Pemanfaatan Algoritma Greedy untuk Penentuan Rute Perjalanan dalam Permainan *Assassin's Creed Unity: Dead Kings*

David Gozaly / 13518118
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13518118@std.stei.itb.ac.id

***Abstrak*—Bermain *game* menjadi salah satu aktivitas yang banyak dilakukan masyarakat di tengah masa PSBB maupun lockdown yang sedang diterapkan di berbagai belahan dunia demi mencegah penyebaran virus Covid-19. Salah saatu permainan yang cukup populer adalah Assassin's Creed, yang memiliki beberapa seri seperti contohnya Assassin's Creed Unity. *Game* ini menawarkan kisah seorang *Assassin* di era Revolusi Prancis, sekitar tahun 1790-an. Permainan modern pada umumnya memiliki peta yang luas, sehingga diperlukan perencanaan rute yang baik untuk dapat menjalankan misi-misi yang ada dalam permainan dengan efektif. Dalam bidang matematika maupun informatika, persoalan penentuan rute sudah tidak asing lagi, dan terdapat banyak sekali algoritma yang dapat digunakan untuk menyelesaikan persoalan ini. Salah satu algoritma yang sering dipilih untuk menyelesaikan persoalan seperti ini adalah algoritma Greedy, yang dapat memberikan solusi dalam waktu yang singkat dan dengan cara penggunaan algoritma yang mudah dibandingkan dengan algoritma lainnya. Makalah ini akan membahas pemanfaatan algoritma Greedy dalam penentuan rute yang sebaiknya dilakukan untuk menyelesaikan misi dalam permainan Assassin's Creed Unity: Dead Kings.**

***Keywords—game, Greedy,* algoritma, misi, Assassin's Creed**

I. PENDAHULUAN

Pandemi Covid-19 yang sedang melanda dunia ini menyebabkan seluruh kegiatan masyarakat menjadi terganggu. Jalan-jalan yang tadinya ramai mendadak sepi. Para pekerja terpaksa bekerja di rumah. Para pelajar terpaksa belajar di rumah. Di satu sisi, kebijakan Pembatasan Sosial Berskala Besar (PSBB) yang diberlakukan di Indonesia, maupun kebijakan lockdown di negara-negara lain menyebabkan kegiatan ekonomi menjadi lumpuh. Namun, dampak positif dari kebijakan-kebijakan ini juga dapat dirasakan, mulai dari tingkat polusi yang menurun drastis, hingga meningkatnya waktu berkumpul keluarga di rumah. Meskipun begitu, larangan keluar rumah bagi yang tidak berkepentingan menyebabkan banyak sekali orang tidak bisa mendapat penghasilan.

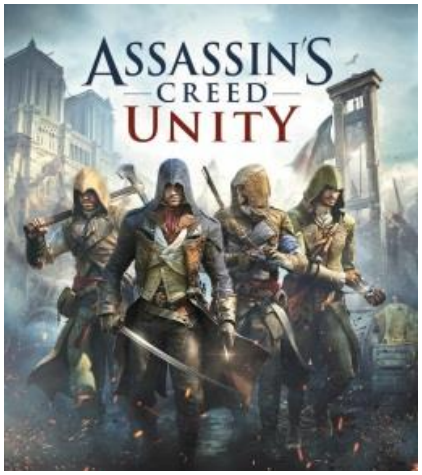
Larangan keluar rumah ini tentunya menyebabkan masalah-masalah baru, terutama bagi kalangan milenial, seperti misalnya kebosanan. Remaja-remaja yang biasa keluar rumah untuk mendapatkan hiburan, mulai dari jalan-jalan di mall hingga nongkrong di *café*, kini hanya bisa berdiam di rumah. Hal ini menyebabkan banyak orang yang mencari hiburan lain, salah satunya adalah bermain *game*.

Berdasarkan analisis data yang dilakukan oleh situs www.gameindustry.biz terhadap data penjualan *game* yang dirilis oleh *Game Sales Data* – ISFE (GSD), diketahui bahwa penjualan *game* baik dari sisi *software* maupun *hardware* mengalami peningkatan yang sangat tajam di masa *lockdown* yang saat ini sedang berlangsung di berbagai belahan dunia. Pada mingu ke-12 tahun ini, yaitu pada tanggal 16 – 22 Maret 2020, penjualan *game* mencapai angka 4.3 juta *copy* secara keseluruhan. Angka ini mengalami peningkatan sebesar 63% dibandingkan dengan minggu sebelumnya. Tentunya, angka ini terus meningkat setiap harinya, terutama dengan lebih banyaknya negara yang menetapkan kebijakan *lockdown*. Salah satu peningkatan terbesar adalah di Prancis, dimana pada minggu pertama kebijakan *lockdown* ditetapkan, penjualan *game* meningkat lebih dari 180% dibandingkan dengan minggu sebelumnya. Hal ini membuktikan bahwa banyak orang yang membutuhkan hiburan, salah satunya dengan bermain *game*, ketika terpaksa tinggal di rumah demi mencegah penyebaran virus Covid-19.

Dikutip dari www.polygon.com, lonjakan pemain ini tidak hanya terjadi pada *game-game* yang memang sedang ramai-ramainya dimainkan, seperti CS:GO, Dota 2, dan PUBG. Tren yang muncul adalah orang-orang lebih teretarik ke judul-judul *game* lain, bukan justru terbawa arus dengan memainkan *game* yang sangat populer saat ini. Banyak faktor yang mempengaruhi dipilihnya suatu *game*. *Game* yang baru dirilis cenderung memiliki banyak peningkatan pemain, misalnya Doom: Eternal. Faktor lainnya yang menyebabkan suatu *game* mengalami peningkatan jumlah pemain adalah ketika *game* tersebut mengalami pengurangan harga atau diskon, bahkan hingga diberikan secara gratis sebagai bentuk dukungan terhadap proses penanganan pandemi virus Covid-19. Salah satu contohnya adalah permainan Tomb Raider *reboot* yang dapat diperoleh dengan cuma-cuma melalui *platform* Steam.

Tren ini tentu saja juga berlaku kepada mahasiswa Institut Teknologi Bandung, terutama mahasiswa jurusan Teknik Informatika yang sering diberi label suka bermain *game* akibat

jurusan yang dipilih. Salah satu permainan yang mengalami peningkatan pemain adalah Assassin's Creed Unity. Hal ini terjadi karena beberapa waktu lalu, *game* ini mengalami perubahan harga yang sangat signifikan pada *platform* Steam.



Gambar 1: Assassin's Creed Unity
Sumber:
https://en.wikipedia.org/wiki/Assassin%27s_Creed_Unity

Terjadi kesalahan pada sistem Steam yang menyebabkan harga permainan Assassin's Creed Unity menjadi teracak di semua negara, tanpa diiringi dengan perubahan mata uang. Sehingga, negara-negara yang beruntung seperti misalnya Indonesia dapat membeli permainan ini dengan nominal harga dari negara lain, namun dalam mata uang rupiah. Permainan Assassin's Creed Unity yang seharusnya dapat dibeli dengan harga 345.000 rupiah dapat didapatkan dengan hanya sekian puluh rupiah saja. Namun, terdapat negara-negara lain yang malah mengalami dampak buruk dari hal ini, misalnya di negara Amerika, dimana permainan ini mengalami perubahan harga menjadi ratusan ribu dollar. Kesalahan yang notabene menyenangkan bagi orang-orang di Indonesia ini menyebabkan banyak sekali orang yang membeli permainan ini, termasuk mahasiswa Teknik Informatika ITB seperti penulis.

II. Landasan Teori

A. *Algoritma Greedy*

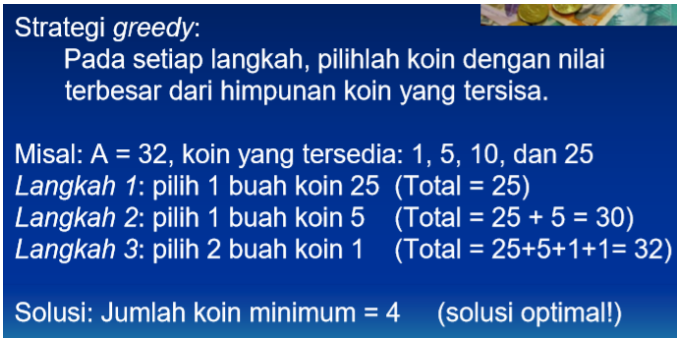
Algoritma Greedy adalah algoritma yang menyelesaikan suatu persoalan secara bertahap, dimana pada setiap tahap, algoritma Greedy memilih solusi terbaik pada tahap tersebut (*local optimum*) dengan harapan bahwa solusi local tersebut menuju ke solusi global yang terbaik. Algoritma Greedy biasa digunakan untuk menyelesaikan persoalan optimasi, yaitu maksimasi dan minimasi.

Algoritma Greedy tidak selalu menghasilkan solusi terbaik, karena algoritma ini hanya mempertimbangkan solusi terbaik untuk langkah selanjutnya, tanpa mempedulikan langkah-langkah berikutnya. Bisa saja langkah yang diambil pada tahap ini menuju kepada solusi yang kurang optimal, meskipun langkah tersebut merupakan langkah terbaik pada tahap itu. Lebih parah lagi, algoritma Greedy bisa saja tidak menghasilkan solusi karena langkah yang dianggap terbaik pada suatu tahap tidak menuju ke solusi (jalan buntu).

Meskipun memiliki berbagai kekurangan, algoritma Greedy tetap banyak digunakan untuk memecahkan masalah karena algoritma ini dapat menghasilkan solusi dengan cepat dan penggunaannya tergolong mudah dibandingkan dengan algoritma lainnya (kompleksitasnya tidak terlalu tinggi). Karena alasan ini pula algoritma Greedy sering digunakan untuk menghasilkan aproksimasi atau solusi hampiran untuk suatu masalah, terutama apabila soolusi terbaik tidak terlalu diperlukan (penggunaan algoritma greedy untuk melakukan aproksimasi solusi lebih baik dibandingkan menggunakan algoritma kompleks untuk mendapatkan solusi eksak yang tidak terlallu dibutuhkan).

Salah satu contoh persoalan optimasi yang dapat diselesaikan dengan algoritma Greedy adalah persoalan penukaran uang, dimana suatu nominal uang akan ditukarkan dengan koin yang tersedia, dan diperlukan jumlah koin minimum untuk penukaran tersebut.

Berikut adalah salah satu contoh penyelesaian persoalan penukaran uang dengan algoritma Greedy.



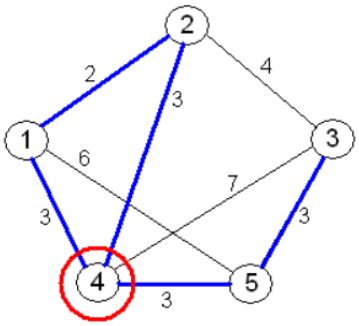
Gambar 2: Pengunaan algoritma Greedy dalam penyelesaian persoalan penukaran uang
Sumber:
http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Algoritma-Greedy-(2020).pdf

Namun, algoritma Greedy tidak selalu menghasilkan solusi optimal, seperti pada contoh berikut.

Gambar 3: contoh solusi tidak optimal yang dihasilkan algoritma Greedy
Sumber:
http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Algoritma-Greedy-(2020).pdf

B. *Travelling Salesman Problem*

Travelling salesman/salesperson problem (TSP) adalah persoalan mencari rute tersingkat yang diperlukan untuk mengunjungi seluruh titik yang ada tepat sekali, dan kembali lagi ke titik asal, apabila diketahui jarak diantara 2 titik untuk setiap titik yang ada. Persoalan TSP ini termasuk dalam NP-hard.



Gambar 4: Contoh persoalan TSP
Sumber: https://commons.wikimedia.org/wiki/File:Example_The_travelling_salesman_problem_(TSP)_penalty_cost_1.gif

Asal mula terciptanya persoalan *Travelling salesman problem* ini tidak jelas. Namun, persoalan TSP ini pertama kali diformulasikan secara matematika sekitar tahun 1800 oleh matematikawan W.R. Hamilton yang berasal dari Irlandia dan Thomas Kirkman yang berasal dari Inggris. Karena itulah persoalan TSP sering juga disebut sebagai persoalan mencari sirkuit Hamilton, dimana sirkuit Hamilton adalah sirkuit dimana setiap simpul dikunjungi tepat sekali, dengan menggunakan salah satu titik sebagai titik awal dan titik akhir.

Persoalan TSP sendiri dapat diselesaikan dengan menggunakan berbagai macam algoritma. Karena itu pula, persoalan TSP sering dijadikan *benchmark* untuk menguji efektifitas / kecepatan suatu algoritma, misalnya dengan membuat jumlah titik atau kota yang dikunjungi sangat banyak.

C. *Assassin's Creed Unity : Dead Kings*

Assassin's Creed Unity adalah permainan *action-adventure* yang merupakan salah satu seri dari permainan Asassin's Creed yang dirilis oleh Ubisoft. Permainan ini dapat dimainkan di PC (Microsoft Windows), Playstation 4, dan Xbox One. Assassin's Creed Unity dirilis pada tahun 2014. Permainan ini bercerita tentang perseteruan berabad-abad diantara Asassins dan Templars, dan kali ini terjadi pada masa revolusi Prancis, sekitar tahun 1790. Permainan ini juga memiliki DLC (*downloadable content*) yang bernama Asassin's Creed Unity: Dead Kings, yang bercerita tentang penemuan sebuah artifak di makam raja Louis IX. DLC ini dirilis secara gratis pada Januari 2015.

Permainan ini menerima respon beragam dari para pencinta seri Assassin's Creed. Sebagian kurang menyukai permainan ini karena tingginya *bug* yang ada di permainan ini ketika pertama kali dirilis. Mulai dari objek-objek yang tidak dapat ditampilkan dengan baik, hingga karakter pemain dan npc yang tersangkut di tempat-tempat yang tidak seharusnya. Berbagai *bug* ini tentunya menjadi bahan *meme* yang sangat menarik dan lucu pada jamannya. Namun berkat kerja keras dari para *developer*, masalah-masalah tersebut tidak terjadi lagi sekarang.

Di sisi lain, banyak orang yang menganggap Assassin's Creed Unity sebagai salah satu permainan Assassin's Creed terbaik, karena berbagai macam alasan. Mulai dari sistem persenjataan dan perlengkapan yang bervariasi, *gameplay* yang menarik, serta sistem *parkour* yang diperbarui dan menjadi jauh lebih baik dibandingkan seri lainnya. Hal ini tentunya menjadi sebuah keunggulan yang sangat besar, terutama karena *parkour* menjadi salah satu daya Tarik utama dari seri Assassin's Creed, dimana pemain bisa dengan bebas memanjat bangunan, berjalan di atas tali, serta melompat dari atap ke atap. Tidak hanya itu, salah satu fitur Assassin's Creed Unity yang dinilai sangat bagus adalah fitur *multiplayer*, dimana pemain bisa menyelesaikan misi bersama-sama dengan teman secara online.

Permainan Assassin's Creed Unity menyediakan berbagai macam pilihan senjata bagi para pemain, yaitu *one-handed swords, long weapons, heavy weapons, pistols,* dan *rifles.* Salah satu senjata terbaik yang terdapat dalam permainan ini adalah *one-handed sword* bernama *Eagle of Suger.* Selain karena memiliki *stats* terbaik, senjata ini juga memiliki efek khusus yaitu 30% kemungkinan untuk membutakan musuh.

Gambar 5: Senjata *Eagle of Suger*
Sumber: dokumentasi pribadi penulis

Senjata ini bisa didapatkan dengan menjalankan serangkaian misi dalam DLC Dead Kings. Terdapat 7 misi yang perlu diselesaikan untuk mendapatkan senjata ini. Pada setiap misi, pemain harus menjelajahi peta permainan untuk menemukan 3 simbol tersembunyi. Lokasi dari simbol pertama dapat diketahui dengan menyelesaikan teka-teki/*riddle* yang didapatkan di awal misi, lalu petunjuk lokasi kedua diberikan setelah menemukan simbol pertama, begitu seterusnya.

Gambar 6: Salah satu simbol yang harus ditemukan pemain
Sumber: dokumentasi pribadi penulis

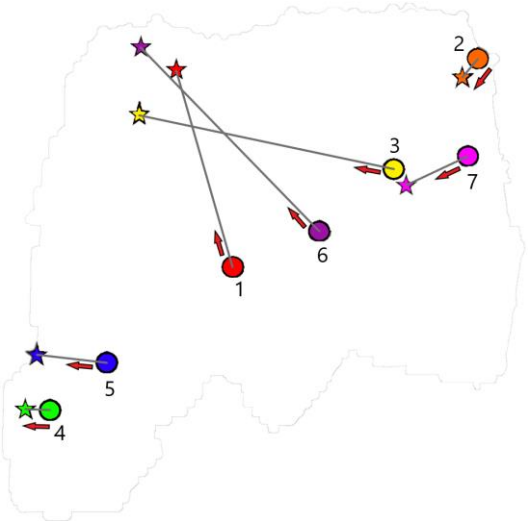
Misi dan simbol-simbol yang perlu ditemukan tersebar di seluruh peta permainan. Karena itu, diperlukan rute yang efektif untuk bisa menyelesaikan seluruh misi dalam waktu yang singkat.

III. Pemanfaatan Algoritma Greedy untuk Menentukan Rute Terbaik dalam Menyelesaikan Misi

Berikut ini adalah peta permainan Asassin's Creed Unity: Dead Kings yang telah ditandai dengan simbol lingkaran dan bintang. Bentuk lingkaran menandakan titik awal setiap misi, sedangkan bentuk bintang menandakan titik akhir suatu misi (lokasi simbol terakhir yang harus ditemukan pada misi tersebut).

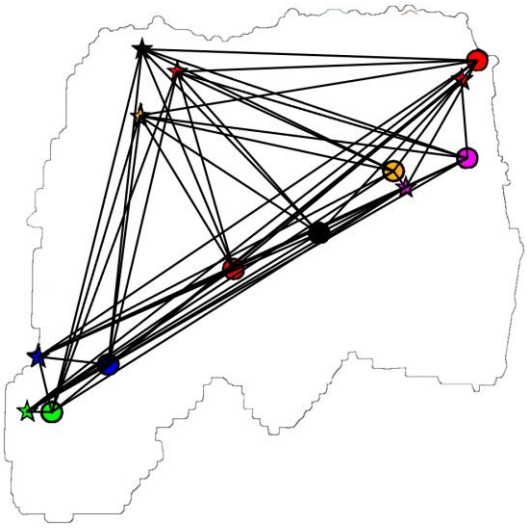
Gambar 7: Peta permainan yang sudah ditandai
Sumber: dokumentasi pribadi penulis

Peta ini disederhanakan menjadi gambar berikut, dengan garis abu-abu sebagai penanda rute dari titik awal misi (lingkaran) ke titik akhir misi (bintang), dan tanda panah sebagai arah perjalanan. Titik awal dan titik akhir dari satu misi memiliki warna yang sama.



Gambar 8: Peta yang disederhanakan
Sumber: dokumentasi pribadi penulis

Berikut adalah graf yang menggambarkan semua kemungkinan rute perjalanan yang dapat dilakukan. Graf ini dibentuk dengan cara menghubungkan setiap titik akhir suatu misi ke seluruh titik awal misi yang ada selain misi tersebut dengan sebuah garis. Sama seperti gambar sebelumnya lingkaran menggambarkan titik awal suatu misi, sedangkan bintang menggambarkan titik akhir suatu misi. Pasangan titik awal dan titik akhir suatu misi ditandai dengan warna yang sama.



Gambar 9: Jalur yang dapat digunakan
Sumber: dokumentasi pribadi penulis

Misi pertama yang dijalankan adalah misi nomor 1, karena merupakan bagian dari cerita utama permainan. Sementara itu, misi nomor 2-7 dapat dimainkan dengan urutan yang bebas, namun pemain hanya bisa menjalankan satu misi dalam satu waktu. Algoritma Greedy akan digunakan untuk menentukan urutan misi yang akan dijalankan.

Persoalan menentukan urutan misi ini mirip dengan persoalan *Travelling Salesman Problem* (TSP), hanya saja rute tidak kembali ke titik awal. Karena itu, diperlukan jarak dari setiap titik akhir suatu misi ke seluruh titik awal misi lainnya. Data ini diperoleh penulis dengan mengukur jarak kedua titik secara manual, yaitu dengan mencetak peta dan mengukur jarak titik akhir suatu misi ke titik awal suatu misi satu persatu. Baris adalah titik akhir misi (*star*), sedangkan kolom adalah titik awal misi (*circle*). Baris dan kolom dengan angka yang sama menunjukkan jarak antara titik awal dan titik akhir suatu misi. Satuan yang digunakan adalah centimeter (cm).

Star\circle	1	2	3	4	5	6	7
1	7.0	10.2	8.1	12.4	10.2	7.3	10.3
2	10.0	0.9	3.9	17.9	15.4	7.1	2.7
3	6.1	11.6	8.8	10.5	8.5	7.3	11.2
4	8.5	19.3	14.9	0.9	3.2	11.6	17.2
5	7.3	17.9	13.6	2.0	2.4	10.4	16.0
6	8.1	11.4	9.5	12.7	10.8	8.7	11.7
7	6.4	4.9	0.7	14.2	11.7	3.3	2.3

Tabel 1: Jarak titik akhir suatu misi ke titik awal setiap misi
Sumber: dokumentasi pribadi penulis

Algoritma Greedy digunakan untuk menentukan urutan misi yang akan dilakukan. Algoritma ini menyelesaikan persoalan secara bertahap, dimana pada setiap tahapnya, akan dipilih misi yang akan dilakukan berikutnya dengan cara memilih misi dengan jarak titik awal terdekat dari titik akhir misi sebelumnya (tidak bisa memilih misi yang sudah dijalankan). Titik awal misi terdekat ini akan menjadi solusi lokal pada suatu tahap, dan diharapkan solusi lokal ini akan menuju ke solusi global yang terbaik, yaitu rute penyelesaian misi yang paling efektif.

Berikut adalah langkah-langkah penyelesaian persoalaln dengan menggunakan algoritma Greedy.

1. Tahap 1:

   Titik yang dipilih : titik 1 (misi 1 wajib dilakukan pertama kali)
   Rute saat ini : 1

2. Tahap 2:

   Data jarak titik akhir misi sebelumnya ke titik awal setiap misi yang belum dijalankan : 10.2, 8.1, 12.4, 10.2, 7.3, 10.3
   Jarak minimum : 7.3 (titik 6)
   Titik yang dipilih : titik 6
   Rute saat ini : 1 – 6

3. Tahap 3:

   Data jarak titik akhir misi sebelumnya ke titik awal setiap misi yang belum dijalankan : 11.4, 9.5, 12.7, 10.8, 11.7
   Jarak minimum : 9.5 (titik 3)
   Titik yang dipilih : titik 3
   Rute saat ini : 1 – 6 – 3

4. Tahap 4:

   Data jarak titik akhir misi sebelumnya ke titik awal setiap misi yang belum dijalankan : 11.6, 10.5, 8.5, 11.2
   Jarak minimum : 8.5 (titik 5)
   Titik yang dipilih : titik 5
   Rute saat ini : 1 – 6 – 3 – 5

5. Tahap 5:

   Data jarak titik akhir misi sebelumnya ke titik awal setiap misi yang belum dijalankan : 17.9, 2.0, 16.0
   Jarak minimum : 2.0 (titik 4)
   Titik yang dipilih : titik 4
   Rute saat ini : 1 – 6 – 3 – 5 – 4

6. Tahap 6:

   Data jarak titik akhir misi sebelumnya ke titik awal setiap misi yang belum dijalankan : 19.3, 17.2
   Jarak minimum : 17.2 (titik 7)
   Titik yang dipilih : titik 7
   Rute saat ini : 1 – 6 – 3 – 5 – 4 – 7

7. Tahap 7:

   Titik yang belum dipilih hanya titik 2
   Jarak minimum : 4.9 (titik 2)
   Titik yang dipilih : titik 2
   Rute yang dihasilkan : 1 – 6 – 3 – 5 – 4 – 7 – 2

Dengan demikian, urutan misi yang dihasilkan dengan menggunakan algoritma Greedy adalah 1 – 6 – 3 – 5 – 4 – 7 – 2 dengan total jarak yang ditempuh sebesar 49.4 cm (tidak termasuk jarak dari titik awal ke titik akhir suatu misi).

Persoalan ini juga dapat diselesaikan dengan menggunakan program. Program dengan algoritma Greedy yang digunakan untuk menyelesaikan persoalan ini dibuat dengan menggunakan bahasa Python 3.8.0 dan menggunakan paradigma pemrograman prosedural.

```
star0 = []
star1 = [99.9, 7.0, 10.2, 8.1, 12.4, 10.2, 7.3, 10.3]
star2 = [99.9, 10.0, 0.9, 3.9, 17.9, 15.4, 7.1, 2.7]
star3 = [99.9, 6.1, 11.6, 8.8, 10.5, 8.5, 7.3, 11.2]
star4 = [99.9, 8.5, 19.3, 14.9, 0.9, 3.2, 11.6, 17.2]
star5 = [99.9, 7.3, 17.9, 13.6, 2.0, 2.4, 10.4, 16.0]
star6 = [99.9, 8.1, 11.4, 9.5, 12.7, 10.8, 8.7, 11.7]
star7 = [99.9, 6.4, 4.9, 0.7, 14.2, 11.7, 3.3, 2.3]
star = [star0, star1, star2, star3, star4, star5, star6, star7]

hasil = [1] # quest pertama adalah titik 1
next = 0
now = 1
i = 0

def terkecil(arraystar, arrayhasil):
    kecil = 0
    for i in range (1, 8):
        if (i not in arrayhasil):
            if (arraystar[i] < arraystar[kecil]):
                kecil = i
    return kecil

while (len(hasil) != 7):
    next = terkecil(star[now], hasil)
    hasil.append(next)
    now = next

print(hasil)
```

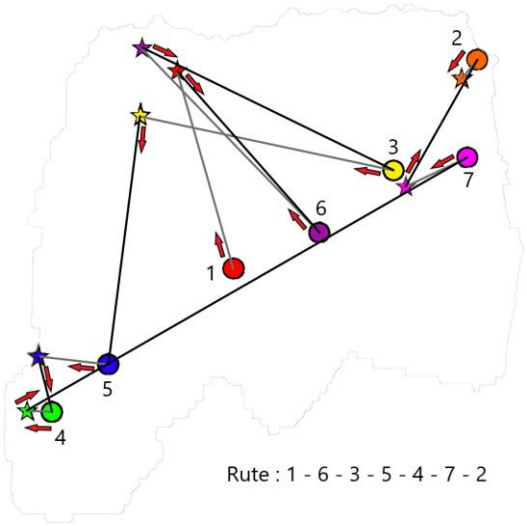
Gambar 10: *Screenshot source code* program
Sumber: dokumentasi pribadi penulis

Program diatas bekerja dengan cara memilih titik dengan jarak terdekat dengan titik sebelumnya pada setiap iterasi. Pemilihan titik terdekat ini menggunakan fungsi ‘terkecil’, dimana fungsi ini menerima masukan sebuah ‘arrayjarak’ yang berisi jarak dari titik akhir suatu misi ke titik awal seluruh misi serta sebuah ‘arrayhasil’ yang berisi titik-titik yang sudah pernah dipilih. Fungsi ‘terkecil’ akan mengembalikan titik awal misi yang memiliki jarak terkecil dengan titik akhir misi sebelumnya, dengan syarat titik tersebut belum pernah dikunjungi sebelumnya (tidak terdapat di array hasil). Variabel ‘now’ menyimpan titik yang sedang dikunjungi sekarang, sedangkan variable ‘next’ menyimpan titik yang dipilih untuk dikunjungi berikutnya. Program menyimpan urutan titik yang dipilih di dalam array ‘hasil’, dimana pada setiap iterasi titiik terdekat akan ditambahkan di akhir array. Hasil program kemudian akan ditampilkan di layar.

```
C:\Users\David Gozaly\Desktop\STIMA\makalah>python acgreedy.py
[1, 6, 3, 5, 4, 7, 2]
```

Gambar 11: *Screenshot output* program
Sumber: dokumentadi pribadi penulis

Program mengeluarkan hasil : 1 – 6 – 3 – 5 – 4 – 7 – 2. Urutan misi yang dihasilkan program serupa dengan urutan misi yang didaptakan dari proses manual yang telah dijabarkan sebelumnya. Rute yang diambil pemain akan ditampilkan pada gambar berikut.



Gambar 12: Rute yang dihasilkan algoritma Greedy
Sumber: dokumentasi pribadi penulis

IV. Kesimpulan

Algoritma Greedy dapat digunakan untuk menentukan rute atau urutan misi yang sebaiknya dilakukan pemain untuk mendapatkan senjata *Eagle of Suger* dalam permainan Assassin's Creed Unity: Dead Kings. Rute yang dihasilkan dengan algoritma Greedy bisa saja bukan merupakan rute yang paling maksimal, namun algoritma ini dipilih karena kemudahan penggunaan dan algoritma Greedy dapat menghasilkan solusi dari permasalahan yang rumit dengan mudah dan dalam waktu yang singkat.

V. Saran

A. Saran Pengembangan

Saran untuk pengembangan program ini adalah mengubah input dari jarak antar titik menjadi lokasi atau koordinat titik dalam bidang Cartesian, lalu kemudian jarak antar titik dihitung menggunakan formula jarak Euclidean oleh program. Hal ini bertujuan agar jarak dapat dihitung otomatis oleh program, sehingga meningkatkan kemudahan penggunaan program.

B. Saran Pemanfaatan

Program atau algoritma Greedy ini dapat digunakan untuk menghasilkan rute perjalanan atau urutan misi yang sebaiknya dilakukan dalam permainan apapun, tidak hanya dalam permainan Assassin's Creed Unity. Lebih dari itu, program atau algoritma ini juga dapat dimanfaatkan untuk berbagai aktivitas di dunia nyata, seperti pemilihan rute perjalanan sesunggunya, misalnya rute perjalanan untuk tukang pos, pengantar paket, dan lain-lain. Rute perjalanan yang efektif tentunya dapat mempermudah pekerjaan manusia serta meminimalisir waktu yang dibutuhkan untuk melakukan pekerjaan tersebut. Hal ini tentunya sangat bermanfaat, terutama dalam masa-masa sulit akibat pandemic Covid-19 ini.

Video Link di Youtube

https://youtu.be/h2bsJiqhQxc

Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena dengan berkat dan kemurahan hati-Nya makalah ini dapat saya selesaikan dengan sebaik-baiknya.

Penulis juga ingin mengucapkan terima kasih kepada kedua orang tua saya yang telah mendukung penulisan makalah ini. Tentunya, saya juga berterima kasih kepada Ibu Masayu Leylia Khodra, S.T., M.T. sebagai dosen pengajar mata kuliah IF2211 Strategi Algoritma kelas K1 karena telah membimbing saya selama proses pembelajaran di semester ini, serta kepada Bapak Dr. Ir. Rinaldi Munir yang telah menyediakan materi pembelajaran di website beliau.

Terima kasih juga penulis ucapkan untuk Ubisoft beserta seluruh developer atas usaha dan kerja kerasnya dalam proses pengembangan permainan Assassin's Creed Unity ini, serta untuk perbaikan yang telah dilakukan terhadap berbagai *bug* yang tadinya terdapat dalam permainan ini. Tentunya penulis juga berterimakasih karena DLC Dead Kings diberikan secara gratis kepada seluruh pemain.

Referensi

[1] Munir, Rinaldi. Algoritma Greedy (revisi 2020). http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Algoritma-Greedy-(2020).pdf. Diakses pada 3 Mei 2020, pukul 15.00.

[2] Goslin, Austen. Steam sets new records amid worldwide lockdown. https://www.polygon.com/2020/3/23/21191397/steam-highest-active-player-record-coronavirus-covid-19. Diakses pada 4 Mei 2020 pukul 17.30.

[3] Dring, Christopher. What is happening with video game sales during coronavirus. https://www.gamesindustry.biz/articles/2020-03-28-what-is-happening-with-video-game-sales-during-coronavirus. Diakses pada 4 Mei 2020 pukul 17.30

[4] Tim penulis GeeksforGeeks. Greedy Algorithms. https://www.geeksforgeeks.org/greedy-algorithms/. Diakses pada 3 Mei 2020, pukul 15.30

[5] Tim penulis GeeksforGeeks. Travelling Salesman Problem. https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/. Diakses pada 3 Mei 2020, pukul 16.00

[6] Tim penulis Assassin's Creed Wiki Fandom. Dead Kings. https://assassinscreed.fandom.com/wiki/Dead_Kings. Diakses pada 3 Mei 2020, pukul 20.00

[7] Tim penulis ssassin's Creed Unity Wiki Guide. Suger's Legacy. https://www.ign.com/wikis/assassins-creed-5-unity/Suger%27s_Legacy. Diakses pada 3 Mei 2020, pukul 21.00

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, 04 Mei 2020

David Gozaly
13518118