Penerapan Algoritma *Backtracking* dalam Penyelesaian *Minigame* Genshin Impact "*Alchemical Ascension*"

Maximilian Sulistiyo - 13522061
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13522061@std.stei.itb.ac.id

***Abstrak*—Genshin Impact merupakan salah satu game action *open world* RPG yang sedang populer saat ini. Secara periodik, permainan ini menggelarkan sebuah *special event* dimana terdapat *minigame* tambahan yang dapat dimainkan. Salah satu *special event* yang pernah digelarkan adalah *event "Alchemical Ascension"* dimana pemain harus meletakkan bahan-bahan secara strategis untuk mendapatkan suatu *potion* dengan *efficacy* sesuai permintaan. Makalah ini mengeksplorasi penerapan algoritma *Backtracking* untuk memecahkan masalah optimisasi ini.**

Kata kunci—Genshin Impact; Alchemical Ascension; Backtracking;

I. PENDAHULUAN

Gambar 1 Logo Utama Genshin Impact
(Sumber [1])

Genshin Impact merupakan permainan *action open world* RPG yang dibuat oleh miHoYo dan dirilis pada tahun 2020 pada platform Windows, iOS, Android, dan Playstation 4. Cerita utama pada permainan ini mengisahkan mengenai seorang *traveller* yang tiba pada dunia baru bernama Tevyat dimana ia terpisah dengan saudaranya dan berusaha untuk bertemu kembali dengannya. Dalam petualangannya seorang *traveller* membantu kota-kota yang ia kunjungi untuk menyelesaikan permasalahan masing-masing kota. Tentu dalam membantu *traveller* juga harus melawan musuh yang mengancam. Agar *traveller* dapat mengalahkan musuh-musuh tersebut maka *traveller* harus melakukan *farming* dan *level up*. Dalam perjalanannya *traveller* didampingi oleh teman-temannya, namun agar kami pemain dapat memainkan teman-teman tersebut maka pemain harus melakukan *gacha*. *Gacha* sendiri dilakukan dengan menukarkan *currency* dalam game bernama *primogems*. *Primogems* ini dapat didapatkan pemain dengan cara memainkan permainan ini secara rutin seperti bermain setiap hari untuk mendapatkan *daily quest* atau juga melakukan *world quest* yang tersedia. Genshin Impact juga menawarkan cara lain untuk mendapatkan *primogems* dimana secara periodik akan dilaksanakan sebuah *event* yang memberikan hadiah *primogems*. Biasanya *event* yang digelarkan merupakan *minigame*. *Minigame* yang digelarkan bervariasi mulai dari melawan musuh, eksplorasi, atau menyelesaikan teka-teki.

Gambar 2 Poster *event Alchemical Ascension*

(Sumber [2])

Salah satu *event* yang pernah digelarkan oleh Genshin Impact adalah *special event* yang digelarkan pada patch 4.5 bernama *Alchemical Ascension*. Pada *event* ini pemain ditugaskan untuk meletakkan bahan-bahan secara strategis pada sebuah board agar mendapatkan *efficacy* tertentu pada *potion* yang dibuat. *Efficacy* yang dapat didapatkan dalam *minigame* ini adalah *wisdom, dexterity, strength, constitution, charisma, dan balance*. Dalam *efficacy* sendiri terdapat dua level yaitu *beginner, intermediate* dan *advanced* dimana perbedaan tingkat menandakan *efficacy* yang lebih dimana pada level *beginner* potion sudah dianggap memiliki *efficacy* dari tipe tertentu.

Gambar 3 Contoh permainan *Alchemical Acension*

(Sumber [3])

Pemain harus memenuhi *efficacy* yang sesuai dengan permintaan market. Oleh karena itu sebuah metodologi peletakkan bahan diperlukan agar dapat memenuhi permintaan. Penulis pun mengusulkan menggunakan algoritma *backtracking* yang telah diajarkan pada mata kuliah IF2211 Strategi Algoritma untuk menyelesaikan permasalahan tersebut

II. LANDASAN TEORI

2.1 *Backtracking*

Algoritma *Backtracking* merupakan perbaikan dari *exhaustive search* dimana pada *exhaustive search* semua kemungkinan solusi dieksplorasi dan diperiksa. Berbeda dengan itu pada *backtracking*, pilihan yang tidak mengarah terhadap solusi akan dihilangkan / tidak dianggap. Algoritma ini diperkenalkan oleh D.H Lehmer pada tahun 1950 dan kemudian oleh R.J Walker, Golomb, dan Baumert disajikan uraian umum mengenai algoritma *backtracking*.

Algoritma ini dapat dilihat sebagai salah satu dari dua hal berikut

1. Sebagai sebuah tahapan dalam algoritma *Depth First Search* yaitu algoritma pencarian mendalam pada sebuah *tree*
2. Sebagai sebuah *metode* pemecahan masalah yang mangkus, terstruktur, dan sistematis dalam penyelesaian masalah optimisasi maupun non-optimisasi

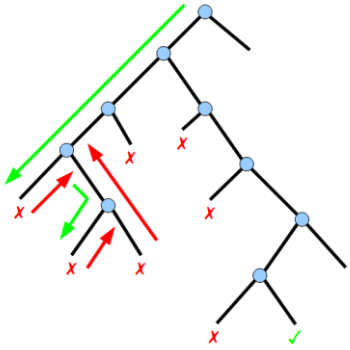
Pada algoritma ini juga terdapat beberapa ciri khas / properti yang umum yaitu:

1. Solusi Persoalan
    - Solusi dari permasalahan dinyatakan sebagai suatu vektor dengan *n-tuple*: X (x_1, x_2, x_3, … x_n), dimana $x_i \in S_i$. Dimana umumnya $S_1 = S_2 = \ldots = S_n$. Artinya vektor *n-tuple* solusi memiliki elemen yang merupakan anggota dari set *S* dimana secara umum semuanya memiliki set yang sama.
    - Contoh : Pada persoalan *1/0* knapsack $S_i = \{0,1\}$, $x_i = 0$ atau 1
2. Fungsi Pembangkit
    - Dinyatakan sebagai predikat T ()
    - Dari nilai-nilai sebelumnya maka *T(x[1], x[2], ..., x[k−1])* akan membangkitan semua nilai untuk x_k
3. Fungsi Pembatas
    - Dinyatakan sebagai predikat B (x_1, x_2, … x_k)
    - Jika B menghasilkan *true* maka (x_1, x_2, … x_k) mengarah terhadap solusi dan tidak melanggar *contraint* yang telah ditetapkan
    - Selebihnya jika B menghasilkan *true* maka akan dibangkitkan nilai untuk x_{k+1}, namun jika B menghasilkan *false* maka (x_1, x_2, … x_k) akan dibuang / tidak dianggap

Kemudian dalam penyelesaian masalah menggunakan *backtracking* maka semua kemungkinan dari solusi disebut ruang solusi atau *solution space*. Ruang solusi tersebut pun di organisasikan menjadi sebuah struktur *tree* dimana tiap *node* pada *tree* merupakan suatu status / *state* dari persoalan. Lintasan dari akar ke daun pun merupakan salah satu solusi yang mungkin dan seluruh lintasan dari akar ke daun membentuk ruang solusi. Pengorganisasian pohon ruang solusi disebut sebagai pohon ruang status / *state space tree*

Dalam pencarian solusi menggunakan *bactracking* memiliki beberapa prinsip yaitu

- Solusi dicari dengan cara membangkitkan semua *node* status hingga terbentuk suatu lintasan dari akar ke daun
- Aturan pembangkitan *node* yang dipakai mengikuti aturan pencarian *Depth First Search*
- Semua *node* yang telah dibangkitkan disebut *live node*
- *Live node* yang sedang diperluas disebut *expand-node*
- Setiap *expand-node* diperluas maka lintasan yang dibangun akan bertambah panjang
- Jika lintasan yang sedang ditelusuri tidak mengarah terhadap solusi maka *expand-node* dimatikan dan menjadi *dead-node*
- Fungsi yang digunakan untuk mematikan *expand-node* disebut fungsi pembatas
- Jika pembentukan lintasan berakhir pada suatu *dead-node* maka akan dilakukan *backtrack* dimana algoritma kembali kepada *node* sebelumnya
- Langkah-langkah ini diteruskan hingga algoritma menemukan *goal-node*



Gambar 4 Visualisasi Algoritma *bactracking*

(Sumber [4])

2.2 *Alchemical Ascension*

Seperti yang telah dijelaskan sebelumnya tujuan utama dari permainan ini ialah untuk mendapatkan *efficacy* tertentu berdasarkan permintaan. Dalam permainan ini pemain diberikan sebuah *cauldron* yang merupakan matriks berukuran *m*×*n* sebagai wadah untuk menempatkan bahan-bahan yang dimiliki. Pada *cauldron* ini juga terdapat beberapa petak yang tidak dapat ditempati oleh bahan-bahan. Setiap bahan yang ditempatkan tidak boleh menimpa satu sama lain, tidak boleh ada bagian yang keluar dari *cauldron*. Namun pada permainan ini diperbolehkan beberapa bahan untuk menimpa satu sama lain. Banyaknya petak yang boleh menimpa satu sama lain tergantung pada level dari pemain pada *event* ini.

Gambar 5 Visualisasi dari *Cauldron*

(Sumber [5])

Bahan-bahan yang digunakan memiliki *efficacy* nya masing-masing dimana setiap jika diletakkan di atas *cauldron* maka akan meningkatkan nilai dari *efficacy* yang bersangkutan. Nilai *efficacy* yang dapat diberikan oleh suatu bahan tergantung pada besarnya bahan tersebut, tidak mengambil bentuk sebagai faktornya. Bentuk dari bahan-bahan yang dapat diletakkan bervariasi. *Efficacy* yang terdapat dalam permainan ini adalah *wisdom, dexterity, strength, constitution, dan charisma*. Kemudian terdapat kategori bahan *balance* dimana jika meletakkan bahan ini maka nilai *efficacy* dari semua nilai akan naik secara merata

Gambar 6 Berbagai Bentuk Bahan

(Sumber [5])

Terdapat 3 level *efficacy* yang dapat dimiliki oleh potion yaitu *beginner, intermediate, dan advanced.* Jika memiliki nilai pada suatu *efficacy* maka sudah langsung dianggap sebagai level *beginner.* Untuk mendapatkan level-level selanjutnya harus memenuhi batas tertentu. Karena pada penulisan makalah ini *event* sudah tidak berlangsung maka penulis mengasumsikan berdasarkan tayangan permainan *event* ini yang beredar di internet dan memutuskan bahwa untuk mendapatkan level *intermediate* maka memerlukan point sebesar 10 dan untuk level *advanced* memerlukan point sebesar 20.

III. Implementasi Struktur Data dan Algoritma

3.1 Struktur *Cauldron*

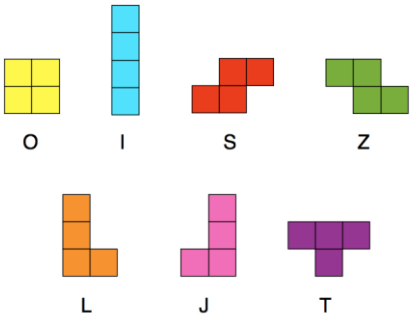
Tentunya kami perlu mendefinisikan struktur dari *cauldron* yang akan digunakan. Seperti yang sudah dijelaskan sebelumnya *cauldron* merupakan matriks berukuran *m*×*n*. Ukuran ini bervariasi berdasarkan level dari *cauldron*, untuk mempermudah penulis menggunakan asumsi bahwa level *cauldron* sudah maksimal sehingga memiliki ukuran *7*×*7*. Maka kami dapat membuat matriks *7*×*7* untuk merepresentasikan status *cauldron*. Setelah itu diperlukan sebuah list untuk mencatat *efficacy* dari *cauldron* pada saat itu, untuk memudahkan maka akan dibuat sebuah list dimana elemen pada indeks ke-i menandakan nilai *efficacy* 0 untuk *wisdom*, 1 untuk *dexterity*, 2 untuk *strength*, 3 untuk *constitution, dan* 4 untuk *charisma*.

Maka singkatnya struktur *node*

- *curr_cauldron*: matriks
- *curr_efficacy*: list of integer

3.2 Struktur Bahan

Setiap bahan memiliki bentuk dan juga nilai *efficacy* yang berbeda. Dikarenakan banyaknya dan ragam bentuk yang berkorespondensi dengan *efficacy* tertentu yang terdapat dalam game, penulis menyederhanakan permasalahan dengan membatasi bentuk yang digunakan yaitu bentuk *tetromino* yang merupakan bentuk yang sering digunakan pada permainan *Tetris*



Gambar 7 Bentuk-Bentuk Tetromino
(Sumber [6])

Tidak terbatas pada bentuk *tetromino* saja, penulis memutuskan untuk memilih bentuk lainnya seperti *block* 1x2 dan juga 3x1. Sekali lagi karena waktu makalah ini ditulis *event* sudah tidak dapat dimainkan maka penulis mengasumsikan bahwa setiap petak menandakan 1 point. Akibat ini dapat dibilang permainan ini menjadi 2D knapsack dimana kami mencari nilai tertinggi dari suatu nilai dengan susunan tertentu. Namun pada kasus ini kami tidak tertarik terhadap optimisasi melainkan susunannya saja agar susunan *cauldron* dapat mendapatkan tingkat *efficacy* tertentu. Urutan penempatan bahan pun sesuai urutan berikut O, I, S, Z, L, J, T, 3x1, 2x1.

3.3 Solusi Persoalan

Solusi dari persoalan ini ialah susunan letak bahan yang memenuhi *efficacy* yang diinginkan. Untuk mempermudah visualisasi *cauldron* maka akan dijadikan angka untuk *efficacy* berbeda dimana 1 menandakan *wisdom*, 2 menandakan bahan *dexterity*, 3 menandakan bahan *strength*, 4 menandakan bahan *constitution*, *dan* 5 menandakan bahan *charisma*.

3.4 Fungsi Pembangkit

Fungsi pembangkit pada algoritma ini akan mencoba menaruh bahan pada cauldron. Rotasi dari bahan pun akan dicoba untuk diletakkan. Peletakkan dari bahan-bahan ini pun akan dimulai dari pojok kiri atas. Syarat dari peletakkan pun ialah dipastikan bahan tidak keluar dari cauldron. Kemudian akan juga dipertimbangkan petak yang dapat menimpa satu sama lain dimana semua kombinasi akan diperiksa dimana tidak ada yang menimpa hingga *n* petak yang menimpa.

3.5 Fungsi Pembatas

Fungsi pembatas pada kasus ini adalah untuk memeriksa apakah point yang diinginkan dapat dicapai dengan kondisi *cauldron* pada status tersebut. Secara *heuristic* maka menghitung jumlah petak kosong yang tersedia dan memeriksa apakah jumlah *efficacy* saat ini ditambah jumlah petak kosong dapat mencapai nilai *efficacy* yang diinginkan.

3.6 Langkah-Langkah Penyelesaian

Setelah mendefinisikan solusi persoalan, fungsi pembangkit, dan fungsi pembatas maka langkah-langkah algoritma ini ialah

1. Menginisialisasi matriks dan juga menandakan petak mana yang tidak dapat di letakkan bahan
2. Mencoba menaruh bahan sesuai urutan yang sudah ditentukan, penempatan bahan akan meiterasi matriks dan mencoba segala rotasi untuk setiap bahan
3. Akan diletakkan terus menerus bahan hingga telah terletak semua, kemudian akan backtrack dan kemudian mencoba bahan dengan bentuk selanjutnya
4. Diperiksa apakah sudah mencapai *efficacy* yang diinginkan, jika sudah maka state saat ini akan di print dan kemudian melanjutkan untuk mencari solusi lain
5. Fungsi pembatas diimplementasikan dimana ketika jumlah petak yang masih kosong ditambah nilai *efficacy* saat ini kurang dari tingkat yang diinginkan, maka tidak mungkin menuju solusi dan dijadikan *dead-node*
6. Hal ini dilakukan hingga menemukan solusi sejumlah yang diinginkan pengguna

IV. Hasil Uji Coba Program

Berikut merupakan contoh-contoh hasil solusi dengan halangan yang telah ditentukan

Gambar 8 Contoh Program

(Sumber Dokumentasi Pribadi)

Untuk lebih lengkapnya maka dapat melihat kode lengkapnya pada link github yang sudah disediakan

V. KESIMPULAN

Penyelesaian dari *minigame event Alchemical Ascension* dapat diselesaikan menggunakan algoritma *backtracking* dimana dengan penggunaan algoritma ini kami para pemain *Genshin Impact* tidak perlu mencoba satu-satu namun tinggal menggunakan program telah dibuat pada makalah ini. Dengan ini juga diharapkan pemain *Genshin Impact* dapat mendapatkan manfaat dan juga belajar sedikit hal mengenai algoritma *backtracking* dalam membaca makalah ini. Perlu diperhatikan juga versi *minigame* yang dibuat oleh penulis merupakan versi yang sangat disederhanakan sehingga diperlukan peningkatan dari segi kemiripan dengan *minigame* asli dan juga peningkatan dari segi hal algoritma agar lebih mangkus dan sangkil.

LINK GITHUB

Github : https://github.com/riyorax/Makalah-Stima

ACKNOWLEDGMENT

Ucapan sebesar besarnya diberikan kepada para dosen pembimbing mata kuliah IF2211 akan bimbingan mereka selama satu semester selama pembelajaran mata kuliah ini. Ucapan juga diberikan kepada orang tua penulis yang selalu mendukung penulis. Terakhir ucapan kepada teman-teman penulis yang telah membantu penulis dalam pembuatan makalah ini.

REFERENCES

[1] https://www.facebook.com/photo/?fbid=2471223049824335&set=a.2382995648647076 diakses pada 11/06/2024

[2] https://genshin-impact.fandom.com/wiki/Alchemical_Ascension diakses pada 11/06/2024

[3] https://dotgg.gg/genshin-impact-alchemical-ascension-event-guide/ diakses pada 11/06/2024

[4] https://www.w3.org/2011/Talks/01-14-steven-phenotype/ diakses pada 11/06/2024

[5] https://www.youtube.com/watch?v=ViN4Kpmxj1Q diakses pada 11/06/2024

[6] https://markmliu.medium.com/the-tetris-proof-60a7a69a8e04 diakses pada 11/06/2024

[7] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf diakses pada 11/06/2024

[8] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian2.pdf diakses pada 11/06/2024

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, 12 Juni 2024

Maximilian Sulistiyo 13522061