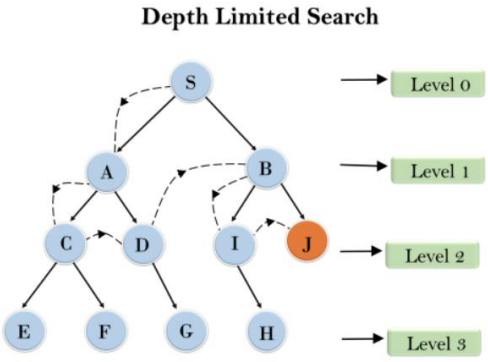

Figure 4. Contoh Pohon DLS Sumber : https://www.javatpoint.com/ai-uninformed-search-algorithms

d) *Iterative Deepening Search*
Algoritma ini bisa dibilang seperti gabungan dari DFS dan BFS. Algoritma ini akan memiliki suatu limit kedalaman, dan tiap iterasinya akan menambah kedalaman tersebut sampai goal node ditemukan. Namun, bisa terlihat kelemahan dari algoritma ini berarti langkah – langkah sebelumnya yang sudah dilakukan akan dilakukan berulang kali. Berikut ilustrasi untuk pohon IDS :

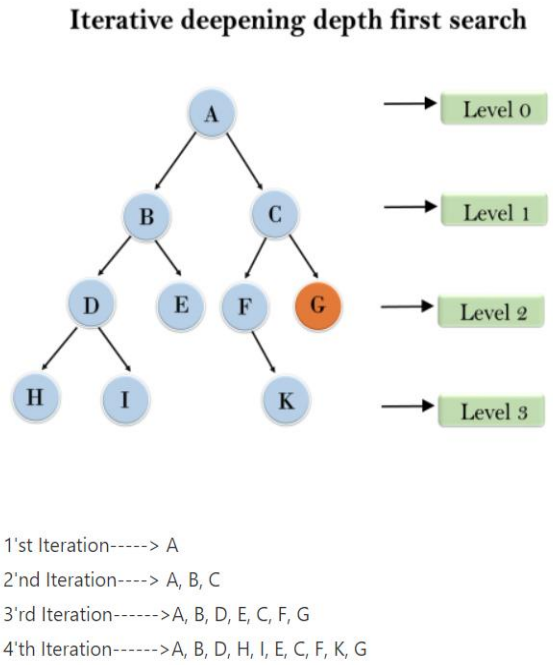

Figure 5. Contoh Pohon IDS Sumber : https://www.javatpoint.com/ai-uninformed-search-algorithms

e) Uniform-cost Search
Uniform-cost Search adalah algoritma yang mirip dengan *Breadth-first Search*, namun bisa digunakan untuk pohon/graf berbobot. Di mana tiap langkahnya akan dibangkitkan simpul dengan *cost* yang paling kecil dari root. Uniform-cost Search ini diimplementasikan menggunakan *Priority Queue* dengan menggunakan bobot. Bobot di sini dapat berupa berbagai macam hal, misal jarak antar kota, lama waktu yang dibutuhkan untuk mencapai node tersebut, dan masih banyak lagi.

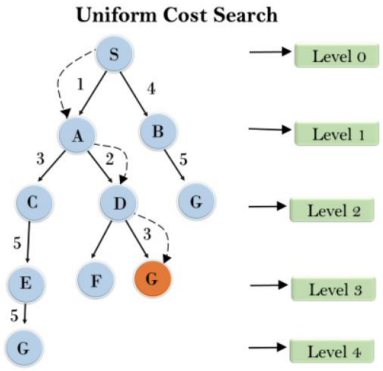

Figure 6. Contoh Pohon UCS Sumber : Sumber : https://www.javatpoint.com/ai-uninformed-search-algorithms

B. *Informed Search Algorithms*

Apabila sebelumnya Uninformed Search Algorithm adalah algoritma "buta" karena tidak mengetahui apa – apa mengenai masalah yang dihadapinya, algoritma ini disebut juga Heuristic search karena kita memiliki informasi mengenai masalah yang dihadapi. Misalnya memiliki informasi seberapa jauh dari goal node, cost untuk suatu jalur, bagaimana bisa mencapai goal node, dan lain lain. Algoritma ini biasanya mempunyai suatu fungsi Heuristik yang memuat cost dari suatu node untuk mencapai goal node sehingga dengan menggunakan informasi ini kita dapat menemukan jalan/rute yang paling optimal untuk mencapai goal. Namun, heuristik yang digunakan belum tentu tepat guna sehingga solusi tidak selalu optimal. Terdapat 2 Algoritma Informed yang akan dibahas, yaitu :

a) Greedy Best-First Search
Greedy Best-First Search merupakan algoritma yang menggunakan sebuah evaluasi atau taksiran pada suatu simpul. Algorita ini memeriksa simpul yang memiliki *cost* terkecil dibutuhkan untuk mencapai solusi yang diinginkan. Cost terkecil ini dihitung menggunakan suatu fungsi heuristik. Hal ini sebenarnya akan mirip dengan UCS, namun di mana UCS hanya misalnya hanya jarak doang, namun menggunakan suatu fungsi heuristik. Algoritma ini memiliki kelemahan berupa *not complete* dengan maksud jarak suatu kota dengan kota lainnya pasti terdapat jarak lurus, namun belum tentu kedua kota berhubungan. Ada juga Get stuck with local minima di mana algoritma hanya berputar di satu tempat dengan cost terkecil tidak sampai simpul goal, dan Irrevocable, yaitu bila simpul sudah dipilih tidak akan berbalik atau diganti.

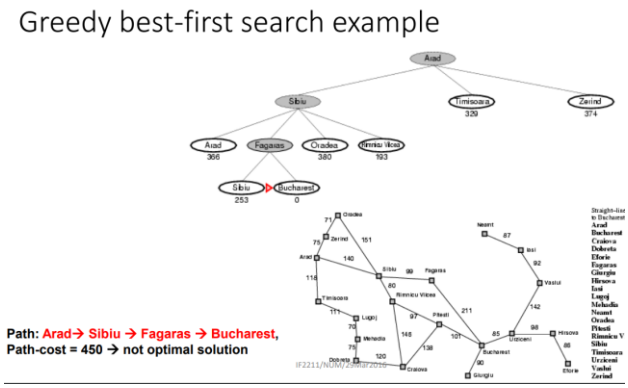

Figure 7. Contoh penggunaan GBFS Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian1-2021.pdf

Misal contoh diatas adalah penggunaan algoritma ini dengan fungsi heuristik yang digunakan adalah jarak lurus dari kota n ke Bucharest.

b) A*

Algoritma A* Search merupakan gabungan dari algoritma *Uniform Cost Search* dengan *Greedy Best-First Search*. Dengan maksud pada UCS fungsi dan jarak yang digunakan adalah jarak antar 2 node dan *Greedy Best-First Search* menggunakan suatu fungsi heuristik misalnya berupa jarak lurus antara suatu simpul ke node solusi. Nah, pada A* Search cost yang digunakan untuk menghitung antar simpul adalah gabungan dari kedua fungsi tersebut.

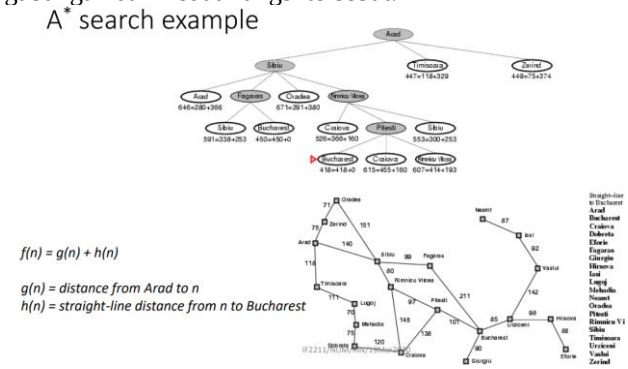

Figure 8. Contoh Penggunaan A search Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf*

Terlihat pada di atas, A* search menggunakan fungsi heuristik berupa jarak dari n ke Arad dan jarak garis lurus dari n ke bucharest

C. Jungle pada Permainan DOTA2

Figure 9. Map DOTA2 dengan penanda Jungle Sumber : https://dota2.fandom.com/wiki/Neutral_creeps

Jungle adalah bagian map dari DOTA2 yang berada di antara lane- lane pemain. Bisa dilihat pada gambar di atas bahwa terdapat 4 jenis Camp yang terdapat pada jungle. Camp itu sendiri adalah suatu daerah yang berisi *Neutral Creep* yang bisa dibunuh oleh pemain untuk mendapatkan exp dan gold. 4 jenis camp yang ada adalah Easy Camp, Medium Camp, Hard Camp,dan Ancient Camp. Pada makalah ini *Camp* yang akan dibahas secara mendalam adalah Camp bagian bawah untuk bagian Radiant.

Setiap jungle ini terdapat sekumpulan *Neutral Creep* yang akan muncul setiap 60 detik apabila Creep sebelumnya sudah dibunuh. Tujuan dari pemain adalah membunuh para *Creep* ini untuk mendapatkan exp serta gold yang nantinya akan menambah kekuatan dengan cara menaikan level mereka ataupun menggunakan gold untuk membeli item yang akan memperkuat hero yang digunakan.

Pada makalah ini Jungle akan dimodelkan menjadi suatu graf berbobot dengan root node merupakan camp paling kiri dan goal node adalah camp paling kanan. Tujuan pemain adalah mencari rute paling efektif untuk mencapai goal node tercepat. Berikut pada bab 3 akan dilakukan pembahasan untuk menemukan hal ini

III. ANALISIS DAN PEMBAHASAN

DOTA2 adalah permainan strategi dimana pemain harus menggunakan cara sebaik mungkin untuk dapat mendapatkan resource ataupun mengalahkan lawan. Salah satu cara yang bisa dilakukan adalah mengumpulkan EXP serta gold. Untuk hal ini bisa digunakan dengan membunuh creep dan yang akan dibahas lebih dalam pada makalah ini adalah rute jungling dari pemain.

Seperti yang sudah dibahas sebelumnya, jungle yang akan dibahas lebih lanjut adalah jungle bagian bawah untuk sisi radiant. Di sini terdapat 6 camps yang bisa player lewati. Camp – camp ini akan dibuat menjadi node. Berikut gambar

Figure 10. Map DOTA2 dengan penanda pada camp Sumber : https://liquipedia.net/dota2/Neutral_Creeps

Untuk pengerjaan, camp pada dota akan diberikan nama. Di sini node awal adalah A (camp paling kiri) dan goal node adalah F (camp paling kanan). Penarikan garis untuk kemungkinan jalur yang bisa ditempuh akan dilakukan dengan 1 camp dan perhitungan jarak untuk bobot akan dilakukan dengan 2 camp terdekatnya. Berikut pemodelan menjadi graf.

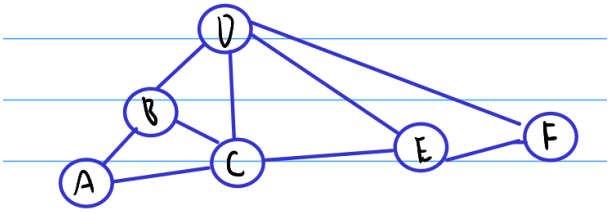

Figure 11. Pemodelan menjadi graf

Lalu akan ditentukan fungsi heuristik yang akan digunakan untuk nanti dilakukan route planning. Untuk ini f(n) = g(n) + h(n) dalam hal ini g(n) adalah jarak sejauh ini dari A ke N dan h(n) adalah jarak langsung dari N ke camp F. Berikut pembobotan yang dilakukan dengan menggunakan jarak antar 2 camp sebagai berikut :

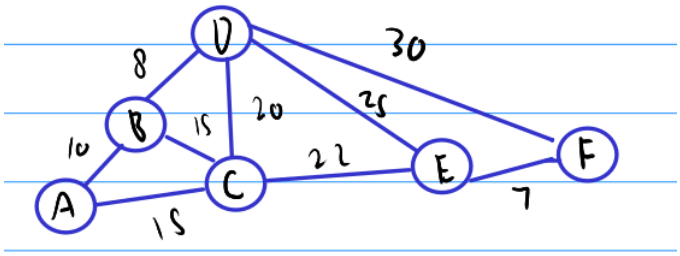

Figure 12. Jarak antara 1 camp dengan 2 camp terdekatnya

Berikut bobot antara 2 camp, pembobotan dihitung dengan lama waktu yang dibutuhkan untuk mencapai camp lainnya. Hal ini akan berbeda dengan jarak lurus langsung karena terdapat banyak obstacle lain sehingga garis mungkin saja terlihat lebih pendek. Berikutnya akan dihitung jarak h(n) yaitu jarak lurus dari n ke camp F

Camp	Jarak Lurus ke F
A	40
B	35
C	25
D	30
E	7

Untuk kasus ini, jungling efisien adalah berarti player dapat cepat mencapai camp paling kanan (camp F) lalu setelah itu player bisa melakukan hal lain. Jadi hal ini akan mencari rute planning paling pendek agar player bisa mencapai F dari A dengan melewati jarak terpendek. Pada kasus ini sebenarnya, lama untuk membersihkan suatu camp akan bergantung pada hero yang digunakan serta *neutral creep* yang muncul. Namun, untuk kemudahan perhitungan akan dianggap hero yang digunakan dapat melakukan *clear* dengan sangat cepat sehingga lama waktu *clear* tidak menjadi pertimbangan\

Setelah sudah mendapatkan bobot heuristik untuk simpul – simpul pada graf. Bisa digunakan algoritma – algoritma yang ada untuk penentuan rute. Berikut akan digunakan beberapa cara :

A. *Greedy Best-First Search*

Algoritma GBFS akan hanya menggunakan fungsi heuristik yang sudah diberikan sebelumnya, berarti pada kasus ini adalah jarak garis lurus dari node n ke goal node atau pada kasus ini adalah node F. Berikut pengerjaan :

Node Hidup	Node Dibangkitkan
A	B(35), C(25)
C(25)	B(35), D(30), E(7)
E(7)	C(25), D(30), F(0)
F	SUDAH GOAL NODE

Berarti dapat dilihat bahwa dengan menggunakan *Greedy Best-First Search* dimana hal yang dipertimbangkan hanyalah nilai heuristik yang sudah diketahui sebelumnya berupa jarak langsung dari current node ke goal node, rute yang didapat adalah A-C-E-F, yang apabila dilihat dari path-cost nya adalah sebesar 15+22+7 = 44

B. *UCS*

Selanjutnya, akan digunakan algoritma UCS, karena algoritma ini merupakan *Uninformed Search*, berarti heuristik tidak diketahui dan hal yang hanya diketahui adalah path – cost total sejauh ini, berikut perhitungan :

Node Hidup	Node Dibangkitkan
A	B(10),C(15)
B(10)	C(15),C(25),D(18)
C(15)	B(30), D(35), E(37), C(25), D(18)
D(18)	C(38), E(43), F(48), B(30), D(35), E(37), C(25)
C(25)	D(45), E(47), C(38), E(43), F(48), B(30), D(35), E(37)
B(30)	D(38), D(45), E(47), C(38), E(43), F(48), D(35), E(37)
D(35)	E(60), F(65), D(38),D(45), E(47), C(38), E(43), F(48), E(37)

E(37)	F(44), E(60), F(65), D(38), D(45), E(47), C(38), E(43), F(48)
C(38)	E(60), F(44), E(60), F(65), D(38), D(45), E(47), E(43), F(48)
D(38)	E(63), F(68), E(60), F(44), E(60), F(65), D(45), E(47), E(43), F(48)
F(44)	SOLUSI DITEMUKAN

Dengan menggunakan algoritma UCS, bisa dilihat rute yang ditemukan adalah A-C-E-F, hal ini sama dengan menggunakan Greedy Best First Search dan mendapat path-cost sebesar 44.

C. *A Star*

Selanjutnya akan digunakan algoritma A Star, pada intinya sebenarnya akan sama dengan kedua algoritma sebelumnya namun bobot yang digunakan adalah tambahan dari bobot kedua algoritma sebelumnya

Node Hidup	Node Dibangkitkan
A	B(45), C(40)
C(40)	B(45), D(65), E(44)
E(44)	F(44), ... Pasti lebih besar
F(44)	SOLUSI DITEMUKAN

Dengan menggunakan algortima A Star, solusi yang ditemukan juga sama, yaitu ACEF dengan nilai beban path route sebesar 44 .Dengan ketiga algoritma diatas, ditemukan bahwa apabila player ingin mencapai jungle F dari jungle A dengan rute terpendek adalah dengan rute A-C-E-F. Ketiga algoritma menemukan jawaban yang sama.

Sebenarnya, masih banyak faktor yang perlu diperhatikan dalam melakukan jungling pada permainan DOTA2, namun faktor tersebut sangatlah banyak dan kompleks. Oleh karena itu juga DOTA 2 merupakan permainan yang perlu banyak waktu dari pemain karena permainannya yang kompleks dan banyak sekali faktor. Menggunakan ketiga algoritma, ditemukan rute jalan jungle dari jungle paling kiri ke paling kanan yang paling efisien adalah A-C-E-F, padahal sebenarnya ada faktor lain seperti lama membersihkan jungle, faktor bahaya dari posisi jungle tersebut, juga masih banyak lagi.

IV. KESIMPULAN

Algoritma Route Pathing yang digunakan pada makalah ini adalah UCS, BCFS, dan A star. Algoritma digunakan untuk menemukan rute jungling yang paling cepat dari jungle paling kiri (camp A) sampai jungle paling kanan (camp F). Dari ketiga algoritma ditemukan bahwa rute yang paling efisien yang bisa player gunakan adalah A-C-E-F dengan cost sebesar 44.

Walaupun hal ini, masih perlu banyak perbaikan yang digunakan pada algoritma. Misalnya adalah heuristik yang lebih baik, karena perlu fungsi heuristik yang lebih tepat guna misalnya menggunakan faktor lama *clear* dari jungle dan faktor tingkat kesusahan membersihkan *jungle*. Kedua hal ini namun sangatlah subjektif dan perlu banyak percobaan sehingga faktor yang dipertimbangkan hanyalah murni dari jarak dan waktu yang diperlukan.

Algoritma yang digunakan melakukan asumsi bahwa player hanya perlu mencapai *camp F* secepat mungkin, dan tidak harus mengunjungi semua camp , hal ini dilakukan dengan pertimbangan bahwa setelah berjalan melewati jungle, Player bisa melakukan hal lain seperti *farming* pada lane, push *objective*, *teamfight*, dan masih banyak lagi, juga masih menyisakan camp lain untuk player lainnya..

ACKNOWLEDGMENT

Penulis mengucapkan syukur kepada Tuhan yang Maha Esa atas rahmat-Nya penulis dapat menyelesaikan Makalah IF2211 Strategi Algoritma – Semester 2 Tahun 2021/2022 dengan tepat waktu. Penulis juga mengucapkan terima kasih kepada teman dan orang tua yang sudah memberi dukungan lewat kata – kata baik maupun doa. Tidak lupa penulis memberikan terima kasih sebesar-besarnya kepada seluruh dosen mata kuliah IF2211 Strategi Algoritma tahun akademik tahun 2021/2222 yang sudah membaikan ilmu sehingga dapat digunakan pada makalah ini.

REFERENCES

[1] Munir, Rinaldi. 2022. Penentuan rute (Bagian 1) https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian1-2021.pdf Diakses 21 Mei 2022

[2] Munir, Rinaldi. 2022. Penentuan rute (Bagian 2) https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf Diakses 21 Mei 2022

[3] https://dota2.fandom.com/wiki/Map Diakses 21 Mei 2022

[4] https://www.javatpoint.com/ai-uninformed-search-algorithms Diakses 22 Mei 2022

[5] https://www.javatpoint.com/ai-informed-search-algorithms Diakses 22 Mei 2022

[6] https://dota2.fandom.com/wiki/Jungling#:~:text=Jungling%20(also%20referred%20to%20as,for%20additional%20Gold%20or%20Experience. Diakses 22 Mei 2022

[7] https://www.geeksforgeeks.org/difference-between-informed-and-uninformed-search-in-ai/ Diakses 22 Mei 2022

[8] https://liquipedia.net/dota2/Neutral_Creeps Diakses 22 Mei 2022

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, 20 Mei 2022

Andhika Arta Aryanto
13520081