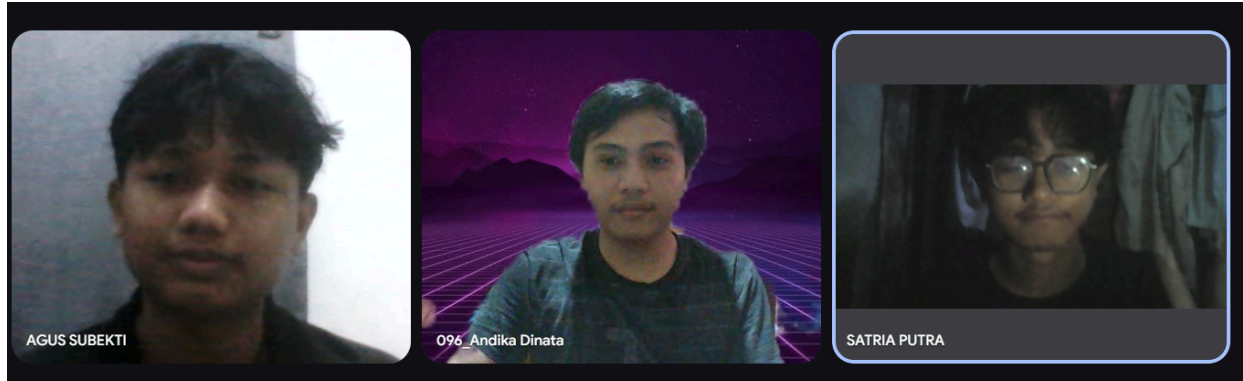


IMPLEMENTASI ALGORITMA GREEDY DALAM PEMECAHAN BOT PERMAINAN DIAMOND

Tugas Besar

Diajukan sebagai syarat menyelesaikan mata kuliah Strategi Algoritma (IF2211) Kelas RE
di Program Studi Teknik Informatika, Fakultas Teknologi Industri, Institut Teknologi Sumatera



Oleh: Kelompok 10 (MY KISAH)

Andika Dinata	123140096
Agus Subekti	123140104
Satria Lemana Putra	123140088

Dosen Pengampu: Imam Eko Wicaksono, S.Si., M.Si.

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SUMATERA
2025**

DAFTAR ISI

BAB I	
DESKRIPSI TUGAS.....	2
Spesifikasi Tugas Besar 1.....	4
BAB II	
LANDASAN TEORI.....	6
2.1 Dasar Teori.....	6
1. Cara Implementasi Program.....	7
2. Menjalankan bot program.....	8
3. Mengembangkan algoritma bot.....	11
BAB III	
APLIKASI STRATEGI GREEDY.....	12
3.1 Proses Mapping.....	12
3.2 Eksplorasi Alternatif Solusi Greedy.....	12
3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy.....	13
3.4 Strategi Greedy yang Dipilih.....	16
BAB IV	
IMPLEMENTASI DAN PENGUJIAN.....	17
4.1 Implementasi Algoritma Greedy.....	17
1. Implementasi Pseudocode.....	17
2. Penjelasan Alur Program.....	20
4.2 Struktur Data yang Digunakan.....	21
4.3 Pengujian Program.....	23
1. Skenario Pengujian.....	23
2. Hasil Pengujian dan Analisis.....	23
BAB V	
KESIMPULAN DAN SARAN.....	26
5.1 Kesimpulan.....	26
5.2 Saran.....	26
LAMPIRAN.....	27
DAFTAR PUSTAKA.....	28

BAB I

DESKRIPSI TUGAS

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.

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

Komponen-komponen dari permainan Diamonds antara lain:

1. Diamonds



Untuk memenangkan pertandingan, kita harus mengumpulkan *diamond* ini sebanyak-banyaknya dengan melewati/melangkahinya. Terdapat 2 jenis *diamond* yaitu *diamond* biru dan *diamond* merah. *Diamond* merah bernilai 2 poin, sedangkan yang biru bernilai 1 poin. *Diamond* akan di-*regenerate* secara berkala dan rasio antara *diamond* merah dan biru ini akan berubah setiap *regeneration*.

2. Red Button/Diamond Button



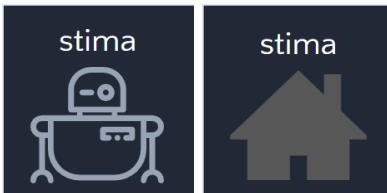
Ketika *red button* ini dilewati/dilangkahi, semua *diamond* (termasuk *red diamond*) akan di-generate kembali pada *board* dengan posisi acak. Posisi *red button* ini juga akan berubah secara acak jika *red button* ini dilangkahi.

3. Teleporters



Terdapat 2 *teleporter* yang saling terhubung satu sama lain. Jika bot melewati sebuah *teleporter* maka bot akan berpindah menuju posisi *teleporter* yang lain.

4. Bots and Bases



Pada game ini kita akan menggerakkan bot untuk mendapatkan *diamond* sebanyak banyaknya. Semua bot memiliki sebuah *Base* dimana *Base* ini akan digunakan untuk menyimpan *diamond* yang sedang dibawa. Apabila *diamond* disimpan ke *base*, *score* bot akan bertambah senilai *diamond* yang dibawa dan *inventory* (akan dijelaskan di bawah) bot menjadi kosong.

5. Inventory

Name	Diamonds	Score	Time
stima	💎💎	0	43s
stima2	💎	0	43s
stima1	💎💎💎💎	0	44s
stima3	💎	0	44s

Bot memiliki *inventory* yang berfungsi sebagai tempat penyimpanan sementara *diamond* yang telah diambil. *Inventory* ini memiliki kapasitas maksimum sehingga sewaktu waktu bisa penuh. Agar *inventory* ini tidak penuh, bot bisa menyimpan isi *inventory* ke *base* agar *inventory* bisa kosong kembali.

Untuk mengetahui *flow* dari game ini, berikut ini adalah cara kerja permainan Diamonds.

1. Pertama, setiap pemain (bot) akan ditempatkan pada *board* secara *random*. Masing-masing bot akan mempunyai *home base*, serta memiliki *score* dan *inventory* awal bernilai nol.
2. Setiap bot diberikan waktu untuk bergerak, waktu yang diberikan semua sama untuk setiap pemain.
3. Objektif utama bot adalah mengambil *diamond-diamond* yang ada di peta sebanyak-banyaknya. Seperti yang sudah disebutkan di atas, *diamond* yang berwarna merah memiliki 2 poin dan *diamond* yang berwarna biru memiliki 1 poin.
4. Setiap bot juga memiliki sebuah *inventory*, dimana *inventory* berfungsi sebagai tempat penyimpanan sementara *diamond* yang telah diambil. *Inventory* ini sewaktu-waktu bisa penuh, maka dari itu bot harus segera kembali ke *home base*.
5. Apabila bot menuju ke posisi *home base*, *score* bot akan bertambah senilai *diamond* yang tersimpan pada *inventory* dan *inventory* bot akan menjadi kosong kembali.
6. Usahakan agar bot anda tidak bertemu dengan bot lawan. Jika bot A menempa posisi bot B, bot B akan dikirim ke *home base* dan semua *diamond* pada *inventory* bot B akan hilang, diambil masuk ke *inventory* bot A (istilahnya *tackle*).
7. Selain itu, terdapat beberapa fitur tambahan seperti *teleporter* dan *red button* yang dapat digunakan apabila anda menuju posisi objek tersebut.
8. Apabila waktu seluruh bot telah berakhir, maka permainan berakhir. *Score* masing-masing pemain akan ditampilkan pada tabel Final Score di sisi kanan layar.

Spesifikasi Tugas Besar 1

- 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.

BAB II

LANDASAN TEORI

2.1 Dasar Teori

Algoritma greedy adalah metode pemecahan masalah yang bekerja secara langkah demi langkah, di mana pada setiap langkahnya, algoritma ini mengambil pilihan terbaik yang tersedia pada saat itu tanpa mempertimbangkan konsekuensi di masa depan, mengikuti prinsip “ambil apa yang bisa kamu dapatkan sekarang!”. Dengan pendekatan ini, algoritma greedy berharap bahwa dengan memilih solusi optimal lokal pada setiap langkah, hasil akhirnya akan mencapai solusi optimal global.

Elemen-elemen algoritma greedy:

1. Himpunan kandidat, C : berisi kandidat yang akan dipilih pada setiap Langkah (misal: simpul/sisi di dalam graf, job, task, koin, benda, karakter, dsb)
2. Himpunan solusi, S : berisi kandidat yang sudah dipilih
3. Fungsi solusi: menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi
4. Fungsi seleksi (selection function): memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik.
5. Fungsi kelayakan (feasible): memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak)
6. Fungsi obyektif : memaksimumkan atau meminimumkan

Jika solusi terbaik mutlak tidak terlalu diperlukan, algoritma greedy dapat digunakan untuk menghasilkan solusi hampiran (approximation) daripada menggunakan algoritma dengan kebutuhan waktu eksponensial untuk menghasilkan solusi eksak. Misalnya, pada persoalan Travelling Salesman Problem (TSP) dengan jumlah simpul (n) yang banyak, algoritma brute force membutuhkan waktu komputasi yang lama untuk menemukan tur dengan bobot minimal. Dengan algoritma greedy, meskipun tur dengan bobot minimal tidak selalu dapat ditemukan, solusi yang dihasilkan dianggap sebagai hampiran solusi optimal. Namun, jika algoritma greedy dapat menghasilkan solusi optimal, keoptimalannya harus dibuktikan secara matematis, yang merupakan tantangan tersendiri. Membuktikan bahwa algoritma greedy tidak selalu optimal lebih mudah dilakukan dengan menunjukkan counterexample, yaitu contoh kasus yang menunjukkan solusi yang diperoleh tidak optimal, seperti pada persoalan penukaran uang, di mana solusinya kadang-kadang optimal dan kadang-kadang tidak.

2.2 Cara Kerja Program

Program bot akan memulai dengan memeriksa apakah bot sudah terdaftar atau belum. Proses ini dilakukan dengan mengirimkan POST request ke endpoint `/api/bots/recover` yang berisi email dan password bot. Jika bot sudah terdaftar, backend akan mengembalikan response code 200 dengan body yang berisi ID bot tersebut. Namun, jika bot belum terdaftar, backend akan memberikan response code 404, menandakan bahwa bot tidak ditemukan.

Jika bot belum terdaftar, program bot akan mendaftarkan bot baru dengan mengirimkan POST request ke endpoint `/api/bots` dengan body berisi email, name, password, dan team. Jika proses pendaftaran berhasil, backend akan mengembalikan response code 200 dengan body yang berisi ID bot yang baru dibuat. ID ini menjadi kunci untuk langkah-langkah selanjutnya dalam permainan.

Setelah ID bot diperoleh, bot dapat bergabung ke board dengan mengirimkan POST request ke endpoint `/api/bots/{id}/join` dengan body berisi `preferredBoardId`, yaitu ID board yang diinginkan. Jika bot berhasil bergabung, backend akan memberikan response code 200 dengan body yang berisi informasi board. Informasi ini penting untuk menentukan langkah bot selanjutnya.

Selama permainan, program bot akan secara berkala menghitung langkah berikutnya berdasarkan kondisi board yang diketahui. Bot akan mengirimkan POST request ke endpoint `/api/bots/{id}/move` dengan body berisi arah yang dipilih, seperti “NORTH”, “SOUTH”, “EAST”, atau “WEST”. Jika langkah berhasil, backend akan mengembalikan response code 200 dengan body berisi kondisi board terbaru setelah langkah tersebut. Proses ini berulang hingga waktu bot habis, di mana bot akan otomatis dikeluarkan dari board.

Sementara itu, program frontend juga akan bekerja secara periodik dengan mengirimkan GET request ke endpoint `/api/boards/{id}` untuk mendapatkan kondisi board terbaru. Dengan demikian, tampilan board pada frontend akan selalu diperbarui sesuai dengan perkembangan permainan, memberikan pengalaman visual yang dinamis bagi pengguna.

1. Cara Implementasi Program

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

NOTE: `next_move` mengembalikan nilai `delta_x` dan `delta_y`, di mana nilai yang diperbolehkan hanyalah (1, 0), (0, 1), (-1, 0), (0, -1). Apabila nilai ilegal

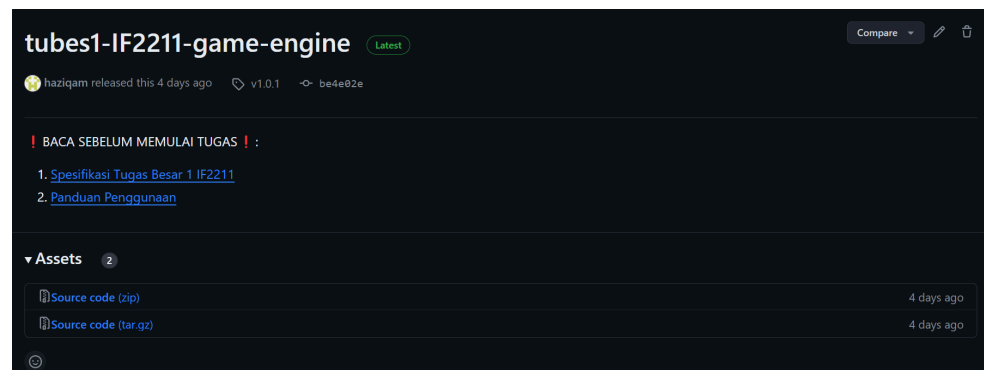
atau di-luar range board, maka move akan diabaikan oleh program

- c. Import kelas yang telah dibuat pada main.py dan daftarkan pada dictionary CONTROLLERS

```
main.py M X
main.py > ...
1  import argparse
2  from time import sleep
3
4  from colorama import Back, Fore, Style, init
5  from game.api import Api
6  from game.board_handler import BoardHandler
7  from game.bot_handler import BotHandler
8  from game.logic.random import RandomLogic
9  from game.util import *
10 from game.logic.base import BaseLogic
11 from game.logic.mybot import MyBot ←
12
13 init()
```

2. Menjalankan bot program

- a. Install game engine nya dulu
1. *Requirement* yang harus di-install
 - Node.js (<https://nodejs.org/en>)
 - Docker desktop
(<https://www.docker.com/products/docker-desktop/>)
 - Yarn
 2. Instalasi dan konfigurasi awal
 - a. Download source code (.zip) pada [release game engine](#)



- b. Extract zip tersebut, lalu masuk ke folder hasil extractnya dan buka terminal

- c. Masuk ke root directory dari project (sesuaikan dengan nama rilis terbaru)

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

- d. Install dependencies menggunakan Yarn

```
yarn
```

- e. 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
```

- f. 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
```

3. Build

```
npm run build
```

4. Run

```
npm run start
```

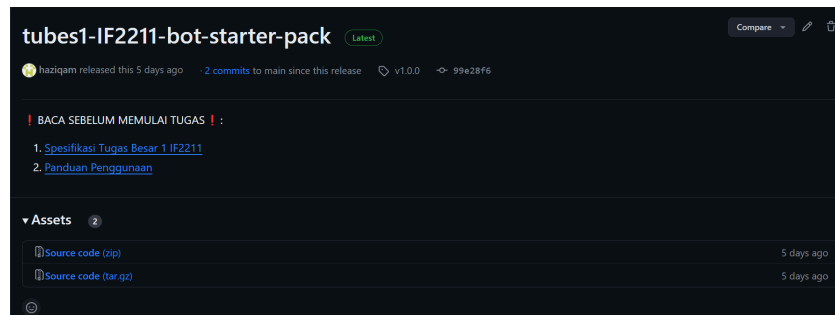
b. Cara menjalankan botnya

1. *Requirement* yang harus di-install

- Python (<https://www.python.org/downloads/>)

2. Instalasi dan konfigurasi awal

- a. Download source code (.zip) pada [release bot starter pack](#)



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

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

- d. Install dependencies menggunakan pip

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

- e. 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
```

- f. 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
1 @echo off
2 start cmd /c "python main.py --logic Random --email=test@email.com --name=stima --password=123456 --team etimo"
3 start cmd /c "python main.py --logic Random --email=test1@email.com --name=stima1 --password=123456 --team etimo"
4 start cmd /c "python main.py --logic Random --email=test2@email.com --name=stima2 --password=123456 --team etimo"
5 start cmd /c "python main.py --logic Random --email=test3@email.com --name=stima3 --password=123456 --team etimo"
6
```

3. Mengembangkan algoritma bot

Dalam pengembangan bot dengan algoritma greedy, terdapat sejumlah batasan, seperti tidak diizinkannya perubahan pada struktur program atau kode di luar file yang berisi logika bot yang kita kembangkan di folder `./game/logic`. Ketika mengembangkan bot di folder `./game/logic`, kita harus memastikan bahwa kelas yang dibuat memiliki fungsi `next_move`. Fungsi `next_move` ini wajib menerima dua parameter, yaitu objek bertipe `GameObject` dan `Board`. Selain itu, fungsi ini harus menghasilkan dua nilai kembalian, yaitu `delta_x` dan `delta_y`, dengan variasi nilai -1, 0, atau 1.

BAB III

APLIKASI STRATEGI *GREEDY*

3.1 Proses *Mapping*

Persoalan permainan Diamonds merupakan masalah optimasi strategis di mana bot bergerak pada peta untuk mengumpulkan diamond bernilai poin dengan memanfaatkan fitur seperti red button dan teleporter, serta menyimpan poin di home base untuk memaksimalkan skor dalam waktu terbatas, di mana setiap keputusan bot harus mempertimbangkan kapasitas inventory dan potensi gangguan dari bot lawan, sehingga dapat dipetakan ke dalam kerangka algoritma dengan himpunan kandidat, solusi, dan fungsi-fungsi yang mengarahkan pada strategi optimal.

- **Himpunan Kandidat:** Semua jalur yang memungkinkan untuk mencapai diamond merah (2 poin) dan diamond biru (1 poin).
- **Himpunan Solusi:** Kumpulan jalur dari himpunan kandidat yang memiliki total poin terbesar dan jarak yang paling pendek.
- **Fungsi Solusi:** Fungsi untuk mengecek apakah jalur yang dipilih dari himpunan kandidat memiliki total poin maksimal (5 poin dalam satu perjalanan).
- **Fungsi Seleksi:** Fungsi untuk memilih jalur yang paling efisien untuk dipilih oleh bot.
- **Fungsi Kelayakan:** Fungsi untuk memeriksa apakah kandidat diamond yang diambil layak diambil oleh bot kami. Dengan melakukan pertimbangan waktu untuk bot kami mengambil diamond dan kembali ke base apakah masih cukup.
- **Fungsi Objektif:** Mendapatkan poin sebanyak-banyaknya.

3.2 Eksplorasi Alternatif Solusi Greedy

Berikut ini adalah solusi alternatif yang kami pikirkan untuk menyelesaikan masalah permainan etimo diamonds.

1. Greed by Jarak Terdekat

Ide utama dari solusi ini adalah mengumpulkan diamond (merah atau biru) terdekat yang membutuhkan langkah paling sedikit selama inventory belum penuh. Tujuannya meminimalkan jumlah langkah untuk efisiensi maksimal.

2. Greed by Poin Tertinggi

Ide utama dari solusi ini adalah memprioritaskan untuk mengumpulkan diamond merah terlebih dahulu untuk memaksimalkan poin yang didapatkan. Tujuannya memaksimalkan poin dengan fokus pada diamond merah.

3. Greed by Density

Ide utama dari solusi ini adalah mengumpulkan diamond dengan poin terbesar dengan jarak terdekat atau memiliki nilai poin dibagi jarak yang paling terbesar. Tujuannya menyeimbangkan poin dan efisiensi gerakan untuk skor optimal.

4. Greed by Kepadatan Diamond

Ide utama dari solusi ini adalah membagi grid (15 x 15) menjadi 4 wilayah (~7 x 7) dan memilih wilayah dengan total poin tertinggi. Tujuannya memaksimalkan efisiensi dengan fokus pada wilayah dengan kepadatan diamond tinggi.

5. Greed by Tackle

Ide utama dari solusi ini adalah memprioritaskan untuk melakukan tackling bot lawan sambil tetap mengumpulkan diamond. Tujuannya memaksimalkan score bot dengan mencuri diamond yang dibawa oleh bot lawan.

6. Greed by Density dengan kalkulasi waktu untuk pulang ke base

Ide utama dari solusi ini adalah mengumpulkan diamond dengan poin terbesar dengan jarak terdekat sambil memprioritaskan untuk pulang ke base dibandingkan memaksakan mengambil diamond jika kalkulasi waktu untuk mengambil dan pulang ke base tidak cukup. Tujuannya menyeimbangkan poin dan efisiensi gerakan dan memprioritaskan untuk mengamankan poin yang sudah dikumpulkan.

7. Greed by Density diamond di sekitar base

Ide utama dari solusi ini adalah mengumpulkan diamond dengan poin terbesar dengan jarak terdekat dengan base, sehingga meminimalkan langkah untuk kembali ke base. Tujuannya menyeimbangkan poin dan efisiensi gerakan, meminimalkan langkah untuk pulang.

3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy

Analisis Efisiensi dan Efektivitas dari alternatif solusi yang telah dibuat sebelumnya:

1. Greed by Jarak Terdekat

Kelebihan dari solusi ini adalah mudah diimplementasikan dan bot kami akan selalu memilih diamond yang paling dekat terlebih dahulu, artinya langkah yang digunakan akan semakin sedikit. Kekurangan dari solusi ini adalah bot kami mengabaikan diamond

merah walaupun perbedaan jaraknya itu sedikit, hal ini membuat bot mendapatkan sedikit poin.

2. Greed by Poin Tertinggi

Kelebihan dari solusi ini adalah mudah diimplementasikan dan bot kami akan selalu memprioritaskan untuk memilih diamond yang memiliki poin tertinggi terlebih dahulu, artinya bot akan selalu memaksimalkan poin yang dikumpulkan. Kekurangan dari solusi ini adalah jika diamond merah jaraknya terlalu jauh maka bot membutuhkan banyak langkah, bot juga memiliki resiko untuk ditackle bot lain yang menargetkan diamond yang sama.

3. Greed by Density

Kelebihan dari solusi ini adalah bot mempertimbangkan poin dan jarak terlebih dahulu, untuk memastikan bot mendapatkan poin maksimal dengan langkah minimal. Kekurangan dari solusi ini adalah lebih sulit diimplementasikan kemudian membutuhkan waktu komputasi yang sedikit lebih lama.

4. Greed by Kepadatan Wilayah

Kelebihan dari solusi ini adalah bot memilih wilayah dengan total poin tertinggi, bot memaksimalkan poin dan meminimalkan langkah karena mengurangi perjalanan antar diamond. Kekurangan dari solusi ini adalah perjalanan ke wilayah dengan kepadatan diamond tertinggi mungkin membutuhkan banyak langkah, bot memiliki resiko ditackle oleh bot musuh yang menargetkan wilayah yang sama.

5. Greed by Tackle

Kelebihan dari solusi ini adalah bot bisa mendapatkan poin lebih besar dengan cara melakukan tackle ke bot musuh yang membawa banyak diamond. Saat peluang tackle tidak ada maka bot akan mengumpulkan diamond terdekat dahulu. Kekurangan dari solusi ini adalah bot kami memiliki resiko untuk ditackle, bot menggunakan banyak langkah yang mungkin kurang efektif dibandingkan mengumpulkan diamond secara langsung.

6. Greed by Density dengan kalkulasi waktu untuk pulang ke base

Kelebihan dari solusi ini adalah bot mempertimbangkan antara poin dan jarak dari diamond dan memiliki fokus utama untuk mengamankan poin terlebih dahulu dibandingkan memaksakan untuk mengambil diamond. Kekurangan dari solusi ini adalah lebih sulit diimplementasikan, dan memakan komputasi yang lebih banyak.

7. Greed by Density diamond di sekitar base

Kelebihan dari solusi ini adalah bot mempertimbangkan antara poin dan jarak dari diamond dan memiliki fokus utama untuk memilih diamond yang lebih dekat dengan base. Kekurangan dari solusi ini adalah lebih sulit diimplementasikan, dan memakan komputasi yang lebih banyak.

Setelah mempertimbangkan beberapa solusi yang sebelumnya telah kami pikirkan. Ternyata menjadikan jarak atau poin tertinggi saja yang menjadi pertimbangan tidak efisien karena bot tidak memaksimalkan score sambil meminimalkan langkah. Jadi solusi nomor 1 dan 2 tidak kami pilih.

Kemudian setelah kami analisa lagi, greed by tackle tidak efisien karena bot musuh bisa jadi memiliki mekanisme untuk melakukan penghindaran. Untuk menerapkan solusi ini juga ada sebuah kendala, yaitu kita tidak mengetahui inventory musuh membawa berapa diamond. Jadi solusi nomor 5 tidak kami pilih.

Selanjutnya kami menganalisa greed by kepadatan wilayah. Ternyata solusi ini yang awalnya terlihat efisien menjadi tidak terlalu efisien karena untuk diterapkan membutuhkan waktu komputasi yang lebih banyak, kemudian bot kami mungkin membutuhkan langkah yang banyak untuk mencapai ke wilayah dengan kepadatan diamond tertinggi. Dan solusi ini udah pasti gagal jika diamond menyebar merata.

Selanjutnya kami mempertimbangkan untuk menggunakan solusi Greed by Density dengan kalkulasi waktu untuk pulang ke base. Sebenarnya ini solusi yang bagus, tapi melakukan estimasi waktu bakal menambah beban komputasi yang lebih signifikan.

Selanjutnya kami mempertimbangkan untuk menggunakan solusi Greed by Density diamond di sekitar base. Solusi ini bagus, tapi tetap ada kemungkinan bot bakal kehabisan waktu untuk pulang ke base.

Berdasarkan hasil analisa solusi yang telah dibuat sebelumnya. Kami memutuskan untuk melakukan kombinasi dari beberapa solusi dan memodifikasi untuk menciptakan solusi yang mengambil kelebihan dari beberapa solusi. Greedy by Tackle, Greedy by Density dengan kalkulasi waktu untuk pulang ke base, Greedy by Density diamond di sekitar base. Untuk bagian Tackle kami akan mengimplementasi bot untuk melakukan tackle jika jarak antara bot kami dan bot musuh itu berdekatan. Jadi bot kami tidak akan terlalu aktif untuk melakukan tackle musuh. Kemudian kami mengkombinasikan antara Greedy by Density diamond di sekitar base dengan kalkulasi waktu untuk pulang ke base. Jadi bot kami memiliki fokus utama mencari diamond di sekitar base agar jarak untuk pulang itu dekat sambil memperkirakan apakah waktu nya cukup untuk pulang dan kembali ke base, jadi bot kami akan mengamankan sisa-sisa diamond dalam detik-detik terakhir masa hidupnya. Kami sebenarnya ingin menggunakan teleporter dan red button namun di dalam implementasi kode bot starter pack di bagian model kami tidak menemukan implementasi properti teleporter dan red diamond. Jadi kami tidak memasukkan objek itu ke dalam strategi kami, dan kami juga tidak bisa menghindari objek tersebut.

3.4 Strategi Greedy yang Dipilih

Dari penjelasan solusi sebelumnya, terlihat bahwa setiap solusi memiliki keunggulan dan kelemahan masing-masing. Kami akhirnya memutuskan untuk mengkombinasikan beberapa solusi. Jadi strategi kami adalah Greedy by Density diamond di sekitar base dengan kalkulasi waktu apakah cukup untuk mengambil dan pulang ke base. Bot kami juga bisa melakukan tackle jika bot musuh berada dekat dengan bot kami. Tapi fokus utama bot kami adalah mengumpulkan diamond dan memaksimalkan score yang bisa didapatkan.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma Greedy

1. Implementasi Pseudocode

```
Unset
CLASS MalingProPlayer EXTENDS BaseLogic:
    PROPERTIES:
        goal_position: Position ← null
        detik_detik_kematian: Boolean ← false

    CONSTRUCTOR():
        goal_position ← null
        detik_detik_kematian ← false
        PRINT "Maling Pro Player siap maling diamond sebanyak-banyaknya!"

    FUNCTION next_move(board_bot: GameObject, board: Board):
        // Inisiasi variabel dari board_bot dan board untuk mempermudah coding
        lokasi_base ← board_bot.properties.base
        lokasi_bot_kita ← board_bot.position
        jumlah_diamond_di_inventory ← board_bot.properties.diamonds
        jumlah_waktu_yang_tersisa ← board_bot.properties.milliseconds_left
        jumlah_maksimal_diamond_di_inventory ←
board_bot.properties.inventory_size
        diamonds ← board.diamonds
        bots ← board.bots

        // Jika bot berada di ambang kematian maka bot akan fokus kembali ke
base
        IF detik_detik_kematian = true THEN
            RETURN get_direction(lokasi_bot_kita.x, lokasi_bot_kita.y,
lokasi_base.x, lokasi_base.y)
        END IF

        // Menghitung jarak dari lokasi bot ke lokasi base
        jarak_ke_base ← |lokasi_bot_kita.x - lokasi_base.x| +
|lokasi_bot_kita.y - lokasi_base.y|

        // Menghitung waktu kembali ke base dengan asumsi 1 langkah itu 1 detik
waktu_kembali_ke_base ← jarak_ke_base × 1000
```

```

// Jika jumlah waktu yang tersisa ≤ waktu untuk kembali ke base
// maka set detik_detik_kematian sebagai true
IF jumlah_waktu_yang_tersisa ≤ waktu_kembali_ke_base THEN
    goal_position ← lokasi_base
    detik_detik_kematian ← true
END IF

// Jika inventory bot penuh maka set tujuan bot ke base
IF jumlah_diamond_di_inventory ≥ jumlah_maksimal_diamond_di_inventory
THEN
    goal_position ← lokasi_base
END IF

// Cek apakah bot musuh ada di dekat bot sebagai bentuk pertahanan
musuh_berada_di_dekat ← false
lokasi_musuh ← null

FOR EACH bot IN bots DO
    IF bot.id ≠ board_bot.id THEN
        dx ← |bot.position.x - lokasi_bot_kita.x|
        dy ← |bot.position.y - lokasi_bot_kita.y|

        // Cek jika bot musuh berada di dekat bot (Manhattan distance =
1)
        IF (dx = 1 and dy = 0) or (dx = 0 and dy = 1) THEN
            musuh_berada_di_dekat ← true
            lokasi_musuh ← bot.position
            BREAK
        END IF
    END IF
END FOR

// Jika ada musuh di dekat bot maka set tujuan ke lokasi musuh
// untuk melakukan tackle
IF musuh_berada_di_dekat = true THEN
    goal_position ← lokasi_musuh
END IF

// Jika bot sudah mencapai tujuan maka reset goal_position
IF goal_position ≠ null and position_equals(goal_position,
lokasi_bot_kita) THEN
    goal_position ← null
END IF

```

```

// Jika bot tidak memiliki tujuan maka lakukan perhitungan
// untuk mencari target diamond
IF goal_position = null THEN
    density_maksimal ← 0
    target_diamond ← null

    // Cari diamond dengan densitas tertinggi dan paling dekat dengan
base
    FOR EACH diamond IN diamonds DO
        // Hitung jarak lokasi diamond dengan lokasi bot
        jarak_bot ← |diamond.position.x - lokasi_bot_kita.x| +
|diamond.position.y - lokasi_bot_kita.y|

        // Hitung jarak lokasi diamond dengan lokasi base
        jarak_base ← |diamond.position.x - lokasi_base.x| +
|diamond.position.y - lokasi_base.y|

        // Tambahkan jarak lokasi diamond dengan lokasi bot dan jarak
// lokasi diamond dengan lokasi base sebagai total_jarak
        total_jarak ← jarak_bot + jarak_base

        // Mencari jumlah point diamond
        diamond_points ← diamond.properties.points

        // Menghitung density dengan membagi jumlah poin diamond
// dengan total jarak ditambah 1
        density ← diamond_points / (total_jarak + 1)

        // Jika density lebih tinggi dibandingkan density yang sudah
// dihitung sebelumnya, maka set density yang sudah dihitung
sebelumnya
        // sebagai density maksimal dan set diamond saat ini sebagai
target
        IF density > density_maksimal THEN
            density_maksimal ← density
            target_diamond ← diamond
        END IF
    END FOR

    // Jika diamond dengan density tertinggi ditemukan maka
// set tujuan bot ke lokasi diamond tersebut
// Jika diamond dengan density tertinggi tidak ditemukan
// maka set tujuan bot ke lokasi base
    IF target_diamond ≠ null THEN

```

```

        goal_position ← target_diamond.position
    ELSE
        goal_position ← lokasi_base
    END IF
END IF

    // Method next_move akan selalu mengembalikan arah dari
    // tujuan bot atau goal_position
    RETURN get_direction(lokasi_bot_kita.x, lokasi_bot_kita.y,
goal_position.x, goal_position.y)

END FUNCTION
END CLASS

```

2. Penjelasan Alur Program

- 1.) Ketika program dimulai, bot akan menginisialisasi dua variabel penting yaitu *goal_position* untuk menyimpan tujuan pergerakan bot dan *detik_detik_kematian* sebagai flag untuk menandai bahwa bot sedang sekarat dan harus cepat-cepat kembali ke base. Method *next_move* akan selalu dipanggil oleh program bot starter pack untuk mendapatkan output arah/direction dari bot.
- 2.) Ketika method *next_move* dipanggil. Program akan mengekstrak informasi penting dari state permainan seperti posisi bot, lokasi base, jumlah diamond di inventory, waktu yang tersisa, dan data diamond serta bot lain di papan. Program mengganti nama variabel supaya mempermudah dalam mengimplementasi atau melakukan coding.
- 3.) Program kemudian melakukan serangkaian pengecekan kondisi berdasarkan prioritas. Prioritas tertinggi adalah ketika bot sekarat, dimana jika flag *detik_detik_kematian* aktif, bot akan langsung bergerak menuju base tanpa mempertimbangkan hal lain.
- 4.) Selanjutnya, program menghitung jarak Manhattan dari lokasi bot saat ini ke lokasi base dan mengkonversinya menjadi waktu yang dibutuhkan dengan asumsi setiap langkah membutuhkan 1 detik (1000 milidetik). Program kemudian mengimplementasikan manajemen waktu yang mengecek apakah waktu yang tersisa cukup untuk kembali ke base. Jika waktu yang tersisa kurang dari atau sama dengan waktu yang dibutuhkan untuk pulang ke base, maka program akan mengaktifkan mode *detik_detik_kematian* dan menetapkan lokasi base sebagai tujuan.

- 5.) Pengecekan berikutnya adalah manajemen inventory, dimana bot akan kembali ke base jika inventory sudah penuh dengan diamond untuk mengosongkan inventory dan mendapatkan poin.
- 6.) Program mengimplementasikan fitur pertahanan melalui deteksi musuh di sekitar bot. Program akan memeriksa semua bot lain di papan dan menghitung jarak Manhattan mereka dengan bot kita. Jika ditemukan bot musuh yang berada tepat di sebelah bot kita (jarak Manhattan = 1), maka bot akan memprioritaskan untuk melakukan tackle dengan menetapkan posisi musuh sebagai tujuan. Strategi ini bertujuan untuk menyerang terlebih dahulu sebelum diserang oleh musuh.
- 7.) Untuk melakukan navigasi, program menggunakan sistem goal-based movement dimana bot akan terus bergerak menuju *goal_position* yang telah ditetapkan. Ketika bot mencapai tujuannya, *goal_position* akan di-reset menjadi null sehingga bot dapat mencari tujuan baru. Jika bot tidak memiliki tujuan, program akan menjalankan algoritma pencarian diamond yang optimal dengan menghitung density setiap diamond yang tersedia di papan.
- 8.) Algoritma pemilihan diamond menggunakan konsep density yang dihitung dengan membagi jumlah poin diamond dengan total jarak (jarak dari bot ke diamond plus jarak dari diamond ke base) ditambah 1. Persamaan ini mempertimbangkan tidak hanya nilai diamond tetapi juga efisiensi perjalanan, dimana diamond yang memiliki nilai tinggi dan berada di lokasi strategis (dekat dengan bot dan tidak terlalu jauh dari base) akan memiliki prioritas tertinggi. Program akan melakukan iterasi pada semua diamond yang tersedia dan memilih diamond dengan density tertinggi sebagai target selanjutnya.
- 9.) Sebagai fallback mechanism, jika tidak ada diamond yang tersedia di papan, bot akan menetapkan base sebagai tujuan untuk memastikan bot selalu memiliki arah pergerakan yang jelas.
- 10.) Pada akhir setiap perhitungan, program akan mengembalikan arah pergerakan yang dihitung menggunakan fungsi *get_direction* berdasarkan lokasi bot dan *goal_position* yang telah ditetapkan.

4.2 Struktur Data yang Digunakan

Program ini dibuat dengan menggunakan paradigma pemrograman berorientasi objek. Dalam paradigma ini, program dibagi menjadi unit-unit yang disebut objek, yang masing-masing

mewakili entitas atau konsep dalam program. Setiap objek memiliki atribut dan metode yang mendefinisikan sifat dan perilaku objek tersebut. Dalam konteks program Bot Etimo Diamonds ini, terdapat beberapa kelas yang mendefinisikan objek-objek dalam program. Misalnya, ada kelas *MalingProPlayer* yang mendefinisikan bot itu sendiri. Bot ini memiliki atribut seperti *goal_position* dan *detik_detik_kematian* yang digunakan untuk menyimpan informasi tentang posisi tujuan bot dan status kondisi kritis waktu untuk kembali ke base.

Selain itu, ada kelas *GameObject*, *Position*, *Board*, dan *Diamond* yang didefinisikan dalam *models.py*. Kelas *GameObject* digunakan untuk merepresentasikan bot dalam permainan dengan properties seperti *id*, *position*, dan properties yang berisi informasi base, diamonds di inventory, *milliseconds_left*, dan *inventory_size*. Kelas *Position* menyimpan koordinat x dan y untuk lokasi objek di papan permainan. Kelas *Board* merepresentasikan keadaan papan permainan yang berisi daftar diamonds dan bots yang tersedia. Kelas *Diamond* digunakan untuk menyimpan informasi tentang diamond di papan dengan properties *position* dan *points*.

Ada beberapa fungsi utilitas yang didefinisikan dalam *util.py*, seperti *get_direction* dan *position_equals*. Fungsi *get_direction* digunakan untuk menghitung arah pergerakan dari posisi saat ini ke posisi tujuan, sedangkan *position_equals* digunakan untuk membandingkan apakah dua posisi adalah sama. Fungsi-fungsi ini digunakan untuk melakukan operasi yang berkaitan dengan navigasi dan perbandingan posisi dalam game. Program juga menggunakan berbagai variabel lokal dalam method *next_move* untuk menyimpan informasi sementara seperti *lokasi_base*, *lokasi_bot_kita*, *jumlah_diamond_di_inventory*, dan variabel-variabel perhitungan seperti *jarak_ke_base*, *density_maksimal*, dan *target_diamond*. Variabel-variabel ini memungkinkan bot untuk melakukan analisis dan pengambilan keputusan yang efisien. Secara keseluruhan, struktur data ini memungkinkan bot kami untuk berinteraksi dengan lingkungan permainannya dan membuat keputusan tentang gerakan berikutnya berdasarkan kondisi saat ini di papan permainan.

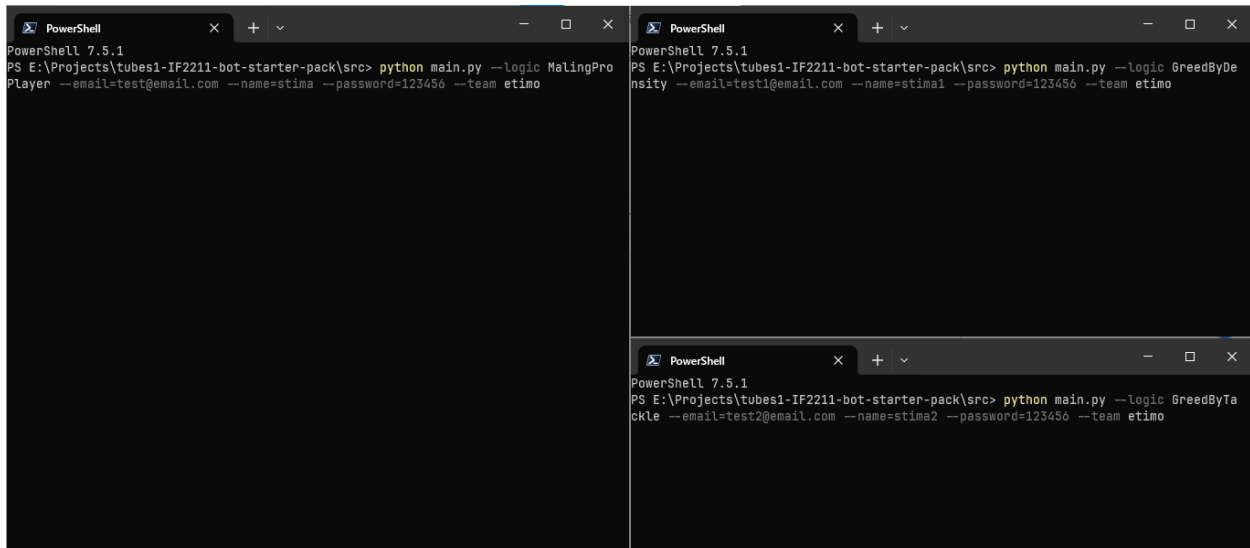
4.3 Pengujian Program

1. Skenario Pengujian

Kami melakukan pengujian bot kami dengan mengintegrasikan bot kami dengan bot yang memiliki algoritma greedy yang kami sebutkan sebelumnya seperti *Greed by Tackle*, *Greed by Density*. Kami juga melakukan pengujian ketika botnya sendirian di papan. Setelah permainan berjalan kami memantau semua aktivitas yang dilakukan oleh bot kami.

2. Hasil Pengujian dan Analisis

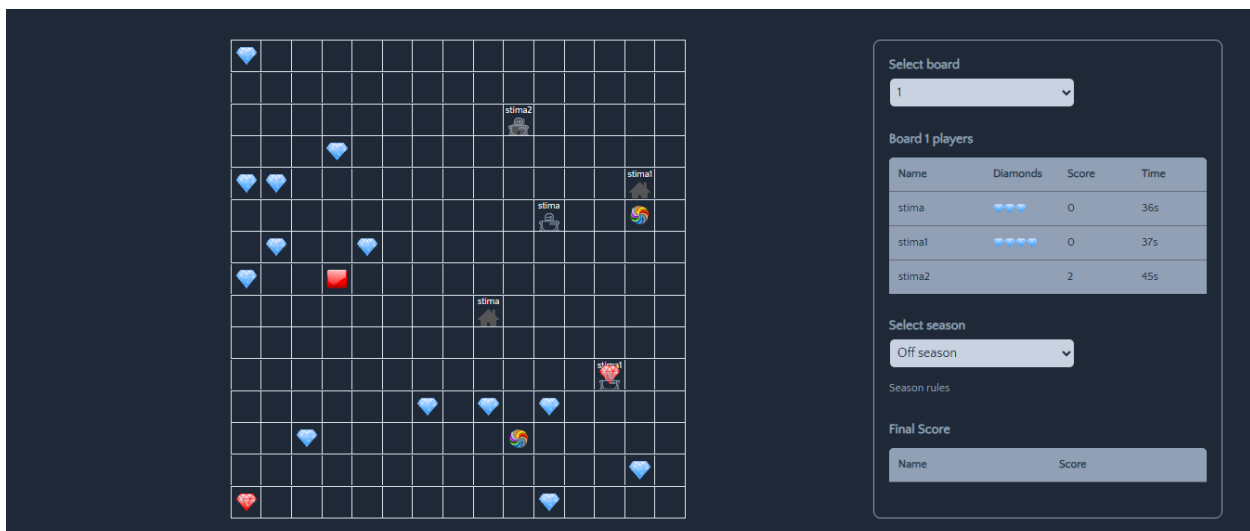
Pertama kami melakukan pengujian dengan cara menjalankan bot kami bersamaan dengan 2 bot lain yang menerapkan algoritma *Greed by Density* dan *Greed by Tackle*.



```
PowerShell 7.5.1
PS E:\Projects\tubes1-IF2211-bot-starter-pack\src> python main.py --logic MalingPro
Player --email=test@email.com --name=stima --password=123456 --team etimo

PowerShell 7.5.1
PS E:\Projects\tubes1-IF2211-bot-starter-pack\src> python main.py --logic GreedByDe
nsity --email=test1@email.com --name=stima1 --password=123456 --team etimo

PowerShell 7.5.1
PS E:\Projects\tubes1-IF2211-bot-starter-pack\src> python main.py --logic GreedByTa
ckle --email=test2@email.com --name=stima2 --password=123456 --team etimo
```



The screenshot displays a game interface with a 10x10 grid. The grid contains several blue diamonds and a red square. Three player icons are visible: stima2, stima, and stima1. The sidebar on the right contains the following information:

Select board: 1

Board 1 players

Name	Diamonds	Score	Time
stima	0	0	36s
stima1	0	0	37s
stima2	2	0	45s

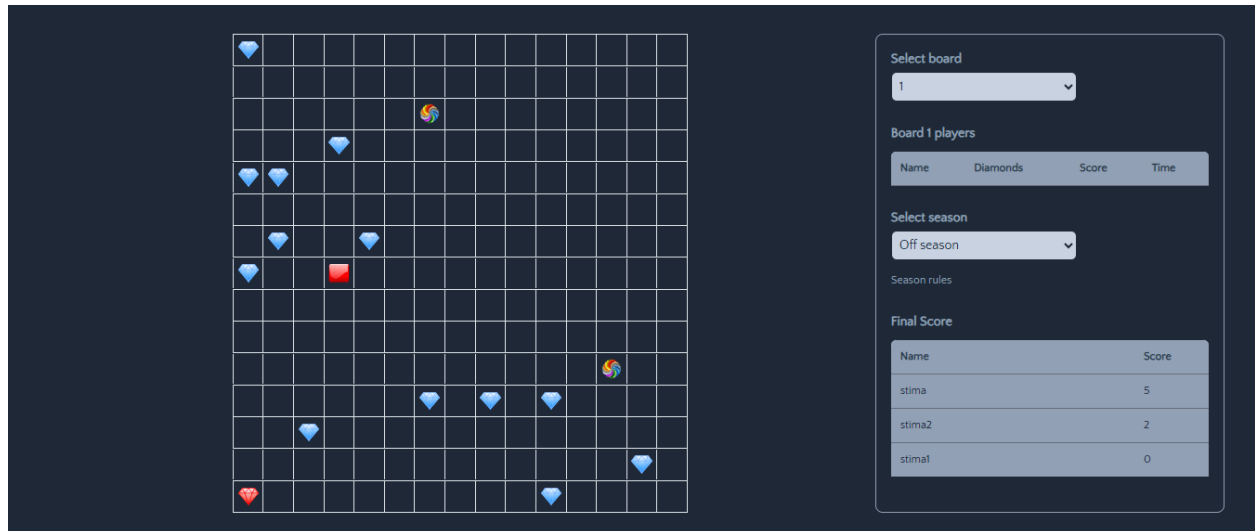
Select season: Off season

Season rules

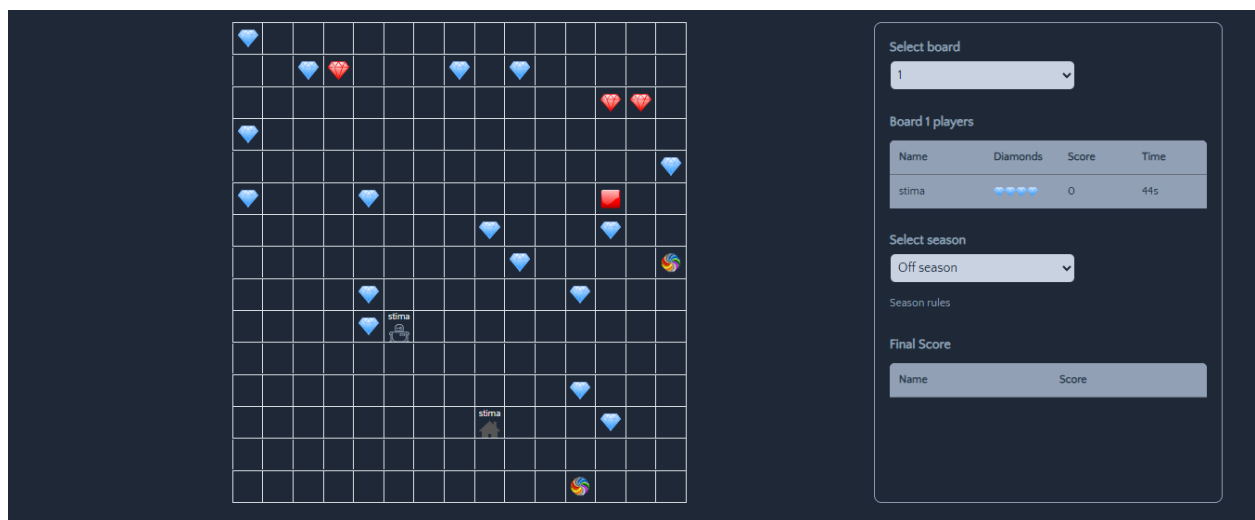
Final Score

Name	Score
------	-------

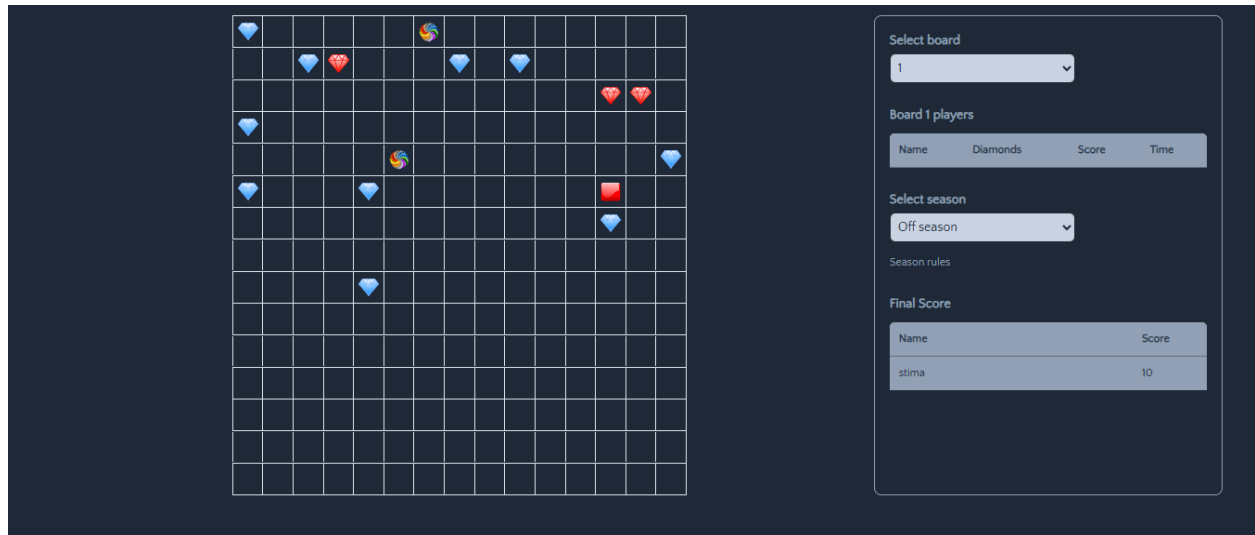
Berdasarkan pengujian menggunakan beberapa bot lain yang menerapkan algoritma greedy yang telah kami uraikan sebelumnya. Bot dengan algoritma yang kami pilih unggul dengan score paling tinggi, padahal base bot kami itu dekat dengan base bot *Greed by Tackle*, yang artinya bot kami sangat rawan ditackle oleh bot tersebut.



Bot dengan nama Stima adalah bot dengan algoritma yang kami pilih, kemudian bot dengan nama stima1 adalah bot dengan algoritma *Greed by Density*. Bot dengan nama stima2 adalah bot dengan algoritma *Greed by Tackle*.



Selanjutnya kami melakukan pengujian bot dengan cara menjalankan bot kami sendiri di papan. Saat melakukan pengujian, bot kami sedikit kurang beruntung karena spawn di wilayah yang jumlah sebaran diamondnya sedikit.



Hasil dari pengujian kami. Bot kami mendapatkan score sebanyak 10 poin ketika sendirian di papan. Hal ini disebabkan karena bot kami memilih diamond yang jaraknya dekat dengan base, kemudian bot kami berusaha mengamankan diamond ketika berada dalam kondisi sekarat.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Ada banyak kasus atau kombinasi kejadian yang mungkin terjadi pada permainan Etimo Diamonds. Untuk mendapatkan kemenangan pada permainan tersebut, algoritma greedy dapat diimplementasikan untuk mendapatkan langkah yang terbaik untuk mengumpulkan poin selama permainan. Algoritma greedy yang dapat digunakan juga tidak hanya satu, tetapi ada beberapa kemungkinan tergantung dari situasi dan kondisi. Dengan melakukan pertimbangan terhadap efisiensi, efektivitas, dan risiko dari kumpulan alternatif algoritma greedy yang ada, kami memutuskan algoritma Greedy by Density diamond di sekitar base dengan tambahan modifikasi dari beberapa solusi yang sudah ada sebagai algoritma terbaik.

5.2 Saran

Saran dari kami untuk pengembang lain yang ingin mengembangkan bot permainan Etimo Diamonds:

1. Kombinasikan beberapa solusi algoritma greedy yang sudah diuraikan untuk menciptakan algoritma greedy yang paling efektif dalam berbagai situasi.
2. Perhatikan object seperti teleporter dan red button agar dapat memaksimalkan diamond yang didapatkan. Kami sebelumnya tidak mengimplementasikan perhitungan untuk object itu karena di dalam source code bot starter tidak disediakan properti untuk mengetahui object tersebut di papan.

LAMPIRAN

A. Repository Github

https://github.com/theglitchpast/tubes1_mykisah

B. Video Penjelasan (link GDrive)

<https://youtu.be/xRYuPhd1s8k>

https://drive.google.com/file/d/1wN1EhyH9qn_ZSNNwK25PsEqMQLHho2-G/view

DAFTAR PUSTAKA

- [1] R. Munir, “Algoritma Greedy (Bagian 1),” Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, Bandung, Indonesia, 2025. [Online]. Tersedia: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-\(2025\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf) [Diakses: 1 Jun. 2025].
- [2] R. Munir, “Algoritma Greedy (Bagian 2),” Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, Bandung, Indonesia, 2025. [Online]. Tersedia: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-\(2025\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-(2025)-Bag2.pdf) [Diakses: 1 Jun. 2025].
- [3] R. Munir, “Algoritma Greedy (Bagian 3),” Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, Bandung, Indonesia, 2025. [Online]. Tersedia: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/06-Algoritma-Greedy-\(2025\)-Bag3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/06-Algoritma-Greedy-(2025)-Bag3.pdf) [Diakses: 1 Jun. 2025].

