Penerapan Algoritma Branch and Bound untuk Mencari Solusi dari *Puzzle* Dungeon 12 pada *Game* EPIC RPG

Daffa Ananda Pratama Resyaly - 13519107
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
daffaresyaly@gmail.com

***Abstraksi*—Sekarang ini, banyak orang yang suka bermain *game online* sehingga pengembang *game* berusaha untuk bersaing satu sama lain untuk membuat *game* yang unik dan disukai oleh banyak orang yang mengakibatkan banyaknya *game online* yang telah dibuat. *Game* tersebut dapat berbasis konsol, PC, dan *mobile*. Beberapa *game* ini ada yang memiliki Real-Time Graphical User Interface (Real-Time GUI) dan ada pula yang tidak. Contoh *game* yang tidak memiliki Real-time GUI adalah *game* berbasis Chatbot, salah satunya adalah EPIC RPG. *Game* EPIC RPG merupakan salah satu jenis Role-Playing Game (RPG) berbasis Chatbot yang dapat dimainkan dalam suatu aplikasi, yaitu Discord. Tujuan dari *game* ini adalah berkompetisi dengan pemain lain untuk menaikkan level setinggi mungkin dan mengumpulkan beberapa *item*, serta menaikkan area dengan melakukan *dungeon*. Beberapa dari *dungeon* yang ada pada EPIC RPG, yaitu mulai dari *dungeon* 10 sampai *dungeon* 15 memiliki suatu *puzzle*. Beberapa dari *puzzle* ini dapat dipecahkan dengan menggunakan suatu algoritma, misalnya algoritma Branch and Bound.**

Kata Kunci—EPIC RPG, Dungeon, Branch and Bound

I. PENDAHULUAN

Pada zaman digital seperti sekarang ini, *game online* memiliki sangat banyak peminat, mulai dari kalangan muda, tua, sampai lanjut usia (lansia). Akibat dari maraknya *game online*, pengembang *game* berusaha dan berkompetisi satu sama lain untuk membuat *game* sebaik mungkin, yang unik dan disukai oleh banyak orang. Walupun begitu, ada juga beberapa orang yang mengembangkan *game* hanya untuk bersenang-senang dan mengisi waktu luang. Akibat dari beberapa hal tersebut, terdapat banyak *game* yang telah dibuat di dunia digital dan keuntungan yang didapat dari *game* ini cukup besar. Setiap *game* ini memiliki *genre*-nya masing-masing. Contoh *genre* dari *game* adalah Action Game, Action-Adventure Game, Role-Playing Game, dan Strategy Game. Setiap *genre game* tersebut memiliki karakteristiknya masing-masing. Ambil contoh *game* ber-*genre* Role-Playing Game (RPG). Pada *game* ini, pemain dapat mengontrol seorang karakter fiksi, dimana karakter tersebut dapat melakukan *quest* tertentu untuk menaikkan levelnya dan mengeksplor dunia *game* lebih luas lagi. Karakteristik umum dari *game* berjenis RPG ini adalah, karakter dapat melakukan *battle* (baik dengan pemain lain maupun dengan Non-Player Character (NPC)), melaksanakan *quest*, memakai *equipment* (contohnya pedang dan perisai), dan menaikkan status dirinya sendiri (misalnya menaikkan level, *attack*, dan *defense*). *Game* ber-*genre* RPG ini juga juga dibagi menjadi beberapa jenis lagi, yaitu Action RPG, Strategy RPG, Adventure RPG, dan MMORPG.

Salah satu dari *game* ber-*genre* RPG adalah EPIC RPG, yang dapat dimainkan pada aplikasi Discord. *Game* EPIC RPG merupakan salah satu *game* berjenis Adventure RPG. *Game* ini berbasis Chatbot, dimana pemain dapat menuliskan *command* yang terdapat di *list command* pada kolom *chat* dan Chatbot akan memberikan respon terhadap *command* yang diberikan oleh pemain. Masing-masing *command* ini memiliki *cooldown* tertentu, tergantung jenis *command*-nya. Contoh dari *command* ini adalah "hunt" yang memiliki *cooldown* selama satu menit. Pada *game* EPIC RPG, terdapat beberapa area yang dapat dieksplor oleh pemain, dimana pada masing-masing area tersebut terdapat monster yang berbeda dan *command* yang dapat dilakukan oleh Chatbot berbeda juga. Area yang dapat dieksplor pada *game* ini adalah area 1 sampai area 15. Untuk dapat naik ke area yang lebih tinggi, pemain perlu melakukan *dungeon*, yaitu mengalahkan *boss* tertentu yang berbeda di setiap area. *Dungeon* memiliki *cooldown* selama dua belas jam.

II. LANDASAN TEORI

A. Algoritma Branch and Bound

Algoritma Branch and Bound merupakan suatu algoritma yang digunakan untuk persoalan optimasi, yaitu meminimalkan atau memaksimalkan suatu fungsi objektif, yang tidak melanggar batasan (*constraint*) persoalan.

Algoritma Branch and Bound bisa dibilang sebagai gabungan dari algoritma *Breadth-First Search* (BFS) dan *least cost search*. Pada BFS murni, simpul berikutnya akan diekspansi berdasarkan urutan pembangkitannya (*First In First Out*). Pada algoritma Branch and Bound, setiap simpul diberi sebuah nilai '*cost*' ĉ(i) yang merupakan nilai taksiran lintasan termurah ke simpul status tujuan yang melalui simpul status i. Simpul berikutnya yang akan diekspansi tidak lagi berdasarkan

urutan pembangkitannya, tetapi simpul yang memiliki *cost* paling kecil (*least cost search*), pada kasus minimasi, atau yang memiliki *cost* terbesar, untuk kasus maksimasi.

Algoritma Branch and Bound memiliki persamaan dengan algoritma Backtracking, yaitu pencarian solusi membentuk pohon ruang status dan simpul yang tidak 'mengarah' ke solusi akan 'dibunuh'. Namun, kedua algoritma ini memiliki perbedaan, yaitu pada algoritma Backtracking, tidak ada batasan terhadap persoalan yang dapat diselesaikan, walaupun umumnya untuk kasus non-optimisasi. Sedangkan algoritma Branch and Bound hanya bisa dilakukan untuk kasus optimisasi. Untuk setiap simpul pada pohon ruang status, ditentukan suatu batas nilai terbaik fungsi objektif pada setiap solusi yang mungkin, dan akan disimpan nilai dari solusi terbaik sejauh ini. Selain itu, pada algoritma Backtracking, pembangkitan simpul umumnya dilakukan dengan *Depth-First Search* (DFS), sedangkan pada algoritma Branch and Bound, pembangkitan simpul dilakukan dengan beberapa aturan tertentu, umumnya *best-first rule*.

Algoritma Branch and Bound menerapkan fungsi pembatas yang memangkas jalur yang dianggap tidak lagi mengarah ke solusi. Secara umum, kriteria pemangkasannya adalah sebagai berikut.

- Nilai simpul tidak lebih baik dari nilai terbaik sejauh ini (*the best solution so far*)
- Simpul tidak merepresentasikan solusi yang '*feasible*' karena ada batasan yang dilanggar
- Solusi pada simpul tersebut hanya terdiri atas satu titik, tidak ada pilihan lain. Bandingkan nilai fungsi objektif dengan solusi terbaik saat ini, yang terbaik yang diambil

Algoritma Branch and Bound dapat dibuat secara umum sebagai berikut.

1. Masukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah simpul solusi (*goal node*), maka solusi telah ditemukan. *Stop*.
2. Jika Q kosong, tidak ada solusi. *Stop*.
3. Jika Q tidak kosong, pilih dari antrian Q simpul i yang mempunyai nilai '*cost*' ĉ(i) paling kecil. Jika terdapat beberapa simpul I yang memenuhi, pilih satu secara sembarang.
4. Jika simpul I adalah simpul solusi, berarti solusi sudah ditemukan, *stop*. Jika simpul I bukan simpul solusi, maka bangkitkan semua anak-anaknya. Jika i tidak mempunyai anak, kembali ke langkah 2.
5. Untuk setiap anak j dari simpul i, hitung ĉ(j), dan masukkan semua anak-anak tersebut ke dalam Q.
6. Kembali ke langkah 2.

B. EPIC RPG

EPIC RPG merupakan sebuah *game* ber-*genre* RPG yang digabung dengan ekonomi, yang telah dibuat sejak Maret 2019. *Game* EPIC RPG merupakan salah satu *game* berjenis Adventure RPG, dimana pemain dapat meningkatkan kemampuan karakternya dengan mengoleksi beberapa *item* yang didapat melalui *hunting* monster tertentu dan membeli *equipment* menggunakan *item* tersebut. *Game* ini memiliki banyak fitur, yaitu *hunting, dungeon, player versus player* (PVP), *gambling, lootbox, leaderboards, prestige*, dll. Tujuan dari *game* ini adalah untuk menaikkan level karakter, mendapatkan *equipment* (*sword* dan *armor*) dan meng-*upgrade*-nya, serta mengalahkan bos di *dungeon* untuk menggapai area yang lebih tinggi sehingga pemain dapat membuka *command* dan fitur baru. *Game* ini berbasiskan Chatbot dan dapat diakses pada aplikasi Discord. Untuk dapat memainkan *game* ini, pemain harus memiliki akun Discord terlebih dahulu. Jika telah memiliki akun, pemain harus membuat suatu *server* pada aplikasi Discord dan mengundang bot yang bernama EPIC RPG ke dalam *server* tersebut atau masuk ke *server* Discord yang telah memiliki bot ini di dalamnya. Setelah itu, pemain dapat memulai menggunakan bot ini dengan menuliskan *command* yang berawalan 'rpg'.

Gambar 1. Logo EPIC RPG

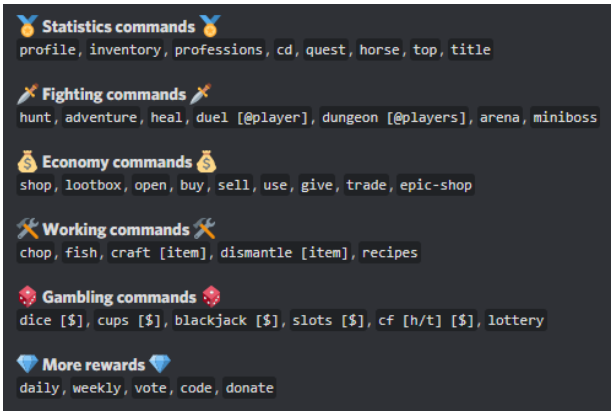
Sebagai bot yang di-*host* pada aplikasi Discord, pemain dari EPIC RPG harus menaati Discord's Terms of Service (TOS) dan Community Guidelines (CG). Selain itu, terdapat pula tiga aturan utama dalam permainan Epic RPG yang harus ditaati oleh pemain.

1. Jangan memperdagangkan atau menjual mata uang dalam *game* untuk apa pun di luar bot, artinya jangan memperdagangkan mata uang di dalam *game* untuk mata uang asli.
2. Jangan menggunakan makro, skrip, atau bentuk otomatisasi apa pun untuk menjalankan *command* dari bot.
3. Jangan menyalahgunakan atau mendapatkan keuntungan dari eksploitasi bug. Bug yang ditemukan dapat dilaporkan di Server Resmi EPIC RPG.

Jika pemain melanggar salah satu dari ketiga aturan tersebut, pemain akan dikenakan sanksi berupa *ban* sementara dari bot (tidak dapat menggunakan bot dalam jangka waktu tertentu). Jika pemain melanggar aturan tersebut secara terus

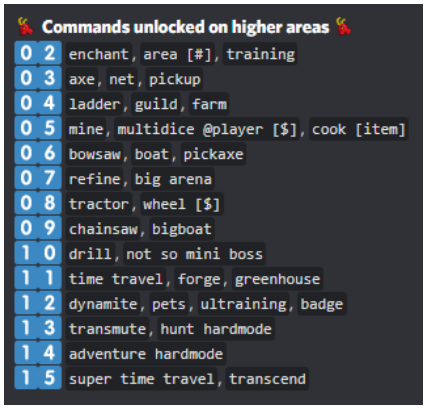
menerus, pemain dapat dikenakan sanksi berupa *ban* secara permanen.

Kita dapat melihat *list of commands* dari *game* EPIC RPG dengan menuliskan 'rpg help'. Bot kemudian akan menampilkan *list of commands* pertama yang berisikan *command* yang dapat bot lakukan secara umum, tidak terbatas pada area pemain, yaitu seperti yang dapat dilihat pada gambar di bawah.



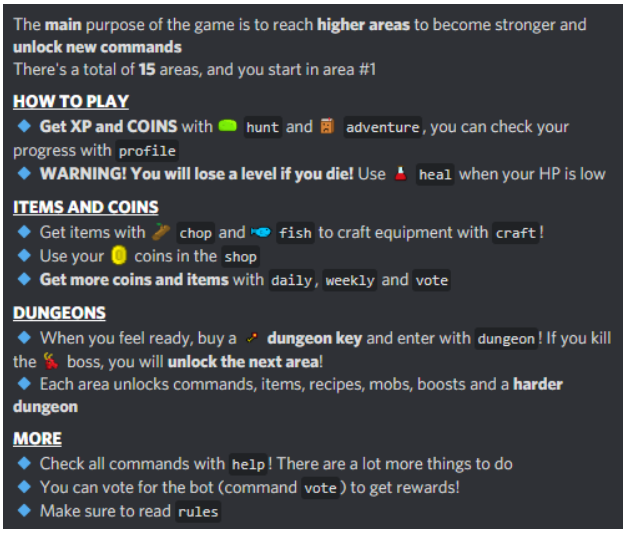
Gambar 2. *List of Commands* 1

Selain itu, bot juga akan menampilkan *list of commands* kedua yang berisi *command* yang dapat dilakukan sesuai area pemain.



Gambar 3. *List of Commands* 2

Pemain yang baru bermain harus menuliskan *command* 'rpg start' terlebih dahulu. Bot akan memberikan respon berupa *guide* untuk pemain baru.



Gambar 4. *Guide* untuk Pemain Baru

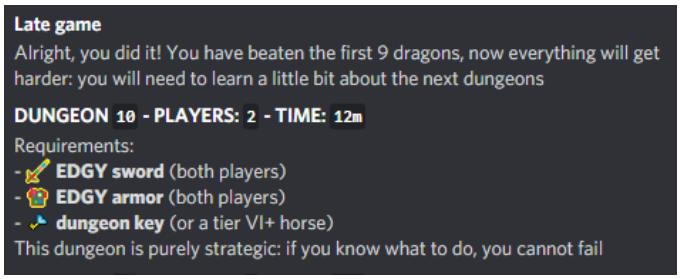
Pemain dapat menggapai area yang lebih tinggi dengan mengalahkan *dragon* di *dungeon*. Untuk melakukan *dungeon*, dibutuhkan sebuah *dungeon key* yang dapat dibeli di 'shop', kecuali untuk pemain yang telah memiliki *horse* dengan *tier* 6 ke atas, dimana ia tidak harus membeli *key* untuk melakukan *dungeon*. Selain itu dibutuhkan pula minimal dua orang pemain dan maksimal empat orang pemain untuk melakukan *dungeon*, kecuali untuk *dungeon* di area 11 dan setelahnya, dimana pemain hanya bisa mengalahkan bos sendirian. *Dragon dungeon* di masing-masing area berbeda satu sama lain, misalnya pada area 1 terdapat Ancient Dragon yang memiliki *life* sebanyak 50 poin per pemain yang melakukan *dungeon* dan *attack* sebesar 37 poin, berbeda dengan area 2 dimana terdapat The Too Ancient Dragon yang memiliki *life* sebanyak 225 poin per pemain yang melakukan *dungeon* dan *attack* sebesar 71 poin. Pemain akan kalah di *dungeon* apabila darahnya dan/atau darah semua anggota timnya telah habis.

Mulai dari area 1 sampai area 9, karakter pemain dapat mengalahkan *dragon* di *dungeon* jika *stat*-nya cukup tinggi untuk mengalahkan *dragon* tersebut (misalnya pada area 1, karakter pemain direkomendasikan untuk memiliki *attack* sebesar 9, *defense* sebesar 14, dan *life* sebesar 120 untuk dapat mengalahkan *dragon* di area tersebut). Dalam bertarung melawan *dragon*, pemain diberikan beberapa pilihan *command* untuk menyerang *dragon*, yang pada umumnya adalah 'bite', 'stab', dan 'power. Waktu pemain untuk bertarung di *dungeon* hanyalah beberapa menit, tergantung area dari pemain tersebut dan jumlah pemain yang mengikuti *dungeon* (misalnya untuk mengalahkan *dragon* di area 1 dengan jumlah pemain sebanyak 2 orang, semua pemain diberikan waktu selama 5 menit untuk mengalahkan *dragon*, 2.5 menit per pemain).

Pemain yang *stat*-nya tidak sesuai dengan yang direkomendasikan dapat meminta bantuan dari pemain lain yang *stat*-nya cukup tinggi untuk mengalahkan *dragon* di *dungeon* tersebut. Namun, pemain yang berada di area yang berbeda dengan pemain lain tidak dapat membantu pemain lain tersebut untuk mengalahkan *dragon* di area pemain itu.

Pemain yang telah mencapai area 10 dapat menggunakan *command* 'rpg start advanced'. Kemudian bot akan

menampilkan beberapa *guide 'late game'* dan informasi *dungeon* di area 10 sampai area 15 untuk pemain tersebut.



Gambar 5. Salah Satu *Guide* '*Late Game*'

Mulai dari area 10 dan seterusnya, *dungeon* akan berupa *puzzle* dan karakter pemain membutuhkan *equipment* tertentu sebagai syarat untuk melakukan *battle dungeon*. Pada area 10, seperti pada Gambar 6, *dungeon* hanya dapat dilakukan oleh dua orang pemain saja dengan syarat setiap pemain harus memiliki *equipment* 'EDGY Sword' dan 'EDGY Armor', serta sebuah *dungeon key* jika pemain tidak memiliki *horse tier* 6 ke atas. *Dungeon* pada area ini memiliki suatu skrip, dimana ketika pemain mengikuti skrip tersebut, pemain tidak akan kalah. Untuk area 11 dan seterusnya, *dungeon* hanya dapat dilakukan oleh seorang pemain. Pemain ini harus memecahkan *puzzle* untuk mengalahkan *dragon* di *dungeon* tersebut.

Dungeon 12 merupakan salah satu *puzzle dungeon* yang dapat dilakukan oleh pemain yang berada di area 12 dan telah memiliki 'ULTRA-EDGY Armor'.

Gambar 6. Contoh Dungeon 12

Dungeon 12 berbasiskan pergerakan, *life*, *defense*, dan pengumpulan *energy orb*. Karakter pemain direpresentasikan dengan 'ULTRA-EDGY Armor' dan berada di sebuah *board* berukuran 3x3. *Board* ini memiliki empat macam *tile*.

1. *Little-white Square* : Berpindah ke *tile* ini akan menambahkan *life* karakter sebanyak 5 poin, tetapi akan mengurangi *energy orb* sebanyak 5 poin pula.
2. *Black Square with White Borders* : Berpindah ke *tile* ini akan mengurangi *life* pemain sebanyak 500 poin.
3. *Black Square* : Berpindah ke *tile* ini akan memberikan pemain sebuah *energy orb*, tetapi akan mengurangi *life* pemain sebanyak 25 poin.
4. *Big-white Square* : Berpindah ke *tile* tidak akan memberi pengaruh apa-apa kepada pemain, tetapi pemain harus berada di *tile* ini untuk dapat melakukan *attack* kepada *dragon* jika pemain telah memiliki 10 buah *energy orb*.

Saat *dungeon* dimulai, karakter pemain akan berada di *Big-white Square* pada bagian tengah dari *board*. Pada setiap *turn*, pemain harus memilih untuk berpindah ke *tile* yang berada di kiri, atas, kanan, atau bawah pemain, atau melakukan *attack* jika pemain telah mengumpulkan 10 buah *energy orb*. Sebagai tambahan, pada setiap *turn*, pemain akan mendapatkan *damage* dari *dragon* sebanyak 30 sampai 60 point, tergantung dari *defense* karakter pemain. Pada setiap *turn* pula, *tile* akan berubah sesuai pola:

Little-white Square > *Black Square with White Borders* > *Black Square* > *Big-white Square* > *Little-white Square* > ...

Namun, *tile* dimana pemain berpindah tidak akan berubah sampai pergerakan pemain berikutnya.

Untuk melakukan *attack* terhadap *dragon*, pemain membutuhkan 10 buah *energy orb* dan harus berada pada *tile Big-white Square*. *Dragon* akan langsung mati ketika terkena *attack* dari pemain.

Untuk mengalahkan *dragon* di Dungeon 12, terdapat pula beberapa aturan dan tips.

1. Ketika pemain memulai *dungeon*, pemain akan berada di *board* yang jenis *tile* pada setiap sisinya *random*.
2. Jika karakter pemain berada di bagian tengah *board*, pemain dapat berpindah ke 4 arah. Jika karakter pemain berada pada pojok suatu *board* (bukan di sudut *board*), pemain hanya dapat berpindah ke 3 arah. Jika karakter pemain berada pada sudut *board*, pemain hanya dapat berpindah ke 2 arah.
3. *Little-white Square* merupakan pilihan *tile* yang baik untuk awal *battle* karena *energy orb* tidak dapat bernilai minus. Namun, ketika pemain telah mengumpulkan lebih dari 1 *energy orb*, *Little-white Square* merupakan salah satu pilihan terburuk karena pemain akan kehilangan banyak *energy orb* ketika berpindah ke *tile* ini. *Black Square with White Borders* juga merupakan salah satu pilihan terburuk karena *life* pemain akan berkurang cukup banyak ketika berpindah ke *tile* ini.
4. *Dragon* akan memberikan *damage* minimum kepada karakter pemain (yaitu sebesar 30 poin) jika karakter pemain memiliki *defense* sebesar 700 poin sehingga karakter pemain dianjurkan untuk memiliki *defense*

sebesar 700 poin atau lebih untuk memulai *dungeon* ini. Karakter pemain juga direkomendasikan untuk memiliki *life* sebesar 901 poin, yang merupakan *life* rata-rata untuk mengalahkan *dragon* di *dungeon* ini. Semakin tinggi *life* karakter, semakin besar pula kesempatan pemain untuk mengalahkan *dragon*.

III. Penerapan Algoritma Branch and Bound pada Dungeon 12

Pemecahan *puzzle* pada Dungeon 12 dapat dilakukan menggunakan Algoritma Branch and Bound. Dengan menggunakan algoritma ini, diharapkan program dapat mencari solusi untuk mendapatkan 10 buah *energy orb* dengan kerugian (kekurangan *life*) seminim mungkin dan keuntungan (sedikitnya jumlah *turn*) semaksimal mungkin. Berikut merupakan rincian pemecahan *puzzle* tersebut.

A. *Inisialisasi Persoalan*

Dalam merancang algoritma Branch and Bound untuk memecahkan *puzzle* Dungeon 12, pertama-tama, inisialisasi persoalan. Misalkan status awal dari *dungeon* adalah sebagai berikut.

Gambar 7. Status Awal Dungeon 12

Ambil ruang solusi persoalan X, dimana himpunan solusinya berupa arah gerak karakter pemain. X = (x_1, x_2, …, x_n), $x_i \in S_i$, $S_1 = S_2 = \ldots = S_n$, S_i = {*up, right, down, left*). *Assign* setiap *tile* dengan nilai tertentu, yaitu:

1. *Black Square* : 4
2. *Big-white Square : 3*
3. *Little-white : 2*
4. *Black Square with White Borders : 1*

Ambil nilai heuristiknya berupa nilai *tile* dimana pemain berpindah, berdasarkan nilai setiap *tile* yang telah di-*assign* sebelumnya. Target telah tercapai ketika pemain telah mengumpulkan 10 *energy orb*. Pada sisi pohon dituliskan arah gerak pemain. Pada setiap simpul dituliskan '*cost*' simpul tersebut, yaitu ĉ(i).

$$\hat{c}(i) = \hat{f}(i) + \hat{g}(i)$$

$\hat{c}(i)$ = ongkos untuk simpul i

$\hat{f}(i)$ = ongkos mencapai simpul i dari akar

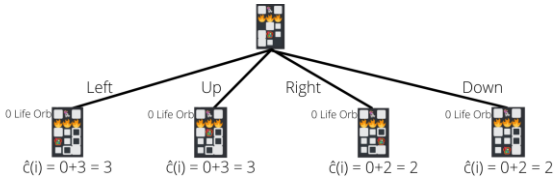
$\hat{g}(i)$ = ongkos mencapai simpul tujuan dari simpul i.

Gambar 8. *Cost* ĉ(i)

Selain itu, pada setiap simpul, dituliskan pula jumlah *energy orb* yang telah dikumpulkan.

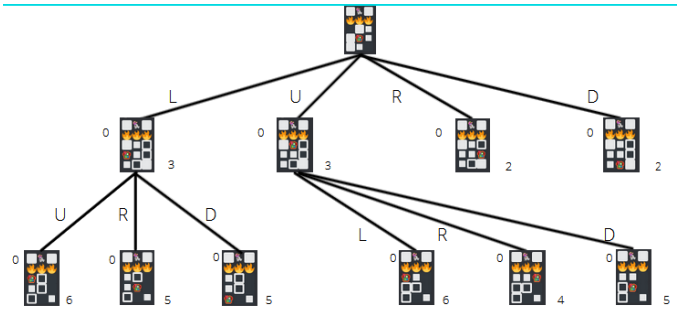
B. *Implementasi Algoritma Branch and Bound*

Berikut merupakan pohon status pencarian awal untuk pencarian solusi menggunakan Algoritma Branch and Bound.



Gambar 9. Pohon Status Pencarian Awal

Dapat dilihat dari pohon tersebut bahwa terdapat sisi yang merupakan arah gerak pemain, jumlah *energy orb* yang telah dikumpulkan, dan *cost* ĉ(i), yaitu jumlah dari nilai *tile* pada *board* akar dan *tile* yang ditempati setelah karakter pemain bergerak. Dari Gambar 9, dapat dilihat bahwa nilai *tile* pada *board* akar adalah 0 dan nilai heuristik *tile* berbeda-beda tergantung jenis *tile* yang ditempati karakter pemain setelah bergerak. Dari pohon status pencarian tersebut, kita dapat menentukan *board* yang diekspan, yaitu *board* yang memiliki *cost* maksimum di antara seluruh *board* yang ada pada simpul, yaitu *board* dimana karakter pemain bergerak ke arah kiri (*left*) dan atas (*up*). *Board* lain yang *cost*-nya lebih kecil akan 'dibuang'.



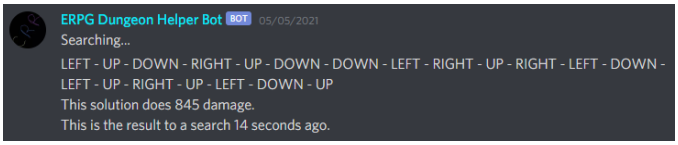
Gambar 10. Pohon Status Pencarian Kedua

Pada pohon status pencarian kedua, dapat dilihat bahwa arah gerak disimbolkan oleh L, U, R, D (*left, up, right, down*) dan pada kiri atas simpul terdapat jumlah *energy orb* yang telah dikumpulkan, sedangkan pada kanan bawah simpul terdapat *cost* dari perpindahan ke sebuah *tile*.

Lakukan pencarian tersebut dengan metode yang sama, berulang kali, sampai tercapai simpul tujuan, yaitu simpul dimana karakter berada di *Big-white Square* pada *board* dan akumulasi *energy orb* pada simpul tersebut sama dengan 10.

Gambar 11. Simpul Tujuan (*Target Node*)

Jika program telah diimplementasikan pada suatu bot lain yang kita punya, bot tersebut akan mencari solusi dari *dungeon* dan menampilkan solusi tersebut dalam bentuk arah gerak karakter. Bot juga akan menampilkan jumlah *life* yang akan berkurang dan waktu pencarian solusi.



Gambar 12. Contoh Solusi dari Dungeon 12

C. Rincian Algoritma

Berikut merupakan rincian tahap pencarian simpul tujuan untuk Dungeon 12 menggunakan Algoritma Branch and Bound.

1. *Assign* setiap sisi dengan arah gerak karakter pemain berdasarkan prioritas (*left*, *up*, *right*, *down*).
2. *Assign* setiap *tile* dengan beberapa nilai, tergantung jenis *tile*-nya.
3. *Assign* simpul pohon dengan gambar setiap *board* setelah pemain bergerak. Tambahkan pula akumulasi *energy orb* dan nilai *cost* ĉ(i) pada setiap simpul.
4. Masukkan simpul akar, yaitu posisi awal karakter pemain di *board*, ke dalam antrian Q.
5. Pilih simpul i dari antrian Q yang memiliki nilai *cost* ĉ(i) terbesar. Jika terdapat beberapa simpul i yang memiliki nilai *cost* ĉ(i) terbesar yang sama, pilih salah satu secara sembarang.
6. Jika simpul i adalah simpul tujuan (*goal node*), yaitu simpul yang akumulasi *life orb*-nya berjumlah 10 dan letak karakter pemain di *Big-white Square*, maka solusi telah ditemukan.
7. Jika simpul i bukan merupakan simpul tujuan (*goal node*), bangkitkan semua anak dari simpul i. Jika i tidak mempunyai anak, kembali ke langkah 2.
8. Untuk setiap anak *j* dari simpul *i*, hitung ĉ(j), dan masukkan semua anak tersebut ke dalam Q.
9. Kembali ke langkah 2.

KESIMPULAN

Dengan menggunakan Algoritma Branch and Bound, kita dapat mencari solusi dari *puzzle* Dungeon 12 dengan solusi optimal, yaitu jumlah *life* yang berkurang minimal dan jumlah *turn* yang dibutuhkan untuk mengalahkan *dragon* juga minimal. Program ini dapat diimplementasikan pada bot Discord yang kita punya.

LINK VIDEO DI YOUTUBE

https://youtu.be/D53WaCnRyo8

REFERENSI

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf
[2] https://www.techopedia.com/definition/27052/role-playing-game-rpg
[3] https://epic-rpg.fandom.com/wiki/EPIC_RPG_Wiki
[4] https://epic-rpg.fandom.com/wiki/Dungeon_12

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.

Jakarta, 10 Mei 2021

Daffa Ananda Pratama Resyaly - 13519107