LAPORAN TUGAS BESAR I
IF2211 STRATEGI ALGORITMA

Disusun oleh :
Christian Justin Hendrawan (13522135)
Muhammad Fatihul Irhab (13522143)
Ikhwan Al Hakim (13522147)

Program Studi Teknik Informatika
Institut Teknologi Bandung
2023

Daftar Isi

BAB I
DESKRIPSI MASALAH

1.1. Tujuan

Tujuan dari pembuatan tugas besar ini adalah sebagai berikut.

1. Buatlah program sederhana dalam bahasa **Python** yang mengimplementasikan **algoritma Greedy** pada bot permainan Diamonds dengan tujuan memenangkan permainan.
2. 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.

1.2. Spesifikasi

Diamonds merupakan suatu programming challenge yang mempertandingkan bot yang anda buat dengan bot dari para pemain lainnya. Setiap pemain akan memiliki sebuah bot dimana tujuan dari bot ini adalah mengumpulkan diamond sebanyak-banyaknya. Cara mengumpulkan diamond tersebut tidak akan sesederhana itu, tentunya akan terdapat berbagai rintangan yang akan membuat permainan ini menjadi lebih seru dan kompleks. Untuk memenangkan pertandingan, setiap pemain harus mengimplementasikan strategi tertentu pada masing-masing bot-nya. Penjelasan lebih lanjut mengenai aturan permainan akan dijelaskan di bawah.

Gambar 1. Tampilan program Etimo Diamonds 2

Pada tugas pertama Strategi Algoritma ini, mahasiswa diminta untuk membuat sebuah bot yang nantinya akan dipertandingkan satu sama lain. Tentunya mahasiswa harus menggunakan strategi greedy dalam membuat bot ini.

Program permainan Diamonds terdiri atas:

1. Game engine, yang secara umum berisi:
    a. Kode backend permainan, yang berisi logic permainan secara keseluruhan serta API yang disediakan untuk berkomunikasi dengan frontend dan program bot
    b. Kode frontend permainan, yang berfungsi untuk memvisualisasikan permainan
2. Bot starter pack, yang secara umum berisi:
    a. Program untuk memanggil API yang tersedia pada backend
    b. Program bot logic (bagian ini yang akan kalian implementasikan dengan algoritma greedy untuk bot kelompok kalian)
    c. Program utama (main) dan utilitas lainnya

BAB II
DASAR TEORI

2.1. Algoritma Greedy

Algoritma *greedy* merupakan pendekatan yang populer dalam pemecahan persoalan optimasi. Persoalan optimasi bertujuan untuk mencari solusi optimal di antara sejumlah solusi yang mungkin. Algoritma greedy menyelesaikan persoalan secara langkah demi langkah, dimana pada setiap langkah, kita memilih pilihan yang tampaknya paling menguntungkan pada saat itu, tanpa mempertimbangkan konsekuensi di masa depan. Harapannya adalah dengan memilih solusi lokal yang optimal pada setiap langkah, kita akan mencapai solusi global yang optimal.

Penting untuk dicatat bahwa pilihan solusi optimal lokal tidak selalu menghasilkan solusi global yang optimal. Ini dapat terjadi karena algoritma greedy tidak mempertimbangkan semua kemungkinan solusi secara menyeluruh dan atribut yang dipertimbangkan tidak tepat dalam menghasilkan solusi optimal. Walaupun algoritma greedy tidak selalu menjamin solusi yang optimal, namun solusi yang diberikan algoritma ini dapat menghasilkan solusi hampiran (*approximation*) yang mana lebih menguntungkan daripada menggunakan algoritma yang kebutuhan waktunya eksponensial untuk menghasilkan solusi yang eksak.

2.2. Elemen Algoritma Greedy

Terdapat beberapa elemen algoritma *greedy*, seperti :

1. Himpunan kandidat (C)
   Himpunan ini berisi kandidat yang akan dipilih pada setiap langkah.
2. Himpunan solusi (S)
   Himpunan yang berisi kandidat yang sudah dipilih
3. Fungsi solusi
   Fungsi ini akan menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi
4. Fungsi seleksi (*selection function*)
   Fungsi yang akan memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik.
5. Fungsi kelayakan (*feasible*)
   Fungsi akan memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak)
6. Fungsi obyektif
   Fungsi ini menjelaskan apakah untuk memaksimumkan atau meminimumkan

2.3. Game Engine Etimo Diamond V2

Untuk memulai menjalankan *game* Etimo Diamond V2 dan mengimplementasikan bot, diperlukan game engine Diamond. Game engine itu sendiri dapat diunduh melalui ini release game engine dan juga diperlukan implementasi bot yang dapat diunduh melalui release bot starter pack. Selain itu, dibutuhkan prasyarat :

1. Node.js
2. Docker Desktop
3. Yarn
4. Python

Untuk memahami cara kerja game, maka terdapat beberapa komponen yang perlu diketahui. Komponen-komponen tersebut adalah sebagai berikut.

1. File Docker

Pada repository game-engine, terdapat 3 file Docker yakni : docker-compose.prod-run.yml,docker-compose.yml, dan docker-compose.prod-build.yml lalu fungsi dari ketiga file docker compose tersebut adalah untuk membangun, menjalankan, dan mengelola aplikasi dalam lingkungan yang terisolasi dan dapat direproduksi, baik di mesin lokal maupun di server produksi.

2. File Prisma

Pada repository game-engine, terdapat folder backend yang memuat folder Prisma. Folder ini bertanggung jawab untuk mendefinisikan struktur database aplikasi, mengisi database aplikasi dengan data awal, dan memungkinkan aplikasi berinteraksi dengan database.

3. Game Engine

Pada repository game-engine, terdapat folder backend yang memuat folder src. Terdapat 3 komponen utama dalam engine ini:

a. Game Object Provider : Fungsi-fungsi ini bereaksi terhadap event di sekitar papan permainan, seperti saat papan diinisialisasi dan saat objek ditambahkan atau dihapus. Semua provider harus mewarisi kelas abstrak AbstractGameObjectProvider. Provider bertanggung jawab utama untuk menghasilkan/menghapus objek game. Misalnya, saat papan diinisialisasi, mereka dapat menghasilkan set objek game berlian yang dapat dikumpulkan.
b. Game Objects: Ini adalah objek aktual yang ditempatkan di papan. Objek game memiliki posisi, mereka dapat bertindak, mereka dapat bertabrakan, mereka dapat bereaksi terhadap objek game lain, dll. Semua objek game

melakukan tindakan menggunakan papan yang ditugaskan sebagai mediator. Semua objek game harus mewarisi kelas abstrak AbstractGameObject.
Game Object yang tersedia pada aplikasi ini adalah berikut.

- Base
- Bot
- Diamond
- Diamond button
- Teleport

c. Boards: Papan dibuat menggunakan konfigurasi dan set provider objek game ("Fitur"). Ini memudahkan pembuatan papan yang berbeda dengan fitur yang berbeda yang diaktifkan. Papan adalah tempat di mana semua objek game ditempatkan dan berinteraksi satu sama lain.

4. Runner

Pada repository game-engine, terdapat folder backend yang memuat folder dist yang memiliki banyak kegunaan, salah satu kegunaannya adalah memulai aplikasi dan memulai server. Namun, ada juga terdapat runner pada main.py di bot-starter-pack yang berguna untuk melakukan set up dan mengontrol bot yang bermain dalam game, membaca argument command line, dan mengandung game loop utama dimana bot menentukan arah gerak selanjutnya.

Struktur file game-engine secara umum yaitu :

- Folder packages : berisi backend, frontend, dan types aplikasi
- Folder scripts : Skrip-skrip yang mengotomatiskan berbagai tugas terkait pembangunan, penyebaran, dan penyiapan aplikasi game
- File-file yarn : digunakan untuk menyesuaikan bagaimana Yarn bekerja untuk aplikasi, termasuk linker yang digunakan, plugin yang dimuat, dan versi Yarn yang digunakan.
- File-file Docker : membangun, menjalankan, dan mengelola aplikasi

2.4. Game Object Etimo Diamond V2

Dalam permainan Etimo Diamond V2 terdapat sejumlah game object yang memiliki peran penting dalam mengubah alur permainan. Oleh karena itu, penting untuk membahas dengan lengkap kegunaan masing-masing game object agar pemain dapat memahami dampaknya dalam permainan. Berikut adalah daftar game object yang ada dalam permainan Etimo Diamond V2.

1. Bot

Dalam permainan Etimo Diamond V2, pengalaman bermain disampaikan melalui bot yang bertindak sesuai dengan logika yang telah kita tentukan.

Bot ini memiliki sejumlah properti yang menjadi panduan dalam merancang strategi permainan. Properti-proper ini meliputi posisi dasar (base), jumlah berlian (diamonds), ukuran inventaris (inventorySize), kemampuan untuk melakukan tackle (canTackle), skor (score), nama (name), waktu gerakan berikutnya yang tersedia (nextMoveAvailableAt), dan ID bot (botId). Tingkat kemenangan dalam permainan bergantung pada seberapa efisien bot kita dalam mengumpulkan berlian. Oleh karena itu, semakin banyak berlian yang berhasil dikumpulkan oleh bot, semakin besar pula peluang kita untuk memenangkan permainan

2. Base
   Masing-masing bot dalam permainan memiliki 1 base. Pada awal permainan, base akan diinisialisasi dan posisinya tidak akan pindah sampai akhir 1 sesi permainan. Kegunaan base adalah tempat di mana bot dapat menyetor berlian yang mereka kumpulkan. Jika bot yang memasuki Base adalah bot yang sama dengan bot yang memiliki Base, maka semua berlian yang dibawa bot akan ditambahkan ke skor bot dan jumlah berlian di bot akan diatur ke 0.

3. Diamond
   Diamond berperan sebagai poin penentu dalam menentukan pemenang permainan. Terdapat dua jenis diamond dalam permainan: diamond yang memberikan 2 poin dan yang memberikan 1 poin. Ketika bot mencapai posisi diamond, diamond tersebut akan ditambahkan ke dalam inventaris bot asalkan masih ada ruang kosong di inventarisnya. Jumlah berlian yang dimiliki oleh bot akan bertambah sesuai dengan nilai poin dari diamond yang diambil, dan diamond tersebut akan dihapus dari papan permainan. Pada awal permainan, posisi diamond akan diinisialisasi, dan ketika semua diamond telah dihapus atau jika ada bot yang mencapai diamond button, diamond-diamond baru akan diinisialisasi kembali dengan posisi yang acak.

4. Diamond Button
   Diamond button merupakan salah satu game object yang memiliki kemampuan untuk mengubah alur permainan secara drastis berkat perannya yang sangat unik. Cara untuk mengaktifkan diamond button adalah dengan menginjak posisi diamond button tersebut. Peran dari diamond button adalah untuk mengubah atau mereset semua posisi diamond yang ada pada papan permainan saat ini, dan kemudian

menginisialisasi diamond-diamond baru dengan posisi yang acak. Setelah diamond button dipencet atau diinjak, posisinya akan berpindah ke lokasi lain.

5. Teleporter
   Teleporter adalah game object yang memiliki dualitas, karena dapat membantu mendekatkan atau menjauhkan posisi bot dari diamond. Pada setiap sesi permainan, terdapat sepasang teleporter yang posisinya acak. Peran dari teleporter, sesuai dengan namanya, adalah untuk mengirimkan bot dari satu teleporter ke teleporter lainnya. Cara mengaktifkan teleporter adalah dengan membuat bot masuk ke salah satu teleporter yang ada di papan permainan. Yang membuat teleporter unik adalah bahwa dalam setiap sesi permainan, kedua teleporter tersebut akan berpindah tempat secara periodik, menjadikannya berbeda dari game object lainnya.

2.5. Cara Menjalankan Program

Berikut adalah langkah-langkah untuk menjalankan program:

1. Melakukan instalasi Node.js, Docker Desktop, dan Yarn
2. Melakukan instalasi release game engine
3. Extract zip tersebut, lalu masuk ke folder hasil extractnya dan buka terminal
4. Masuk ke root directory dari project (sesuaikan dengan nama rilis terbaru) :

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

5. Install dependencies menggunakan Yarn :

```
yarn
```

6. Setup default environment variable dengan menjalankan script berikut
   Untuk Windows :

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

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

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

8. Lalu jalankan script berikut. Untuk Windows :

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

9. Ketik `npm run build` pada terminal game-engine

```
npm run build
```

10. Setelah itu masukkan

```
npm run start
```

11. Kunjungi *frontend* melalui http://localhost:8082/

Cara melakukan set up bot:

1. Requirement yang harus di instal adalah python.
2. Download source code (.zip) pada release bot starter pack
3. Extract zip tersebut, lalu masuk ke folder hasil extractnya dan buka terminal
4. Masuk ke root directory dari project (sesuaikan dengan nama rilis terbaru) : `cd tubes1-IF2110-bot-starter-pack-1.0.1`
5. Install dependencies menggunakan pip : `pip install -r requirements.txt`

Cara menjalankan bot:

1. 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
```

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

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

4. 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"

```

Cara membuat bot logic

1. Buka IDE yang kalian miliki, lalu bukalah folder tubes1-IF2211-bot-starter-pack-1.0.1

2. Lalu pergi ke bagian folder game -> folder logic dan buatlah file python dimana kamu dapat mengimplementasikan logika bot yang kamu ingini, selain itu kamu juga dapat melihat logika pada random.py sebagai referensi.

Cara mengimpor bot logic

1. Buka IDE yang kalian miliki, lalu bukalah folder tubes1-IF2211-bot-starter-pack-1.0.1
2. Setelah itu, pergi ke bagian main.py dan seharusnya terlihat CONTROLLERS yang hanya terdapat 1 logic seperti ini

```
import argparse
from time import sleep

from colorama import Back, Fore, Style, init
from game.api import Api
from game.board_handler import BoardHandler
from game.bot_handler import BotHandler
from game.logic.random import RandomLogic
from game.util import *
from game.logic.base import BaseLogic

init()
BASE_URL = "http://localhost:3000/api"
DEFAULT_BOARD_ID = 1
CONTROLLERS = {
    "Random": RandomLogic,
}
```

3. Import bot logic kalian ke file main, setelah itu masukkan bot logic kalian ke CONTROLLERS, jangan lupa juga untuk memberi nama seperti berikut.

```
from game.logic.base import BaseLogic
from game.logic.uzi import Uzi

init()
BASE_URL = "http://localhost:3000/api"
DEFAULT_BOARD_ID = 1
CONTROLLERS = {
    "Random": RandomLogic, "Uzi" : Uzi
}
```

4. Lakukan run bot seperti yang sudah dijelaskan di atas, jangan lupa mengganti nama bot, email, serta password.

2.6. Implementasi Algoritma Greedy Ke Dalam Bot

Implementasi algoritma greedy ke dalam bot pada kelompok kami dilakukan dengan membuat sebuah kelas bernama Pulang yang menjadi wadah bagi metode-metode logika bot. Salah satu metode yang dinamai purge akan mengimplementasikan algoritma greedy dengan selalu memilih opsi yang tampaknya paling menguntungkan pada setiap langkahnya. Pemilihan yang dilakukan oleh bot bersifat greedy berdasarkan jarak diamond ke bot dan nilai diamond yang akan dijelaskan lebih lanjut pada Bab 3.

Untuk metode purge, semua proses yang perlu dilakukan akan dijalankan. Prosesnya adalah bot akan melakukan iterasi untuk setiap diamond pada papan permainan (board). Kemudian, untuk masing-masing diamond akan dihitung rasio. Rasio ini merupakan hasil dari diamond point / jarak. Jarak yang dihitung adalah jarak Manhattan terdekat dari posisi bot ke diamond. Setelah rasio untuk semua diamond didapatkan, bot akan memilih rasio terbesar sebagai aksi yang dinilai paling optimal saat itu dan menjalankannya. Sebagai ciri khas dari algoritma greedy, bot kami akan selalu melihat game state saat ini, tidak dapat belajar dari hasil aksi yang dilakukan pada game state sebelumnya, dan juga tidak dapat memprediksi game state yang akan datang.

BAB III
APLIKASI STRATEGI GREEDY

3.1 Alternatif Solusi Greedy

3.1.1 Algoritma Greedy Berdasarkan Diamond Terdekat

Algoritma greedy berdasarkan diamond terdekat merupakan metode untuk menentukan pergerakan bot berdasarkan jarak real-time antara bot dan diamond. Algoritma ini secara terus-menerus mengevaluasi posisi diamond untuk menemukan yang terdekat, dan kemudian menetapkan posisi tersebut sebagai target pergerakan bot. Sehingga proses algoritma ini berfokus pada pengambilan keputusan berdasarkan kriteria jarak terpendek antara bot dan diamond, yang mengoptimalkan pencapaian tujuan. Dengan demikian, bot akan selalu bergerak menuju diamond terdekat, meminimalkan waktu tempuh dan meningkatkan efisiensi pengumpulan diamond.

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 pergerakan robot terbaik berdasarkan jarak antara bot dan diamond

b. Analisis Efisiensi

    Algoritma ini menggunakan dua pendekatan yaitu mencari posisi diamond dan balik ke base keduanya menggunakan target position sebagai parameter pergerakan arah bot. Bot akan balik ke base ketika bot sudah mengumpulkan diamond sebanyak kapasitas bot, lalu target position dari bot akan diganti dengan posisi dari base bot tersebut. Mencari diamond dengan parameter diamond terdekat, menggunakan algoritma dengan skema seperti ini yaitu program akan mengambil list posisi diamond yang ada di board, lalu melakukan perulangan setiap posisi diamond yang ada di list tersebut dan menghitung jarak dari posisi bot saat ini. Sehingga akan didapatkan posisi diamond dengan jarak terdekat dengan bot, lalu posisi diamond tersebut akan dijadikan sebagai target position untuk arah pergerakan bot tersebut. Kompleksitas dari algoritma ini adalah $T(n) = O(n)$.

c. Analisis Efektivitas

Strategi ini efektif apabila:

- Di dalam board hanya ada satu bot saja yang berjalan
- Diamond didalam board saling berdekatan

Strategi ini tidak efektif apabila:

- Di dalam board terdapat bot lainnya yang jaraknya lebih dekat dengan diamond yang kita incar
- Terdapat banyak diamond bernilai 2 di board

3.1.2 Algoritma Greedy Berdasarkan Jarak Setiap Bot Terhadap Diamond

Algoritma greedy berdasarkan jarak setiap bot terhadap diamond merupakan metode untuk menentukan pergerakan bot berdasarkan jarak real-time antara setiap bot dan diamond. Algoritma ini secara terus-menerus mengevaluasi jarak setiap bot terhadap diamond, untuk menentukan posisi diamond terdekat terhadap bot kita dibandingkan dengan bot musuh, kemudian menetapkan posisi tersebut sebagai target pergerakan bot kita. Sehingga proses algoritma ini berfokus pada pengambilan keputusan berdasarkan kriteria jarak terdekat antara suatu diamond terhadap bot kita dibandingkan dengan bot musuh. Dengan demikian bot akan bergerak ke arah diamond, dimana tidak ada bot dengan jarak yang lebih dekat dengan diamond tersebut, hal ini akan memungkinkan bot selalu mendapatkan diamond sebelum bot musuh mengambilnya.

a. Proses Mapping
   - Himpunan kandidat : Semua *command* yang tersedia
   - Himpunan solusi : Semua *command* yang dipilih
   - Fungsi solusi : Memeriksa *command* yang dipilih memiliki jarak terpendek dari bot ke diamond dibanding dengan bot lain
   - Fungsi seleksi : Memilih *command* yang akan dilakukan dengan parameter jarak terpendek dari bot ke diamond dibanding dengan bot lain
   - Fungsi kelayakan : Memeriksa apakah command yang dipilih valid dan dapat dilakukan bot
   - Fungsi objektif : Mencari pergerakan bot terbaik berdasarkan jarak terpendek dari bot ke diamond dibanding dengan bot lain

b. Analisis Efisiensi

   Algoritma ini menggunakan tiga pendekatan yaitu mencari posisi diamond, mencari posisi setiap bot, dan balik ke base keduanya menggunakan target position sebagai parameter pergerakan arah bot. Bot akan balik ke base ketika bot sudah mengumpulkan diamond sebanyak kapasitas bot, lalu target position dari bot akan diganti dengan posisi dari base bot tersebut. Algoritma ini akan mengambil list dari diamond yang ada di board, lalu akan mengambil list bot yang ada di board. Kemudian akan dilakukan perulangan sebanyak list diamond yang ada di board, dan dilakukan lagi

didalamnya perulangan sebanyak list bot yang ada di board. Kemudian dilakukan perhitungan jarak setiap diamond terhadap setiap bot yang ada di board. Sehingga kita mendapatkan jarak setiap diamond terhadap setiap bot, kita akan memilih hasil dimana bot kita memiliki jarak terpendek dibanding dengan bot lain. Hasil tersebut akan berupa posisi diamond yang akan menjadi target position dari pergerakan bot kita. Kompleksitas dari algoritma ini adalah T(n) = O(n^2)

c. Analisis Efektivitas
   Strategi ini efektif apabila:
   - Terdapat banyak bot di board
   - Persebaran diamond tersebar merata
   - Jumlah diamond di board melebihi jumlah bot di board

   Strategi ini tidak efektif apabila:
   - Persebaran diamond tersebar ke satu titik
   - Jumlah diamond di board kurang dari jumlah bot di board
   - Jarak diamond yang memenuhi syarat jauh dari posisi bot

3.1.3 Algoritma Greedy Berdasarkan Mentackle Bot Musuh

Algoritma greedy berdasarkan mentackle bot musuh merupakan metode untuk menentukan pergerakan bot berdasarkan jarak bot musuh terdekat dan banyak diamond yang sedang dibawa oleh bot musuh tersebut. Algoritma ini akan secara terus-menerus mengevaluasi jarak bot terhadap bot musuh dan banyak diamond yang dimiliki bot musuh, untuk menentukan bot musuh mana yang akan kita kejar dengan menjadikan posisi dari bot musuh tersebut sebagai target pergerakan bot kita, kemudian kita tackle untuk mendapatkan diamond nya. Sehingga proses algoritma ini berfokus pada jarak bot musuh terhadap kita dan banyak dari diamond yang dimiliki bot musuh tersebut. Dengan demikian bot akan bergerak menuju posisi bot musuh dengan tujuan untuk mentackle bot musuh tersebut sehingga mendapatkan diamond dari bot tersebut, kemudian balik ke base untuk mengamankan diamond yang sudah dimiliki.

a. Proses Mapping
   - Himpunan kandidat : Semua *command* yang tersedia
   - Himpunan solusi : Semua *command* yang dipilih
   - Fungsi solusi : Memeriksa *command* yang dipilih memiliki jarak terpendek dari bot ke bot lain yang memenuhi syarat
   - Fungsi seleksi : Memilih *command* yang akan dilakukan dengan parameter jarak terpendek dari bot ke bot lain yang memenuhi syarat
   - Fungsi kelayakan : Memeriksa apakah *command* yang dipilih valid dan dapat dilakukan bot

- Fungsi objektif : Mencari pergerakan bot terbaik untuk mentackle bot lain lalu mengambil diamond bot tersebut

b. Analisis Efisiensi

Algoritma ini menggunakan dua pendekatan yaitu mencari posisi setiap bot dan balik ke base keduanya menggunakan target position sebagai parameter pergerakan arah bot. Bot akan balik ke base ketika bot sudah mengumpulkan diamond sebanyak kapasitas bot, lalu target position dari bot akan diganti dengan posisi dari base bot tersebut. Algoritma ini akan mengambil list bot yang ada di board, lalu melakukan perulangan sebanyak list bot yang ada, dengan skema membandingkan setiap posisi lain terhadap posisi kita dan dicek berapa diamond yang dimiliki bot lain tersebut, jika memenuhi maka akan dicek jaraknya. Sehingga akan didapatkan jarak terpendek terhadap bot lain yang memenuhi syarat. Hasil tersebut berupa posisi bot yang akan menjadi target position bot kita, untuk kita tackle. Kompleksitas dari algoritma ini adalah $T(n) = O(n)$

c. Analisis Efektivitas

Strategi ini efektif apabila:

- Terdapat banyak bot di board
- Pergerakan bot musuh tidak sama dengan pergerakan bot kita
- Posisi base setiap base jauh dari diamond yang beredar

Strategi ini tidak efektif apabila:

- Terdapat sedikit bot di board
- Pergerakan bot musuh sama dengan pergerakan bot kita
- Posisi bot musuh lebih dekat dengan posisi base musuh dibandingkan dengan bot kita

3.1.4 Algoritma Greedy Berdasarkan Jarak Diamond dan Nilai Diamond

Algoritma greedy berdasarkan jarak diamond dan nilai diamond merupakan metode untuk menentukan pergerakan bot berdasarkan value tertinggi yang didapat dengan merumuskan jarak diamond terhadap bot dan nilai dari diamond tersebut. Algoritma ini akan terus menerus mengevaluasi value yang dimiliki suatu diamond terhadap bot. Sehingga proses algoritma ini akan berfokus pada value tertinggi yang dimiliki suatu diamond untuk menentukan pengambilan keputusan pergerakan bot. Dengan demikian bot akan bergerak terhadap posisi diamond yang memiliki value tertinggi, sehingga pergerakan dari bot lebih optimal.

a. Proses Mapping

- Himpunan kandidat : Semua *command* yang tersedia
- Himpunan solusi : Semua *command* yang dipilih

- Fungsi solusi : Memeriksa *command* yang dipilih memiliki value yang tertinggi berdasarkan rumus yang telah dibuat
- Fungsi seleksi : Memilih *command* yang akan dilakukan dengan parameter value tertinggi yang bisa didapat
- Fungsi kelayakan : Memeriksa apakah *command* yang dipilih valid dan dapat dilakukan bot
- Fungsi objektif : Mencari pergerakan bot terbaik berdasarkan value tertinggi yang bisa bot dapat

b. Analisis Efisiensi

Algoritma ini menggunakan tiga pendekatan yaitu mencari posisi diamond, nilai diamond, dan balik ke base keduanya menggunakan target position sebagai parameter pergerakan arah bot. Bot akan balik ke base ketika bot sudah mengumpulkan diamond sebanyak kapasitas bot, lalu target position dari bot akan diganti dengan posisi dari base bot tersebut. Algoritma ini akan mengambil list dari diamond yang ada di board. Kemudian kita algoritma ini akan melakukan perulangan sebanyak list diamond, dengan setiap perulangannya akan menghitung value sesuai rumus yang ditetapkan yang terdiri dari jarak diamond terhadap bot kita dan nilai diamond. Sehingga kita akan mendapatkan value dari setiap perhitungan, kemudian algoritma ini akan mengambil value tertinggi yang bisa didapat bot pada saat itu. Hasil ini akan berupa posisi diamond dengan value tertinggi tadi setelah dilakukan perhitungan tersebut, sehingga bot akan bergerak ke posisi diamond tersebut. Kompleksitas dari algoritma ini adalah $T(n) = O(n)$

c. Analisis Efektivitas

Strategi ini efektif apabila:

- Diamond didalam board saling berdekatan
- Terdapat banyak diamond bernilai 2 di board
- Jarak antar bot cukup berjauhan

Strategi ini tidak efektif apabila:

- Di dalam board terdapat bot lainnya yang jaraknya lebih dekat dengan diamond yang kita incar

3.2 Strategi Greedy yang Diimplementasikan

Berdasarkan alternatif solusi greedy yang telah dijelaskan pada bagian 3.1, terdapat beberapa pendekatan greedy yang dapat diterapkan pada bot. Setiap solusi memiliki fokus dan implementasi yang berbeda, juga menghasilkan efisiensi dan efektivitas yang bervariasi. Dalam implementasi bot, dipilih algoritma greedy berdasarkan jarak dan nilai diamond, yang telah dijelaskan pada bagian 3.1.4. Pada pengimplementasian algoritma tersebut, ditambahkan beberapa algoritma penting untuk mendukung optimalisasi strategi greedy tersebut.

Beberapa algoritma tambahan yang dimasukkan melibatkan strategi khusus:

1. Algoritma Go Home: memeriksa kondisi tertentu, jika terpenuhi, bot akan diarahkan pulang ke base untuk menyimpan diamond. Ini memastikan keamanan dan akumulasi diamond yang telah dikumpulkan.
2. Algoritma Avoid Obstacle: Memeriksa apakah rute bot memiliki teleport. Jika ya, bot akan menghindari teleport tersebut, mengoptimalkan pergerakan dan tujuan akhirnya.
3. Algoritma Reset Board: Mengecek jumlah diamond di papan permainan. Jika kurang dari 3, bot diarahkan ke diamond button untuk mereset papan. Ini dapat menjadi strategi penting untuk mengubah kondisi permainan.

Berikut beberapa poin penting mengenai algoritma yang diimplementasikan :

1. Memeriksa apakah kapasitas bot telah mencapai batas maksimum. Jika sudah, bot akan berhenti mencari diamond dan memprioritaskan kembali ke base.
2. Memeriksa jumlah diamond, jika jumlah diamond di board kurang dari 3, prioritas bot beralih untuk mencapai posisi diamond button.
3. Bot akan mencari diamond berdasarkan algoritma greedy yang dijelaskan pada 3.1.4
4. Jika bot melewati daerah base saat mencari diamond, prioritasnya akan beralih untuk kembali ke base sebelum melanjutkan pencarian.
5. Jika dalam perjalan bot terdapat teleport, maka bot akan berusaha menghindari teleport tersebut
6. Jika waktu permainan kurang dari 10 detik, bot akan membatasi pergerakannya hanya di sekitar wilayah base.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1. Repositori Github

Bot yang telah kami buat dapat diakses pada laman repositori Github berikut: https://github.com/Nerggg/Tubes1_bingung

4.2. Implementasi dalam Pseudocode

Berikut adalah implementasi algoritma greedy yang terdapat pada file `pulang.py`.

4.2.1. Prosedur go_home

```
// Fungsi ini digunakan untuk pulang ke base jika beberapa kondisi terpenuhi
procedure go_home(output self, input board_bot: List<GameObject>, input board:
List<Board>):
    props <- board_bot.properties

    // Mendapatkan waktu yang tersisa dalam milisecond
    current_time <- props.milliseconds_left

    // Mengubah waktu yang tersisa dari milisecond ke detik
    new_time <- current_time div 1000

    base <- props.base

    // Menghitung estimasi waktu yang dibutuhkan untuk sampai ke base dengan
manhattan distance
    estimated_time_to_base <- abs(base.x - board_bot.position.x) + abs(base.y -
board_bot.position.y)

    // Jika waktu yang tersisa kurang dari 10 detik dan estimasi waktu yang
dibutuhkan untuk sampai ke base lebih besar sama dengan dari waktu yang tersisa
    // atau jika diamond yang dimiliki bot adalah 3 atau 4 dan estimasi waktu yang
dibutuhkan untuk sampai ke base kurang dari 4
    // Maka goal positionnya adalah base
    if current_time < 10000 and estimated_time_to_base >= new_time or
(props.diamonds in [3,4] and estimated_time_to_base < 4 ) then
        self.goal_position <- base
    else then
        self.goal_position <- None
```

4.2.2. Prosedur purge

```
// Fungsi ini digunakan untuk mencari diamond dengan rasio terbesar pada setiap
langkahnnya
```

```
// yang mana merupakan implementasi dari algoritma greedy
procedure purge(output self, input board_bot: List<GameObject>, input board:
List<Board>):
    // Mendapatkan semua diamond yang ada di board dengan cara mengiterasi semua
game object pada board
    // lalu mencari type yang diinginkan yakni "DiamondGameObject"
    diamonds <- [go for each go in board.game_objects if go.type =
"DiamondGameObject"]

    // Menginisialisasi max_ratio dengan -1 dan min_index dengan None
    max_ratio <- -1
    min_index <- None

    // Untuk setiap diamond, kalkulasi rasio dari diamond point dengan jarak
manhattan antara bot dan diamond
    for each i, diamond in diamonds do
        bot_diamonds <- board_bot.properties.diamonds // Mendapatkan jumlah diamond
yang dimiliki oleh bot
        diamond_points <- diamond.properties.points // Mendapatkan point dari
diamond

        if (bot_diamonds + diamond_points > 5) then // Jika jumlah diamond yang
dimiliki bot setelah mengambil diamond ini lebih dari 5, maka skip diamond ini
            continue

        // Melakukan kalkulasi jarak manhattan antara bot dan diamond
        // Dengan cara mengurangi x dan y koordinat dari posisi diamond dengan
posisi bot
        // Lalu hasilnya diabsolutkan agar tidak ada nilai negatif
        x_dist <- abs(board_bot.position.x - diamond.position.x)
        y_dist <- abs(board_bot.position.y - diamond.position.y)
        manhattan_dist <- x_dist + y_dist

        // Jika jarak manhattannya 0, berarti bot sudah berada di posisi diamond,
yang berarti
        // bot tidak lagi perlu melakukan kalkulasi ratio untuk diamond yang ini,
jadi diamond yang ini akan diskip perhitungannya
        // dan akan langsung dilanjutkan ke diamond selanjutnya
        if manhattan_dist = 0 then
            continue

        // Mengkalkulasi rasio dari diamond point dengan jarak dari bot ke diamond
        // Rasio ini akan digunakan untuk menentukan diamond mana yang akan
dijadikan goal position
        // Diamond dengan point yang lebih tinggi dan jarak yang lebih dekat akan
memiliki rasio lebih besar
        ratio <- diamond_points div manhattan_dist
```

```
        // Jika rasio ini adalah rasio terbesar yang ditemukan sejauh ini, maka
update max_ratio dan min_index
        // max_ratio digunakan untuk menyimpan rasio terbesar yang ditemukan sejauh
ini
        // min_index digunakan untuk menyimpan index dari diamond yang memiliki
rasio terbesar
        if ratio > max_ratio then
            max_ratio <- ratio
            min_index <- i

    // Jika min_index tidak None, maka kita sudah menemukan dimaond dengan rasio
terbesar
    // Maka kita set goal positionnya menjadi posisi dari diamond tersebut
    if min_index != NULL then
        self.goal_position <- board.diamonds[min_index].position
```

4.2.3. Prosedur avoid_obstacle

```
// Fungsi ini digunakan untuk menyesuaikan goal position dari bot agar dapat
terhindar dari obstacle (Dalam kasus ini adalah teleporter)
procedure avoid_obstacle(output self, input board_bot: List<GameObject>, input
board: List<Board>):

    // Mendapatkan semua teleporter yang ada di board dengan melakukan iterasi
untuk semua game object di board
    // dan menyeleksi game object dengan type "TeleporterGameObject"
    teleporters <- [go for each go in board.game_objects if go.type =
"TeleporterGameObject"]

    // Jika tidak ada teleporter di board, maka langsung return
    if not teleporters then
        return

    // Jika ada teleporter maka akan diiterasi untuk masing-masing teleporter
    for teleporter in teleporters do
        // Ini akan melakukan pengecekan apakah bot berada dalam garis lurus antara
teleporter dan goal position
        if self.is_in_line(board_bot.position, teleporter.position,
self.goal_position) then
            // Jika teleporter berada di kiri bot, maka goal positionnya akan
diubah ke kanan bot
            if teleporter.position.x > board_bot.position.x then
                self.goal_position.x <- board_bot.position.x - 1
            // Jika teleporter berada di kanan bot, maka goal positionnya akan
diubah ke kiri bot
```

```
            elif teleporter.position.x < board_bot.position.x then
                self.goal_position.x <- board_bot.position.x + 1
            // Jika teleporter berada di atas bot, maka goal positionnya akan
diubah ke bawah bot
            if teleporter.position.y > board_bot.position.y then
                self.goal_position.y <- board_bot.position.y - 1
            // Jika teleporter berada di bawah bot, maka goal positionnya akan
diubah ke atas bot
            elif teleporter.position.y < board_bot.position.y then
                self.goal_position.y <- board_bot.position.y + 1
```

4.2.4. Fungsi is_in_line

```
// Fungsi ini digunakan untuk mengecek apakah 3 posisi berada dalam satu garis
lurus
function is_in_line(self, pos1, pos2, pos3) -> boolean:
    // Mengkalkulasi perbedaan koordinat x dan y antara pos1 dan pos2
    dx1 <- pos2.x - pos1.x
    dy1 <- pos2.y - pos1.y

    // Setelah itu, dilakukan kalkulasi perbedaan koordinat x dan y antara pos2 dan
pos3
    dx2 <- pos3.x - pos2.x
    dy2 <- pos3.y - pos2.y

    // Jika posisi berada dalam satu garis lurus, maka dx1, dx2, dy1, dan dy2 akan
bernilai 0
    if dx1 = dx2 = 0 then
        -> true
    if dy1 = dy2 = 0 then
        -> true


    // Jika dx1 dan dx2 tidak 0, dan dy1/dx1 sama dengan dy2/dx2, maka posisi
berada dalam satu garis lurus
    if dx1 != 0 and dx2 != 0 and dy1/dx1 = dy2/dx2 then
        -> true

    // Jika tidak memenuhi kondisi, maka posisi tidak berada dalam satu garis lurus
    -> false
```

4.2.5. Fungsi next_move

```
// Fungsi ini digunakan untuk melakukan kalkulasi untuk gerakan selanjutnya dari
bot
function next_move(self, board_bot: GameObject, board: Board) -> int, int:
```

```
    // Mendapatkan properties dari bot
    props <- board_bot.properties

    // Jika bot sudah mengumpulkan 5 diamond, maka goal positionnya adalah base
    if props.diamonds = 5 then
        base <- props.base
        self.goal_position <- base

    else then
        // Memanggil metode go_home untuk mengecek apakah bot harus pulang ke base
        // Jika iya, maka goal positionnya adalah base
        self.go_home(board_bot, board)
        if self.goal_position is None then
            // Memanggil metode purge untuk mencari diamond dengan rasio terbesar
            // yang akan dijadikan goal position
            self.purge(board_bot,board)

    // Jika bot sudah mempunyai posisi goal, maka akan dihitung gerakan untuk
mendekati posisi tersebut
    if self.goal_position then
        // Akan dipanggil fungsi avoid_obstacle untuk menyesuaikan posisi goal dari
bot agar dapat terhindar dari obstacle
        self.avoid_obstacle(board_bot, board)

        // Mendapatkan posisi bot sekarang
        current_position <- board_bot.position
        cur_x <- current_position.x
        cur_y <- current_position.y

        // Memanggil fungsi get_direction untuk mendapatkan delta x dan delta y
yang akan digunakan untuk bergerak dari posisi sekarang ke posisi goal
        delta_x, delta_y <- get_direction(cur_x, cur_y, self.goal_position.x,
self.goal_position.y)

        // Jika bot tidak bergerak semenjak langkah sebelumnya, maka botnya stuck
        if (cur_x, cur_y) = self.previous_position then

            // Cara bot keluar dari posisi stuck adalah dengan mencoba mengganti
arah. Misalkan arah gerak sebelumnya ada horizontal,
            // maka akan dicoba gerak secara vertikal, dan sebaliknya
            if delta_x != 0 then
                delta_y <- delta_x * self.turn_direction
                delta_x <- 0
            elif delta_y != 0 then
                delta_x <- delta_y * self.turn_direction
                delta_y <- 0
```

```
            // Akan mengganti arah putaran untuk langkah selanjutnya
            self.turn_direction <- -self.turn_direction

        // Akan menyimpan posisi sekarang sebagai previous position untuk langkah
selanjutnya
        self.previous_position <- (cur_x, cur_y)

        // Akan mengembalikan delta x dan delta y yang akan digunakan untuk
bergerak
        -> delta_x, delta_y

    // Jika bot tidak mempunyai posisi goal, maka akan mengembalikan 0, 0 yang
berarti bot tidak akan bergerak
    -> 0, 0
```

4.3. Struktur Data Program

Bahasa pemrograman yang digunakan untuk pembuatan bot ini adalah bahasa Python. Dalam bot itu sendiri, terdapat juga *class* yang harus diutilisasikan agar bot dapat berfungsi sesuai yang diinginkan. *Class* yang digunakan dalam pengembangan bot adalah sebagai berikut:

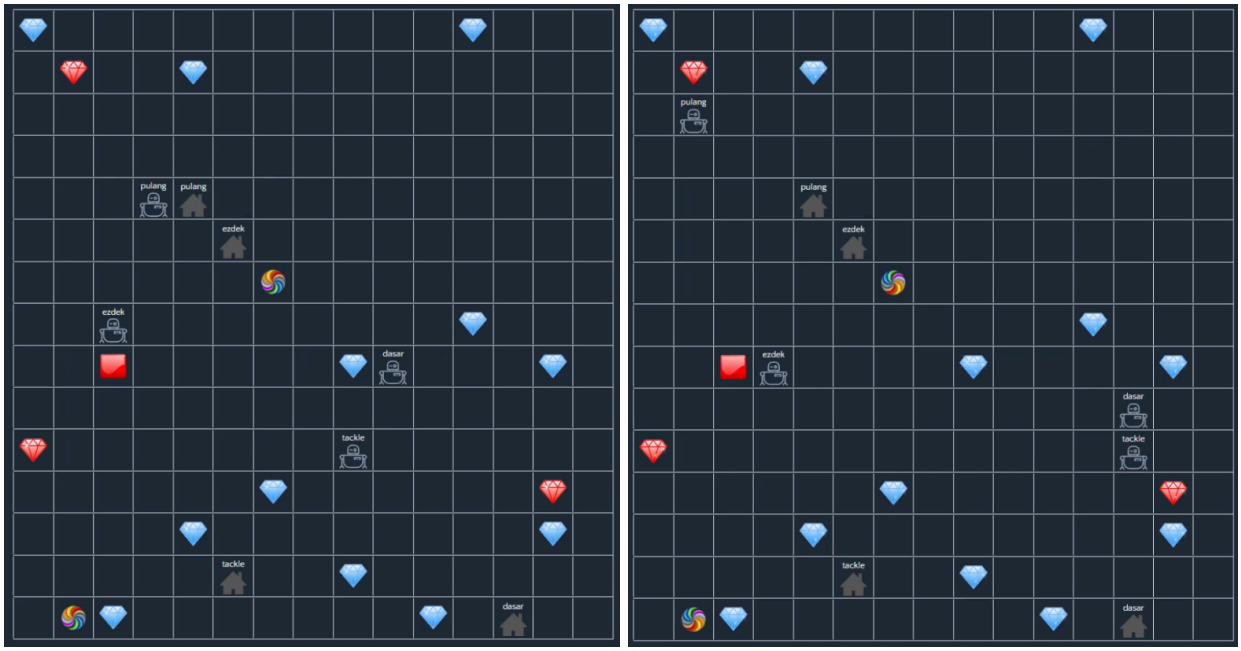
- *Class* Pulang
  Class ini berfungsi sebagai bot utama kami. Atribut yang ada pada *class* ini adalah goal_position yang berisi posisi yang ingin dicapai oleh bot, previous_position yang berisi posisi sebelumnya dari bot, dan turn_direction yang berisi arah putaran bot. Sedangkan method yang ada adalah go_home, purge, avoid_obstacle, is_in_line, dan next_move.
- *Class* Position
  Class ini berisi informasi posisi suatu objek. Terdapat dua atribut dalam *class* ini, yaitu posisi x dan posisi y.
- *Class* Properties
  Class ini menyimpan berbagai properti untuk masing-masing objek pada permainan, yaitu points, pair_id, diamonds, score, name, inventory_size, can_tackle, milliseconds_left, time_joined, dan base.
- *Class* Base
  Class ini menyimpan lokasi base masing-masing bot sebagai *class* position.
- *Class* GameObject
  Class ini berisi seluruh objek-objek yang ada pada permainan. Setiap objek pada *class* ini memiliki atribut id, *class* position, type, dan *class* properties.
- *Class* Board
  Class ini berfungsi untuk menyimpan informasi tentang papan permainan. Atribut yang ada pada *class* ini adalah id, width, height, array bernama features yang berisi *class*

Feature, minimum_delay_between_moves, dan array bernama game_objects yang berisi *class* GameObject

4.4. Analisis dan Pengujian

Kami melakukan pengujian bot dengan cara mempertarungkan bot utama kami yaitu pulang yang menerapkan algoritma greedy berdasarkan daerah dengan densitas diamond tertinggi dengan bot buatan kami lainnya, yaitu dasar yang hanya menerapkan algoritma greedy berdasarkan jarak diamond terdekat, ezdek yang menerapkan algoritma greedy berdasarkan nilai diamond, dan tackle yang kami beri algoritma khusus mentackle lawan.
Kami melakukan pengujian dan analisis terhadap beberapa perilaku bot, yaitu

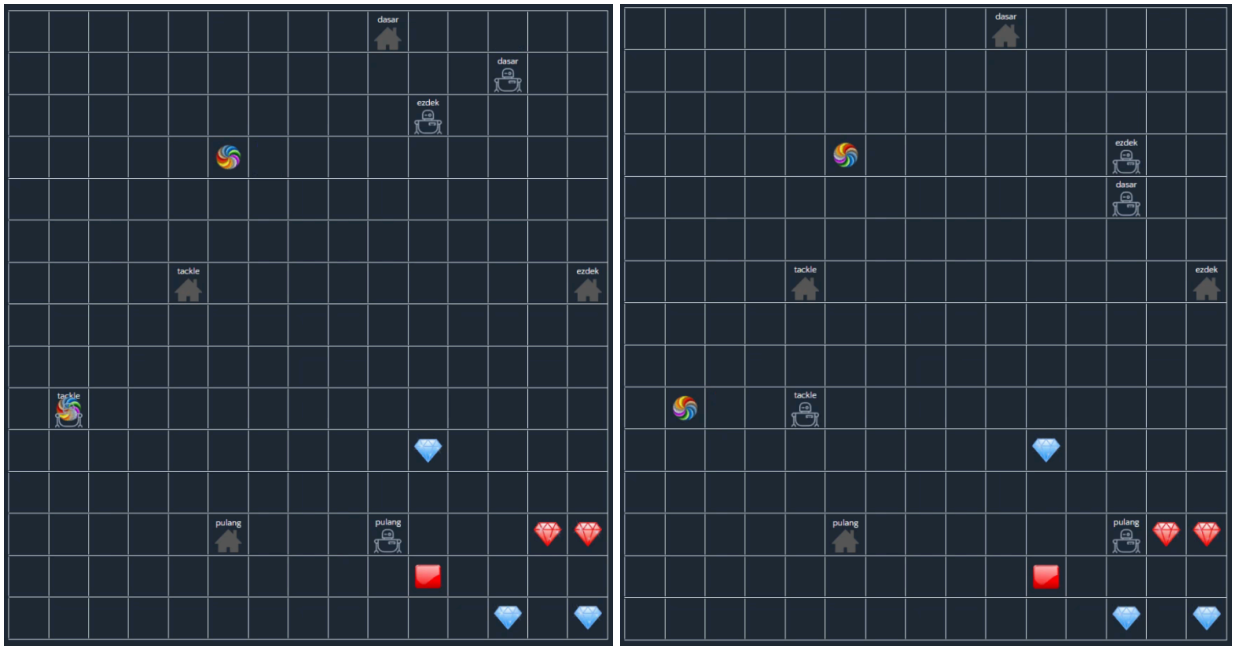
4.4.1. Fokus ke daerah dengan densitas diamond tertinggi



Gambar 2 & 3. Tampilan perilaku mengejar diamond berdasarkan densitas daerah

Dapat dilihat pada gambar di sebelah kiri, diamond biru yang berada di sebelah kanan berada di jarak yang lebih dekat ke bot pulang dibandingkan dengan bot merah yang berada di sebelah kiri. Tetapi bot pulang tetap lebih memilih daerah kiri karena daerah tersebut memiliki densitas diamond lebih tinggi dibandingkan dengan daerah kanan.

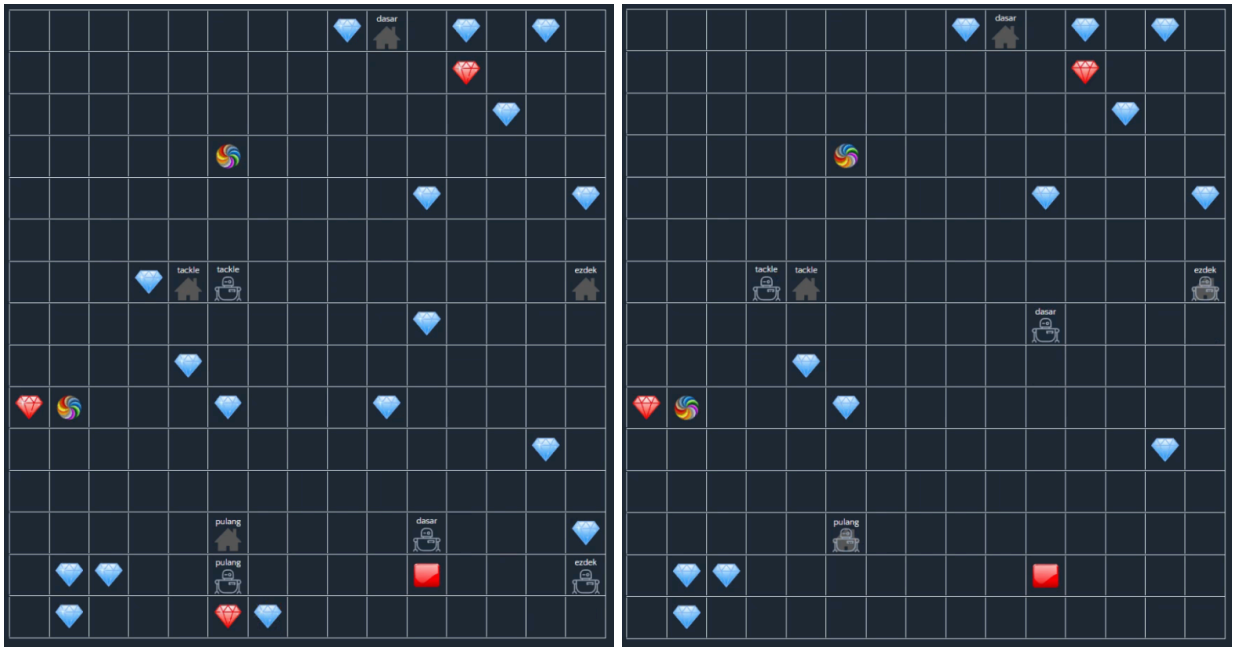
Berikut adalah contoh kejadian lain dimana bot memiliki perilaku yang sama



Gambar 4 & 5. Tampilan perilaku mengejar diamond berdasarkan densitas daerah

Terdapat diamond biru yang jaraknya lebih dekat dari bot pulang, tetapi bot pulang lebih memilih dua diamond merah yang terletak lebih jauh dari posisinya.

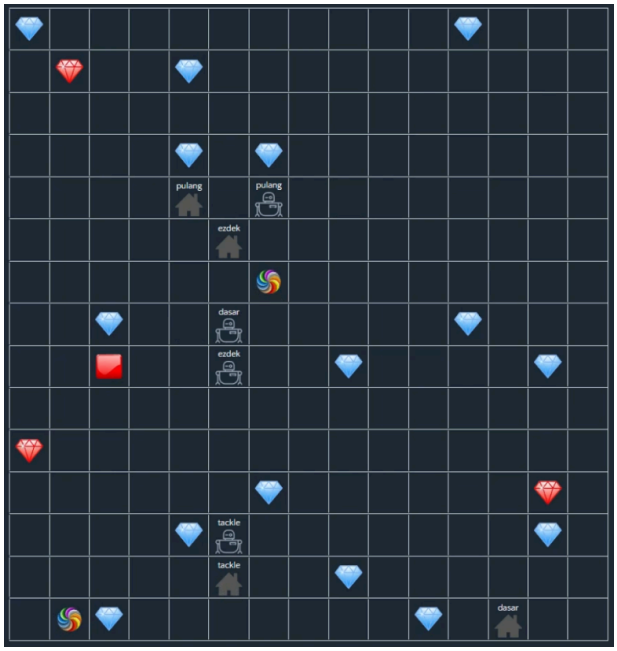
4.4.2. Langsung kembali ke base ketika waktu sudah mau habis



Gambar 6 & 7. Tampilan perilaku kembali ke base ketika waktu sudah mau habis

Pada kejadian ini, ada dua diamond yang berada di dekat base bot pulang. Setelah diamond tersebut diambil, bot pulang masih memiliki ruang di inventory-nya. Tetapi karena waktu sudah mau habis, bot pulang langsung kembali ke base untuk mengembalikan diamond yang dimilikinya karena jika dia memilih untuk mengambil lebih banyak diamond dengan waktu yang sedikit, maka diamond yang berada di inventory-nya tidak akan dihitung dan akan menjadi sia-sia.

4.4.3. Menghindari portal



Gambar 8. Tampilan perilaku menghindari portal

Pada kasus ini, bot pulang ingin mengembalikan diamondnya ke base. Namun, terdapat portal yang menghalangi jalannya. Apabila dia memilih untuk jalan lurus saja, maka dia akan menabrak portal dan akhirnya mengambil jalan lebih jauh lagi untuk pulang ke base. Dengan adanya perilaku ini, bot pulang akan mengambil jalan memutar jika ada portal yang berada diantara jalannya dengan base nya dan dapat dilihat pada gambar, perilaku tersebut berhasil dijalankan.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Melalui penyelesaian tugas besar IF2211 Strategi Algoritma, kami berhasil mengimplementasikan bot permainan 'Etimo Diamonds V2' dengan menggunakan strategi Greedy. Dari pengalaman ini, kami menyimpulkan bahwa algoritma greedy dapat efektif digunakan untuk mencapai solusi lokal maksimum atau optimum dari kondisi saat ini.

Algoritma Greedy juga dapat dianggap sebagai algoritma optimasi yang cukup baik dalam pengambilan keputusan untuk pembuatan bot. Hal ini karena dalam banyak kasus, solusi optimal lokal yang dipilih cukup mendekati solusi optimal global. Namun, penting untuk diingat bahwa terkadang algoritma greedy mungkin gagal menemukan solusi optimum global, seperti yang terlihat dalam beberapa pertandingan di mana bot gagal mendapatkan keuntungan maksimal akibat lokasi spawn yang tidak menguntungkan.

Sebagai alternatif, kami menyadari bahwa exhaustive search merupakan pendekatan lain yang dapat digunakan. Meskipun memakan waktu lebih lama karena mengecek setiap heading, action, 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

Saran untuk kelompok kami diantaranya:

- Lebih mendalami game engine, karena banyak isu-isu yang tidak bisa di-solve karena kurang pengetahuan tentang game engine yang digunakan
- Lebih teratur dalam menjadwalkan pengerjaan Tugas besar
- Lebih banyak sesi brainstorming diantara sesi pengerjaan
- Mengerjakan bot tidak mendekati deadline agar bot dapat lebih sempurna.

BAB VI
DAFTAR PUSTAKA

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

1. Algoritma Greedy (itb.ac.id) (Diakses pada 3 Maret, 2024)
2. Algoritma Greedy (itb.ac.id) (Diakses pada 3 Maret, 2024)
3. Algoritma Greedy (itb.ac.id) (Diakses pada 3 Maret, 2024)
4. Etimo/diamonds2: 💎Diamonds v2 (github.com) (Diakses pada 27 Februari, 2024)

Link Repository GitHub :
https://github.com/Nerggg/Tubes1_bingung

Link Youtube Video:
https://youtu.be/zSz0QU0fojI