LAPORAN TUGAS BESAR 1

IF2211 - STRATEGI ALGORITMA

Dosen:

Ir. Rila Mandala, M.Eng., Ph.D.

Monterico Adrian, S.T., M.T.

Mahasiswa:

Farhan Raditya Aji (13522142)

M. Zaidan Sa'dun R. (13522146)

Rafif Ardhinto Ichwantoro (13522159)

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER I TAHUN 2023/2024

DAFTAR ISI

BAB I

Deskripsi Masalah

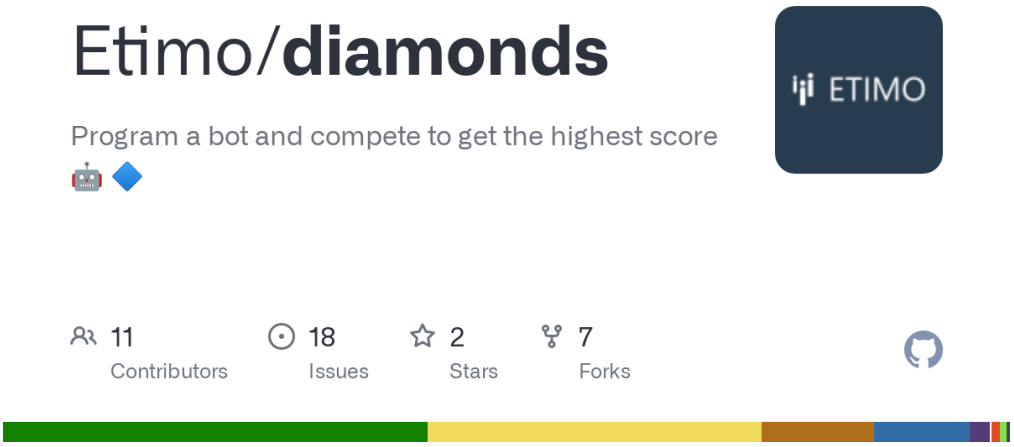
1.1 Tujuan

Tujuan dari pembuatan tugas besar ini adalah sebagai berikut,

1. Membuat program sederhana dalam bahasa Python dengan mengimplementasikan algoritma Greedy pada bot permainan etimo diamonds untuk memenangkan permainan.
2. Strategi greedy terbaik yang harus diterapkan oleh setiap kelompok harus terfokus pada tujuan utama permainan, yaitu untuk mengumpulkan sebanyak mungkin berlian dan mencegah bot lain mengambilnya. Hal ini sangat penting karena setiap bot dari setiap kelompok akan bersaing dalam kompetisi Tubes 1.

1.2 Spesifikasi

Diamonds adalah sebuah tantangan pemrograman di mana bot yang Anda buat akan bersaing dengan bot dari pemain lain. Setiap pemain memiliki bot yang harus mengumpulkan sebanyak mungkin berlian. Namun, perjalanan untuk mengumpulkan berlian tidaklah mudah karena berbagai rintangan akan ditemui, membuat permainan menjadi lebih menarik dan kompleks. Untuk meraih kemenangan, setiap pemain harus menerapkan strategi khusus pada bot mereka masing-masing. Detail tentang aturan permainan akan dijelaskan lebih lanjut di bawah ini.



Buatlah program sederhana dalam bahasa **Python** yang mengimplementasikan ***algoritma Greedy*** pada *bot* permainan Diamonds dengan tujuan memenangkan permainan.

- Tugas dikerjakan berkelompok dengan anggota minimal 2 orang dan maksimal 3 orang, boleh lintas kelas dan lintas kampus.
- Strategi *greedy* yang diimplementasikan setiap kelompok harus dikaitkan dengan fungsi objektif dari permainan ini, yaitu memenangkan permainan dengan memperoleh *diamond* sebanyak banyak nya dan jangan sampai *diamond* tersebut diambil oleh bot lain. Buatlah strategi *greedy* terbaik, karena setiap bot dari masing-masing kelompok akan diadu dalam kompetisi Tubes 1.
- Strategi *greedy* yang kelompok anda buat harus dijelaskan dan ditulis secara eksplisit pada laporan, karena akan diperiksa saat demo apakah strategi yang dituliskan sesuai dengan yang diimplementasikan. Tiap kelompok dapat menggunakan kreativitas yang bermacam macam dalam menyusun strategi *greedy* untuk memenangkan permainan. Implementasi pemain harus dapat dijalankan pada *game engine* yang telah disebutkan diatas serta dapat dikompetisikan dengan bot dari kelompok lain.
- Program harus mengandung komentar yang jelas, dan untuk setiap strategi *greedy* yang disebutkan, harus dilengkapi dengan kode sumber yang dibuat.
- Mahasiswa dilarang menggunakan kode program yang diunduh dari Internet. Mahasiswa harus membuat program sendiri, diperbolehkan untuk belajar dari program yang sudah ada.
- Mahasiswa dianggap sudah melihat dokumentasi dari *game engine,* sehingga tidak terjadi kesalahpahaman spesifikasi antara mahasiswa dan asisten.
- BONUS (maks 10): Membuat video tentang aplikasi *greedy* pada bot serta simulasinya pada game kemudian mengunggahnya di Youtube. Video dibuat harus memiliki audio dan menampilkan wajah dari setiap anggota kelompok. Untuk contoh

video tubes stima tahun-tahun sebelumnya dapat dilihat di Youtube dengan kata kunci “Tubes Stima”, “strategi algoritma”, “Tugas besar stima”, dll.

- Jika terdapat kesulitan selama mengerjakan tugas besar sehingga memerlukan bimbingan, maka dapat melakukan asistensi tugas besar kepada asisten (opsional). Dengan catatan asistensi hanya bersifat membimbing, bukan memberikan “jawaban”.
- Terdapat juga demo dari program yang telah dibuat. Pengumuman tentang demo menunggu pemberitahuan lebih lanjut dari asisten.
- Bot yang telah dibuat akan dikompetisikan dengan kelompok lain dan disaksikan oleh seluruh peserta kuliah. Terdapat hadiah menarik bagi kelompok yang memenangkan kompetisi.
- Setiap kelompok harap mengisi nama kelompok dan anggotanya pada link berikut, paling lambat **Sabtu, 24 Februari pukul 22.11 WIB**.
  Pendataan Kelompok Tubes 1 Stima
- Diwajibkan untuk memilih asisten meskipun tidak melakukan asistensi, karena asisten yang dipilih akan menjadi asisten saat asistensi (opsional) dan demo tugas besar. Pemilihan asisten dapat dilakukan pada link berikut, paling lambat **Sabtu, 24 Februari pukul 22.11 WIB**.
  Pendataan Kelompok Tubes 1 Stima
- Program disimpan dalam repository yang bernama **Tubes1_NamaKelompok** dengan nama kelompok sesuai dengan yang di sheets diatas. Berikut merupakan struktur dari isi repository tersebut:
  a. Folder src berisi **semua bagian yang ada di bot-starter pack**
  b. Folder doc berisi laporan tugas besar dengan format **NamaKelompok.pdf**
  c. README untuk tata cara penggunaan yang minimal berisi:
     i. Penjelasan singkat algoritma greedy yang diimplementasikan
     ii. Requirement program dan instalasi tertentu bila ada
     iii. Command atau langkah-langkah dalam meng-compile atau build program
     iv. Author (identitas pembuat)
- Laporan dikumpulkan hari **Sabtu, 9 Maret 2024** pada alamat Google Form berikut paling lambat pukul **23.59** :
  https://bit.ly/tubes1stima24
- Adapun pertanyaan terkait tugas besar ini bisa disampaikan melalui QnA berikut: https://bit.ly/qnastima24

BAB II

Landasan Teori

2.1 Dasar Teori

Algoritma greedy merupakan pendekatan dalam pemrograman yang mengatasi masalah optimasi dengan cara yang tampaknya rakus. Pendekatan ini berfokus pada pengambilan keputusan saat ini dengan harapan bahwa setiap langkah akan membawa kita lebih dekat ke solusi akhir yang optimal. Pada dasarnya, algoritma greedy berusaha mencari solusi terbaik dari suatu masalah dengan mempertimbangkan pilihan yang tersedia pada setiap langkahnya, dan memilih pilihan yang paling optimal pada saat itu, tanpa memperhitungkan konsekuensi jangka panjang dari pilihan tersebut. Contoh penerapan algoritma greedy adalah dalam masalah pemilihan koin. Jika seseorang ingin memberikan kembalian untuk suatu jumlah uang dengan jumlah koin yang minimal, maka algoritma greedy akan memilih koin dengan nilai tertinggi yang masih bisa digunakan untuk memberikan kembalian, sampai jumlah kembalian yang diberikan sesuai dengan jumlah uang yang diminta. Meskipun algoritma greedy ini tidak selalu menghasilkan solusi yang optimal secara keseluruhan, namun seringkali algoritma ini menghasilkan solusi yang cukup baik dan efisien.

2.2 Etimo Diamond V2

Etimo Diamond V2 adalah game engine yang melibatkan bot dengan logic bot masing-masing dalam persaingan dengan bot dari pemain lainnya. Setiap bot yang dimiliki oleh pemain memiliki tujuan untuk mengumpulkan sebanyak mungkin diamond dalam permainan. Namun, proses pengumpulan diamond tidak akan mudah karena terdapat berbagai rintangan yang membuat permainan menjadi menarik dan kompleks. Untuk menjadi pemenang dalam pertandingan, setiap pemain harus mengimplementasikan strategi khusus pada bot-nya agar dapat mengatasi rintangan dan mengumpulkan diamond sebanyak mungkin. Dengan demikian, kreativitas dan kecerdasan dalam merancang strategi bot akan menjadi kunci untuk meraih kemenangan dalam Diamonds.

2.2.1 Cara Menjalankan Game Engine

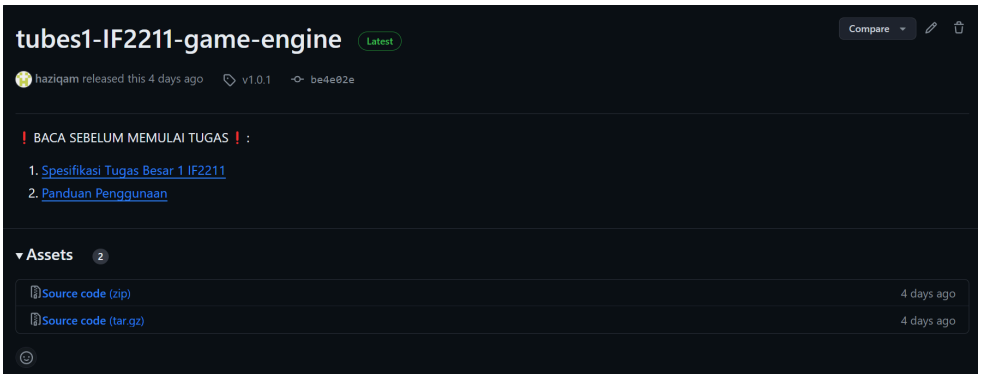
a. *Requirement* yang harus di-install

- Node.js (https://nodejs.org/en)
- Docker desktop (https://www.docker.com/products/docker-desktop/)
- Yarn

```
npm install --global yarn
```

b. Instalasi dan konfigurasi awal

1) Download source code (.zip) pada release game engine



2) Extract zip tersebut, lalu masuk ke folder hasil extractnya dan buka terminal
3) Masuk ke root directory dari project (sesuaikan dengan nama rilis terbaru)

```
cd tubes1-IF2110-game-engine-1.1.0
```

4) Install dependencies menggunakan Yarn

```
yarn
```

5) Setup default environment variable dengan menjalankan script berikut
Untuk Windows

```
./scripts/copy-env.bat
```

Untuk Linux / (possibly) macOS

```
chmod +x ./scripts/copy-env.sh
./scripts/copy-env.sh
```

6) Setup local database (buka aplikasi docker desktop terlebih dahulu, lalu jalankan command berikut di terminal)

```
docker compose up -d database
```

Lalu jalankan script berikut. Untuk Windows

```
./scripts/setup-db-prisma.bat
```

Untuk Linux / (possibly) macOS

```
chmod +x ./scripts/setup-db-prisma.sh
./scripts/setup-db-prisma.sh
```

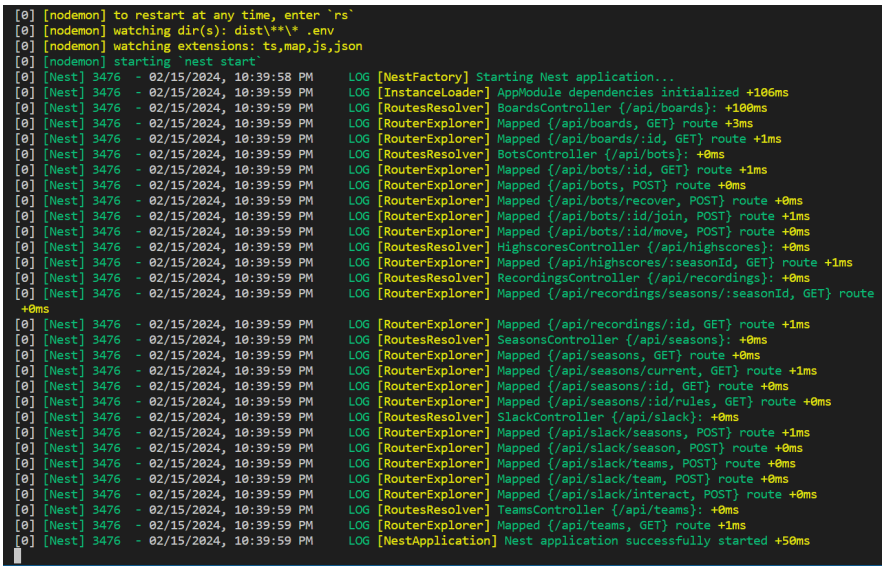
c. *Build*

```
npm run build
```

d. *Run*

```
npm run start
```

Jika berhasil, tampilan terminal akan terlihat seperti gambar di bawah ini.



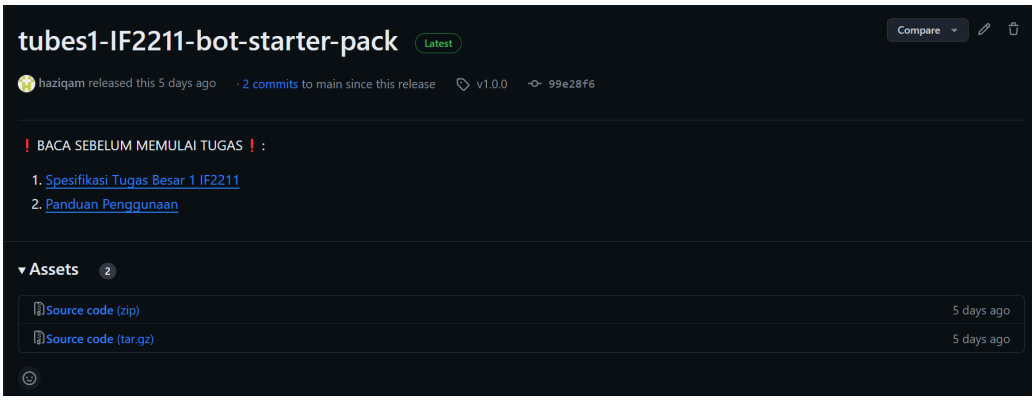
Kunjungi *frontend* melalui http://localhost:8082/. Berikut adalah tampilan awal *frontend*.

2.2.2 Cara Menjalankan Bot

a. *Requirement* yang harus di-install
   - Python (https://www.python.org/downloads/)

b. Instalasi dan konfigurasi awal

1) Download source code (.zip) pada release bot starter pack



2) Extract zip tersebut, lalu masuk ke folder hasil extractnya dan buka terminal

3) Masuk ke root directory dari project (sesuaikan dengan nama rilis terbaru)

```
cd tubes1-IF2110-bot-starter-pack-1.0.1
```

4) Install dependencies menggunakan pip

```
pip install -r requirements.txt
```

c. *Run*

Untuk menjalankan satu bot (pada contoh ini, kita menjalankan satu bot dengan logic yang terdapat pada file game/logic/random.py)

```
python main.py --logic Random --email=your_email@example.com
--name=your_name --password=your_password --team etimo
```

Untuk menjalankan beberapa bot sekaligus (pada contoh ini, kita menjalankan 4 bot dengan logic yang sama, yaitu game/logic/random.py)

- Untuk windows

```
./run-bots.bat
```

- Untuk Linux / (possibly) macOS

```
./run-bots.sh
```

Kalian dapat menyesuaikan *script* yang ada pada run-bots.bat atau run-bots.sh dari segi **logic yang digunakan, email, nama,** dan **password**

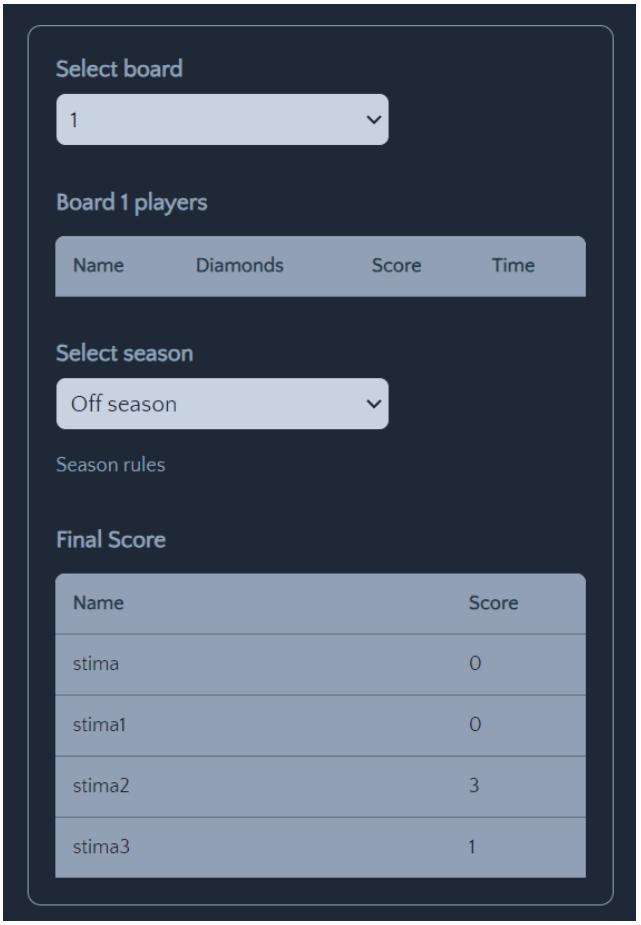
run-bots.bat

```
@echo off
start cmd /c "python main.py --logic Random --email=test@email.com --name=stima --password=123456 --team etimo"
start cmd /c "python main.py --logic Random --email=test1@email.com --name=stima1 --password=123456 --team etimo"
start cmd /c "python main.py --logic Random --email=test2@email.com --name=stima2 --password=123456 --team etimo"
start cmd /c "python main.py --logic Random --email=test3@email.com --name=stima3 --password=123456 --team etimo"

```

Perhatikan bot yang telah bergabung melalui *frontend*

Saat permainan selesai, final score akan muncul pada sisi kanan bawah.

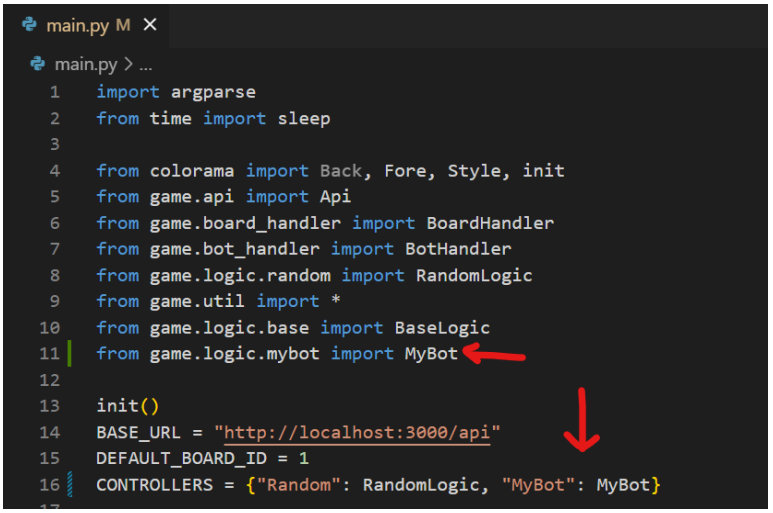


2.2.3 Cara Implementasi Bot

1) Buatlah folder baru pada direktori /game/logic (misalnya mybot.py)
2) Buatlah kelas yang meng-inherit kelas BaseLogic, lalu implementasikan constructor dan method next_move pada kelas tersebut

```
mybot.py U
game > logic > mybot.py > ...
1   from game.logic.base import BaseLogic
2   from game.models import Board, GameObject
3
4
5   class MyBot(BaseLogic):
6       def __init__(self):
7           # Initialize attributes necessary
8           self.my_attribute = 0
9
10      def next_move(self, board_bot: GameObject, board: Board):
11          # Calculate next move
12          delta_x = 1
13          delta_y = 0
14          return delta_x, delta_y
15
```

3) Import kelas yang telah dibuat pada main.py dan daftarkan pada dictionary CONTROLLERS



4) Jalankan program seperti step c pada bagian 2 (sesuaikan argumen logic pada command/script tersebut menjadi nama bot yang telah terdaftar pada CONTROLLERS). Anda bisa menjalankan satu bot saja atau beberapa bot menggunakan .bat atau .sh script.

```
python main.py --logic MyBot --email=your_email@example.com
--name=your_name --password=your_password --team etimo
```

2.3 Implementasi Algoritma Greedy ke Dalam Bot

Implementasi algoritma greedy yang diterapkan pada bot kami dilakukan pada satu kelas yang menjadi wadah *method* logika bot. Salah satu metode bernama nextmove mengimplementasikan algoritma greedy dengan langkah yang paling menguntungkan untuk mendapatkan diamond terbanyak. Pemaparan algoritma greedy yang digunakan pada bot ini akan dipaparkan lebih detail pada Bab 3.

Pada metode nextmove, proses akan dilakukan sesuai prioritas yang telah kami tentukan. Proses yang awal dilakukan pada metode ini adalah melakukan iterasi pengecekan objek pada papan. Lalu, bot akan memfokuskan mencari diamond terdekat dengan mempertimbangkan jarak manhattan. Berikutnya dia juga memeriksa apakah memasuki teleport adalah hal yang menguntungkan. Sebagai karakteristik khas dari algoritma greedy, bot kami akan terus memperhatikan kondisi permainan saat ini, tidak memiliki kemampuan untuk memperoleh pengetahuan dari tindakan yang telah dilakukan pada kondisi permainan

sebelumnya, dan juga tidak mampu mengantisipasi kondisi permainan yang akan muncul selanjutnya.

BAB III

Aplikasi Algoritma Greedy

3.1 Alternatif Solusi Greedy

3.1.1 Alternatif Greedy berdasarkan Langkah ke Objek terdekat

Algoritma Greedy mencari posisi objek terdekat dengan bot, Algoritma ini diimplementasikan untuk mencari arah pergerakan bot. Proses pencarian dilakukan berdasarkan jarak sebelum bot mulai bergerak terhadap objek terdekat. Algoritma greedy ini bertujuan mengambil diamond sebanyak-banyaknya dengan mengestimasi jarak antara bot dan objek. Selain itu konsep greedy ini memungkinkan meminimalisir waktu pengumpulan diamond dan efektivitas pergerakan bot. Objek yang dipilih bisa berupa diamond dan teleport.

a. Proses Mapping
    - Himpunan kandidat : Semua command yang tersedia
    - Himpunan Solusi: Semua command yang dipilih
    - Fungsi Solusi: Memeriksa command yang dipilih memiliki jarak terdekat antara bot dan diamond
    - Fungsi Seleksi: Memilih command yang akan dilakukan dengan parameter jarak terdekat antara bot dan diamond
    - Fungsi Kelayakan: Memeriksa apakah command yang dipilih valid dan dapat dilakukan bot
    - Fungsi Objektif:Mencari langkah terbaik berdasarkan jarak antara bot dan diamond
b. Analisis Efisiensi

    Algoritma ini menggunakan konsep target position sebagai titik acuan pergerakan bot. Bot akan menjadikan posisi objek terdekat sebagai goal position, setiap mencapai goal, goal bot akan mereset ke ‘None’ yang berarti

bot tidak memiliki target. setelah itu, bot akan mencari lagi posisi diamond terdekat sampai diamond penuh. algoritma menggunakan skema list of objek yang didalamnya terdapat tipe objek dan posisi objek yang akan digunakan sebagai parameter. setiap objek diperiksa jarak terdekat dengan fungsi yang mencari jarak petak langkah bot ke objek lalu setiap jarak yang didapatkan dibandingkan jaraknya dicari yang terdekat. Algoritma memeriksa tipe objek dan memiliki pemeriksaan lebih lanjut jika yang terpilih teleport. Jika bot sudah memiliki 4 diamond proses pencarian diamond mengecualikan diamond merah. Kompleksitas dari algoritma ini adalah T(n) = O(n).

c. Analisis Efektivitas

Algoritma ini efektif jika :

- Diamond selalu berdekatan
- Terdapat banyak diamond dekat rumah

Algoritma ini tidak efektif jika:

- Diamond terdekat tidak terdapat banyak diamond di dekatnya
- Bot musuh lebih dekat daripada diamond yang kita sasar

3.1.2 Alternatif Greedy berdasarkan Pencarian Diamond melalui Teleport

Algoritma greedy ini mencari teleport yang efektif diambil dengan parameter banyak diamond di sekitar teleport. proses pencarian objek teleport menggunakan algoritma greedy 3.1.1. ketika tipe objek terdeteksi teleport algoritma langsung mendeteksi daerah objek-objek terdekat teleport dengan jarak tertentu dengan parameter petak langkah. Objek yang masuk dalam jangkauan akan dihitung. jika jumlah objek yang terhitung memenuhi batas yang ditentukan, bot akan langsung memasang posisi teleport sebagai target. Bot tidak akan kembali ke teleport sebelum mengambil paling tidak 2 diamond kecuali dalam kondisi tas penuh.

a. Proses Mapping:

- Himpunan kandidat : Semua command yang tersedia
- Himpunan Solusi: Semua command yang dipilih
- Fungsi Solusi: Memeriksa command yang dipilih memiliki jarak terdekat ke diamond melalui teleport
- Fungsi Seleksi: Memilih command yang akan dilakukan dengan parameter jarak terdekat ke diamond melalui teleport
- Fungsi Kelayakan: Memeriksa apakah command yang dipilih valid dan dapat dilakukan bot
- Fungsi Objektif: Mencari jalan menuju base terbaik berdasarkan jarak bot ke diamond melalui teleport

b. Analisis efisiensi

Algoritma ini menggunakan jumlah diamond di pintu keluar teleport sebagai acuan. Algoritma greedy ini bertujuan untuk mengefisiensi jarak memperoleh diamond.Dalam beberapa kondisi dibanding bot harus berjalan jauh dari ujung board ke ujung board jika terdapat portal yang meungkinkan meraih diamond memasuki portal dapat meminimalisir jumlah langkah. dengan memberikan perhitungan jumlah diamond yang terdapat disekitar portal memungkinkan bot meraih diamond lebih banyak dan pengambilan portal tidak. kompleksitas algoritma ini O(n^2)

c. Analisis Efektivitas

Algoritma greedy Efektif jika :

- Jumlah diamond terdekat tersisa sedikit
- Tidak ada yang menekan tombol reset
- jumlah diamond yang dikumpulkan setelah teleport penuh

Algoritma greedy tidak Efektif jika:

- Diamond yang diambil tidak mencukupi tas
- terdapat diamond yang banyak di area sekitar bot

3.1.3 Alternatif Greedy berdasarkan Penentuan Jalan Menuju Base melalui Teleport

Algoritma greedy ini akan menghitung jarak dari bot ke base. Proses ini akan mengecek dulu dimana posisi teleport berada. Setelah didapat posisi teleport pertama maka akan dicari kembali posisi teleport kedua dimana pintu keluarnya. Setelah mendapatkan kedua teleportnya maka akan dihitung dahulu dari bot menuju base apakah lebih dekat melalui teleport atau tidak, jika lebih cepat maka masuk dan jika tidak maka tidak masuk.

a. Proses Mapping
    - Himpunan kandidat : Semua command yang tersedia
    - Himpunan Solusi: Semua command yang dipilih
    - Fungsi Solusi: Memeriksa command yang dipilih memiliki jarak terdekat ke base melalui teleport
    - Fungsi Seleksi: Memilih command yang akan dilakukan dengan parameter jarak terdekat ke base melalui teleport
    - Fungsi Kelayakan: Memeriksa apakah command yang dipilih valid dan dapat dilakukan bot
    - Fungsi Objektif: Mencari jalan menuju base terbaik berdasarkan jarak bot ke base melalui teleport
b. Analisis Efisiensi

    Algoritma ini menggunakan pendekatan goal position dan posisi bot sekarang sebagai acuan. Jika goal position diset pada base, maka program akan langsung mengecek posisi teleport berada. Setelah dicari dan didapat, maka program akan menghitung dan membandingkan jarak antara bot menuju base melalui teleport dan tidak melalui teleport. Proses perhitungan jarak dilakukan dengan menggunakan jarak manhattan. Setelah dilakukan perhitungan ,jika dihasilkan melalui teleport akan lebih cepat maka boolean isTeleport akan bernilai “true” dan perhitungan next_move akan berfokus lebih dahulu untuk masuk ke teleport. Jika sudah masuk ke teleport maka isTeleport akan berubah menjadi nilai “false” dan bot akan kembali menuju base. Namun jika didapatkan bahwa tidak melalui teleport akan lebih cepat maka program akan langsung menuju base. kompleksitas algoritma ini O(n^2)

c. Analisis Efektivitas

Algoritma ini efektif jika :

- Posisi teleport dekat dengan base
- Posisi teleport dari bot juga dekat

Algoritma ini tidak efektif jika:

- Teleport masuknya jauh
- dan Teleport keluarnya juga tidak dekat dengan base

3.1.4 Alternatif Greedy Berdasarkan jarak ke base dengan kondisi tertentu

Algoritma greedy menghitung jarak base dengan bot dengan kondisi ketika bot telah mengantongi diamond sebanyak 3. pada proses ini Algoritma akan memeriksa jarak bot ke base dengan parameter langkah petak. Algoritma akan memilih jarak terdekat antara jarak bot ke base dan jarak bot ke objek lain selain base. Jika jarak ke base lebih dekat ke rumah maka bot akan langsung menarget base sebagai destinasi tujuan.

a. Proses Mapping
    - Himpunan kandidat : Semua command yang tersedia
    - Himpunan Solusi: Semua command yang dipilih
    - Fungsi Solusi: Memeriksa command yang dipilih memiliki jarak terdekat bot ke base ketika keadaan tertentu
    - Fungsi Seleksi: Memilih command yang akan dilakukan dengan parameter jarak terdekat bot ke base ketika kondisi tertentu
    - Fungsi Kelayakan: Memeriksa apakah command yang dipilih valid dan dapat dilakukan bot
    - Fungsi Objektif: Mencari pergerakan bot terbaik ke base pada kondisi tertentu

b. Analisis Efisiensi

Konsep greedy yang dipakai dalam algoritma ini dengan tujuan mengamankan diamond yang telah dikumpulkan. Demi menghemat waktu dan efektivitas algoritma mempertimbangkan jarak ke base dengan jarak ke objek terdekat. algoritma greedy ini sangat efisien dalam kondisi ketika diamond bersisah berjauhan dengan base agar diamond yang sudah dikantongi tidak menghilang karena ditekel lawan bot lebih memilih balik ke rumah terlebih dahulu untuk mengamankan diamond. Mencari posisi base tidak memerlukan proses loop yang panjang karena terdapat class base. Namun pencarian objek terdekat dilakukan sama dengan Algoritma 3.1.1. Kompleksitas dari algoritma ini adalah T(n) = O(n)

c. Analisis Efektivitas

Algoritma ini efektif jika:

- posisi base dekat sangat dekat dengan bot
- terdapat banyak bot musuh di sekitar

Algoritma ini tidak efektif jika:

- terdapat diamond yang cukup untuk memenuhi tas
- perbedaan jarak antara ke base dan diamond tidak terlalu jauh

3.1.5 Alternatif Greedy Berdasarkan Diamond yang Sudah diperoleh

Algoritma greedy ini Memaksa pulang bot jika waktu permainan tersisah 10 detik dan tas bot berisi paling tidak 2 diamond. algoritma ini mencegah bot terlena dengan waktu hingga tidak sempat pulang. Dengan algoritma ini dapat mengoptimalkan pendapatan diamond dalam permainan.

a. Proses Mapping
    - Himpunan kandidat : Semua command yang tersedia
    - Himpunan Solusi: Semua command yang dipilih
    - Fungsi Solusi: Memeriksa command yang dipilih memiliki jarak terdekat bot ke base ketika keadaan tertentu

- Fungsi Seleksi: Memilih command yang akan dilakukan dengan parameter jarak terdekat bot ke base ketika kondisi tertentu
- Fungsi Kelayakan: Memeriksa apakah command yang dipilih valid dan dapat dilakukan bot
- Fungsi Objektif: Mencari pergerakan bot terbaik ke base pada kondisi tertentu

b. Analisis Efisiensi

Algoritma ini menggunakan sisa waktu permainan sebagai parameter acuan. Jadi ketika permainan tersisah 10 detik maka bot akan langsung berubah target menjadi base. algoritma ini mengoptimalkan pendapatan diamond bot dalam permainan karena mencegah kemungkinan bot tidak sempat kembali. Kemungkinan bot tidak sempat kembali karena bot berjalan terlalu menjauh dari base.

c. Analisis Efektivitas

Algoritma greedy efektif jika :

- bot mengambil diamond sebanyak mungkin
- tidak terdapat posisi diamond yang berdekatan dengan bot saat sebelum pulang

Algoritma greedy tidak efektif jika:

- Terdapat diamond dekat bot yang seharusnya waktu cukup untuk mengambil
- bot hanya mengantongi 2 diamond

3.2 Solusi Greedy yang Diterapkan dan Pertimbangannya

Setiap solusi yang telah dijelaskan memiliki keunggulan dan kelemahan tertentu. Karena setiap ronde permainan memiliki situasi yang tidak pasti, kami memutuskan bahwa bot harus mampu mempertimbangkan pendekatan yang sesuai dengan kondisi yang sedang terjadi dan memilih tindakan yang paling tepat berdasarkan pendekatan greedy.

Pada implementasi program bot kami, kami memutuskan untuk menggunakan alur pemrograman dengan urutan prioritas dari tertinggi hingga terendah sebagai berikut :

1. Bot mencari diamond terdekat
2. Bot mencari diamond terdekat melalui teleport
3. Bot melewati teleport untuk menempuh jarak ke base lebih dekat
4. Bot segera ke base jika waktu tersisa 10 detik

Hal utama yang kami prioritaskan pertama yaitu pencarian bot terdekat secara langsung. Hal ini kami dapatkan melalui analisis secara berkelanjutan melalui pertandingan bot yang satu dengan yang lainnya. Solusi greedy ini sudah cukup efektif, tetapi jika hanya menerapkan algoritma ini maka bot bisa saja terjebak pada teleport atau rugi waktu karena terlalu lama mengejar diamond yang bisa dibilang cukup jauh dari base.

Lalu, aplikasi greedy yang kami terapkan berikutnya adalah mencari diamond terdekat melalui teleport. Hal ini sangat menguntungkan untuk efisiensi waktu untuk mencapai diamond serta menghindari bot menggunakan teleport dengan cara yang salah atau dengan kata lain merugikan.

Berikutnya, yaitu menggunakan teleport untuk menuju base. Hal ini tentunya untuk mengefisiensikan waktu lagi yang ada demi memanfaatkan teleport yang ada jika titik ujungnya berada di dekat base. Aplikasi greedy ini sudah kami coba terapkan dan cukup memperbesar kemungkinan kemenangan.

Aplikasi greedy yang terakhir yaitu bot ditujukan ke base jika waktu tersisa 10 detik. Hal ini dilakukan untuk memberi waktu bot untuk menyimpan diamond yang ada sebelum

waktu habis yang nantinya bisa saja diamondnya sia-sia jika masih saja terus mencari diamond.

Setelah kami melakukan aplikasi greedy terdapat juga beberapa greedy yang menurut kami kurang efektif, diantaranya yaitu

1. Memfokuskan untuk men tackle lawan

   Pada dasarnya setelah kami melakukan percobaan berulang kali, hal ini justru membuang waktu ketika target bot yang akan kita makan kabur tetapi kita masih menargetkan bot tersebut. Ketika bot target sudah mendapat diamond, kita justru menghabiskan waktu mengejarnya dan tidak memiliki kepastian dalam mentackelnya

2. Menekan tombol reset diamond:

   Pada kasus ini kami berpikir bahwa tombol ini justru juga menambah kemungkinan bot musuh untuk menyusuli bot yang sudah kami simpan

BAB IV

Implementasi dan Pengujian

4.1 Implementasi Algoritma Greedy pada Program

4.1.1 Repositori Github

Implementasi Bot yang telah kami buat dapat dilihat pada link github berikut :

zaidanav/Tubes1_Istiqomah (github.com)

4.1.2 Implementasi Program dalam Pseudocode

4.1.2.1 Prosedur Inisialisasi

```
procedure Inisialisasi(output self) :
    directions ← [(1, 0), (0, 1), (-1, 0), (0, -1)]  // pergerakan bot ke arah kanan, atas, kiri, bawah
   goal_position: Posisi ← nil  // posisi tujuan bot
   teleport: Posisi ← nil  // posisi teleport
   isTeleport ← false  // status teleport, bot masuk atau tidak
   current_direction ← 0  // arah pergerakan bot sekarang
   idTeleport ←1  // id teleport
   isDiamond ← 0 // diamond yang diambil
```

4.1.2.2 Fungsi pytagoras

```
 // Menghitung jarak antara dua titik menggunakan rumus pytagoras
function pytagoras(input x1 : int, input  y1: int, input  x2: int, input  y2: int) → int :
   hasil ← akar dari ((x1 - x2) pangkat 2 + (y1 - y2) pangkat 2)
   → hasil
```

4.1.2.3 Fungsi countSteps

```
// Fungsi untuk menghitung jumlah langkah yang diperlukan untuk mencapai tujuan
function countSteps(output self, input goal: Position,input current: Position) → int :
     → |(goal.x - current.x)| + |(goal.y - current.y)|
```

4.1.2.4 Fungsi count

```
function count(self,board: Board, center, int rad) → int:
    count ← 0
    for each game_object in board.game_objects:
        if game_object.type  "DiamondGameObject" then
            if self.countSteps(center, game_object.position) <= rad then
                count += 1
    return count
```

4.1.2.5 Fungsi Diamond

```
function Diamond(input board_bot: GameObject,input board: Board) → (int,int):
    props ← board_bot.properties
    current_position ← board_bot.position  // posisi bot sekarang
    jarak ← 0 // inisialisasi jarak antara bot dengan diamond terdekat
    x ←  -1 // inisialisasi koordinat x diamond terdekat
    y ←  -1  // inisialisasi koordinat y diamond terdekat

    for each objek in board.game_objects do  // iterasi semua objek di board
        if objek = “DiamondGameObject” then // jika objek adalah diamond
            cek ←  false
            k traversal [0..len(board.diamonds)-1] do  // iterasi semua diamond di board
                if board.diamonds[k].position.x = objek.position.x and
board.diamonds[k].position.y = objek.position.y and board.diamonds[k].properties.points +
props.diamonds > 5 then
                    // jumlah diamond bot = 4 dan terdekat ternyata diamond merah yang 2 poin
maka akan di skip
                    cek ←  true
            if cek then
                continue //skip diamond merah jika akan melebihi 5 poin
            if jarak = 0 then //jika jarak masih 0 maka diinisialisasi dengan diamond yang
pertama ditemukan
                jarak ←  pytagoras(current_position.x, current_position.y, objek.position.x,
objek.position.y)
                x ←  objek.position.x
                y ←  objek.position.y
            else  // jika jarak tidak 0 maka akan di cek apakah jarak dengan diamond yang
ditemukan lebih kecil dari jarak sebelumnya
                if jarak > pytagoras(current_position.x, current_position.y, objek.position.x,
objek.position.y) then
                    // jika ya maka jarak diupdate dengan jarak yang lebih kecil
                    jarak ←  pytagoras(current_position.x, current_position.y, objek.position.x,
objek.position.y)
                    x ←  objek.position.x
                    y ←  objek.position.x

    →(y, x) //mengembalikan koordinat diamond terdekat
```

4.1.2.6 Fungsi NewCheckSekitar

```
funciton NewCheckSekitar(output self ,input board_bot: GameObject, input board:
Board) → (int.int) :
   props ← board_bot.properties
   current_position ← board_bot.position  // posisi bot sekarang
   jarak ← 0 // inisialisasi jarak antara bot dengan diamond terdekat
   x ←  -1 // inisialisasi koordinat x diamond terdekat
   y ←  -1  // inisialisasi koordinat y diamond terdekat

   // iterasi semua objek di board
  for each objek in board.game_objects then
      cek ← false
      if objek =“DiamondGameObject” or  objek = “TeleportGameObject” then
         // Jika objek adalah TeleportGameObject
         if objek = “TeleportGameObject” then
            // Iterasi untuk setiap tele dalam game_objects di papan
            for each tele in board.game_objects di papan do
               // Jika tele adalah TeleportGameObject dan id tele tidak sama dengan id objek:
               if tele = “TeleportGameObject” and tele.id != objek.id then
                  // Jika jumlah diamond di sekitar tele kurang dari 4
                  if count(board, tele.position, 6) < 3 then
                     // jika jumlah diamond di sekitar teleport kurang dari 3 maka akan di skip
                     continue

         else if objek = “DiamondGameObject” then
            // Iterasi untuk setiap k dalam rentang dari 0 hingga panjang dari diamonds di
papan
            k traversal [0..len(board.diamonds)-1] do
               // # iterasi semua diamond di board
               if board.diamonds[k].position.x = objek.position.x and
board.diamonds[k].position.y = objek.position.y and board.diamonds[k].properties.points +
props.diamonds > 5 then
                  cek ← true //jumlah diamond bot = 4 dan terdekat ternyata diamond merah
yang 2 poin maka akan di skip
         if cek then
            continue //skip diamond merah jika akan melebihi 5 poin
         if jarak = 0 then //jika jarak masih 0 maka diinisialisasi dengan diamond yang
pertama ditemukan
            jarak ←  pytagoras(current_position.x, current_position.y, objek.position.x,
objek.position.y)
            x ←  objek.position.x
            y ←  objek.position.y
         else  // jika jarak tidak 0 maka akan di cek apakah jarak dengan diamond yang
ditemukan lebih kecil dari jarak sebelumnya
            if jarak > pytagoras(current_position.x, current_position.y, objek.position.x,
objek.position.y) then
               // jika ya maka jarak diupdate dengan jarak yang lebih kecil
               jarak ←  pytagoras(current_position.x, current_position.y, objek.position.x,
```

```
objek.position.y)
                x ← objek.position.x
                y ← objek.position.x

  →(y, x) //mengembalikan koordinat diamond terdekat
```

4.1.2.7 Fungsi isObjectTeleport

```
function isObjectTeleport(self, board: Board, x, y) → boolean:
        for each game_object in board.game_objects:
              if game_object.position.x == x and game_object.position.y == y and
game_object.type == "TeleportGameObject" then :
                    → True
              → False
```

4.1.2.8 Fungsi isTaken

```
function isTaken(output self,input board: Board, input x : int, input y : int) → boolean :
      // Fungsi untuk mengecek apakah di koordinat x, y objek diamond masih ada atau
sudah diambil
      for each game_object in board.game_objects do
         if game_object.position.x = x and game_object.position.y = y and game_object.type
!= "DiamondGameObject" then
            → True
      → False
```

4.1.2.9 Fungsi isTeleportReset

```
Fungsi isTeleportReset(output self, board: Board,Goal: Position,Id: int):
      # Fungsi untuk mengecek apakah teleport sudah di reset atau belum
      for each objek in board.game_objects:
         if objek.type = "TeleportGameObject" and objek.id = Id then
            if objek.position.x != Goal.x and objek.position.y != Goal.y then
               → True
            else:
               → False
```

4.1.2.10 Fungsi next_move

```
  Fungsi  next_move(self, board_bot: GameObject, board: Board):
      # Fungsi untuk menghitung langkah selanjutnya yang akan diambil oleh bot
      print("Goal Position: ", self.goal_position)
      props ← board_bot.properties
      current_position ← board_bot.position # inisialisasi posisi bot sekarang
      sessionlegth ← props.milliseconds_left # inisialisasi sisa waktu
```

```
        base_x ← board_bot.properties.base.x # inisialisasi koordinat x base
        base_y ← board_bot.properties.base.y # inisialisasi koordinat y base

        if self.isTeleport: # jika bot diarahkan melalui teleport agar lebih dekat menuju base
            print("Teleporting to: ", self.teleport)
            delta_x, delta_y ←
get_direction(current_position.x,current_position.y,self.teleport.x,self.teleport.y) #
menghitung langkah yang diperlukan untuk menuju teleport
            if current_position.x + delta_x = self.teleport.x and current_position.y + delta_y =
self.teleport.y then
                # jika bot sudah berada di teleport maka status teleport di reset
                self.isTeleport ← False
                self.teleport ← None
                self.idTeleport ← -1
                self.goal_position ← None
            else if self.isTeleportReset(board, self.teleport, self.idTeleport) then
                # jika teleport sudah di reset maka status teleport di reset
                self.isTeleport ← False
                self.teleport ← None
                self.idTeleport ← -1
                self.goal_position ← None
            # mengembalikan langkah yang diperlukan untuk menuju teleport
            → delta_x, delta_y

        if self.goal_position = Position(base_y, base_x) then # jika tujuan bot adalah base
            output("Moving to base")
            if self.goal_position.x = board_bot.position.x and self.goal_position.y =
board_bot.position.y then
                # jika bot sudah berada di base maka tujuan bot di reset
                self.goal_position ← None

            else:
                # cek apakah ada teleport yang akan mempercepat bot menuju base
                temp ← False
                for each i, tele1 in board.game_objects:
                    if tele1.type = "TeleportGameObject"  then
                        for each i, tele2 in board.game_objects:
                            if tele2.type = "TeleportGameObject" and tele2.id != tele1.id then
                                if self.countSteps(current_position, tele1.position) +
self.countSteps(tele2.position, Position(base_y, base_x)) <
self.countSteps(current_position, Position(base_y, base_x))then
                                    # jika ada teleport yang mempercepat bot menuju base maka bot
diarahkan menuju teleport
                                    # status teleport di set menjadi true
                                    # id teleport di set menjadi id teleport yang diarahkan
                                    # tujuan bot di set menjadi teleport yang diarahkan
                                    self.teleport ← Position(tele1.position.y, tele1.position.x)
                                    self.isTeleport ← True
```

```
                        self.idTeleport ← tele1.id
                        temp ← True
                        break
                if temp then
                   break

      else if self.goal_position != None and self.goal_position != Position(base_y, base_x)
then
         # jika tujuan bot bukan base
         if self.goal_position.x = board_bot.position.x and self.goal_position.y =
board_bot.position.y then
            # jika bot sudah berada di tujuan maka tujuan bot di reset
            self.goal_position ← None
         else if self.isTaken(board, self.goal_position.x, self.goal_position.y) then
            # jika tujuan bot sudah diambil maka tujuan bot di reset
            self.goal_position ← None

      if sessionlegth < 10000 and props.diamonds > 2 then
         # jika sisa waktu kurang dari 10 detik dan jumlah diamond bot lebih dari 2
         # maka bot diarahkan menuju base
         self.goal_position ← Position(base_y, base_x)

      if self.goal_position = NIL then
         if props.diamonds = 5 then
            # jika jumlah diamond bot = 5
            # maka bot diarahkan menuju base
            self.goal_position ← Position(base_y, base_x)

         else:
            # mencari objek selain diamond jika telah mengambil diamond lebih dari 2
            if self.isDiamond = 2 then
               check ← self.NewCheckSekitar(board_bot, board) # mencari diamond atau
teleport terdekat
               if self.isObjectTeleport(board, check[1], check[0]) then # jika objek terdekat
adalah teleport
                  self.isDiamond ← False
                  self.isTeleport ← True
                  self.teleport ← Position(check[0], check[1]) # tujuan bot di set menjadi
teleport terdekat
                  self.isDiamond ← 0 # status diamond di reset
            else:
               check ← self.Diamond(board_bot, board)
               self.isDiamond ← self.isDiamond + 1 # bot telah mengambil 1 diamond

            if props.diamonds >= 3 then
               # jika jumlah diamond bot lebih dari 3
               if self.countSteps(current_position, Position(check[0], check[1])) <=
self.countSteps(current_position, board_bot.properties.base) then
```

```
                # jika jarak antara bot dengan diamond terdekat lebih kecil dari jarak antara
bot dengan base
                self.goal_position ← Position(check[0], check[1])
                →
get_direction(current_position.x,current_position.y,self.goal_position.x,self.goal_position.y
)
            else:
                # jika jarak antara bot dengan diamond terdekat lebih besar dari jarak antara
bot dengan base
                self.goal_position ← Position(base_y, base_x)
                self.previous_goal ← Position(base_y, base_x)
                →
get_direction(current_position.x,current_position.y,self.goal_position.x,self.goal_position.y
)
        elif check[0] != -1:
            # jika diamond terdekat ditemukan
            self.goal_position ← Position(check[0], check[1])
            self.previous_goal ← Position(check[0], check[1])
            return
get_direction(current_position.x,current_position.y,self.goal_position.x,self.goal_position.y
)

    if self.goal_position.x = -1 and self.goal_position.y == -1 then
        # jika tujuan bot tidak ditemukan
        # maka bot diarahkan menuju red button
        for each i , red in board.game_objects:
            if red.type = "DiamondButtonGameObject" then
                self.goal_position ← Position(red.position.y, red.position.x)
                break
    →
get_direction(current_position.x,current_position.y,self.goal_position.x,self.goal_position.y
)  # mengembalikan langkah yang diperlukan untuk menuju tujuan bot
```

4.2 Struktur Data Program

Pembuatan bot dalam game engine Etimo Diamond V2 ini menggunakan bahasa pemrograman python. Pada pembuatan bot ini dibutuhkan beberapa class dalam implementasinya agar berjalan sesuai yang diharapkan. Class-class yang dibutuhkan adalah sebagai berikut:

a. Class Fantom

Class ini adalah kelas utama dalam penetuan logika dari bot yang kami dibuat. Penentuan dari langkah apa yang harus diambil ada pada kelas ini. Atribut dari kelas ini ada directions, goal_position, teleport, isTeleport, current_direction, idTeleport, dan previous_goal.

b. Class Position

Class ini digunakan untuk representasi posisi game objek atau bot dalam bentuk koordinat pada board permainannya.

c. Class Base

Class ini digunakan untuk representasi posisi base dari bot dalam bentuk koordinat pada board permainan.

d. Class Properties

Class ini digunakan untuk mengakses properties dalam permainan seperti points,pair_id,diamonds,score,name,inventory_size ,can_tackle, millisecond_left,time_joined, dan base.

e. Class GameObject

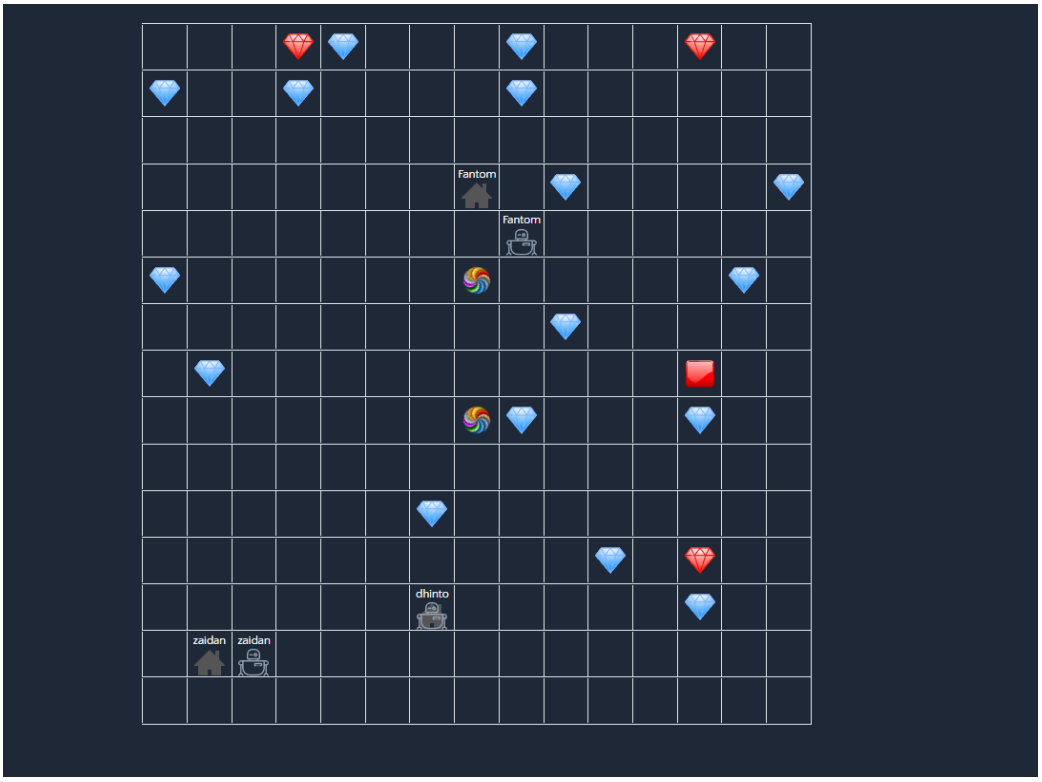
Class ini sebagai representasi dari tiap objeknya, karena class ini berisi type objeknya nya,posisi,id,dan properties.

f. Class Board

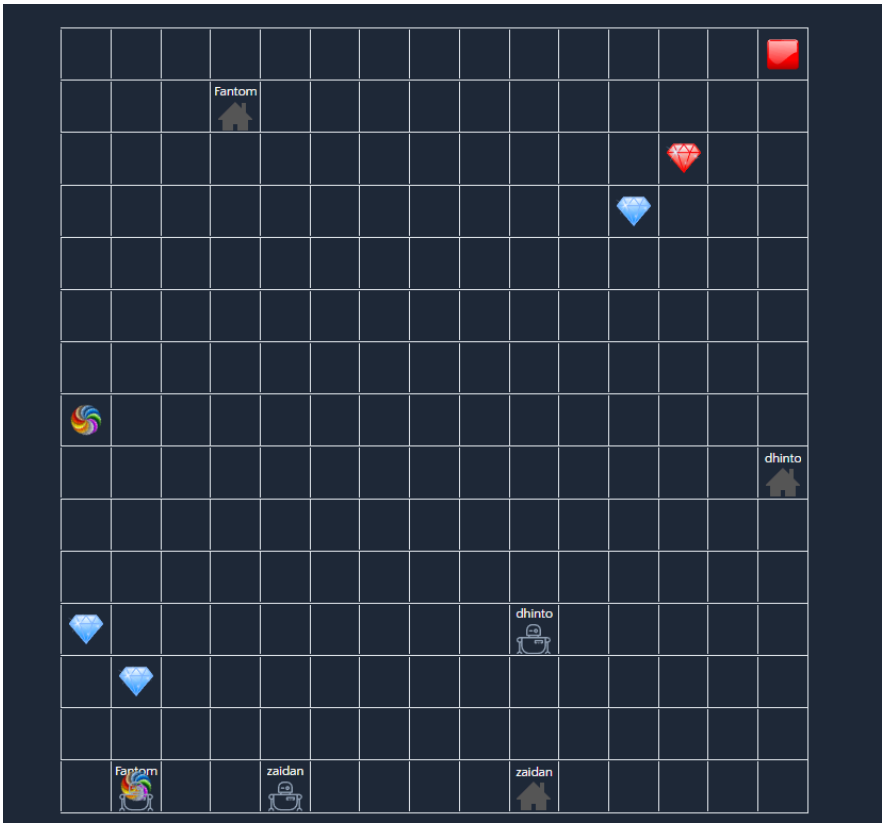
Class ini berisi seluruh elemen elemen dan objek yang ada pada papan permainan.

4.3 Analisis dan Pengujian

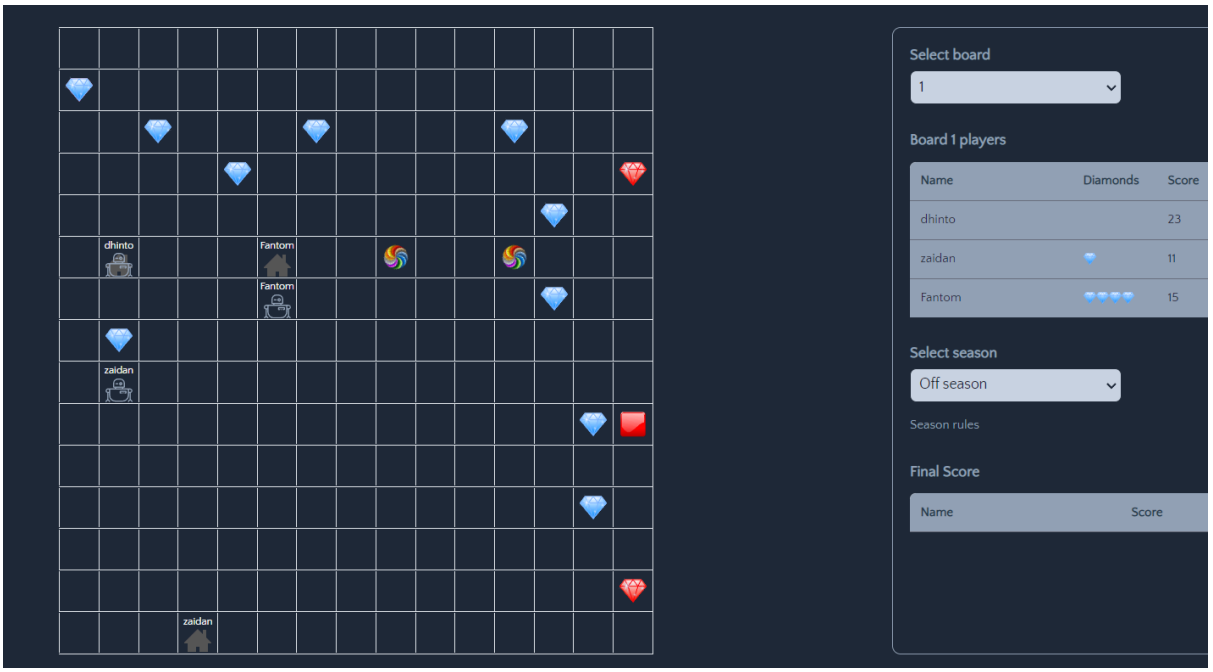
Kami melakukan analisis dan pengujian dengan mengoperasikan permainan dan memasukkan versi terbaru bot kami bersama dengan versi sebelumnya. Selanjutnya, kami mengamati perilaku dan tindakan yang dilakukan oleh bot kami menggunakan visualizer yang disediakan. Berikut adalah ringkasan analisis yang kami lakukan terhadap hasil pertandingan bot kami.



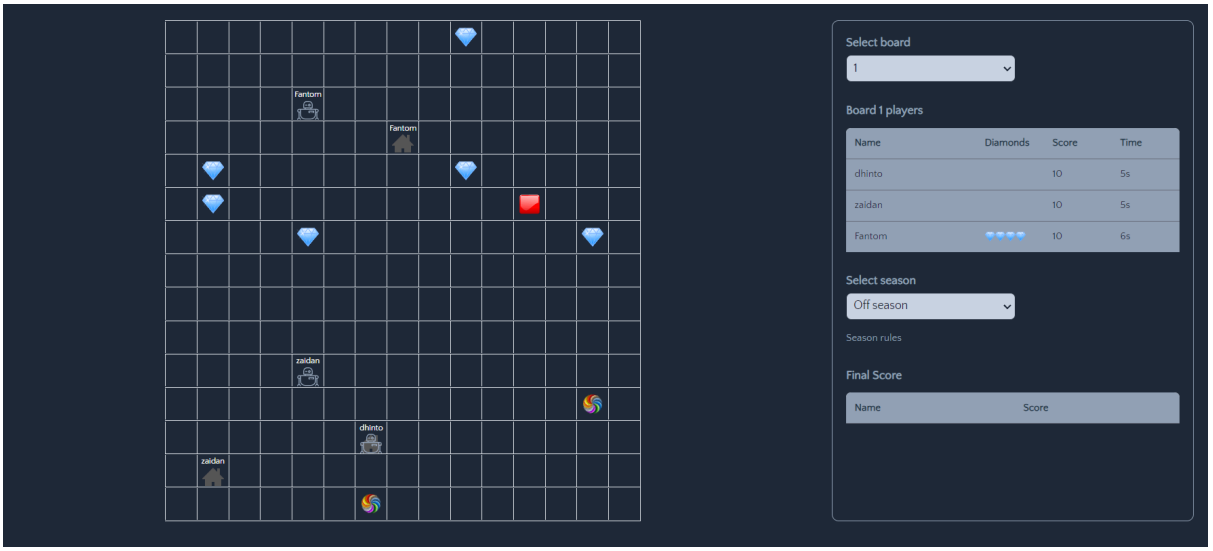
Pada gambar di atas fantom merupakan versi terbaru bot kami yang melakukan aplikasi dari program greedy yang kami buat yaitu mencari diamond terdekat.



Lalu, ketika bertanding dengan bot lain, fantom juga dapat menggunakan teleport untuk mencapai diamond terdekat



Bot juga berhasil menerapkan ketika bot akan pulang menggunakan teleport jika jarak ke base lebih dekat



Terakhir bot dapat mengimplementasikan kondisi ketika harus ke base yaitu 10 detik terakhir.

BAB V

Kesimpulan dan Saran

5.1 Kesimpulan

Dari penyelesaian tugas besar IF2211 Strategi Algoritma, kami berhasil mengimplementasikan bot untuk permainan 'Etimo Diamonds V2' menggunakan strategi Greedy. Pengalaman ini menunjukkan bahwa algoritma Greedy dapat efektif dalam mencapai solusi lokal maksimum atau optimum dari kondisi saat itu. Greedy juga dapat dianggap sebagai algoritma optimasi yang baik dalam pengambilan keputusan untuk pembuatan bot, karena seringkali solusi lokal yang dipilih cukup mendekati solusi optimal global.

Namun, perlu diingat bahwa terkadang algoritma Greedy dapat gagal menemukan solusi optimal global, seperti yang terlihat dalam beberapa pertandingan di mana bot gagal memperoleh keuntungan maksimal akibat lokasi spawn yang tidak menguntungkan. Sebagai alternatif, kita menyadari bahwa exhaustive search merupakan pendekatan lain yang dapat digunakan. Meskipun memakan lebih banyak waktu karena mengecek setiap kemungkinan langkah dan potensi keuntungan dari setiap aksi, exhaustive search dapat memberikan solusi yang lebih optimal.

Oleh karena itu, pemilihan strategi algoritma harus disesuaikan dengan kebutuhan dan karakteristik permainan yang bersangkutan.

5.2 Saran

Beberapa saran untuk kelompok kami termasuk:

1. Memperdalam pemahaman tentang game engine yang digunakan, karena banyak isu yang tidak dapat diselesaikan karena kurangnya pengetahuan tentang game engine tersebut.

2. Memperbaiki perencanaan jadwal pengerjaan Tugas Besar dengan lebih cermat.

3. Meningkatkan jumlah sesi brainstorming antara sesi pengerjaan untuk memperoleh lebih banyak ide dan perspektif.

4. Menghindari mengerjakan bot secara mendekati deadline agar ada lebih banyak waktu untuk menyempurnakan bot.

Lampiran

Link Repositori Github:

zaidanav/Tubes1_Istiqomah (github.com)

Link Video:

https://youtu.be/R8j0P02IXHQ?si=TK-FRg_c3V1m9O1P

Daftar Pustaka

Berikut adalah daftar refrensi yang dipakai dalam pengerjaan tugas besar ini.

1. Algoritma Greedy v1 (itb.ac.id) (Diakses pada 4 Maret, 2024)

2. Algoritma Greedy v2 (itb.ac.id) (Diakses pada 4 Maret, 2024)

3. Algoritma Greedy v3 (itb.ac.id) (Diakses pada 4 Maret, 2024)

4. Etimo/diamonds2: 💎Diamonds v2 (github.com) (Diakses pada 27 Februari, 2024)