

IF2211 – STRATEGI ALGORITMA

**PEMANFAATAN ALGORITMA GREEDY DALAM
BOT PERMAINAN “DIAMONDS”**

LAPORAN TUGAS BESAR 1



Oleh :
susugratis

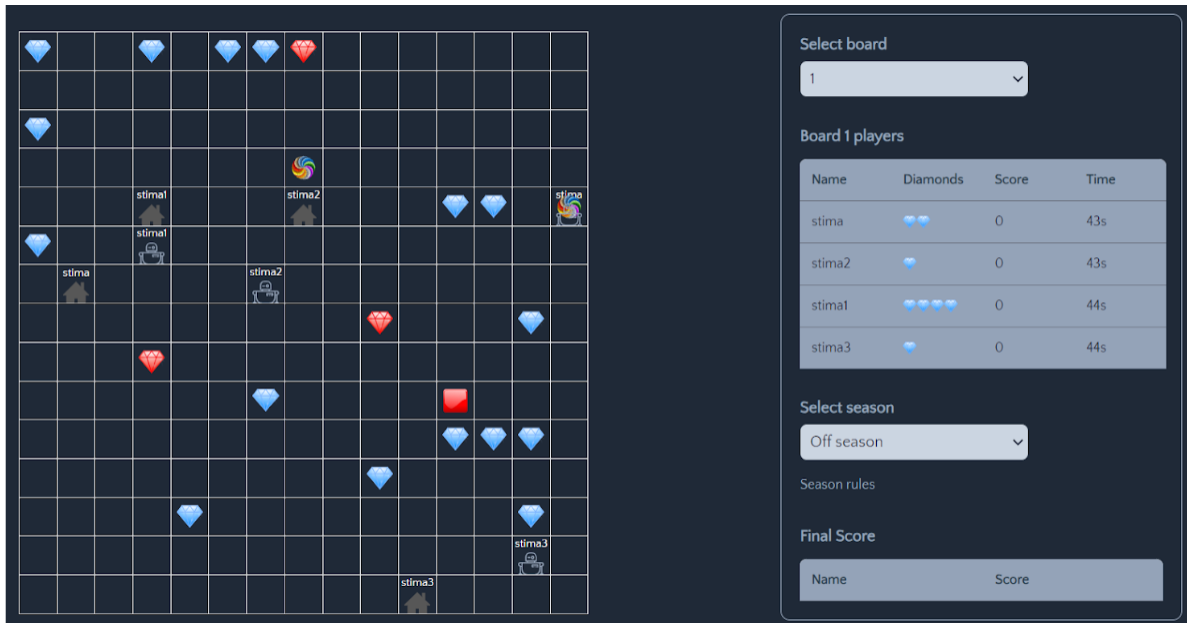
Aland Mulia Pratama	13522124
Mohamad Akmal Ramadan	13522161
Atqiya Haydar Luqman	13522163

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024

DAFTAR ISI

BAB I	2
DESKRIPSI TUGAS	2
BAB II.....	7
LANDASAN TEORI	7
2.1. Algoritma Greedy	7
2.2. Game Engine Diamonds	8
2.3. Struktur Game Engine	8
2.4. Bot Engine Diamonds.....	9
2.5. Alur Permainan Diamonds by Etimo	9
2.5. 1. Menjalankan <i>Game Engine</i> Diamonds by Etimo	9
2.5. 2. Mengembangkan <i>Bot Engine</i> Permainan Dianonds by Etimo	10
2.6. Kesalahan umum dalam Konfigurasi engine Diamonds by Etimo	11
BAB III	13
Aplikasi Strategy Greedy	13
3.1. Eksplorasi Alternatif Solusi <i>Greedy</i>	13
3.1.1 <i>Greedy by Diamonds</i>	13
3.1.2 <i>Greedy by Diamond Game Button</i>	15
3.1.3 Greedy by Base	17
3.1.4 Greedy by Avoid Teleporter	19
3.1.5 Greedy by Milliseconds Left	20
3.2. Strategi <i>Greedy</i> yang Dipilih	22
BAB IV	25
IMPLEMENTASI DAN PENGUJIAN	25
4.1. Implementasi dalam <i>Pseudocode</i>	25
4.2. Struktur Data Program	30
4.3. Analisis dan Pengujian	32
BAB V PENUTUP	37
5.1. Kesimpulan.....	37
5.2. Saran.....	37
5.3. Refleksi.....	37

BAB I DESKRIPSI TUGAS



Gambar 1.1. Permainan *programming challenge* Diamonds

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

Untuk mengimplementasikan algoritma pada bot tersebut, mahasiswa dapat menggunakan *game engine* dan membuat bot dari *bot starter pack* yang telah tersedia pada pranala berikut.

- **Game engine** : <https://github.com/haziqam/tubes1-IF2211-game-engine/releases/tag/v1.1.0>
- **Bot starter pack** : <https://github.com/haziqam/tubes1-IF2211-bot-starter-pack/releases/tag/v1.0.1>

Komponen-komponen dari permainan Diamonds antara lain:

1. Diamonds



Gambar 1.2. Komponen Diamonds

Untuk memenangkan pertandingan, kita harus mengumpulkan *diamond* ini sebanyak-banyaknya dengan melewati/melangkahnya. 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



Gambar 1.3. Komponen Reset 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.

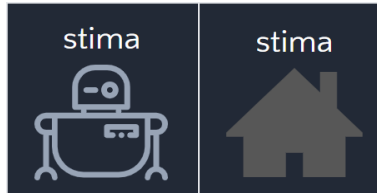
3. Teleporters



Gambar 1.3. Komponen 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



Gambar 1.4. Komponen Teleporters

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

Board 1 players			
Name	Diamonds	Score	Time
stima	💎💎	0	43s
stima2	💎	0	43s
stima1	💎💎💎💎	0	44s
stima3	💎	0	44s

Gambar 1.5. Inventory Bots

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 permainan yang digunakan pada tugas besar ini disesuaikan dengan spesifikasi yang disediakan oleh game engine Diamonds. Beberapa spesifikasi bot permainan Diamonds adalah sebagai berikut:

1. Buatlah program sederhana dalam bahasa Python yang mengimplementasikan algoritma Greedy pada bot permainan Diamonds dengan tujuan memenangkan permainan.
2. Tugas dikerjakan berkelompok dengan anggota minimal 2 orang dan maksimal 3 orang, boleh lintas kelas dan lintas kampus.
3. 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.
4. 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.
5. Program harus mengandung komentar yang jelas, dan untuk setiap strategi greedy yang disebutkan, harus dilengkapi dengan kode sumber yang dibuat.
6. Mahasiswa dilarang menggunakan kode program yang diunduh dari Internet. Mahasiswa harus membuat program sendiri, diperbolehkan untuk belajar dari program yang sudah ada.
7. Mahasiswa dianggap sudah melihat dokumentasi dari game engine, sehingga tidak terjadi kesalahpahaman spesifikasi antara mahasiswa dan asisten.
8. 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.
9. 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”.

IF2211	Strategi Algoritma	Tugas Besar 1	5
--------	--------------------	---------------	---

10. Terdapat juga demo dari program yang telah dibuat. Pengumuman tentang demo menunggu pemberitahuan lebih lanjut dari asisten.
11. 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. Algoritma Greedy

Algoritma *greedy* adalah metode yang paling populer dan sederhana untuk memecahkan persoalan optimasi langkah. Persoalan optimasi yang akan digunakan pada bot permainan Etime *Diamonds* adalah mengambil diamonds sebanyak-banyaknya untuk dikumpulkan ke dalam base bots. Algoritma *greedy* adalah algoritma yang memecahkan persoalan yang dibagi secara langkah per langkah sehingga pada setiap langkah akan mengambil pilihan yang terbaik sesuai dengan ketentuan yang dibuat pada programnya. Dengan pemilihan optimum lokal, algoritma *greedy* mengharapkan untuk mendapatkan optimum global.

Permasalahan yang diselesaikan menggunakan algoritma *greedy* dapat dipecah menjadi berbagai elemen-elemen seperti berikut:

1. Himpunan kandidat
Himpunan ini berisi kandidat yang akan dipilih pada setiap Langkah (misal: simpul/sisi di dalam graf, *tools*, task, koin, objek, karakter, dsb)
2. Himpunan solusi
kumpulan elemen yang dipilih dari himpunan kandidat untuk membentuk solusi akhir.
3. Fungsi solusi
fungsi yang mengevaluasi solusi yang berada pada himpunan solusi dengan tujuan untuk menentukan solusi yang optimum.
4. Fungsi seleksi (*selection function*)
Fungsi yang akan memilih kandidat berdasarkan suatu strategi atau ketentuan tertentu. Fungsi ini merupakan suatu pendekatan heuristic.
5. Fungsi kelayakan (*feasible*)
Fungsi akan memeriksa kandidat apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak).
6. Fungsi obyektif
Fungsi ini bertujuan untuk menjelaskan tujuan dari algoritma seperti memaksimumkan atau meminimumkan.

Algoritma *greedy* tidak selalu memberikan solusi yang optimal meskipun dia memilih Keputusan terbaik dalam waktu tersebut. Algoritma *greedy* biasanya digunakan untuk menghasilkan solusi hampiran (*approximation*) dimana waktu/kecepatan lebih diutamakan dibandingkan ketepatan/akurasi. Misalnya dalam mencari persoalan *Travelling Salesman Person* dibutuhkan waktu yang lama untuk menemukan solusi eksak atau pasti menggunakan algoritma *bruteforce*. Dengan menggunakan algoritma *greedy* kita dapat mendapatkan solusi yang tidak selalu optimal tapi dapat dianggap sebagai hampiran solusi optimal. Algoritma *greedy* yang optimal harus dilakukan pembuktian secara matematis.

Dalam permasalahan knapsack, perlu dilakukan berbagai alternatif solusi seperti *greedy by weight*, *greedy by profit*, dan *greedy by density*. Dari berbagai alternatif fungsi solusi ini akan dilakukan suatu seleksi berdasarkan output yang memberikan hasil maksimum. Dengan mempertimbangkan berbagai kasus, solusi dari algoritma *greedy* dapat semakin mendekati solusi optimal atau bahkan akan optimal terus menerus.

Algoritma *greedy* tidak selalu optimal dalam memecahkan masalah. Perlu dilakukan analisa yang mendalam dalam menerapkan algoritma *greedy* agar solusi dari *greedy* dapat mendekati optimum atau bahkan optimum. Hal penting dalam menerapkan algoritma *greedy* adalah dengan cara memperhatikan input *counterexample* dan mencoba untuk melakukan suatu penanganan khusus untuk *counterexample*. *Counterexample* harus diuji dan ditangani agar setidaknya program yang kita rancang tidak memberikan solusi yang jauh dari kata optimal.

2.2. Game Engine Diamonds

Game engine dari permainan Diamonds dapat ditemukan pada tautan <https://github.com/haziqam/tubes1-IF2211-game-engine/releases/tag/v1.1.0>. Game engine ini telah dipersiapkan oleh tim Asisten Lab IRK sehingga kita hanya perlu melakukan konfigurasi untuk menggunakannya. Ada beberapa *dependencies* yang digunakan pada game engine ini seperti:

1. Node.js
2. Docker desktop
3. Yarn

Game engine Diamonds ini tidak menggunakan engine compose sehingga kita memerlukan Docker untuk memproses dan menjalankan seluruh komponen dari game engine ini. Pada game engine ini, Docker juga membantu dalam proses berjalannya aplikasi secara independent dan konsisten di dalam mesin local.

2.3. Struktur Game Engine

Penting untuk mengetahui komponen-komponen dari game engine Diamonds karena akan sangat berguna bagi pengembangan bot dalam permainan Diamonds tersebut. Game engine berdasarkan tautan *releases* pada subbab 2.2. memiliki beberapa komponen yang strukturnya sendiri terdiri dari:

1. Folder *frontend* – berisi tampilan antarmuka pengguna untuk permainan Diamonds
2. Folder *backend* – berisi konfigurasi pengolahan data dari folder *types* dengan *frontend* seperti komponen-komponen dalam game yaitu Diamonds, Teleporter, Reset Button, Base, dan juga bots.
3. Folder *type* – penyimpanan *game objects* seperti base, bot, diamond, dummy-bot, teleport, teleport relocation, dan reset button.

2.4.Bot Engine Diamonds

Terdapat dua (2) file penting yang terdapat pada direktori utama file bot engine diamonds berdasarkan tautan *releases* pada subbab 2.2. File yang dibutuhkan penting untuk diketahui karena memberikan kita pemahaman tentang cara kerja bot engine diluar logika pemrograman yang kita rancang. File pendukung yang penting dalam bot engine tersebut adalah:

1. main.py – Membantu bot dalam permainan Diamonds untuk berkomunikasi dengan server melalui API. Bot ini menerima beberapa argumen baris perintah untuk mengkonfigurasi cara bermainnya.
2. decode.py – Menerima data dalam bentuk dictionary atau daftar dictionary. File ini mengonversi kunci-kunci dalam data tersebut menjadi snake case menggunakan `_keys_to_snake_case()` dan melakukan rekursi untuk mengonversi kunci-kunci dalam dictionary yang bersarang (nested) jika ada.

Pada bot engine ini terdapat juga folder game yang akan mendukung dan menopang keberjalanan logika program yang telah dirancang. Pada laporan ini akan dijelaskan file yang penting dalam folder game seperti util dan juga folder logic. Pada tugas besar ini, kita akan diminta untuk mengimplementasikan logika program algoritma *greedy* yang akan dirancang.

2.5.Alur Permainan Diamonds by Etimo

Terdapat beberapa prasyarat/*dependencies* yang harus diunduh dan/atau diinstalasi sebelum mengembangkan bot permainan diamonds dan juga menjalankan *game engine* Diamonds by Etimo. Prasyarat/*dependencies* tersebut adalah:

1. Game Engine
 - Node.js (<https://nodejs.org/en>)
 - Docker desktop (<https://www.docker.com/products/docker-desktop/>)
 - Yarn
2. Bot engine
 - Python

2.5.1. Menjalankan Game Engine Diamonds by Etimo

Pada permainan diamonds by Etimo, kita tidak memerlukan melakukan konfigurasi maupun perubahan apapun pada folder *game engine*. Sebelum menjalankan game engine, pastikan bahwa Node.js, Docker desktop, dan juga yarn telah terinstall pada perangkat anda. Setelah prasyarat untuk *game engine* telah terinstall silahkan mengikuti langkah-langkah berikut untuk menjalani *game engine*:

1. Download source code .zip pada tautan berikut <https://github.com/haziqam/tubes1-IF2211-game-engine/releases/tag/v1.1.0>
2. Extract zip tersebut, lalu masuk ke folder hasil extractnya dan buka terminal.

3. Masuk ke root directory dari project

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

4. Instalasi dependencies

```
yarn
```

5. Konfigurasi *environment variable* dengan menjalankan script berikut

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

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

```
compose up -d database
```

7. Jalankan script berikut untuk windows

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

8. Lakukan perintah build

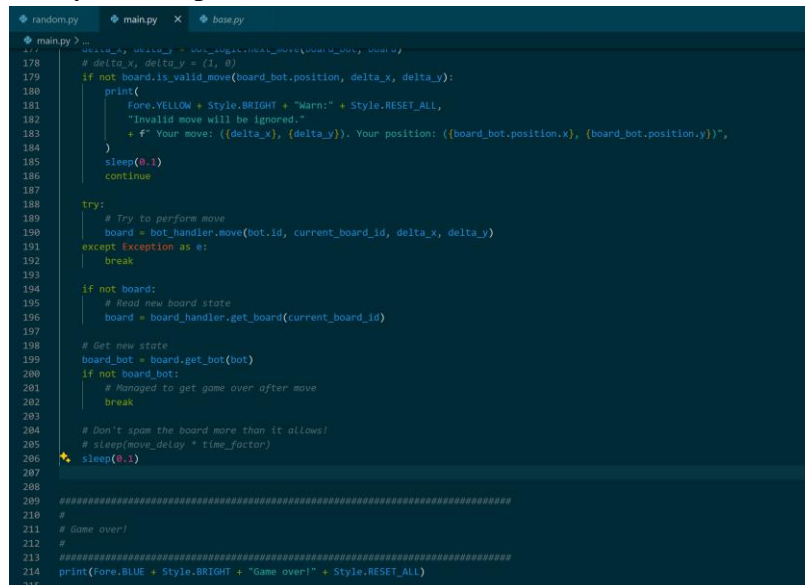
```
npm run build
```

9. Lakukan perintah start

```
npm run start
```

2.5.2. Mengembangkan *Bot Engine* Permainan Diamonds by Etimo

Sesuai dengan folder bot engine yang dapat diunduh melalui tautan yang terdapat pada Bab 1, pengembangan *bot engine* dapat dilakukan melalui file `./game/logic/random.py`. Kita juga dapat mengatur waktu setiap langkah bot berjalan pada file `./main.py` dengan mengubah *value* pada fungsi `Pustaka` yaitu `sleep(value)`.

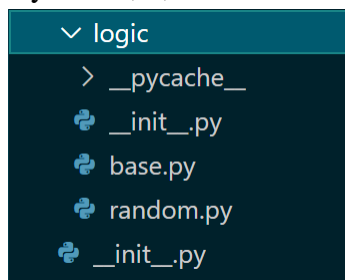


```
178 # delta_x, delta_y = (1, 0)
179 if not board.is_valid_move(board_bot.position, delta_x, delta_y):
180     print(
181         Fore.YELLOW + Style.BRIGHT + "Warn:" + Style.RESET_ALL,
182         "Invalid move will be ignored."
183         + f" Your move: {(delta_x), (delta_y)}. Your position: {(board_bot.position.x), (board_bot.position.y)}",
184     )
185     sleep(0.1)
186     continue
187
188 try:
189     # Try to perform move
190     board = bot_handler.move(bot.id, current_board_id, delta_x, delta_y)
191 except Exception as e:
192     break
193
194 if not board:
195     # Read new board state
196     board = board_handler.get_board(current_board_id)
197
198 # Get new state
199 board_bot = board.get_bot(bot)
200 if not board_bot:
201     # Managed to get game over after move
202     break
203
204 # Don't spam the board more than it allows!
205 # sleep(move_delay * time_factor)
206 sleep(0.1)
207
208
209 #####
210 #
211 # Game over!
212 #
213 #####
214 print(Fore.BLUE + Style.BRIGHT + "Game over!" + Style.RESET_ALL)
215
```

Gambar 2.5.2.1 Cuplikan main.py dengan fungsi Pustaka `sleep(value)`

Pada tugas pengembangan bot dengan algoritma *greedy* memiliki beberapa batasan seperti kita tidak diperbolehkan untuk mengubah struktur program maupun program selain file yang berisi program bot yang dikembangkan pada folder `./game/logic` oleh kelompok kami. Dalam melakukan uji coba, kita dapat mengubah *value* pada `sleep(value)` agar kita dapat lebih mudah menemui kemungkinan *error* atau *worst case* dari algoritma *greedy* yang telah dikembangkan dengan jumlah langkah yang lebih banyak untuk dilalui oleh bot.

Saat mengembangkan bot dalam folder `./game/logic` harus dipastikan bahwa terdapat fungsi next move dalam class yang kita rancang. Fungsi next move yang kita rancang juga harus menerima 2 parameter yaitu variabel dengan class ``GameObject`` dan juga ``Board``. Fungsi next move yang telah dibuat juga harus mengembalikan atau *return* dua value yaitu `delta_x` dan `delta_y` yang memiliki variasi nilai yaitu -1, 0, dan 1.

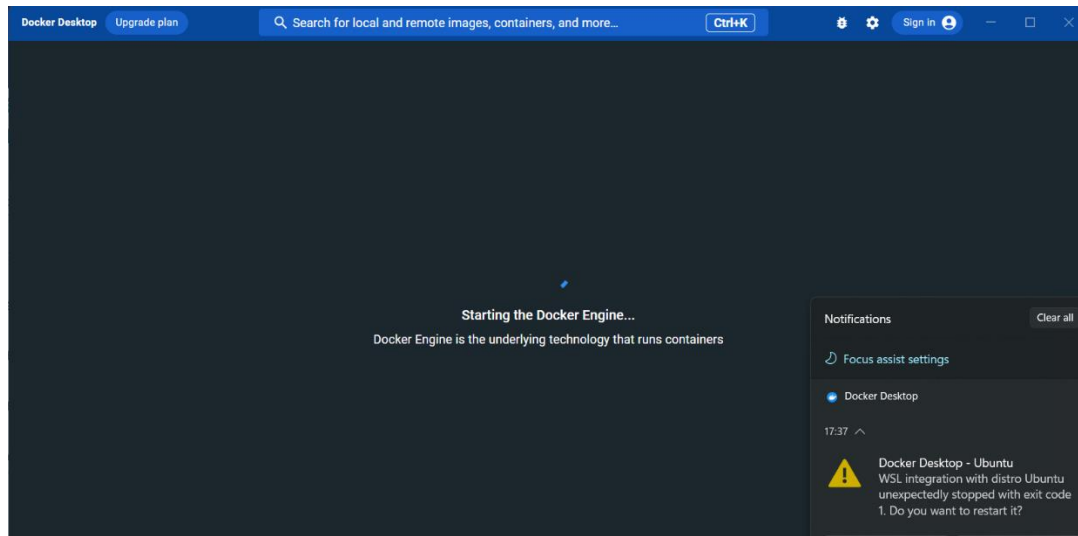


Gambar 2.5.2.2 File yang terdapat pada program logic

Untuk menjalankan satu bot (pada contoh ini, kita menjalankan satu bot dengan logic yang terdapat pada file `game/logic/random.py`) kita dapat menjalankan perintah `python main.py --logic Random --email=your_email@example.com --name=your_name --password=your_password --team etimo` dari root directory program *bot engine*. Kita dapat menjalankan beberapa bot sekaligus dengan cara menjalankan perintah `./run-bots.bat` untuk windows pada root directory program *bot engine*. Kita diperbolehkan untuk melakukan konfigurasi pada file `./run-bots.bat` sesuai dengan nama logika yang telah dikembangkan dan juga nama bot yang kita inginkan pada program.

2.6.Kesalahan umum dalam Konfigurasi engine Diamonds by Etimo

Kesalahan umum yang biasa dialami dalam melakukan konfigurasi *engine* Diamonds by Etimo ini sering dialami pada saat instalasi docker yang kemungkinan dialami oleh beberapa orang. Kesalahan ini berupa aplikasi docker desktop yang *stuck* pada tampilan starting engine.



Gambar 2.6.1 Docker desktop tertahan pada kondisi *starting*

Biasanya masalah tersebut dapat selesai dengan menghapus docker *desktop* dan melakukan instalasi ulang untuk docker *desktop* tersebut. Terdapat suatu kondisi dimana cara sebelumnya tidak menyelesaikan masalah sehingga harus dilakukan pengecekan apakah sudah melakukan instalasi linux dan WSL. Apabila WSL dan linux sudah terinstalasi pada perangkat, maka harus dilakukan pengecekan apakah terdapat folder konfigurasi docker pada linux. Masalah seperti ini dapat diatasi dengan cara melakukan instalasi ulang Windows Subsystem Linux (WSL) dan setelah itu lakukan instalasi ulang untuk docker desktop.

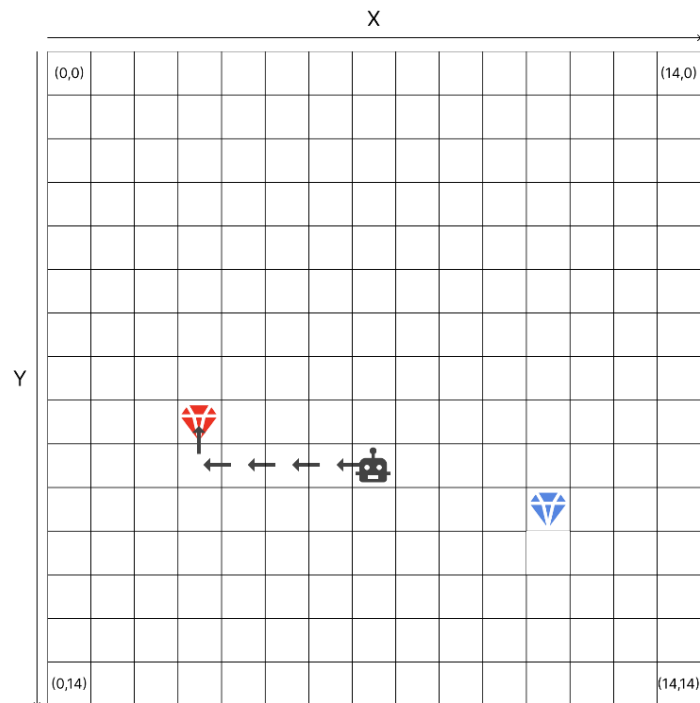
BAB III

Aplikasi Strategy Greedy

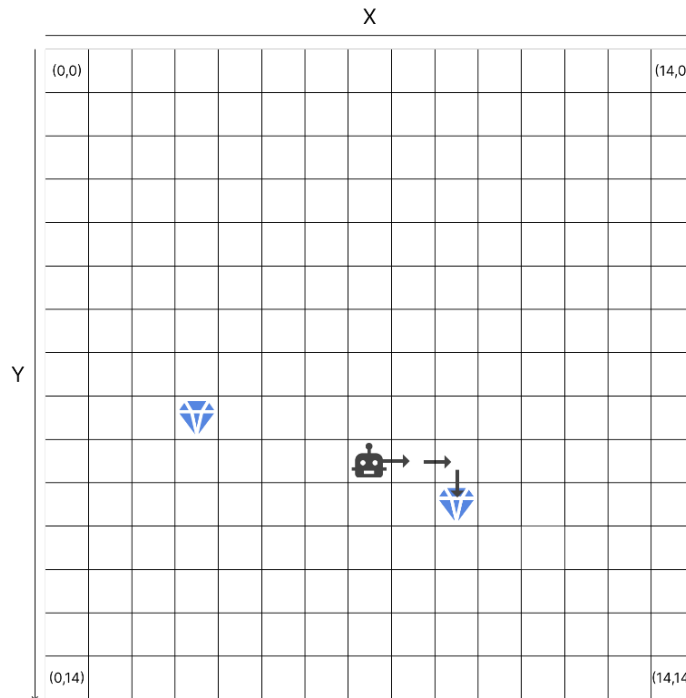
3.1. Eksplorasi Alternatif Solusi *Greedy*

3.1.1 *Greedy by Diamonds*

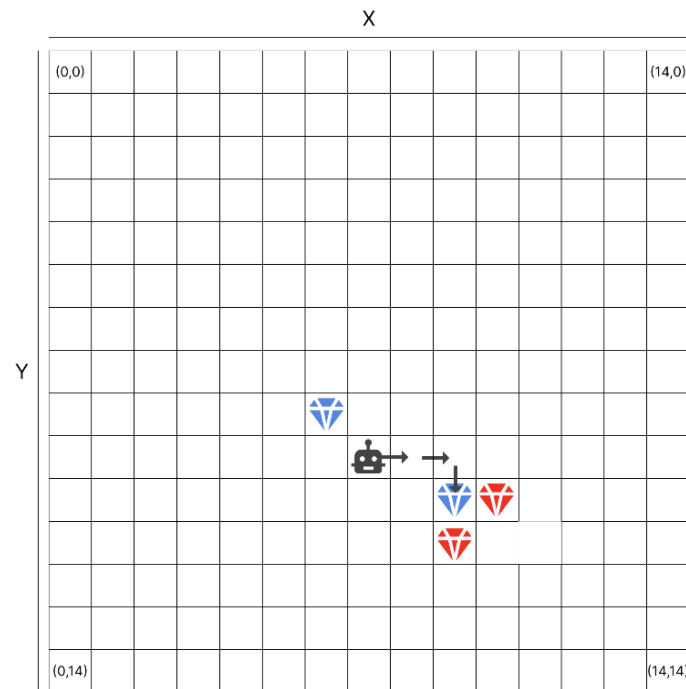
Strategi *Greedy by Diamonds* mencari hasil skor maksimum berdasarkan pengumpulan diamond-diamond yang terletak pada *game board*. Cara untuk mendapatkan skor pada game ini adalah dengan mengumpulkan diamond. Diamond terbagi menjadi dua jenis yaitu diamond biru dengan value satu dan diamond merah dengan value dua. Dengan memprioritaskan value yang dapat dicapai oleh bot, diamond-diamond yang berdekatan dianggap sebagai suatu value yang diakumulasi dengan pendekatan rekursif. Strategi *Greedy by Diamonds* ini juga mempertimbangkan sisa kapasitas inventory dari bot. Misalnya apabila inventory dari bot tersisa dua value, maka value yang diakumulasi dengan menggunakan pendekatan rekursif maksimal akan bernilai dua.



Gambar 3.1.1.1.1 Ilustrasi bot akan mengambil diamond dengan value tertinggi



Gambar 3.1.1.2 Ilustrasi bot akan mengambil diamond dengan jarak terdekat



Gambar 3.1.1.3 Ilustrai bot akan lebih memilih untuk mengambil diamond yang berkumpul
(asumsi inventory bot masih tersisa ≥ 3)

a. Mapping Elemen *Greedy*

- i. Himpunan Kandidat : Diamond biru, diamond merah, reset button, teleporter, base
- ii. Himpunan Solusi : Diamond biru, diamond merah

- iii. Fungsi Solusi : Diamond yang memiliki jarak terdekat dengan bot, kumpulan diamond-diamond yang saling berdekatan satu sama lain dengan perbedaan jarak satu petak membentuk suatu nilai yang besar
 - iv. Fungsi Seleksi : Pilih diamond yang paling dekat dengan bot atau diamond yang membentuk suatu nilai yang besar
 - v. Fungsi Kelayakan : Memeriksa apakah diamond dapat diraih oleh bot ketika inventory belum penuh dan waktu habis
 - vi. Fungsi Objektif : Mencari hasil skor yang maksimal
- b. Analisis Efisiensi Solusi
- Dengan menggunakan strategi ini, bot akan mencari suatu diamond dan koordinat yang sudah dikunjungi akan dicatat dan tidak akan dikunjungi lagi setelahnya. Strategi ini bergantung pada cara rekursif pencarian diamond yang saling berdekatan, yaitu berapa banyak panggilan rekursif yang dibuat dan bagaimana kedalaman rekursi terjadi. Kompleksitas waktu algoritma:

$$T(n) = O(n \log n)$$

n = jumlah langkah untuk menemukan diamond

Beberapa aspek seperti koordinat yang sudah dikunjungi, informasi tentang diamond yang telah dikumpulkan, dan stack call dalam rekursi akan memengaruhi ruang penyimpanan algoritma. Kompleksitas memori algoritma:

$$S(O) = O(N + M)$$

n = jumlah petak yang sudah dikunjungi, m = jumlah diamond yang sudah diambil

- c. Analisis Efektivitas Solusi
- Strategi *Greedy by Diamonds* akan tereksekusi dengan baik apabila :
- Tidak ada bot musuh yang mengincar diamond yang sama
 - Terdapat diamond yang berdekatan dengan value total bernilai lima, misalkan dua buah diamond merah dan satu buah diamond biru dan inventory bot akan langsung terisi penuh
- Strategi *Greedy by Diamonds* akan terkesekusi dengan buruk apabila :
- Ketika bot musuh mengincar diamond yang sama
 - Ketika tidak ada diamond yang saling berdekatan
 - Apabila rute yang ingin dilalui oleh bot terdapat 2 teleporter, bot akan mengulang rute yang sama yaitu melalui teleporter secara terus menerus

3.1.2 Greedy by Diamond Game Button

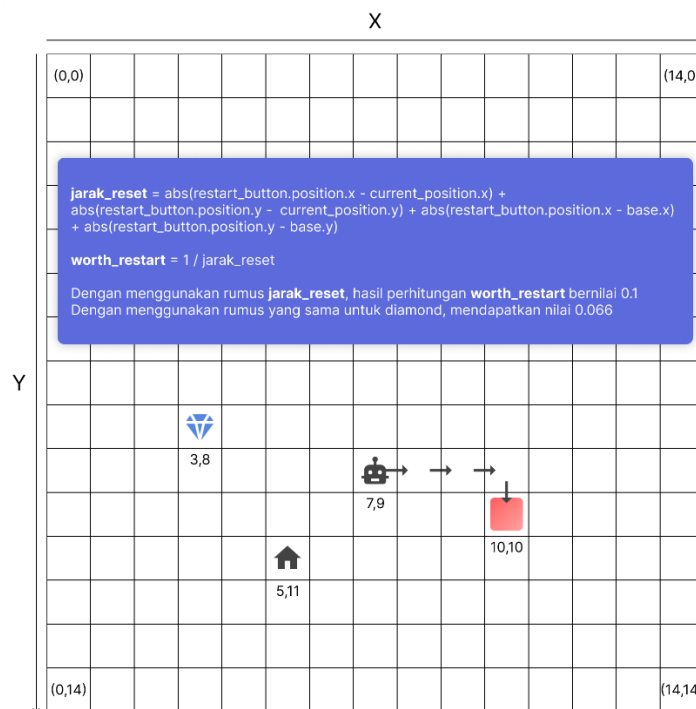
IF2211	Strategi Algoritma	Tugas Besar 1	15
--------	--------------------	---------------	----

Strategi *Greedy by Diamond Game Button* mencari reset button yang terdapat pada *game board* dengan tujuan untuk mereset map ketika tidak ada diamond yang sudah di dekat dengan bot dan diamond sudah sangat sedikit dalam map. Reset button memiliki value yang sama dengan diamond biru yaitu 1. Pada algoritma ini kita menentukan asumsi mengenai skala prioritas yang harus diambil oleh bot, urutan prioritasnya yaitu diamond merah, dan diamond biru sama dengan reset button. Nilai jarak reset button ditentukan dengan rumus

$$\begin{aligned} \text{jarak_reset} = & \text{abs}(\text{restart_button.position.y} - \text{current_position.x}) + \\ & \text{abs}(\text{restart_button.position.y} - \text{current_position.y}) + \\ & \text{abs}(\text{restart_button.position.x} - \text{base.x}) + \\ & \text{abs}(\text{restart_button.position.y} - \text{base.y}) \end{aligned}$$

Nilai prioritas dari reset button adalah

$$\text{worth_restart} = 1 / \text{jarak_reset}$$



Gambar 3.1.2.1 Ilustrasi *Greedy by Diamond Game Button*
(asumsi diamond sudah sangat sedikit dalam map)

a. Mapping Elemen *Greedy*

- i. Himpunan Kandidat : Diamond biru, diamond merah, reset button, teleporter, base
- ii. Himpunan Solusi : Reset button

- iii. Fungsi Solusi : Reset button yang dekat dengan bot ketika sudah hampir tidak ada diamond yang tersedia
- iv. Fungsi Seleksi : Pilih reset button yang ada dan berada dekat dengan bot
- v. Fungsi Kelayakan : Memeriksa apakah reset button mungkin untuk diraih
- vi. Fungsi Objektif : Meraih reset button dan mengulang kembali isi *game board* dan me-regenerate diamond

b. Analisis Efisiensi Solusi

Dengan menggunakan strategi ini, ketika diamond sudah hampir habis dan diamond yang berada di sekitar bot sudah tidak ada. Bot akan mencari reset button untuk mengulang kembali *board game*. Strategi ini menggunakan algoritma yang sama dengan *Greedy by Diamonds* hanya saja pada strategi ini dilakukan penentuan skala prioritas dan pemilihan keputusan berdasarkan kondisi yang sudah disebutkan. Kompleksitas waktu algoritma:

$$T(n) = O(n \log n)$$

n = jumlah petak

Penggunaan memori pada algoritma ini terdapat dalam tiga aspek yaitu koordinat yang sudah dikunjungi, jumlah diamond yang sudah diambil, dan jumlah reset button. Jumlah reset button dapat diabaikan karena dalam satu permainan jumlah reset button hanya ada satu. Kompleksitas memori algoritma:

$$S(O) = O(N + M)$$

n = jumlah petak yang sudah dikunjungi, m = jumlah diamond yang sudah diambil

c. Analisis Efektivitas Solusi

Strategi *Greedy by Game Button* akan tereksekusi dengan baik apabila :

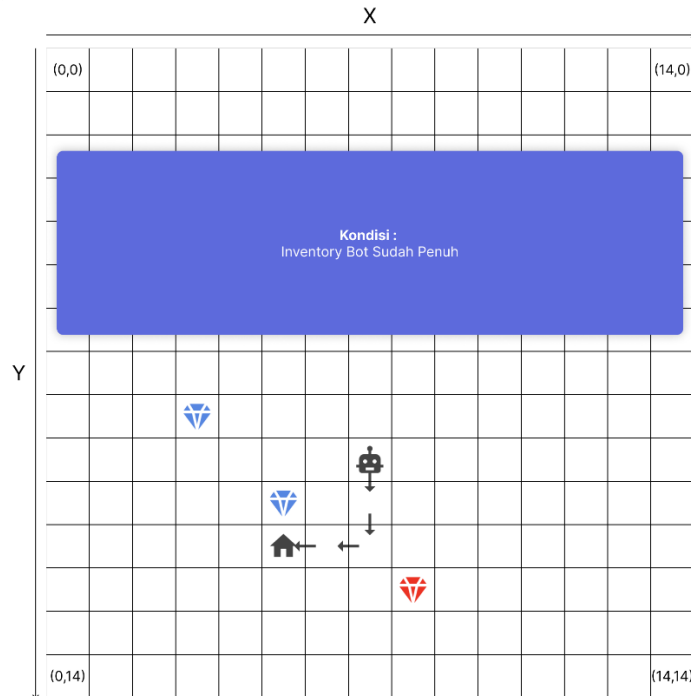
- Reset button tidak sangat jauh dengan bot
- Tidak ada bot musuh yang sedang menuju reset button

Strategi *Greedy by Game Button* akan tereksekusi dengan buruk apabila :

- Reset button berada sangat jauh dengan bot
- Ada bot musuh yang sedang menuju reset button

3.1.3 Greedy by Base

Strategi *Greedy by Base* membuat bot menjadikan tujuan sekarnya adalah base ketika inventory bot sudah penuh atau waktu untuk mendapatkan diamond sudah tidak cukup.



Gambar 3.1.3.1 Ilustrasi *Greedy by Base*

a. Mapping Elemen *Greedy*

- i. Himpunan Kandidat : Diamond merah, diamond biru, reset button, teleporter, base
- ii. Himpunan Solusi : Base
- iii. Fungsi Solusi : Base ketika inventory bot sudah penuh dan tidak ada waktu untuk mengambil diamond kembali
- iv. Fungsi Seleksi : Mengatur tujuan bot menjadi base
- v. Fungsi Kelayakan : Memeriksa apakah base dekat dengan bot sehingga bot dapat kembali ke base
- vi. Fungsi Objektif : Bot kembali ke base untuk dapat menyimpan value diamond

b. Analisis Efisiensi Solusi

Dengan menggunakan strategi ini, bot akan kembali ke base apabila inventory sudah penuh atau waktu yang tersisa untuk mengambil diamond sudah habis. Strategi ini menggunakan algoritma yang bergantung pada suatu kondisi yang sudah disebutkan. Algoritma ini memanfaatkan *if else statement*. Kompleksitas waktu algoritma:

$$T(n) = O(1)$$

Penggunaan memori pada algoritma ini terdapat pada koordinat base. Algoritma perlu menyimpan informasi tentang koordinat base di peta permainan. Jumlah memori yang

diperlukan untuk menyimpan koordinat base diabaikan karena hanya ada satu base milik bot dalam permainan. Kompleksitas memori algoritma:

$$S(O) = O(1)$$

c. Analisis Efektivitas Solusi

Strategi *Greedy by Base* akan tereksekusi dengan baik apabila :

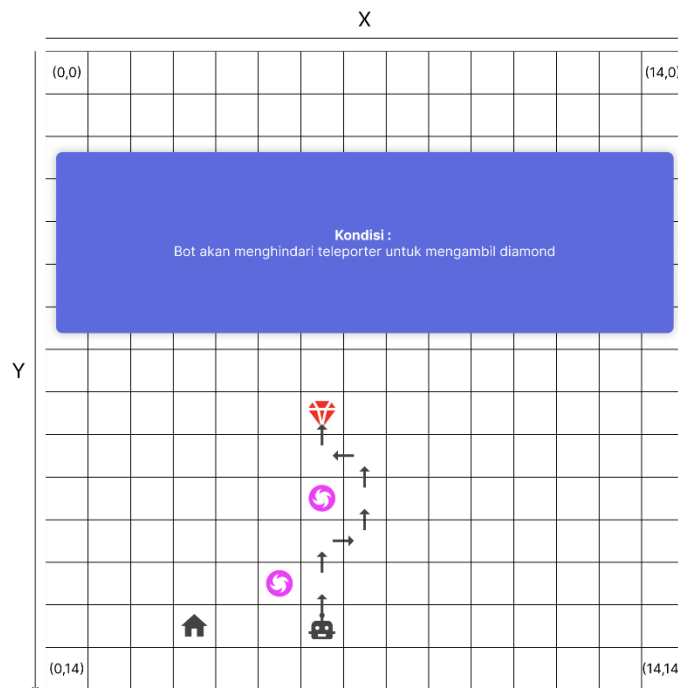
- Tidak ada bot musuh yang berada di jalur bot kita menuju base

Strategi *Greedy by Base* akan tereksekusi dengan buruk apabila :

- Ada bot musuh yang berada di jalur bot kita menuju base

3.1.4 Greedy by Avoid Teleporter

Strategi *Greedy by Avoid Teleporter* digunakan untuk menghindari teleport ketika bot sedang mencari diamond. Dengan menghindari teleport bot dapat memaksimalkan rute yang dipilih sehingga bot tidak mengulangi rute yang sama berulang ulang. Strategi ini digunakan untuk mengatasi *worst case* yang berada pada *Greedy by Diamonds*, yaitu pengulangan (*looping*) pada koordinat yang sudah dikunjungi.



Gambar 3.1.4.1 Ilustrasi *Greedy by Avoid Teleporter*

a. Mapping Elemen *Greedy*

- i. Himpunan Kandidat : Diamond merah, diamond biru, reset button, teleporter, base
- ii. Himpunan Solusi : Teleporter
- iii. Fungsi Solusi : Teleporter yang dihindari bot

- iv. Fungsi Seleksi : Menghindari teleporter
- v. Fungsi Kelayakan : Memeriksa apakah jalur bot dihalangi oleh teleporter untuk bot dapat menghindari teleporter
- vi. Fungsi Objektif : Bot menghindari teleporter

b. Analisis Efisiensi Solusi

Dengan menggunakan strategi ini, bot dapat menghindari teleport yang berada di jalurnya untuk mencegah terjadinya pengulangan rute yang sama. Kompleksitas waktu algoritma ini adalah

$$T(n) = O(1)$$

Penggunaan memori pada algoritma ini terdapat dalam koordinat teleporter. Algoritma perlu menyimpan informasi tentang koordinat teleporter dalam peta permainan untuk menentukan apakah jalur bot dihalangi oleh teleporter. Jumlah memori yang diperlukan dapat diabaikan karena hanya ada satu pasangan teleporter pada permainan. Kompleksitas memori algoritma:

$$S(O) = O(1)$$

c. Analisis Efektivitas Solusi

Strategi *Greedy by Avoid Teleporter* akan tereksekusi dengan baik apabila :

- Tidak ada bot yang muncul dari teleporter

Strategi *Greedy by Avoid Teleporter* akan tereksekusi dengan buruk apabila :

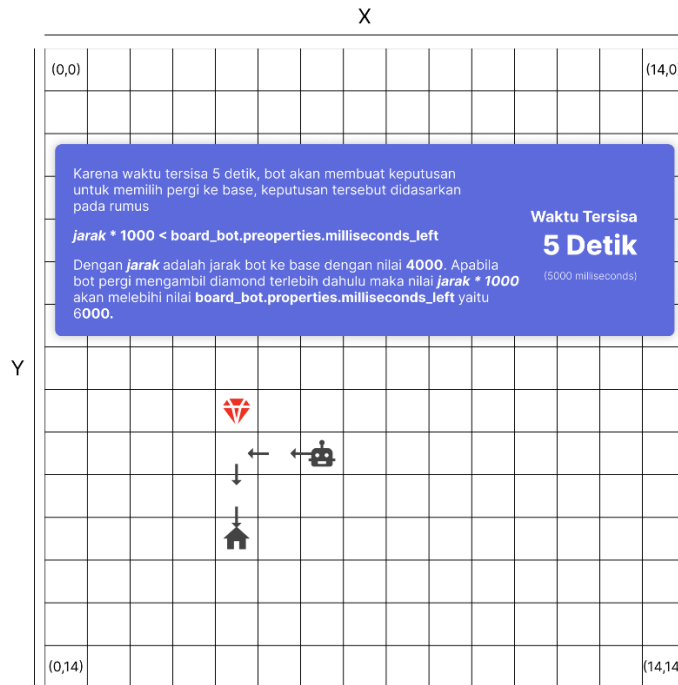
- Ada bot yang muncul dari teleporter

3.1.5 Greedy by Milliseconds Left

Strategi *Greedy by Milliseconds Left* membuat bot untuk menentukan keputusan berdasarkan sisa waktu yang tersedia. Apabila waktu cukup untuk mengambil suatu diamond, bot akan mengambil diamond tersebut. Apabila waktu tidak mencukupi, bot akan kembali ke base. *By default*, setiap langkah dari bot akan dijalankan selama satu detik. Perhitungan ini dilakukan dengan rumus

$$Jarak * 1000 < board_bot.properties.milliseconds_left$$

IF2211	Strategi Algoritma	Tugas Besar 1	20
--------	--------------------	---------------	----



Gambar 3.1.5.1 Ilustrasi *Greedy by Milliseconds Left*

a. Mapping Elemen *Greedy*

- i. Himpunan Kandidat : Diamond merah, diamond biru, reset button, teleporter, base
- ii. Himpunan Solusi : Diamond merah, diamond biru, base
- iii. Fungsi Solusi : Diamond apabila waktu masih mencukupi, base apabila waktu sudah tidak cukup
- iv. Fungsi Seleksi : Diamond yang dapat diambil atau kembali ke base
- v. Fungsi Kelayakan : Memeriksa waktu untuk menentukan keputusan
- vi. Fungsi Objektif : Mengambil diamond apabila waktu masih mencukup atau kembali ke base apabila waktu sudah tidak cukup

b. Analisis Efisiensi Solusi

Dengan menggunakan strategi ini, bot dapat menentukan keputusan apakah masih memungkinkan untuk mengambil diamond atau kembali ke base apabila waktu sudah tidak mencukupi. Kompleksitas waktu algoritma:

Ketika bot mengambil diamond :

$$T(n) = O(n \log n)$$

Ketika bot kembali ke base:

$$T(n) = O(1)$$

Penggunaan memori ketika bot mengambil diamond berbeda dengan ketika bot pergi ke base. Kompleksitas memori:

Ketika bot mengambil diamond:

$$S(O) = O(N + M)$$

n = jumlah petak yang sudah dikunjungi, m = jumlah diamond yang sudah diambil

Ketika bot kembali ke base:

$$S(O) = O(1)$$

c. Analisis Efektivitas Solusi

Strategi *Greedy by Milliseconds Left* akan tereksekusi dengan baik apabila :

- Waktu permainan tersisa sedikit lagi

Strategi *Greedy by Avoid Teleporter* akan tereksekusi dengan baik apabila :

- Waktu permainan tersisa banyak

3.2. Strategi *Greedy* yang Dipilih

Berdasarkan solusi-solusi yang telah dijelaskan sebelumnya, terdapat kelebihan dan kekurangan dari masing-masing solusi. Dikarenakan pada setiap ronde game mempunyai situasi yang tidak tentu, maka kami menentukan bot dapat mempertimbangkan pendekatan greedy yang sesuai dengan situasi yang sedang terjadi dan menentukan action apa yang akan dipilih.

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

1. Mengambil diamond-diamond yang berkumpul kemudian menjadikan kumpulan diamond tersebut sebagai value yang besar
2. Mengambil diamond yang terdekat dengan bot apabila waktu masih ada waktu tersisa
3. Menekan reset button apabila jumlah diamond di sekitar bot sudah tidak ada
4. Kembali ke base apabila inventory sudah penuh
5. Kembali ke base apabila waktu sudah tidak memungkinkan bot untuk mengambil diamond
6. Menghindari teleporter agar tidak mengunjungi koordinat yang sudah dikunjungi

Poin-poin diatas dibuat berdasarkan solusi-solusi greedy sebelumnya. Poin 1 merepresentasikan *Greedy by Diamonds*. Poin 2 merepresentasikan *Greedy by Diamonds* dan *Greedy by Milliseconds Left*. Poin 3 merepresentasikan *Greedy by Diamond Game Button*. Poin 4

merepresentasikan *Greedy by Base*. Poin 5 merepresentasikan *Greedy by Base* dan *Greedy by Milliseconds Left*. Point 6 merepresentasikan *Greedy by Avoiding Teleporter*.

Ketika mulai permainan dan inventory bot masih kosong, bot akan menggunakan algoritma *Greedy By Diamonds* untuk mengambil diamond dengan tujuan dapat mengumpulkan value yang besar dari diamond yang telah diambil. Bot dapat mengambil diamond secara satuan atau mengambil diamond yang saling berkumpul.

Ketika inventory bot sudah penuh, bot akan menggunakan algoritma *Greedy by Base* untuk menyimpan semua value dari diamond yang telah diambil ke dalam base.

Algoritma *Greedy by Diamond Reset Button* digunakan ketika diamond sekitar bot sudah habis atau diamond yang berada dalam map sudah hampir habis. Tujuan dari reset button ini adalah untuk mengulang kembali tampilan peta sehingga muncul kembali diamond-diamond tersebar di peta.

Untuk memperlancar jalur yang ditempuh oleh bot, algoritma *Greedy by Avoiding Teleporter* digunakan untuk menghindari teleporter yang ada pada map, algoritma ini digunakan agar bot tidak mengular koordinat petak yang sudah dikunjungi oleh bot sebelumnya.

Ketika waktu permainan sudah semakin sedikit, bot akan menentukan dua pilihan dengan menggunakan algoritma *Greedy by Milliseconds Left* yaitu apabila waktu masih mencukupi bot akan mengambil diamond, apabila sebaliknya maka bot akan kembali ke base.

Dengan menggunakan semua kombinasi algoritma *Greedy* bot dapat mendapatkan hasil yang maksimum dalam memperoleh skor pada permainan yang kemudian memenangkan permainan. Strategi *Greedy* dirancang secara strategis sehingga dapat digunakan pada situasi dan kondisi yang berbeda-beda.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi dalam *Pseudocode*

Dalam bot permainan Diamonds kami akan menjelaskan implementasi strategi algoritma *greedy* bot permainan “Diamonds” dalam bentuk *pseudocode*. Dalam implementasi program bot permainan diamonds menggunakan bahasa pemrograman python kami membuat dua kelas utama yaitu kelas `move` dan juga kelas `logic`. Dalam setiap kelas, implementasi strategi algoritma *greedy* kami buat dengan pendekatan fungsional dan kami akan membahas implementasi algoritma program dalam *pseudocode* dimulai dari kelas kelas `move` lalu kelas `logic`.

a) Kelas `move`

Kelas ini memiliki tujuan utama untuk melakukan inisiasi terhadap kedua teleport dan juga reset button yang terdapat dalam papan permainan.

Procedure initialize object

```
procedure initialize_object(move self):  
    # Inisialisasi atribut objek  
    future_move_x ← 0  
    future_move_y ← 0  
    initiating ← False
```

Procedure initialize

```
procedure initialize(move self, Board board): // inisiasi teleport dan  
reset button  
    for i traversal [0..len(board.game_objects)]:  
        if board.game_objects[i].type = "TeleportGameObject" then  
            self.teleport2 ← i  
        else if board.game_objects[i].type = "DiamondButtonGameObject" then  
            self.restart_button ← i  
    self.teleport1 ← self.teleport2 - 1
```

b) Kelas `logic`

Kelas ini berisi strategi algoritma *greedy* yang telah dipaparkan pada bab 3. Kelas ini terdiri dari inisiasi object dan juga beberapa *method* / fungsi yaitu *get_direction*, *recursive_search*, dan *next_move*. Fungsi *get_direction* mengembalikan dua nilai dalam bentuk integer yaitu *delta_x* dan juga *delta_y*. Dalam fungsi *get_direction* diterapkan strategi algoritma *greedy by avoiding teleporters*. Fungsi *recursive_search* digunakan untuk menerapkan strategi algoritma *greedy by diamonds* dengan mengembalikan nilai diamond di sekitarnya dan juga nilai dari diamond itu sendiri. Fungsi *next_move* digunakan untuk mengembalikan nilai *delta_x* dan *delta_y* untuk bot permainan.

Procedure initialize object

```
procedure initialize_object(Logic self):  
    // Inisialisasi atribut objek
```

```

max_distance_base ← 0
goal_position ← None
max_distance_bot ← 0
move ← move()

```

Function get_direction

```

Procedure get_direction(integer current_x, integer current_y, integer
teleport1_x, integer teleport1_y, integer teleport2_x, integer teleport2_y,
integer base)
    delta_x ← clamp(goal_position.x - current_x, -1, 1)
    delta_y ← clamp(goal_position.y - current_y, -1, 1)
    move.future_move_x ← 0
    move.future_move_y ← 0

    if delta_x ≠ 0 && delta_y ≠ 0 then
        if (current_x + delta_x, current_y) not in [(teleport1_x,
teleport1_y), (teleport2_x, teleport2_y)] and (current_x + delta_x, current_y
+ delta_y) not in [(teleport1_x, teleport1_y), (teleport2_x, teleport2_y)]
then
            delta_y ← 0
            else if (current_x + delta_x, current_y + delta_y) in [(teleport1_x,
teleport1_y), (teleport2_x, teleport2_y)] then
                delta_x ← 0
            else
                delta_x ← 0
            else if delta_x ≠ 0 && delta_y = 0 then
                if (current_x + delta_x, current_y) in [(teleport1_x, teleport1_y),
(teleport2_x, teleport2_y)] then
                    move.future_move_x ← delta_x
                    delta_x ← 0
                    if current_y ← 14 then
                        delta_y ← 1
                    else
                        delta_y ← 1
                else if delta_x = 0 && delta_y ≠ 0 then
                    if (current_x, current_y + delta_y) in [(teleport1_x, teleport1_y),
(teleport2_x, teleport2_y)] then
                        move.future_move_y ← delta_y
                        delta_y ← 0
                        if current_x = 14 then
                            delta_x ← -1
                        else
                            delta_x ← 1
                    else
                        delta_x ← 1

    → delta_x, delta_y

```

Function recursive_search

```

procedure recursive_search(diamonds, current_x, current_y, base,
bot_position, visited)
    if current_x > 14 || current_y > 14 || current_x < 0 or current_y < 0
then

```

```

        → 0, visited
    if (current_x, current_y) in visited then
        → 0, visited

    visited.add((current_x, current_y))
    points ← 0
    ada ← False

    for each diamond in diamonds
        if diamond.position.x = current_x && diamond.position.y = current_y
then
            ada ← True
            points += diamond.properties.points
            max_distance_base ← max(max_distance_base, abs(base.x -
current_x) + abs(base.y - current_y))
            max_distance_bot ← max(max_distance_bot, abs(bot_position.x -
current_x) + abs(bot_position.y - current_y))
            end

            if ada then
                if (current_x + 1, current_y) not in visited && current_x + 1 < 15
then
                    points += recursive_search(diamonds, current_x + 1, current_y,
base, bot_position, visited)[0]
                if (current_x - 1, current_y) not in visited && current_x - 1 ≥ 0
then
                    points += recursive_search(diamonds, current_x - 1, current_y,
base, bot_position, visited)[0]
                if (current_x, current_y + 1) not in visited && current_y + 1 < 15
then
                    points += recursive_search(diamonds, current_x, current_y + 1,
base, bot_position, visited)[0]
                if (current_x, current_y - 1) not in visited && current_y - 1 ≥ 0
then
                    points += recursive_search(diamonds, current_x, current_y - 1,
base, bot_position, visited)[0]
                if (current_x - 1, current_y - 1) not in visited && current_x - 1 ≥
0 and current_y - 1 ≥ 0 then
                    points += recursive_search(diamonds, current_x - 1, current_y -
1, base, bot_position, visited)[0]
                if (current_x + 1, current_y - 1) not in visited && current_x + 1 <
15 and current_y - 1 ≥ 0 then
                    points += recursive_search(diamonds, current_x + 1, current_y -
1, base, bot_position, visited)[0]
                if (current_x - 1, current_y + 1) not in visited && current_x - 1 ≥
0 and current_y + 1 < 15 then
                    points += recursive_search(diamonds, current_x - 1, current_y +
1, base, bot_position, visited)[0]
                if (current_x + 1, current_y + 1) not in visited && current_x + 1 <
15 and current_y + 1 < 15 then
                    points += recursive_search(diamonds, current_x + 1, current_y +
1, base, bot_position, visited)[0]

            → points, visited

```

Function next_move

```
function next_move(Logic self, GameObject board_bot, Board board)
    if self.move.initiating = False then //jika move belum diinisiasi
        self.move.initialize(board)

    //jika future_move_x atau future_move_y tidak sama dengan 0
    if self.move.future_move_x ≠ 0 || self.move.future_move_y ≠ 0 then
        delta_x ← self.move.future_move_x
        delta_y ← self.move.future_move_y
        self.move.future_move_x ← 0
        self.move.future_move_y ← 0
        return delta_x, delta_y
    else // jika future_move_x dan future_move_y sama dengan 0
        teleport1 ← board.game_objects[self.move.teleport1]
        teleport2 ← board.game_objects[self.move.teleport2]
        restart_button ← board.game_objects[self.move.restart_button]
        visited ← set()
        props ← board_bot.properties
        current_position ← board_bot.position
        base ← props.base
        jarak_base ← abs(base.x - current_position.x) + abs(base.y -
current_position.y)
        jarak_reset ← abs(restart_button.position.x -
current_position.x) + abs(restart_button.position.y -
current_position.y) + abs(restart_button.position.x - base.x) +
abs(restart_button.position.y - base.y)
        if props.diamonds = 5 then // jika diamond yang ditemukan sudah
5
            // jika jarak_reset lebih kecil dari jarak antara bot dan
base
                if jarak_reset ≤ jarak_base + 1 then
                    self.goal_position = restart_button.position
                else // jika jarak_reset lebih besar dari jarak antara bot
dan base
                    self.goal_position = base
            else // jika diamond yang ditemukan belum 5
                diamonds ← board.diamonds
                worth ← 0
                min_jarak ← 999
                calculated_diamonds ← []
                for diamond in diamonds do // mencari diamond yang paling
dekat dengan bot dan base
                    // jika diamond yang ditemukan sudah 4 dan point diamond
yang ditemukan adalah 2
                        if props.diamonds = 4 && diamond.properties.points = 2
then
                            continue
                        if diamond.position in calculated_diamonds then // jika
diamond sudah dihitung
                            continue
                    // jika posisi bot di salah satu sudut papan dan
terdapat teleport di sekitar bot
```

```

        else if (current_position.x, current_position.y,
teleport1.position.x, teleport2.position.y) in [(0, 0, 1, 1), (0, 14, 1,
13), (14, 0, 13, 1), (14, 14, 13, 13)] then
            continue
            point, visited ← self.recursive_search(
                diamonds, diamond.position.x, diamond.position.y,
base, current_position, visited)
            calculated_diamonds.append(diamond.position)
            jarak ← self.max_distance_base + self.max_distance_bot
            if jarak ≠ 0 then
                if point > 5 - props.diamonds then //jika point
lebih besar dari 5 dikurangi diamonds pada inventory
                    point ← 5 - props.diamonds

            cari_terdekat ← False
            value ← point / jarak
            temp ← worth
            worth ← max(worth, value)
            min_jarak ← min(min_jarak, jarak)
            if worth = value then // jika worth sama dengan
value
                self.goal_position ← diamond.position
                // jika waktu yang tersisa kurang dari 30 detik
dan jarak lebih besar dari waktu yang tersisa dibagi 1000
                if props.milliseconds_left < 30000 then
                    if min_jarak == jarak then
                        self.goal_position = diamond.position
                    if jarak > props.milliseconds_left / 1000
and props.diamonds != 0 then
                        if jarak_reset ≤ jarak_base + 4 and
jarak_reset+4 ≤ props.milliseconds_left/1000 then
                            self.goal_position ←
restart_button.position
                        else
                            self.goal_position ← base
                            cari_terdekat ← True
                            // jika jarak_base kurang dari sama dengan jarak
                            if jarak_base ≤ jarak && current_position ≠ base
and props.diamonds ≠ 0 then
                                if (current_position.x > base.x >
diamond.position.x || current_position.y > base.y > diamond.position.y ||
current_position.x < base.x < diamond.position.x || current_position.y <
base.y < diamond.position.y) then
                                    self.goal_position ← base
                                else if jarak_reset ≠ 0 then //jika jarak_reset
tidak sama dengan 0
                                    worth_restart ← 1 / jarak_reset
                                    # jika worth kurang dari worth_restart dan
posisi restart_button tidak sama dengan posisi bot
                                    if worth < worth_restart &&
restart_button.position ≠ current_position then
                                        self.goal_position ←
restart_button.position
                                    self.max_distance_base ← 0
                                    self.max_distance_bot ← 0

```

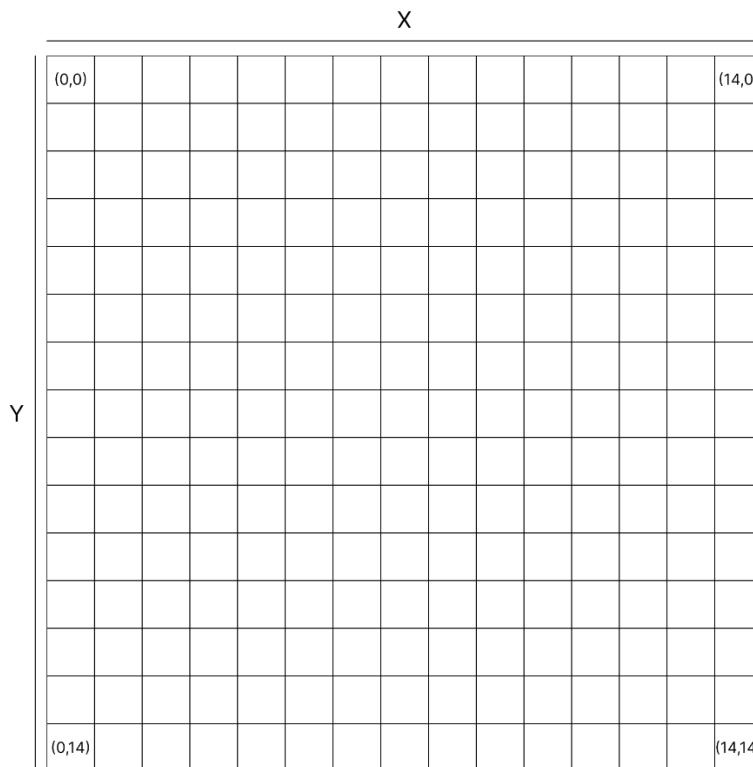
```

        delta_x, delta_y ← self.get_direction(//mencari arah gerak bot
        current_position.x,
        current_position.y,
        teleport1.position.x,
        teleport1.position.y,
        teleport2.position.x,
        teleport2.position.y,
        base,
    )
    → delta_x, delta_y

```

4.2. Struktur Data Program

Dalam bot permainan Diamonds yang dibuat dengan Bahasa python, penting sekali untuk mempelajari struktur data yang terdapat dalam permainan ini. Struktur data yang paling utama dari permainan ini adalah peta permainan Diamonds yang berukuran 15 x 15 petak.



Gambar 4.2.1 Ilustrasi peta permainan beserta koordinat

Ilustrasi peta permainan diamonds bisa dilihat pada gambar 4.2.1 beserta dengan pembagian kordinatnya. Dapat dilihat bahwa posisi koordinat (0,0) berada pada pojok kiri atas dimana posisi x dan juga y sama dengan 0. Saat bot permainan bergerak ke bawah maka nilai y akan bertambah sedangkan saat bot bergerak ke atas maka nilai y akan berkurang. Sekarang kita akan melihat konfigurasi untuk sumbu x papan permainan. Berdasarkan ilustrasi, dapat dilihat bahwa saat bot bergerak ke kanan maka nilai x akan bertambah satu.

Kita akan membahas kelas-kelas yang terdapat dalam program bot diamonds setelah mengetahui konfigurasi terkait papan permainan Diamonds dan juga penempatan posisi x dan y dalam papan permainan. Selama pengembangan bot, kami menerapkan pendekatan pemrograman berorientasi objek dimana variabel dan data kami enkapsulasi ke dalam kelas-kelas. Kami juga menambahkan beberapa *methods* dan juga fungsi tambahan pada kelas bot yang telah kami rancang. Bot engine Diamonds ini terdiri dari dua kelas utama yang akan sangat berguna dalam pengembangan bot karena berisi data-data penting yang akan digunakan sebagai pertimbangan algoritma *greedy* bot permainan yang dirancang. Program ini terdiri dari dua kelas utama yaitu ``GameObject`` dan juga ``Board``.

c) GameObject

Kelas ``GameObject`` ini merupakan kelas yang mayoritas berisi data mengenai informasi bot yang kita miliki. Informasi ini terdiri dari base yang dimiliki oleh bot, jumlah waktu yang tersisa, jumlah diamonds pada inventory bot dan masih banyak lagi. Kami akan menjelaskan struktur terkait dari kelas ``GameObject`` yang mendukung dalam pengembangan program bot.

- Properties

Atribut properties ini memiliki beberapa tipe data dan yang akan kita bahas pada laporan tugas besar ini adalah diamonds, base, inventory_size, milliseconds_left, dan can_tackle.

- diamonds

Diamonds menyimpan informasi berupa jumlah diamonds yang dimiliki bot saat ini pada inventorynya. Diamonds ini memiliki tipe data integer dan memiliki range nilai dari 0 sampai dengan 5.

- base

Base menyimpan informasi terkait posisi base dari bot yang kita kembangkan. Base memiliki tipe data tuple yaitu (integer, integer) dan kedua tuple tersebut memiliki range nilai 0 sampai dengan 14. Informasi ini dapat dikenali sebagai koordinat base dari bot.

- inventory_size

Inventory_size merupakan informasi terkait ukuran inventory dari bot. inventory_size memiliki tipe data integer dengan nilai 5 yang digunakan sebagai default saat berkompetisi dengan bot lain.

- milliseconds_left

milliseconds_left memiliki informasi tentang sisa waktu dari permainan Diamonds. Milliseconds_left memiliki tipe data integer dengan perhitungan satu detik sama dengan seribu milliseconds_left.

- `can_tackle`
`can_tackle` merupakan informasi boolean tentang apakah bot yang kita kembangkan bisa di tackle. `Can_tackle` memberikan kita informasi mengenai entity yang dapat ditackle oleh bot kita.

- **Id**
Atribut ID berisi informasi mengenai dimana indeks bot kita berada pada kelas ``Board`` pada atribut `game_object`.
- **Position**
Atribut position bertipe data tuple yaitu `(integer,integer)` atau bisa lebih dikenal dengan posisi dari bot yang kita kembangkan. Dalam atribut position, kita dapat mengakses nilai x dan juga nilai y dari bot tersebut.

d) Board

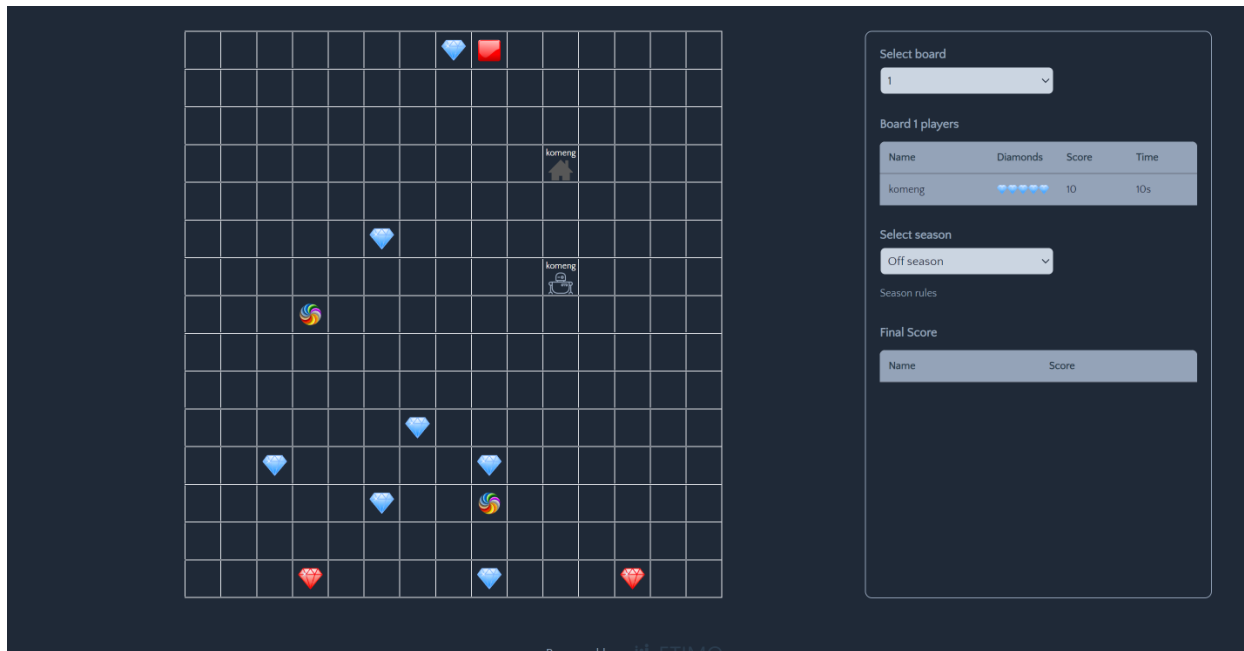
Kelas ``Board`` ini memberikan informasi terkait komponen-komponen yang terdapat dalam papan permainan diamonds. Pada kali ini kami akan membahas dua atribut utama dalam board ini yaitu diamonds dan juga `game_objects`.

- **diamonds**
Atribut diamonds ini merupakan array yang berisi posisi-posisi dari seluruh diamonds yang berada pada papan permainan. Pengaksesan posisi x dan y dari diamond indeks ke 1 dapat dilakukan dengan cara berturut-turut
`board.diamonds[maks_indeks].position.x` dan
`board.diamonds[maks_indeks].position.y`.
- **game_objects**
`game_objects` merupakan atribut yang berisi seluruh komponen yang terdapat dalam papan permainan. Reset button dapat diakses dengan `board.game_objects[2].position` sedangkan untuk kedua teleport button dapat diakses dengan `board.game_objects[0].position` dan `board.game_objects[1].position`.

4.3. Analisis dan Pengujian

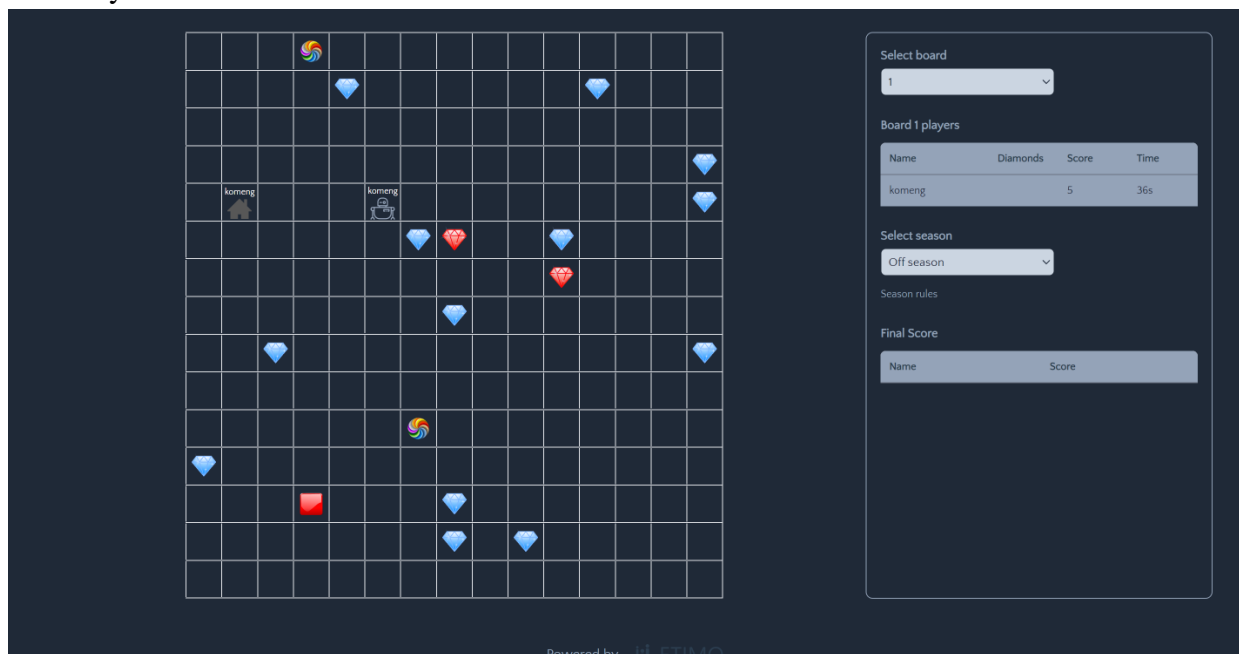
Kami melakukan evaluasi dan pengujian permainan dengan mengintegrasikan bot kami, bersama dengan bot referensi yang telah kami perbarui sedikit. Setelah menjalankan permainan, kami secara cermat memantau perilaku dan semua tindakan yang dilakukan oleh bot kami menggunakan visualisasi yang tersedia.

Berikut adalah analisis yang kami lakukan terhadap suatu pertandingan bot kami:



Gambar 4.3.1 Pengujian strategi algoritma *greedy by base*

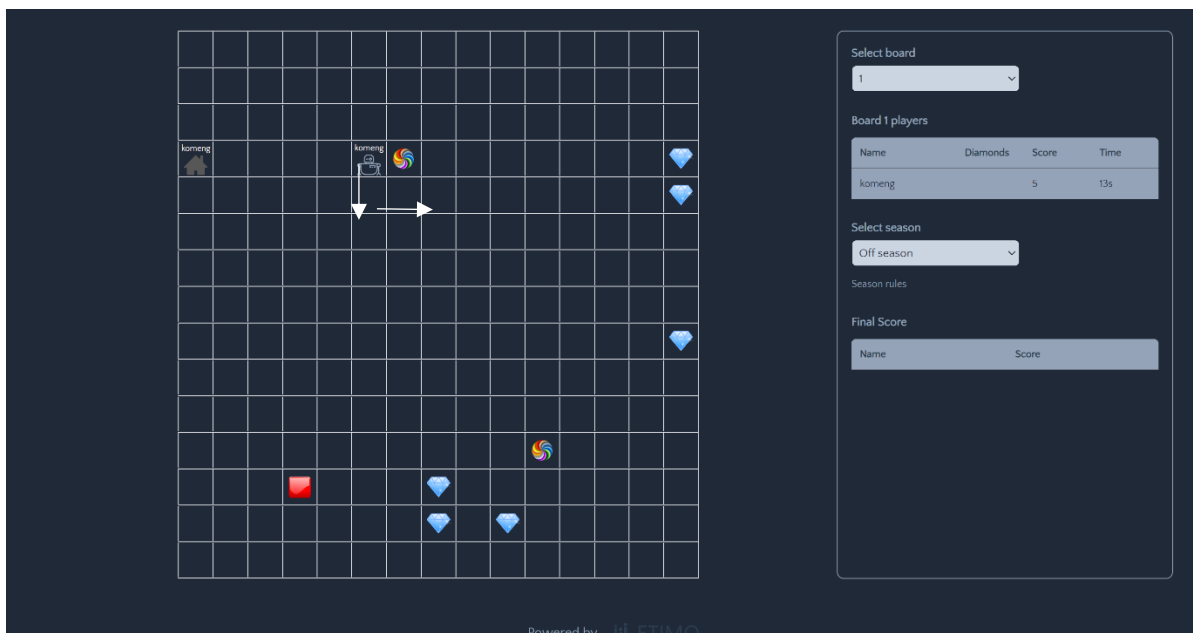
Berdasarkan gambar 4.3.1 dapat dilihat bahwa bot permainan yang telah dikembangkan menggunakan algoritma *greedy* telah bergerak sesuai dengan harapan. Pada gambar 4.3.1 dapat dilihat bahwa bot akan kembali ke base ketika inventory yang dimiliki sudah penuh yaitu saat inventory terisi 5 value dari diamonds.



Gambar 4.3.2 Pengujian strategi algoritma *greedy by diamonds* (diamonds yang saling berdekatan)

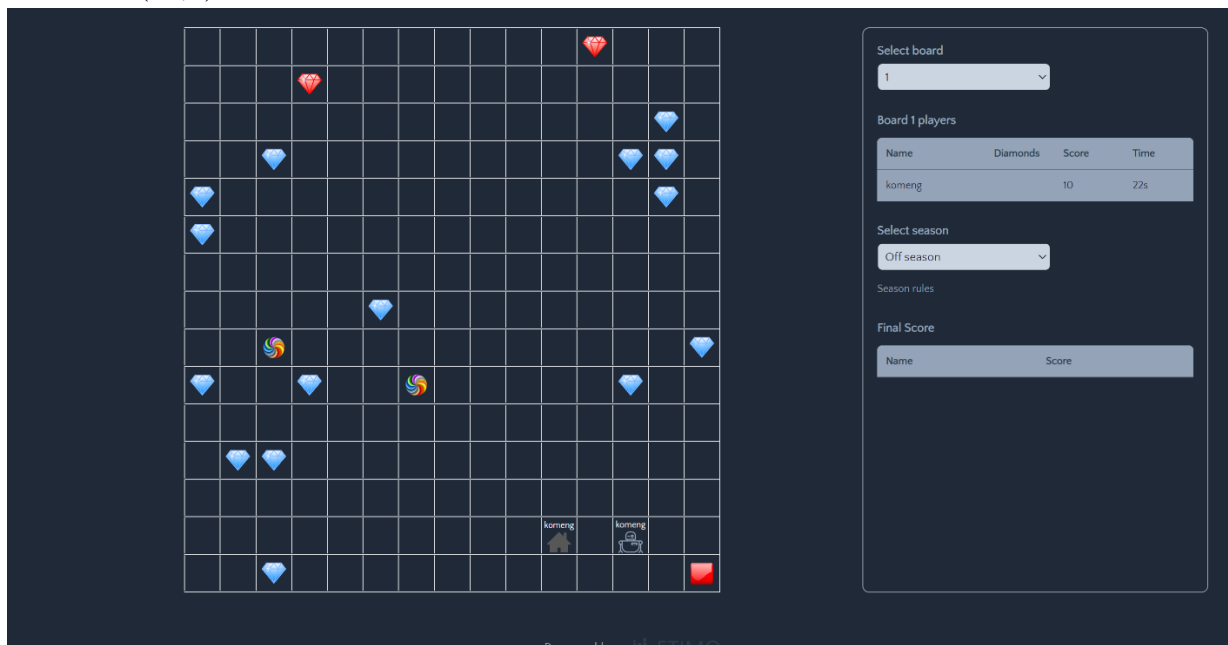
Pada gambar 4.3.2 bot memiliki posisi awal yang berada pada base dan terletak di koordinat (1,4). Bot memiliki dua pilihan diamond terdekat dari base atau tempat dia berada yaitu diamonds pada koordinat (2,8) dan juga (6,5). Gambar 4.3.2 menunjukkan bahwa bot telah mengambil keputusan diamond yang ingin diambil sesuai dengan harapan. Harapan pada uji coba kali ini

adalah bot mengambil pilihan sesuai dengan kepantasan dan nilai dari diamond di sekitarnya berdasarkan perhitungan pada Bab 3.



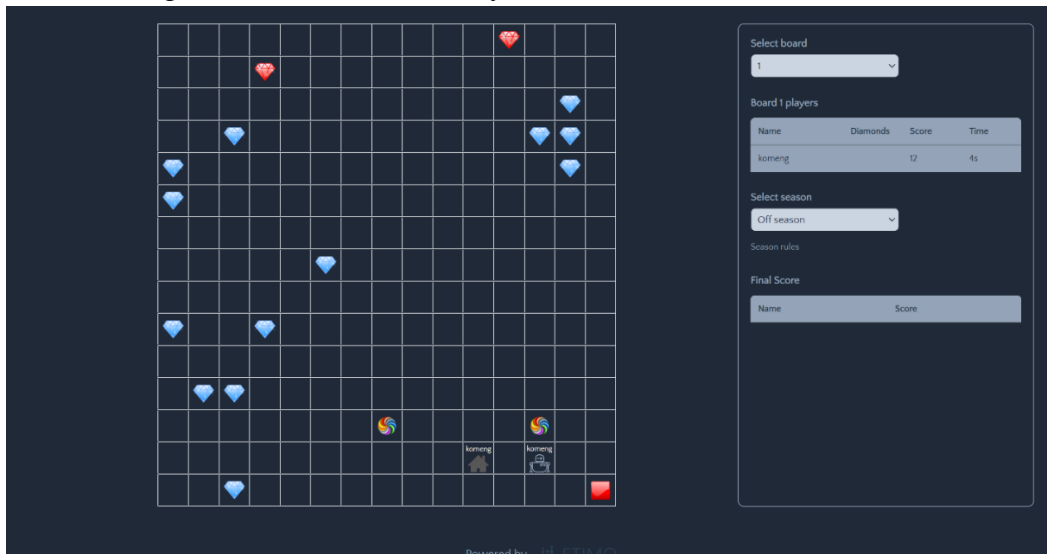
Gambar 4.3.3 Pengujian strategi algoritma *greedy by avoid teleporters*

Bot permainan “Diamonds” awalnya memiliki tujuan posisi pada diamond biru yang terletak pada koordinat (14,3). Namun, karena pada posisi kanan atau $\delta_x = 1$ terdapat teleporter bot permainan merubah arah gerak sesuai dengan tanda panah pada gambar 4.3.3. Perubahan arah ini akan mengakibatkan pengambilan diamonds yang seharusnya tertuju pada koordinat (14,3) menjadi diamonds pada koordinat (14,4) lalu diamonds yang berada pada koordinat (14,3).



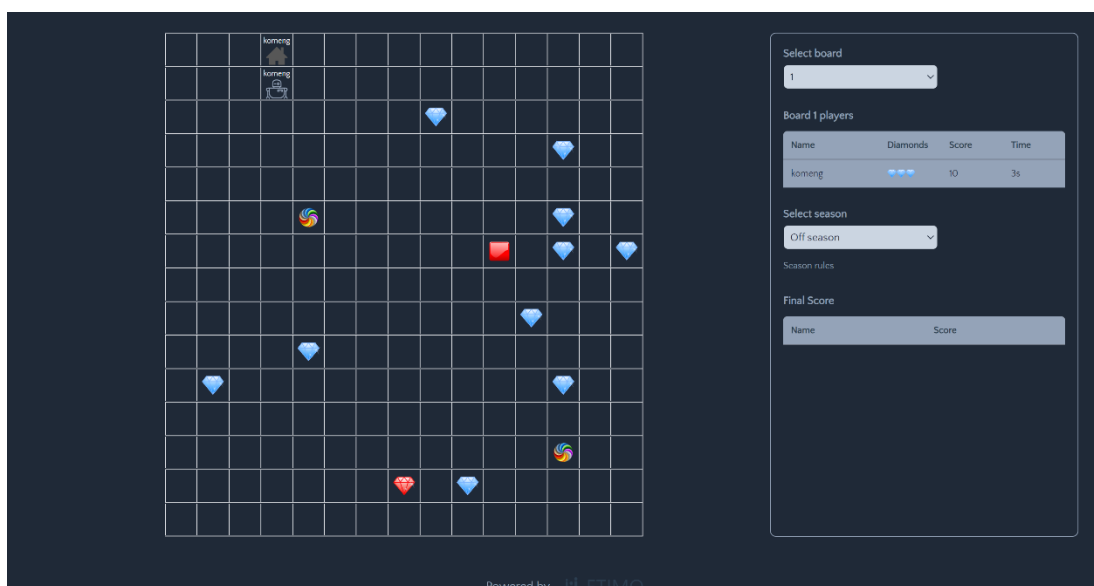
Gambar 4.3.4 Pengujian skala prioritas diamonds dengan *reset button*

Dapat dilihat pada gambar 4.3.4 bahwa bot permainan “Diamonds” memiliki posisi awal pada base atau dengan koordinat (10,13). Pada kasus ini, bot permainan “Diamonds” memiliki 2 pilihan yang paling mendekati keputusan bot yaitu diamonds pada koordinat (12,9) dengan jarak 6 langkah dan juga *reset button* pada koordinat (14,14) dengan jarak 5 langkah. Bot akan mengambil diamonds pada koordinat (14,4) meskipun memiliki jarak yang lebih jauh daripada *reset button* yang disebabkan oleh penentuan skala prioritas *reset button* dengan nilai 0.9 yang lebih kecil dibandingkan nilai diamonds biru yaitu 1.



Gambar 4.3.5 Pengujian skala algoritma *greedy by reset button*

Ketika sudah tidak ada lagi diamonds lain yang lebih dekat dibandingkan dengan *reset button* maka bot akan melakukan pengambilan *reset button*. Pada gambar 4.3.5 bot akan mengambil *reset button* untuk mengulang tata letak atribut serta penambahan atribut sesuai dengan tata letak yang baru.



Gambar 4.3.6 Pengujian skala algoritma *greedy by milliseconds left*

Final Score	
Name	Score
komeng	13

Gambar 4.3.7 Hasil akhir score bot permainan "Diamonds"

Saat waktu yang digunakan untuk mengambil diamonds lain tidak cukup untuk waktu yang tersisa pada permainan maka strategi algoritma *greedy by milliseconds left* akan membuat bot kembali ke base yang berada pada koordinat (0,3). Pada gambar 4.3.6 dapat dilihat bahwa strategi algoritma *greedy by milliseconds left* telah berhasil diterapkan dalam bot permainan "Diamonds". Hasil akhir yang diperoleh dalam permainan ini adalah 13 yang dapat dilihat pada gambar 4.3.7 yang berarti bot sampai pada base tepat waktu dan sebelum permainan berakhir.

BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan laporan tugas besar ini, kami berhasil merealisasikan strategi algoritma *greedy* dalam bot permainan Diamonds dengan objektif utama yaitu mengambil diamond sebanyak-banyaknya. Algoritma *greedy* memiliki semboyan penting yaitu “*take what you can get now*” sehingga penentuan keputusan yang dilakukan oleh bot permainan harus dilakukan pada saat itu dan pada kondisi saat itu juga. Pada algoritma *greedy* kita diharuskan untuk mencari optimum lokal dengan harapan menghasilkan optimum global. Karena algoritma *greedy* menggunakan pendekatan yang sederhana maka sering kali terjadi kasus *counterexample* dari program yang dirancang. Dalam mengatasi *counterexample* dari strategi algoritma *greedy* diperlukan penggabungan beberapa alternatif strategi algoritma *greedy*. Penggabungan strategi algoritma *greedy* bertujuan agar masing-masing dari strategi algoritma *greedy* dapat menutupi *counterexample* satu sama lain. Program akhir yang kami buat dalam repository github terlampir sudah melalui tahap pengujian dan telah menghasilkan strategi terbaik menurut kelompok kami.

5.2. Saran

Terdapat ruang pengembangan yang cukup luas dalam pengembangan bot permainan diamonds seperti melakukan elaborasi strategi algoritma *greedy* lainnya agar efektivitas bot permainan diamonds dapat ditingkatkan. Perlu juga dilakukan pengujian lebih lanjut untuk menemukan *counterexample* lainnya pada program bot diamonds yang telah kami buat. Dapat dilakukan juga pembuktian terkait solusi yang dihasilkan program bot diamonds apakah hampiran dari solusi optimal ataupun solusi yang optimal pada kasus tertentu. Pada panduan penggunaan game engine diamonds kami memiliki saran untuk menambahkan kesalahan – kesalahan umum yang mungkin terjadi saat melakukan konfigurasi. Hal ini dilakukan untuk meminimalisir terjadinya kesalahan pada saat melakukan instalasi terutama saat melakukan konfigurasi docker.

5.3. Refleksi

Beberapa hal yang perlu dipertimbangkan untuk kelompok kami adalah meningkatkan efektivitas strategi algoritma yang telah kami buat dengan tujuan memperoleh strategi yang lebih optimal. Selain itu, perlu juga dilakukan peningkatan dalam pembagian tugas agar waktu yang dialokasikan untuk setiap tugas lebih merata di antara anggota kelompok.

LAMPIRAN

Pranala *Repository* GitHub :

https://github.com/akmalrmn/Tubes1_susugratis

Tautan video penjelasan strategi algoritma *greedy* bot permainan “Diamonds” :

<https://www.youtube.com/watch?v=mLt-mygD1M0>

Uji coba pertandingan bot dengan kelompok lain:

