Penerapan Algoritma *Branch and Bound* dalam Penentuan Rute Farming 600 Mora Tercepat pada Game Genshin Impact Versi 1.5

Nabil Nabighah 13519168
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail : 13519168@std.stei.itb.ac.id

***Abstrak*—Perkembangan teknologi sangatlah pesat saat ini. Sudah banyak produk yang dihasilkan karena teknologi salah satunya adalah permainan atau game. Game memiliki banyak jenisnya, salah satunya adalah game dengan jenis action RPG. Salah satu game yang memenuhi jenis tersebut adalah Game Genshin Impact. Pada game tersebut terdapat sebuah mekanik untuk melawan musuh yang akan memberikan pemain sejumlah uang dalam bentuk *in-game currency* yang disebut Mora. Untuk mengumpulkan mora tersebut diperlukan rute tercepat agar efisien. Dengan algoritma branch and bound dapat dicari rute tercepat tersebut.**

Keywords—Genshin Impact; Branch and Bound; Rute

I. PENDAHULUAN

Perkembangan zaman telah mempengaruhi berbagai bidang di kehidupan. Salah satunya adalah bidang hiburan. Banyak sekali hiburan yang menggunakan teknologi, contohnya adalah permainan atau *game*. Game memiliki banyak jenisnya. Salah satu yang banyak penggemarnya adalah game berjenis RPG (Role-Playing Game). Game jenis ini adalah game dengan mekanik yaitu pemain akan memainkan karakter di dalam game. Biasanya permainan dengan jenis ini akan disertai dengan cerita-cerita naratif.

Gambar 1 Genshin Impact Ganyu

Sumber : https://www.facebook.com/Genshinimpact

Sampai saat ini, game muncul dalam berbagai macam bentuk dan *platform* untuk memainkannya. Genshin Impact adalah salah satu game yang memiliki genre RPG *open world*. *Open world* yang berarti memiliki banyak cara untuk menyelesaikan suatu objektif pada game tersebut. Genshin Impact dirilis tanggal 28 September 2020. Game ini termasuk game yang populer saat ini. Game ini termasuk game *multi-platform* artinya game ini dapat dimainkan di *platform* manapun seperti Windows, Android, PlayStation 4, iOS, PlayStation 5, dan Nintendo Switch. Pada game ini pemain dapat melakukan banyak hal di dalamnya seperti menyelesaikan quest, melawan *mob*, menyelesaikan *dugeon*, melawan *boss*, mendesain sebuah tempat, dan berpetualang menjelajahi dunia tersebut.

Gambar 2 *Battle* Melawan *Elite Mob*

Sumber : Dokumen Penulis

Salah satu mekanik pada game ini adalah melawan *mob*. *Mob* adalah lawan yang dianggap sangat mudah untuk dilawan. Biasanya mob memiliki darah yang lebih sedikit dari lawan jenis lain. Ketika pemain berhasil menang melawan *mob* tersebut, pemain akan mendapatkan mora. Mora adalah mata uang pada game ini. Mora dapat digunakan untuk berbagai macam hal seperti membeli senjata, *item*, makanan, menaikkan level karakter yang dimainkan oleh pemain, menaikkan *skill* yang dimiliki sebuah karakter, *enhance* senjata, dan membuat *item*. Dari mekanik tersebut turunlah sebuah mekanik yang disebut farming. Farming adalah sebuah teknik yang dilakukan secara berulang-ulang untuk mendapatkan suatu hal di dalam game seperti uang (Mora). Setelah berhasil melawan *mob* beberapa *mob* akan men*drop* mora sejumlah 600. Mob tersebut adalah elite *mob* dan tersebar di beberapa wilayah pada game ini. Farming sangat

baik jika dilakukan secara efisien. Farming efisien haruslah melihat beberapa faktor, salah satunya yang paling penting adalah faktor waktu. Karena *mob* ini tersebar diberbagai wilayah maka diperlukan rute tercepat agar dapat melawan *mob* ini dan mengumpulkan mora. Rute tercepat ini dapat diperoleh dengan menggunakan algoritma *branch and bound*. Rute dimulai dari sebuah wilayah ke wilayah lainnya dengan tepat hanya satu kali.

II. Landasan Teori

A. *Algoritma Branch and Bound*

Algoritma branch and bound atau BnB adalah sebuah desain dari algoritma yang digunakan untuk memecahkan persoalan optimisasi. Persoalan optimisasi adalah persoalan yang meminimalkan atau memaksimalkan tanpa melanggar suatu batasan (*constraints*) persoalan tersebut. Pada algoritma BnB membutuhkan beberapa hal yaitu, nilai dari solusi terbaik sejauh ini dan sebuah jalan yang membuktikan untuk setiap simpul pada pohon ruang status diperlukan suatu cara untuk menentukan batas (bound) nilai terbaik fungsi objektif pada setiap solusi yang mungkin, dengan menambahkan komponen pada solusi sementara yang direpresentasikan oleh simpul. Pada algoritma BnB, sama seperti algoritma backtracking yaitu melakukan "pemangkasan" terhadap jalur yang dianggap tidak mengarah ke solusi. Kriteria pemangkasan secara umum yaitu :

1. Nilai simpul tidak lebih baik dari nilai terbaik sejauh ini
2. Simpul dianggap tidak merepresentasikan solusi yang 'feasible' karena terdapat batasan yang dilanggar
3. Solusi pada simpul tersebut hanya terdiri atas satu titik. Artinya tidak ada pilihan lain.

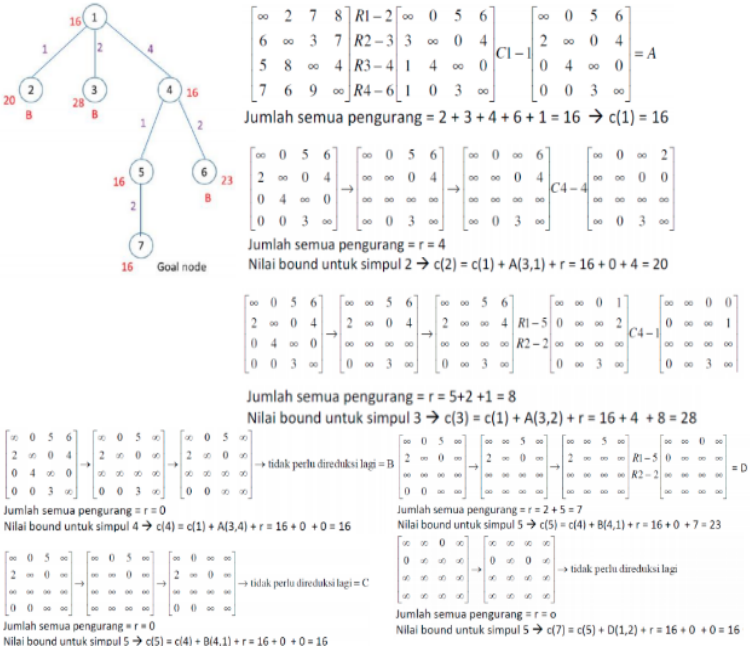
Langkah pengerjaan algoritma BnB pada makalah ini adalah sebagai berikut.

1. Hitung cost simpul akar
2. Apabila simpul adalah solusi maka stop
3. Masukkan anak-anak dari simpul tersebut kedalam antrian prioritas (Priority Qeueu) dengan menghitung seluruh cost pada setiap simpul anak.
4. Ambil simpul dengan cost terkecil dari antrian prioritas kemudian cek apakah solusi atau bukan
5. Ulangi langkah 2

B. *Reduced Cost Matrix*

Reduced cost matrix atau matriks ongkos tereduksi adalah matriks yang elemen-elemennya berisi ongkos (cost) atau nilai yang tereduksi. Tereduksi dalam hal ini berarti setiap baris dan kolom pada matriks ongkos tersebut memiliki paling sedikit satu buah angka nol dan elemen-elemen pada matriks tersebut semunya elemen non-negatif.

Pada makalah ini, algoritma Branch and Bound dipakai untuk menyelesaikan masalah travelling salesman problem (TSP). Perlu diketahui hal yang paling penting pada algoritma Branch and Bound adalah penentuan cost, maka ada beberapa cara menentukan cost algoritma Branch and Bound untuk persoalan TSP. Beberapa cara tersebut antara lain menggunakan matriks ongkos tereduksi, dan bobot minimum tur lengkap. Pada makalah ini penentuan cost akan menggunakan metode matriks ongkos-tereduksi.



Gambar 3 Metode matriks ongkos tereduksi

Sumber :
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branchand-Bound-2021-Bagian2.pdf

Misalkan A adalah matriks tereduksi untuk simpul R. Sedangkan S adalah anak dari simpul R sehingga sisi (R, S) pada pohon ruang status berkoresponden dengan sisi (i, j) pada perjalanan. Langkah-langkah menghitung matriks ongkos-tereduksi adalah sebagai berikut.

1. Jika S bukan simpul daun maka
2. Ubah semua nilai pada baris i dan kolom j menjadi ∞.
3. Ubah A(j, 1) menjadi ∞.
4. Reduksi kembali semua baris dan kolom pada matriks A kecuali untuk elemen.
5. Hitung nilai r dimana r adalah total semua pengurang ketika mereduksi matriks.
6. Hitung cost pada simpul tersebut

C. *Travelling Salesman Problem (TSP)*

Persoalan travelling salesman problem atau TSP adalah sebuah persoalan yang termasuk persoalan yang melibatkan proses optimasi. Persoalan ini melibatkan permasalahan maksimum dan minimum. TSP adalah persoalan mengenai penentuan sebuah rute yang minimum sedemikian sehingga tempat - tempat yang dilaluinya tepat hanya sekali kemudian kembali ke tempat asal keberangkatan.

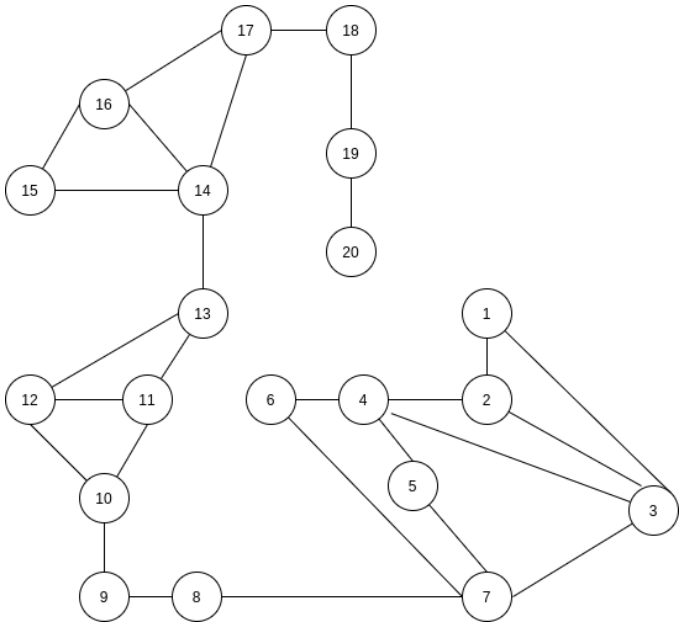
III. Penerapan Algoritma dan Pengujian

A. Membuat Graf Tidak Berarah

Gambar 4 Sebagian Peta Game Genshin Impact

Sumber : Dokumen Penulis

Gambar 4 menunjukkan sebagian kecil dari peta pada game Genshin Impact versi 1.5. Pada gambar tersebut terdapat banyak titik berwarna merah yang menandakan simpul - simpul yang akan digunakan untuk graf berarah. Titik berwarna biru menunjukkan bahwa rute dimulai dari titik tersebut. Dari gambar tersebut diperoleh graf berarah seperti berikut



Gambar 5 Graf Tak Berarah Rute Farming

Sumber : Dokumen Penulis

Penomoran pada graf tak berarah menyatakan nomor simpul - simpul pada gambar 4, simpul 1 menunjukkan simpul awal yang pada gambar 4 ditunjukkan dengan titik berwarna biru. Graf tersebut tidak semua simpul terdapat sisi untuk ke setiap simpul lainnya karena penulis mempertimbangkan waktu yang dibutuhkan terlalu besar sehingga simpul tersebut dapat dikatakan tidak bertetangga dengan simpul yang jauh.

B. Membuat Matriks Ongkos

Perhitungan ongkos dari simpul satu ke simpul lainnya dengan menggunakan waktu. Satuan waktu yang digunakan adalah satuan detik. Waktu diukur dari sejak karakter bergerak hingga karakter melakukan damage terhadap musuh pertama kali. Apabila waktu yang didapatkan terlalu lama atau lebih besar dari 70 detik maka jarak simpul satu ke simpul lainnya adalah tak hingga atau bisa dikatakan simpul satu ke simpul lainnya tidak bertetangga. Dengan metode seperti ini, diperoleh matriks ongkos awal adalah sebagai berikut.

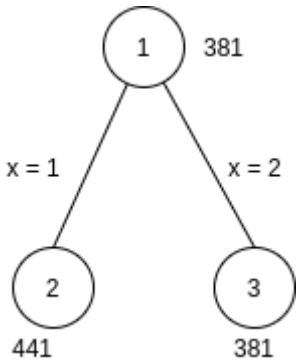
∞	10	60	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	60	20	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
70	10	∞	20	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	10	60	∞	25	20	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	20	∞	17	27	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	20	25	∞	17	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	25	20	∞	27	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	17	∞	25	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	25	∞	20	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	20	∞	20	18	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	20	∞	20	30	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	20	20	∞	30	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	20	18	∞	18	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	18	∞	18	30	40	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	18	∞	30	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	30	18	∞	10	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	18	∞	10	∞	14	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	14	∞	21	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	21	∞	18
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	18	∞

sehingga dapat diperoleh matriks ongkos reduksi untuk simpul awal atau simpul akar adalah sebagai berikut.

∞	0	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
0	∞	0	7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
60	0	∞	7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	0	0	∞	10	10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	0	∞	0	10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	0	3	∞	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	0	0	∞	2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	0	∞	6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	0	∞	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	0	∞	2	0	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	0	∞	0	10	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	0	0	∞	10	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	2	0	∞	0	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	∞	0	12	22	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	∞	12	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	20	8	∞	0	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	8	∞	0	∞	1	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	∞	7	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	∞	0
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	∞

Dari matriks ongkos reduksi untuk simpul akar terebut, diperoleh ongkos untuk simpul akar sebesar 381. Dari matriks

ongkos tereduksi untuk simpul akar tersebut dapat diperoleh pohon ruang status untuk awal yaitu sebagai berikut.



Gambar 6 Pohon Ruang Status Simpul Awal

Sumber : Dokumen Penulis

Perlu diketahui bahwa nomor pada gambar 6 bukanlah menyatakan simpul tetapi menyatakan urutan langkah. X pada gambar 6 menunjukkan index simpul jadi ketika X = 1 itu berarti simpul x+1 . Dari gambar 6, Terihat jelas bahwa cost anak yang paling kecil adalah simpul ke 3. sehingga pasti jawaban program harus terdapat jalur dari simpul 1 ke simpul 3.

C. Design Algoritma Branch and Bound

Penulis mendesign algoritma branch and bound dengan sebuah array simpul hidup dan sebuah array simpul ekspand. Array simpul hidup berisi seluruh anak-anak dari simpul ekspand. Array simpul hidup diimplementasikan dengan menggunakan arrray PriorityQueue. Branch and Bound akan menentukan cost terkecil dari simpul. Sebagai contoh kasus menggunakan gambar 6, simpul akar memiliki cost 381 kemudian simpul anaknya memiliki cost 441 dan 381. Algoritma branch and bound akan memiliki anak dengan cost 381 dan memasukkannya kedalam array fungsi ekspand. Kemudian anak dengan cost 381 menghasilkan anak kembali dan menghitung cost anaknya begitu seterusnya hingga diperoleh simpul solusi. Ketika sampai ke simpul solusi simpul-simpul dari akar hingga ke simpul solusi tersebut adalah jalur atau jawaban untuk algoritma branch and bound ini.

D. Pembuatan Program

Persoalan rute farming ini memiliki banyak simpul sehingga memerlukan program untuk menghitung cost setiap simpul dengan menggunakan matriks ongkos tereduksi. Penulis membuat program ini dalam bahasa Python. Simpul - simpul pada graf disimpan dalam sebuah class bernama class Node. Isi dari kelas node adalah atribut dan metode. Atribut kelas node antara lain matrix, cost, dan array jalur. Atribut matriks berguna untuk menyimpan matriks tereduksi dari sebuah simpul, atribut cost berguna untuk menyimpan cost simpul tersebut, begitu juga atribut array jalur untuk menyimpan jalur simpul dimulai dari akar hingga ke simpul tersebut. Terdapat beberapa metode untuk menunjang algoritma ini. Pada kelas Node terdapat metode reduceKolom dan reduceBaris keduanya memiliki fungsi yang sama yaitu untuk mengubah baris dan kolom menjadi baris dan kolom yang tereduksi. Terdapat pula fungsi bnb yaitu fungsi algoritma bnb dan fungsi costSimpul untuk membuat matriks tereduksi dari simpul dan menghitung cost dari matriks tersebut.

```
class Node(object):
    def __init__(self,matrix,cost,jalur):...
    def __lt__(self, other):...
    def __le__(self, other):...
    def __eq__(self, other):...
    def __ne__(self, other):...
    def __gt__(self, other):...
    def __ge__(self, other):...
    def getBaris(self):...
    def getKolom(self):...
    def isReduced(self):...
    def reduceKolom(self):...
    def reduceBaris(self):...
    def setJalur(self,jalur):...
    def getMatrix(self):...
    def getCost(self):...
    def getJalur(self):...
```

Gambar 7 Kelas Node pada Program Python

Sumber : Dokumen Penulis

```
def costSimpul(tempNode):
    idxakhir = tempNode.jalur[len(tempNode.jalur)-1]
    idxkeduaakhir = tempNode.jalur[len(tempNode.jalur)-2]
    elementCostMatrix = tempNode.matrix[idxkeduaakhir][idxakhir]
    #(a)
    for kolom in range(tempNode.getBaris()):
        tempNode.matrix[idxkeduaakhir][kolom] = inf
    for baris in range(tempNode.getBaris()):
        tempNode.matrix[baris][idxakhir] = inf
    #(b)
    # isiidxakhir = tempNode.jalur[idxakhir]
    # isiidxkeduaakhir = tempNode.jalur[idxkeduaakhir]
    tempNode.matrix[idxakhir][idxkeduaakhir] = inf
    #(c)
    if(not(tempNode.isReduced())):
        tempNode.reduceBaris()
        tempNode.reduceKolom()
    tempNode.cost += elementCostMatrix
```

Gambar 8 Fungsi costSimpul

Sumber : Dokumen Penulis

```
def bnb(simpulTemp,simpulHidup):
    #buat anak
    idxakhir = len(simpulTemp.jalur)-1
    baris = simpulTemp.jalur[idxakhir]
    for kolom in range(simpulTemp.getKolom()):
        if(simpulTemp.matrix[baris][kolom] != inf):

            #buat Node baru masukkan ke simpul hidup
            tempMatrix = copy.deepcopy(simpulTemp.getMatrix())
            tempCost = simpulTemp.getCost()
            tempJalur = copy.deepcopy(simpulTemp.getJalur())
            tempJalur.append(kolom)
            tempNode = Node(tempMatrix,tempCost,tempJalur)
            tempJalur = []
            costSimpul(tempNode)
            simpulHidup.put(tempNode)
```

Gambar 9 Fungsi Algoritma Branch and Bound

Sumber : Dokumen Penulis

Fungsi program bnb ini untuk mencari anak-anak pada simpul ekspand kemudian memasukkannya kedalam simpul hidup terurut secara membesar berdasarkan cost menggunakan array Priority Queue.

```
#simpul akar/cost awal
simpul = Node(matrixTemp,0,[0])
if(not(simpul.isReduced())):
    simpul.reduceBaris()
    simpul.reduceKolom()
simpulHidup.put(simpul)
jalurJawaban = simpul.jalur
```

Program akan menghitung terlebih dahulu cost pada simpul akar kemudian memasukkan simpul akar tersebut kedalam simpul hidup.

Gambar 10 Program Mengatasi Simpul Akar

Sumber : Dokumen Penulis

```
while(not(simpulHidup.empty()) and len(jalurJawaban) < simpul.getBaris()):
    #ambil simpul hidup yg pertama pada simpul Hidup
    simpulTemp = simpulHidup.get()
    jalurJawaban = copy.deepcopy(simpulTemp.jalur)
    costJawaban = simpulTemp.cost
    bnb(simpulTemp,simpulHidup)
akar = jalurJawaban[0]
jalurJawaban.append(akar)
for i in range(len(jalurJawaban)):
    if(i != len(jalurJawaban)-1):
        print(jalurJawaban[i]+1, end=" -> ")
    else:
        print(jalurJawaban[i]+1)
print("cost total : ", costJawaban)
```

Gambar 11 Program Driver Branch and Bound

Sumber : Dokumen Penulis

Program akan melakukan traversal secara terus menerus hingga suatu saat ketika simpul hidup kosong atau ketika simpul jawaban banyaknya sama dengan jumlah simpul pada graf. Apabila telah setelah traversal maka program akan menampilkan jalurnya.

E. Pengujian Program

Cara kerja program :

1. Masukkan nama file txt yang berisi matriks ongkos yang telah dibuat secara manual pada gambar 12 matriks menggunakan spasi sebagai pembeda baris dan kolom dan menggunakan ongkos -1 sebagai penanda bahwa jarak antara simpul tersebut ke simpul lainnya sangat jauh sehingga dapat dibilang infinity atau tak terhingga.

```
-1 10 60 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
10 -1 60 20 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
70 10 -1 20 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 10 60 -1 25 20 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 20 -1 17 27 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 20 25 -1 17 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 25 20 -1 27 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 17 -1 25 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 25 -1 20 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 20 -1 20 18 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 20 -1 20 30 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 20 20 -1 30 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 20 18 -1 18 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 18 -1 18 30 40 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 18 -1 30 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 30 18 -1 10 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 18 -1 10 -1 14 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 14 -1 21 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 21 -1 18
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 18 -1
```

```
kali@kali:~/MakalahStima$ /usr/bin/python3 /home/kali/MakalahStima/main.py
nama file :matrixCost.txt
```

Gambar 12 Contoh matriks ongkos

Sumber : Dokumen Penulis

2. Program akan mulai melakukan traversal sesuai dengan program python pada gambar 11.
3. Setelah selesai maka program akan menampilkan rute tercepat tersebut.

```
1 -> 3 -> 2 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10
-> 12 -> 11 -> 13 -> 14 -> 15 -> 16 -> 17 -> 18 -> 19 -> 20
cost total :  436
Waktu eksekusi :  0.0887753963470459 detik
```

Gambar 13 Hasil Eksekusi Program

Sumber : Dokumen Penulis

Dari gambar 13 dapat dilihat bahwa algoritma yang telah dibuat memberikan hasil jalur rute tercepat yaitu jalur dari simpul 1 -> 3 -> 2 -> 4 -> 5-> 6 -> 7 -> 8 -> 9 -> 10 -> 12 -> 11 -> 13 -> 14 -> 15 -> 16 -> 17 -> 18 -> 19 -> 20. Rute tersebut memberikan cost total sebanyak 436.

Program algoritma branch and bound yang telah dibuat dalam bahasa Python telah cukup mangkus dalam segi kompleksitas waktu untuk simpul berukuran 20 simpul daripada algoritma - algoritma lainnya seperti *brute force*. Hal ini karena pada algoritma branch and bound jumlah eksekusi yang dilakukan lebih sedikit dibandingkan algoritma *brute force*.

Program yang telah dibuat masih memiliki kelemahan, salah satunya adalah simpul yang ingin menjadi akar hanyalah simpul nomor 1 sehingga simpul yang ingin menjadi akar perlu diletakkan pada index 0 di file txt. Apabila ingin mengganti nomor simpul awal program tersebut perlu diganti pada bagian simpul akar. Data pada program ini masih memiliki kelemahan karena di game tersebut ada mekanisme *sprint* yang berarti dapat menambah kecepatan lari dalam beberapa waktu dan *gliding* yaitu membuat karakter terbang menggunakan sebuah sayap. *sprint* dan *gliding* memiliki batas

waktu yang berbeda-beda setiap pemain karena sprint dan *gliding* memerlukan stamina yang setiap pemain berbeda-beda jumlah staminanya sesuai dengan level pemain sehingga data yang digunakan mungkin masih memiliki beberapa tidak cocokan dengan kenyataannya. Perlu diketahui juga bahwa game Genshin Impact adalah game yang selalu diperbaharui setiap beberapa bulan sekali sehingga bisa jadi dimasa depan akan ada penambahan simpul - simpul untuk farming 600 mora ini.

IV. Kesimpulan

Algoritma branch and bound dapat mencari pencarian rute tercepat berserta costnya dengan menggunakan metode matriks ongkos tereduksi. Diperoleh rute tercepat untuk melakukan farming pada game Genshin Impact adalah sebagai berikut.

Gambar 14 Rute Pada Map Genshin Impact Sesuai Hasil yang diperoleh

Secara keseluruhan algoritma branch and bound dapat menghitung jarak tercepat untuk farming 600 mora. Tetapi algoritma ini mungkin saja bukan algoritma terbaik untuk menghitung persoalan optimasi karena mungkin ada algoritma lain yang lebih baik untuk menyelesaikannya.

Video Link at Youtube

Video mengenai demo masalah pada makalah ini dapat dilihat pada link berikut https://youtu.be/h0ZA6VJEeTQ

Ucapan Terima Kasih

Penulis mengucapkan puji dan syukur kepada Allah Azza wa Jalla atas kehendaknya penulis dapat menyelesaikan karya tulis ini. Selain itu, penulis mengucapkan terima kasih kepada Bapak Rinaldi Munir dan seluruh dosen mata kuliah Strategi Algoritma IF2211 Semester 2 2020/2021 yang telah membimbing penulis untuk menyelesaikan karya tulis ini. Tak lupa penulis mengucapkan terima kasih kepada teman - teman penulis yang telah menyemangati penulis dalam penyusunan karya tulis ini.

Referensi

[1] Levitin, "Intriduction to the Design and Analysis of Algorithms", 2011

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branchand-Bound-2021-Bagian2.pdf

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf

[4] https://genshin-impact-map.appsample.com/#/

[5] https://genshin-impact.fandom.com/

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, 11 Mei 2021

Nabil Nabighah 13519168