Penerapan Algoritma *Branch and Bound* Dalam Penentuan Rute Terpendek Dalam Permainan *Apex Legends*.

Iqbal Naufal - 13518074
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): iqbalph17@gmail.com

***Abstrak*—Permainan merupakan bagian dari keseharian yang dijalani oleh masyarakat saat ini. Permainan dilakukan sebagai kegiatan untuk melepaskan penat dari kesibukan sehari-harinya. Salah satu permainan yang sedang populer saat ini adalah *Apex Legends*. *Apex Legends* adalah sebuah permainan *FPS* (*First Person Shooter*) dengan sistem *Battle Royale*. Pada permainan ini, peta dari permainan yang disediakan sangat luas. Untuk mengelilingi semua tempat dengan rute terpendek dapat dicari menggunakan algoritma *Branch and Bound*.**

Kata Kunci—Permainan; Rute; Branch and Bound; TSP; FPS

I. PENDAHULUAN

Masyarakat saat ini berada pada kondisi yang tidak stabil. Masyarakat berada pada kondisi tersebut dikarenakan adanya PSBB (Pembatasan Sosial Berskala Besar). PSBB ini tentu dilaksanakan dengan tujuan dan maksud yang bermanfaat bagi masyarakat. Namun dikarenakan adanya PSBB, jadwal beberapa masyarakat menjadi lebih padat namun menyisakan banyak waktu luang. Banyak bagian dari masyarakat yang menggunakan sebagian dari waktu luang tersebut untuk melepaskan stresnya lewat permainan atau *Game*.

Permainan atau *Game* dilakukan oleh pemainnya dalam berbagai macam alasan. Ada yang melakukannya sebagai profesi dan ada juga yang melakukannya hanya untuk bersenang-senang. Permainan yang telah tersebar di dunia ini terdapat banyak sekali jenis atau genre, yang salah satu paling terkenal adalah FPS (*First Person Shooter*). Permainan FPS ini adalah jenis permainan dimana pemain dalam perspektif orang pertama dan menembak musuh dengan senjata api.

Salah satu sistem permainan FPS adalah *Battle Royale*. *Battle Royale* adalah sistem permainan FPS dimana pemainnya akan dimasukkan ke dalam sebuah peta permainan dan akan dikompetisikan dengan pemain lain. Pada peta permainan pemain akan mengelilingi peta permainan untuk mencari barang-barang yang akan membantunya dalam melawan pemain lain. Pada peta tersebut juga pemain dapat bertemu dengan lawan kapan saja dan dimana saja. Pemain dikatakan menang apabila pemain menjadi pemain terakhir yang masih hidup di peta tersebut.

Permainan zaman sekarang kini muncul dalam berbagai macam bentuk. Permainan sendiri sekarang terdapat pada berbagai macam *platform* dan salah satu yang sangat umum digunakan adalah *platform* komputer. Sudah jutaan atau bahkan miliaran permainan yang sudah terdistribusi di dunia ini dan salah satu yang populer adalah permainan *Apex Legends*.

Permainan *Apex Legends* ini merupakan salah satu permainan berjenis FPS dengan sistem *Battle Royale* yang sangat populer dikalangan masyarakat. Permainan ini seperti yang dijelaskan pada sistem *Battle Royale* adalah permainan dimana pemain menjadi satu-satunya yang bertahan hidup pada pertandingan saat itu. Pada pertandingan tersebut, pemain akan bertemu dengan berbagai macam kondisi tempat dan musuh yang merupakan pemain lawan juga dan harus menyesuaikan diri. Pemain akan mendapatkan keunggulan dari musuh apabila pemain mendapatkan barang-barang yang lebih bagus dibandingkan
musuh.

Gambar 1. Gambar promosi permainan *Apex Legends*.
Sumber : https://www.ea.com/games/apex-legends/play-now-for-free
Waktu Akses : 3 Mei 2020 pukul 13:27 WIB.

Pada permainan, pemain akan dibawa dengan sebuah pesawat dengan lintasan yang berupa garis lurus diatas peta permainan. Setelah itu, pemain dapat memilih untuk terjun dari pesawat pada tempat yang ditentukannya. Pada tempat yang ditentukannya tersebut, pemain dapat langsung mencari barang-barang yang akan digunakan untuk melawan musuh seperti baju zirah, helm, tas, perisai, dan tentu senjata api beserta pelurunya.

Namun peta yang terdapat pada permainan ini sangat luas sehingga pemain seringkali tidak mendapatkan barang-barang yang dibutuhkannya sebelum bertemu musuh yang mengakibatkan pemain mati terlebih dahulu. Solusi dari masalah tersebut dapat diselesaikan dengan rute terpendek dari keliling seluruh tempat untuk mencari barang-barang pada peta. Rute terpendek dapat dicari menggunakan algoritma *Branch and Bound* (BB).

II. LANDASAN TEORI

A. *Algoritma Branch and Bound (BB)*

Algoritma *Branch and Bound* (BB) adalah sebuah algoritma optimasi yang digunakan pada struktur graf. Algoritma ini pertama kali di usulkan oleh A. H. Land dan A. G. Doig pada tahun 1960 untuk *discrete programming*. Algoritma ini digunakan untuk mengoptimasi sebuah fungsi objektif, baik meminimalkan ataupun memaksimalkan, dengan tidak melanggar sebuah batasan dari persoalan.

Algoritma ini berdasarkan pada algoritma BFS (*Breadth First Search*) dengan penambahan elemen *least cost search*. Pada dasarnya algoritma BB ini terdapat 2 perbedaan dengan algoritma BFS yaitu, setiap simpul diberi *cost* dengan sebuah nilai yang menjadi taksiran lintasan termurah ke simpul tujuan dari simpul tersebut dan simpul yang kemudian di-*expand* tidak berdasarkan urutan pembangkitan melainkan berdasarkan *cost* dari tiap simpul yang telah di tentukan.

Algoritma global dari algoritma BB ini adalah :

1. Masukkan simpul akar ke dalam antrian Q. Jika simpul akar merupakan solusi, maka solusi telah ditemukan. Stop.
2. Jika Q kosong, tidak ada solusi. Stop.
3. Jika Q tidak kosong, keluarkan simpul yang berada di antrian Q dengan nilai *cost* terkecil. Jika terdapat beberapa, ambil secara acak.
4. Jika simpul yang dikeluarkan merupakan simpul solusi, maka solusi telah ditemukan. Stop. Jika simpul yang dikeluarkan bukan simpul solusi, maka bangkitkan anak-anak simpulnya. Jika simpul yang dikeluarkan tidak mempunyai anak, kembali ke langkah 2.
5. Untuk setiap anak dari simpul yang dikeluarkan, hitung *cost* dari simpul-simpulnya dan masukkan ke dalam antrian berdasarkan urutan *cost* yang telah dihitung.
6. Kembali ke langkah 2.

B. *Travelling Salesman Problem (TSP)*

Penentuan rute terpendek merupakan sebuah masalah yang sangat sering ditemukan pada kehidupan sehari-hari. Masalah ini merupakan sebuah kategori masalah tersendiri yang disebut *Travelling Salesman Problem* (TSP). Masalah ini merupakan sebuah masalah yang dapat diselesaikan dengan berbagai macam algoritma yang salah satunya adalah algoritma *Branch and Bound.* Masalah ini merupakan sebuah masalah dimana solusinya adalah rute terpendek dengan batasan semua lokasi yang tersedia harus sudah dikunjungi tepat sekali kemudian kembali ke lokasi awal. Rute terpendek bukan hanya jarak yang minimum, namun dapat berbentuk biaya yang minimum.

C. *Matriks ongkos-tereduksi*

Pada pengerjaan solusi dari masalah TSP, terdapat 2 cara pengerjaan algoritma BB. Salah satu caranya adalah matriks ongkos-tereduksi. Matriks ongkos-tereduksi adalah, sesuai dengan namanya, matriks ongkos yang di reduksi. Matriks tereduksi adalah matriks dimana seluruh baris dan kolomnya memiliki minimal 1 buah 0 dan semua elemen lainnya non-negatif.

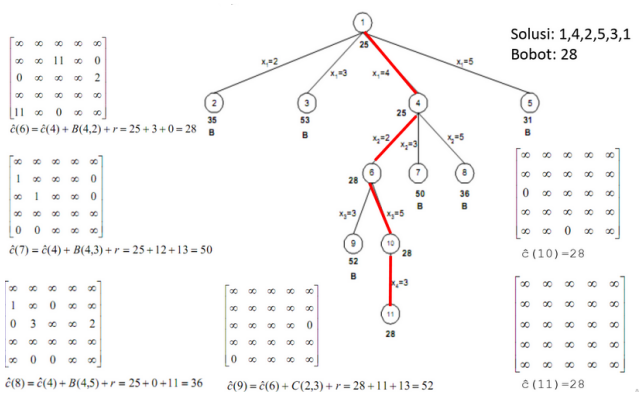
$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix} \begin{matrix} R_1 - 10 \\ R_2 - 2 \\ R_3 - 2 \\ R_4 - 3 \\ R_5 - 4 \end{matrix} \begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 13 & \infty & 14 & 2 & 0 \\ 1 & 3 & \infty & 0 & 2 \\ 16 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \end{bmatrix}$$

$$\begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 13 & \infty & 14 & 2 & 0 \\ 1 & 3 & \infty & 0 & 2 \\ 16 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \end{bmatrix} \begin{matrix} \\ \\ C_1 - 1 \\ C_3 - 3 \\ \end{matrix} \begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix} = A$$

Gambar 2. Langkah mereduksi matriks.
Sumber : http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Branch-&-Bound-(2018).pdf
Waktu Akses : 3 Mei 2020 Pukul 14:55 WIB.

Misalkan A adalah matriks yang tereduksi untuk simpul R, S adalah anak dari simpul R sehingga sisi (R, S) pada pohon ruang status terdapat pada sisi (i, j) pada perjalanan. Langkah-langkah pengerjaannya adalah :

1. Buat matriks ongkos menjadi matriks tereduksi A dan hitung jumlah pengurang setiap baris dan kolom. Jumlah tersebut menjadi *cost* simpul akar.
2. Jika S bukan simpul solusi, lanjut ke langkah 3.
3. Ubah semua nilai pada baris i dan kolom j menjadi ∞.
4. Ubah A(j, i) menjadi ∞.
5. Ubah matriks tersebut menjadi matriks tereduksi dan hitung jumlah pengurang.
6. Misalkan r adalah jumlah pengurang, cost dari simpul S adalah cost simpul sebelumnya ditambah r ditambah nilai pada A(i, j).
7. Ulang hingga seluruh simpul solusi optimum ditemukan.
8. Jika ada simpul yang bukan merupakan simpul solusi namun costnya sudah melebihi cost simpul solusi, potong simpul tersebut.



Gambar 3. Langkah pengerjaan hingga solusi Matriks tereduksi
Sumber : http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Branch-&-Bound-(2018).pdf
Waktu Akses : 3 Mei 2020 Pukul 15:36 WIB.

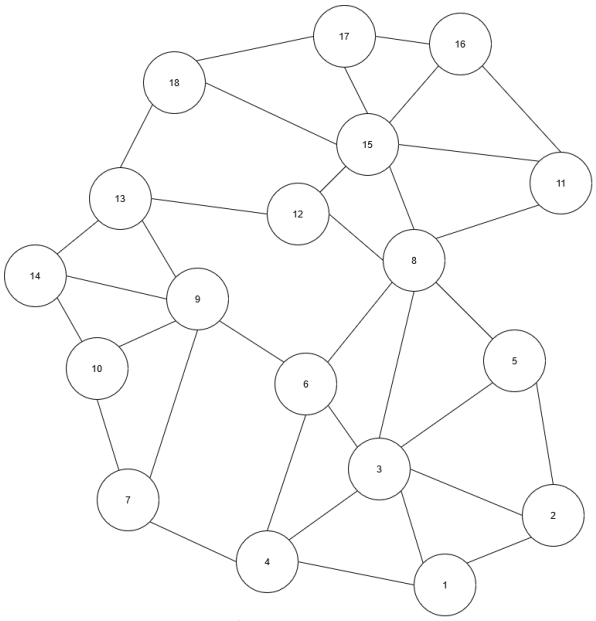
III. IMPLEMENTASI ALGORITMA

A. *Perancangan Graf Tak-Berarah*

Pada permainan *Apex Legends* diberikan peta permainan seperti gambar dibawah ini :

Gambar 4. Peta Permainan
Sumber : https://www.rockpapershotgun.com/2020/02/11/apex-legends-map-guide-season-4-worlds-edge-best-locations-and-loot/
Waktu Akses : 3 Mei 2020 Pukul 17:17 WIB.

Setelah melihat peta dan memperkirakan lokasi-lokasi yang terdapat barang-barang yang dibutuhkan, didapatkan graf tak-berarah :



Gambar 5. Graf tak-berarah dari Peta Permainan

Penamaan simpul dari graf merupakan penamaan secara acak dan hanya bersifat memudahkan penulisan, bukan merupakan urutan rute. Dan sisi dari setiap simpul mempertimbangkan berbagai macam rintangan pada peta yang tidak bisa dilalui sehingga tidak semua simpul dapat bersisian dengan semua simpul lainnya.

B. *Perancangan Matriks ongkos beserta Matriks tereduksi*

Dengan menghitung ongkos setiap sisi dari graf dengan cara taksiran jarak lurus dari lokasi satu ke lokasi lainnya, didapatkan matriks ongkos :

∞	20	20	34	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
20	∞	35	∞	30	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
20	35	∞	24	31	14	∞	42	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
34	∞	24	∞	∞	34	27	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	30	31	∞	∞	∞	∞	22	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	14	34	∞	∞	∞	28	21	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	27	∞	∞	∞	∞	43	20	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	42	∞	22	28	∞	∞	∞	∞	31	19	∞	∞	19	∞	∞	∞
∞	∞	∞	∞	∞	21	43	∞	∞	17	∞	∞	18	29	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	20	∞	17	∞	∞	∞	∞	14	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	31	∞	∞	∞	∞	∞	∞	39	32	∞	∞
∞	∞	∞	∞	∞	∞	∞	19	∞	∞	∞	∞	32	∞	10	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	18	∞	∞	32	∞	14	∞	∞	∞	20
∞	∞	∞	∞	∞	∞	∞	∞	29	14	∞	∞	14	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	19	∞	∞	39	10	∞	∞	∞	21	14	40
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	32	∞	∞	∞	21	∞	15	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	14	15	∞	33
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	20	∞	40	∞	33	∞

Sedemikian sehingga dengan melakukan reduksi baris-kolom didapatkan matriks tereduksi :

∞	0	0	7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
0	∞	15	∞	7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	21	∞	3	14	0	∞	28	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	0	∞	∞	10	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	8	9	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	0	13	∞	∞	∞	14	4	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	0	∞	∞	∞	∞	20	0	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	23	∞	0	9	∞	∞	∞	∞	0	0	∞	∞	0	∞	∞	∞
∞	∞	∞	∞	∞	4	23	∞	∞	0	∞	∞	1	12	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	3	∞	0	∞	∞	∞	∞	0	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	8	0	∞	∞
∞	∞	∞	∞	∞	∞	∞	9	∞	∞	∞	∞	22	∞	0	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	1	∞	∞	18	∞	0	∞	∞	∞	0
∞	∞	∞	∞	∞	∞	∞	∞	12	0	∞	∞	0	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	9	∞	∞	17	0	∞	∞	∞	10	4	24
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	5	∞	∞	∞	6	∞	0	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	0	∞	13
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	∞	20	∞	13	∞

Dengan *cost* simpul akar adalah jumlah pengurang matriks tereduksi berjumlah 347.

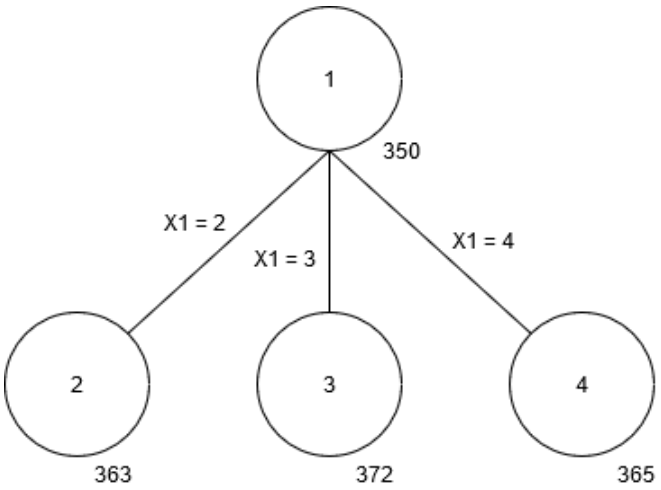
C. *Perancangan Pohon Status*

Dari hasil pereduksi matriks, didapatkan cost simpul akar bernilai 350. Anggap rute berawal dari lokasi 1, sehingga simpul 1 diekspansi lalu didapatkan anak dari simpul akar S ∈ {2,3,4}. Untuk setiap simpul dihitung setiap cost dari matriks yang telah diubah, sehingga untuk kasus simpul 2 didapatkan matriks tereduksi :

∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	8	∞	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
0	∞	∞	3	14	0	∞	25	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
4	∞	0	∞	∞	10	17	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	9	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	0	13	∞	∞	∞	14	4	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	0	∞	∞	∞	∞	20	0	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	23	∞	0	9	∞	∞	∞	∞	0	0	∞	∞	0	∞	∞	∞
∞	∞	∞	∞	∞	4	20	∞	∞	0	∞	∞	1	12	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	0	∞	0	∞	∞	∞	∞	0	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	8	0	∞	∞
∞	∞	∞	∞	∞	∞	∞	9	∞	∞	∞	∞	22	∞	0	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	1	∞	∞	18	∞	0	∞	∞	∞	0
∞	∞	∞	∞	∞	∞	∞	∞	12	0	∞	∞	0	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	9	∞	∞	17	0	∞	∞	∞	10	4	24
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	5	∞	∞	∞	6	∞	0	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	0	∞	13
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	∞	20	∞	13	∞

Dengan total cost bernilai cost sebelumnya + A(i, j) + total pengurang = 347 + 0 + 13 = 360.

Demikian juga cara untuk mengekspansi simpul 3 dan 4, sehingga didapatkan cost dari simpul 3 bernilai 347 + 0 + 22 = 369 dan cost simpul 4 bernilai 347 + 7 + 8 = 362. Sehingga didapatkan pohon status :



Gambar 6. Pohon Status awal Persoalan

D. *Algoritma Branch and Bound*

Dengan algoritma *Branch and Bound* (BB), kita dapat menentukan simpul selanjutnya yang di ekspan yaitu simpul dengan nilai cost terkecil, simpul 2. Dengan cara yang sama, simpul 2 memiliki simpul anak S ∈ {5,6} dengan X_1 ∈ {3, 5}. Kemudian dihitung untuk masing-masing simpul anak nilai costnya. Didapatkan cost simpul 5 = 376 dan cost simpul 6 = 363. Sehingga sesuai dengan algoritma BB yang di-ekspan selanjutnya adalah simpul 4. Dan akan terus diulang hingga ditemukan rute terpendek dari persoalan TSP ini.

E. *Implementasi Program*

Pada kasus TSP ini, karena matriks ongkos yang berukuran cukup besar, dapat diselesaikan dengan program berbahasa python yang terdapat dibawah ini :

```
from queue import PriorityQueue
import copy
import operator
```

Gambar 7. Header program

Program menggunakan kakas PriorityQueue yang telah disediakan bahasa pemrograman untuk mempermudah program

```
class Matriks(object):
    cost = 0
    path = []
    mat = [[]]

    def __init__(self, c, p, m):
        self.cost = c
        self.path = p
        self.mat = m
    def __gt__(self, other):
        return operator.__gt__(self.cost, other.cost)
    def __eq__(self, other):
        return operator.__eq__(self.cost, other.cost)
    def __lt__(self, other):
        return operator.__lt__(self.cost, other.cost)
    def __ge__(self, other):
        return operator.__ge__(self.cost, other.cost)
    def __le__(self, other):
        return operator.__le__(self.cost, other.cost)
    def __ne__(self, other):
        return operator.__ne__(self.cost, other.cost)
    def set_cost(self, c):
        self.cost = c
    def set_path(self, p):
        self.path = p
```

Gambar 8. Implementasi Kelas Matriks

Pada kelas matriks ini terdapat beberapa atribut seperti cost yang bernilai dari cost pada simpul pohon status yang bersesuaian. Path yang merupakan rute yang diambil selama perjalanan. Mat yang merupakan matriks ongkos tereduksi pada simpul di pohon status yang bersesuaian. Kelas matriks menyediakan method-method primitif untuk menyokong kelas tersebut.

```
def reducedRow(a, i):
    r = False
    j = 0
    for j in range (len(a[0])):
        if (a[i][j] == 0):
            r = True
            break
    return r

def reducedCol(a, j):
    r = False
    i = 0
    for i in range (len(a[0])):
        if (a[i][j] == 0):
            r = True
            break
    return r

def reducedMat(a):
    x = True
    for i in range (len(a[0])):
        for j in range (len(a[0])):
            y = reducedRow(a, i)
            z = reducedCol(a, j)
            x = y and z
            if (x==False):
                break
        if (x==False):
            break
    return x
```

Gambar 9. Fungsi Pengecekan Matriks

Pada fungsi-fungsi diatas disediakan fungsi untuk mengecek apakah matriks yang dimasukkan sebagai parameter telah berbentuk matriks tereduksi atau tidak.

```
def changeMat(a, i, j):
    x = float("inf")
    t = len(a.mat[0])
    c = a.cost
    p = a.path
    p.append(j)
    z = a.mat[i][j]
    for b in range (t):
        a.mat[b][j] = x
    for b in range (t):
        a.mat[i][b] = x
    a.mat[j][i] = x
    a.set_cost(c+z)
    reduceMat(a)
```

Gambar 10. Fungsi untuk Mengambil sisi yang bersesuaian pada graf

Fungsi diatas merupakan implementasi pada langkah algoritma matriks ongkos-tereduksi pada saat mengambil satu sisi dari graf sehingga mengubah matriks sesuai ketentuan algoritma.

```
def reduceRow(a, i):
    min = float("inf")
    for j in range (len(a[0])):
        if (a[i][j]<min):
            min = a[i][j]
    if (min != float("inf")):
        for j in range (len(a[0])):
            a[i][j]-=min
        return min
    else:
        return 0

def reduceCol(a, j):
    min = float("inf")
    for i in range (len(a[0])):
        if (a[i][j]<min):
            min = a[i][j]
    if (min != float("inf")):
        for i in range (len(a[0])):
            a[i][j]-=min
        return min
    else:
        return 0

def reduceMat(a):
    temp = a.cost
    for i in range (len(a.mat[0])):
        temp+=reduceRow(a.mat, i)
    for j in range (len(a.mat[0])):
        temp+=reduceCol(a.mat, j)
    a.set_cost(temp)
```

Gambar 10. Fungsi Mereduksi Matriks

Pada fungsi-fungsi diatas disediakan fungsi untuk mengubah sebuah matriks menjadi matriks tereduksi dengan cost matriks tersebut berubah sesuai dengan pereduksi.

```
m = [
    [x,     20, 20, 34, x,  x,  x,  x,  x,  x,  x,  x,  x,  x,  x,  x,  x],
    [20,    x,  35, x,  30, x,  x,  x,  x,  x,  x,  x,  x,  x,  x,  x,  x],
    [20,    35, x,  24, 31, 14, x,  42, x,  x,  x,  x,  x,  x,  x,  x,  x],
    [34,    x,  24, x,  x,  34, 27, x,  x,  x,  x,  x,  x,  x,  x,  x,  x],
    [x,     30, 31, x,  x,  x,  x,  22, x,  x,  x,  x,  x,  x,  x,  x,  x],
    [x,     x,  14, 34, x,  x,  x,  28, 21, x,  x,  x,  x,  x,  x,  x,  x],
    [x,     x,  x,  27, x,  x,  x,  x,  43, 20, x,  x,  x,  x,  x,  x,  x],
    [x,     x,  42, x,  22, 28, x,  x,  x,  x,  31, 19, x,  x,  19, x,  x],
    [x,     x,  x,  x,  x,  21, 43, x,  x,  17, x,  x,  18, 29, x,  x,  x],
    [x,     x,  x,  x,  x,  x,  20, x,  17, x,  x,  x,  x,  14, x,  x,  x],
    [x,     x,  x,  x,  x,  x,  x,  31, x,  x,  x,  x,  x,  x,  39, 32, x],
    [x,     x,  x,  x,  x,  x,  x,  19, x,  x,  x,  x,  32, x,  10, x,  x],
    [x,     x,  x,  x,  x,  x,  x,  x,  18, x,  x,  32, x,  14, x,  x,  x],
    [x,     x,  x,  x,  x,  x,  x,  x,  29, 14, x,  x,  14, x,  x,  x,  x],
    [x,     x,  x,  x,  x,  x,  x,  19, x,  x,  39, 10, x,  x,  x,  21, 14, 40],
    [x,     x,  x,  x,  x,  x,  x,  x,  x,  x,  32, x,  x,  x,  21, x,  15, x],
    [x,     x,  x,  x,  x,  x,  x,  x,  x,  x,  x,  x,  x,  x,  14, 15, x,  33],
    [x,     x,  x,  x,  x,  x,  x,  x,  x,  x,  x,  x,  20,  x, 40,  x, 33,  x]
    ]
p = [0]
ma = Matriks(0, p, m)
reduceMat(ma)
pq = PriorityQueue(0)
pq.put(ma)
temp = ma.path
l=0
```

Gambar 11. Inisialisasi Matriks Ongkos dan PriorityQueue

Pada gambar dapat dilihat merupakan matriks ongkos awal yang didapatkan dari perhitungan jarak pada graf tak-berarah. Setelah jarak untuk setiap simpul didapatkan, untuk setiap sisi i, j dimasukkan ke dalam matriks pada baris i dan kolom j. Karena graf tak-berarah, matriks simetris terhadap sumbu diagonal.

Selain itu dapat dilihat kelas Matriks dan PriorityQueue diinisialisasi. Matriks yang diinisialisasi memiliki cost awal sesuai dari hasil jumlah pereduksi dari langkah mereduksi matriks.

```
while (len(temp) < 18 and not pq.empty()):
    a = pq.get()
    temp = copy.deepcopy(a.path)
    c = a.cost
    if (len(temp) == 18):
        break
    i = a.path[len(a.path)-1]
    for j in range (18):
        if (a.mat[i][j] != float("inf")):
            q = a.cost
            w = copy.deepcopy(a.path)
            e = copy.deepcopy(a.mat)
            matr = Matriks(q, w, e)
            changeMat(matr, i, j)
            pq.put(matr)
            l+=1
for i in range (18):
    temp[i]+=1
print("Rute yang ditempuh:")
print(temp)
print("ongkos total: ", c)
print("total langkah: ", l)
```

Gambar 12. Algoritma BB beserta Output

Pada potongan program diatas dapat dilihat algoritma *Branch and Bound* untuk mencari rute terpendek dari persoalan TSP ini.

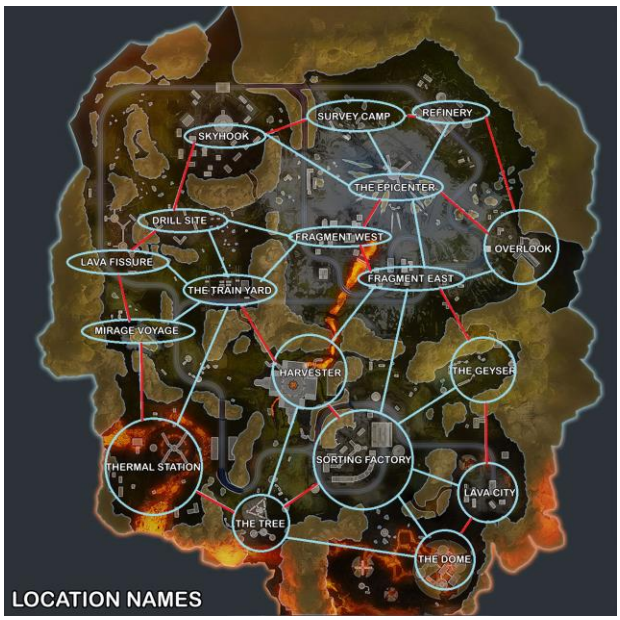
```
Rute yang ditempuh:
[1, 2, 5, 8, 12, 15, 11, 16, 17, 18, 13, 14, 10, 7, 4, 3, 6, 9]
ongkos total:  401
total langkah:  2227
```

Gambar 13. Keluaran Program

Pada gambar diatas didapatkan hasil keluaran dengan rute terpendeknya adalah 1->2->5->8->12->15->11->16->17->18->13->14->10->7->4->3->6->9 dengan ongkos yang dibutuhkannya adalah 401 beserta pembangkitan simpul dilakukan sebanyak 2227 kali.

IV. KESIMPULAN

Dari pencarian rute terpendek menggunakan algoritma *Branch and Bound* dan Matriks ongkos-tereduksi didapatkan rute terpendek beserta ongkosnya. Sehingga dapat dilihat pada peta permainan rute terpendeknya :



Gambar 14. Rute perjalanan Terpendek

Dengan ongkos total 401. Sehingga dapat disimpulkan bahwa dengan menggunakan algoritma *Branch and Bound* dapat ditemukan rute terpendek dari sebuah persoalan TSP. Namun tentu pada kasus tertentu untuk permainan ini juga masih belum diimplementasikan adanya pengecilan peta permainan setiap selang waktu tertentu yang tentunya akan memengaruhi penentuan rute yang optimal.

Fungsi jarak yang ditentukan juga belum tentu optimal dikarenakan adanya belokan-belokan dari jalan-jalan tertentu pada peta yang mengakibatkan jarak kurang tepat, selain itu juga harusnya diperhitungkan kemungkinan adanya pemain lain pada tempat tempat tertentu berdasarkan uji coba.

Namun, jika semuanya dapat diperhitungkan, algoritma *Branch and Bound* dapat di implementasikan dalam permainan *Apex Legends* sehingga pemain dapat dengan mudah

menentukan tujuan selanjutnya untuk mencapai barang-barang yang optimal untuk melawan pemain lain dan memenangkan permainan.

TAUTAN VIDEO PADA YOUTUBE

https://youtu.be/4Bhq43ZNp8I

UCAPAN TERIMA KASIH

Penulis ingin mengucapkan terima kasih sebesar-besarnya pada Tuhan Yang Maha Esa dimana atas karunia dan hikmah-Nya lah penulis dapat menyelesaikan tulisan ini. Penulis juga mengucapkan terima kasih kepada Ibu Nur Ulfa Maulidevi beserta tim dosen pengampu mata kuliah Strategi Algoritma karena tanpa bimbingan dan ajaran Ibu dan Bapak sekalian lah penulis dapat menyelesaikan tulisan dan mendapatkan banyak sekali ilmu yang bermanfaat kelak. Penulis juga mengucapkan banyak terima kasih kepada keluarga dan teman-teman yang menyemangati penulis untuk menyelesaikan tulisan ini.

REFERENSI

[1] Munir, Rinaldi. "Algoritma *Branch and Bound*" http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Branch-&-Bound-(2018).pdf, Waktu Akses : 3 Mei 2020.

[2] Toms, Ollie. "Apex Legends map guide (Season 4) - World's Edge best locations and loot" https://www.rockpapershotgun.com/2020/02/11/apex-legends-map-guide-season-4-worlds-edge-best-locations-and-loot/ Waktu Akses : 3 Mei 2020.

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 2020

Iqbal Naufal, 13518074