

LAPORAN TUGAS BESAR I

IF2211 Strategi Algoritma

Pemanfaatan Algoritma *Greedy* dalam Pembuatan Bot Permainan *Diamonds*



Disusun oleh:

“The Omega”

Erdianti Wiga Putri Andini 13522053

Zahira Dina Amalia 13522085

Abdullah Mubarak 13522101

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

Daftar Isi

BAB I.....	4
DESKRIPSI TUGAS.....	4
1.1 Deskripsi Tugas.....	4
1.2 Spesifikasi.....	7
BAB II.....	8
LANDASAN TEORI.....	8
2.1 Dasar Teori Algoritma Greedy secara Umum.....	8
2.2 Elemen-elemen dalam Algoritma Greedy.....	9
2.2.1 Himpunan kandidat (C).....	9
2.2.2 Himpunan solusi (S).....	9
2.2.3 Fungsi solusi.....	9
2.2.4 Fungsi seleksi (selection function).....	9
2.2.5 Fungsi kelayakan (feasible).....	9
2.2.6 Fungsi objektif.....	9
2.3 Game Engine.....	9
2.4 Komponen Program Etimo Diamonds.....	10
2.5 Alur Jalan Program Permainan Etimo Diamonds.....	10
2.6 Cara Menjalankan Bot dan Implementasi Algoritma Greedy pada Bot.....	11
2.7 Mekanisme Teknis Permainan Etimo Diamonds.....	11
BAB III.....	13
APLIKASI STRATEGI GREEDY.....	13
3. 1 Mapping.....	13
3. 2 Eksplorasi Alternatif Solusi Greedy.....	13
3.2.1 Greedy by Distance.....	13
3.2.2 Greedy by Diamond Type.....	15
3.2.3 Greedy by Attack.....	16

3.2.4 Greedy by Avoidant.....	17
3. 3. Strategi Greedy yang Dipilih.....	19
BAB IV.....	21
IMPLEMENTASI DAN PENGUJIAN.....	21
4.1 Implementasi Algoritma Greedy.....	21
4.1.1 Prosedur __init__.....	21
4.1.2 Fungsi getDistance.....	21
4.1.3 Fungsi isInDomain.....	21
4.1.4 Fungsi getPortal.....	21
4.1.5 Fungsi logicPortal.....	22
4.1.6 Fungsi logicDiamond.....	22
4.1.7 Prosedur greedyNearestDiamond.....	23
4.1.8 Prosedur logicBasePortal.....	24
4.1.9 Prosedur logicRedButtonPortal.....	24
4.1.10 Prosedur greedyDiamondPortal.....	25
4.1.11 Fungsi diamondInArea.....	27
4.1.12 Prosedur next_move.....	27
4.2 Penjelasan Struktur Data.....	29
4.3 Analisis dan Pengujian.....	30
BAB V.....	36
KESIMPULAN DAN SARAN.....	36
5. 1 Kesimpulan.....	36
5.2 Saran.....	36
LAMPIRAN.....	37
DAFTAR PUSTAKA.....	38

BAB I

DESKRIPSI TUGAS

1.1 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. Penjelasan lebih lanjut mengenai aturan permainan akan dijelaskan di bawah.

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 : [Game Engine Git](#)
- Bot starter pack : [Bot Starter Pack Git](#)

Komponen-komponen dari permainan Diamonds antara lain:



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.



Red Button/Diamond Button, ketika 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.



Teleporters, terdapat 2 teleporter yang saling terhubung satu sama lain. Jika bot melewati sebuah teleporter maka bot akan berpindah menuju posisi teleporter yang lain.



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.

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.

Berikut ini adalah cara kerja permainan Diamonds.

1. 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 menimpa 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.

1.2 Spesifikasi

Pada tugas ini kami diminta untuk membuat program sederhana dalam bahasa Python yang mengimplementasikan algoritma Greedy pada bot permainan Diamonds dengan tujuan memenangkan permainan. Program perlu dibuat dengan spesifikasi sebagai berikut.

- 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.
- Bisa memmbuat 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.

BAB II

LANDASAN TEORI

2.1 Dasar Teori Algoritma Greedy secara Umum

Algoritma greedy adalah metode yang sederhana namun populer dalam menyelesaikan masalah optimasi. Algoritma ini beroperasi dengan memilih pilihan terbaik pada setiap langkahnya tanpa mempertimbangkan implikasi di masa depan, dengan keyakinan bahwa memilih opsi terbaik di setiap langkah akan membawa kita menuju solusi optimal secara keseluruhan. Persoalan optimasi yang diselesaikan oleh algoritma greedy dapat dibagi menjadi dua jenis: maksimasi dan minimasi. Algoritma ini didasarkan pada pengambilan keputusan lokal yang optimal pada setiap langkahnya, tanpa memperhatikan kemungkinan konsekuensi di langkah-langkah berikutnya. Sebagai contoh, dalam persoalan penukaran uang, algoritma greedy akan memilih koin dengan nilai tertinggi yang tersedia pada setiap iterasi.

Algoritma greedy melibatkan beberapa elemen penting, antara lain himpunan kandidat (kumpulan pilihan yang tersedia untuk dipilih pada setiap langkah), himpunan solusi (kumpulan pilihan yang sudah dipilih), fungsi solusi (yang memeriksa apakah himpunan kandidat memberikan solusi yang diinginkan), fungsi seleksi (memilih kandidat berdasarkan strategi greedy), fungsi kelayakan (memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi), dan fungsi objektif (memaksimalkan atau meminimalkan nilai yang ingin dicapai).

Meskipun algoritma greedy sering kali dapat memberikan solusi hampiran yang baik dalam waktu komputasi yang relatif singkat dibandingkan dengan algoritma yang eksak, membuktikan optimalitasnya secara matematis bisa menjadi tantangan. Terkadang, algoritma ini dapat menghasilkan solusi yang tidak optimal tergantung pada sifat dari masalah yang diselesaikan. Meskipun demikian, algoritma greedy tetap menjadi alat yang berguna dalam menyelesaikan masalah optimasi dengan cepat, meskipun dengan risiko tidak mendapatkan solusi optimal dalam beberapa kasus.

Secara keseluruhan, algoritma greedy merupakan alat yang sangat berguna dalam menyelesaikan berbagai masalah optimasi. Meskipun tidak selalu menghasilkan solusi

optimal, algoritma ini seringkali memberikan solusi yang baik dengan kecepatan yang optimal pula. Karena itu, algoritma greedy menjadi sangat penting bagi para pemrogram dalam menciptakan solusi yang efektif dan efisien. Dalam konteks pembuatan program bot dalam permainan, penggunaan algoritma greedy membantu dalam mendapatkan aksi bot yang paling efektif dan efisien.

2.2 Elemen-elemen dalam Algoritma Greedy

Algoritma greedy terdiri dari beberapa komponen, antara lain:

2.2.1 Himpunan kandidat (C)

Himpunan ini berisi kandidat yang akan dipilih pada setiap Langkah (misal: simpul/sisi di dalam graf, job, task, koin, benda, karakter, dsb).

2.2.2 Himpunan solusi (S)

Himpunan yang berisi kandidat yang sudah dipilih.

2.2.3 Fungsi solusi

Fungsi ini akan menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi.

2.2.4 Fungsi seleksi (*selection function*)

Fungsi yang akan memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik.

2.2.5 Fungsi kelayakan (*feasible*)

Fungsi akan memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak).

2.2.6 Fungsi objektif

Fungsi ini menjelaskan apakah untuk memaksimumkan atau meminimumkan.

2.3 Game Engine

Game Engine adalah sebuah sistem perangkat lunak yang berperan dalam mengembangkan dan menciptakan video game. Sebagai sebuah library, game engine menyediakan fungsi-fungsi inti yang mempermudah pengembangan dalam proses

pembuatan game. Penggunaan game engine tidak hanya terbatas untuk satu permainan saja, namun pengembang permainan dapat membuat lebih dari satu permainan dengan mengoptimalkan proses pengembangan melalui adaptasi atau pemanfaatan game engine yang telah ada sebelumnya. Game engine merupakan sebuah Software Development Environment (SDE) yang membantu dalam pembuatan video game. Terdiri dari berbagai komponen perangkat keras dan lunak seperti 2D dan 3D graphics rendering, animasi, Artificial Intelligence (AI), manajemen memori, audio, dan lain-lain, game engine memfasilitasi seluruh proses desain dan pengembangan video game. Dengan menggunakan game engine sebagai pondasi, pengembang dapat mengurangi biaya pengembangan video game karena tidak perlu membuat semua komponen dari nol. Selain itu, satu game engine dapat digunakan untuk mengembangkan berbagai jenis video game tanpa memerlukan banyak perubahan.

2.4 Komponen Program Etimo Diamonds

Program ini terdiri atas dua komponen yaitu *game engine* dan *bot starter pack*. *Game Engine* permainan Etimo Diamonds ini bersumber dari tautan repository <https://github.com/haziqam/tubes1-IF2211-game-engine/releases/tag/v1.1.0>. Sedangkan *Bot Starter Pack* untuk program ini bersumber dari tautan repository <https://github.com/haziqam/tubes1-IF2211-bot-starter-pack/releases/tag/v1.0.1>.

2.5 Alur Jalan Program Permainan Etimo Diamonds

Untuk menjalankan permainan bot ini, diperlukan beberapa tools yang perlu diinstal terlebih dahulu sebelum menjalankan program yaitu Node.js untuk *frontend website*, Docker Desktop sebagai *framework backend*, dan Yarn untuk menginstal semua dependensi yang diperlukan. Setelah dependensi terinstal, perlu mengatur variabel lingkungan default dengan menjalankan script yang sesuai dengan sistem operasi komputer. Untuk Windows, perlu menjalankan script '`copy-env.bat`', dan untuk Linux atau macOS, perlu menjalankan script '`copy-env.sh`'. Selanjutnya, perlu disiapkan database lokal dengan menjalankan Docker Compose menggunakan perintah '`docker compose up -d database`', dan kemudian menjalankan script '`setup-db-prisma`' sesuai dengan sistem operasi komputer.

Setelah semua konfigurasi awal selesai, pastikan aplikasi Docker Desktop sudah berjalan sebelum menjalankan *game engine*. Selanjutnya kita dapat melakukan *build* dengan menjalankan perintah '`npm run build`', dan kemudian menjalankan *game engine* dengan

perintah `npm run start`. Jika berhasil, kita akan melihat tampilan terminal sesuai dengan yang diharapkan, dan dapat mengunjungi frontend melalui <http://localhost:8082/>.

2.6 Cara Menjalankan Bot dan Implementasi Algoritma Greedy pada Bot

Untuk menjalankan bot, pastikan komputer telah terinstall dengan Python dan memiliki semua persyaratan yang tercantum. Setelah mengunduh source code bot starter pack dan mengekstraknya, buka terminal dan navigasikan ke direktori root proyek bot. Instal semua dependensi yang diperlukan dengan menjalankan `pip install -r requirements.txt`. Untuk menjalankan bot, pastikan komputer telah terinstall Python dan persyaratan yang dibutuhkan. Instal semua dependensi yang diperlukan dan jalankan bot dengan menggunakan perintah `python main.py` dengan menentukan logic bot, email, nama, password, dan tim yang sesuai. Untuk menjalankan beberapa bot, gunakan skrip `run-bots.bat` untuk Windows atau `run-bots.sh` untuk Linux atau macOS.

Kita juga dapat mengimplementasikan bot baru dengan membuat kelas yang menginherit kelas `BaseLogic` dan mengimplementasikan constructor dan method `next_move`. Import kelas yang dibuat ke `main.py` dan daftarkan pada dictionary `CONTROLLERS`. Setelah itu, jalankan program seperti yang dijelaskan sebelumnya dengan argumen logic yang sesuai. Langkah-langkah tersebut dirancang untuk memastikan bahwa *game engine* dan bot berjalan dengan baik tanpa masalah dan dapat dipertimbangkan untuk digunakan dalam kompetisi.

2.7 Mekanisme Teknis Permainan Etimo Diamonds

Permainan ini merupakan permainan berbasis web, sehingga setiap aksi yang dilakukan – mulai dari mendaftarkan bot hingga menjalankan aksi bot – akan memerlukan HTTP request terhadap API endpoint tertentu yang disediakan oleh *backend*. Berikut adalah urutan requests yang terjadi dari awal mula permainan.

1. Program bot akan mengecek apakah bot sudah terdaftar atau belum, dengan mengirimkan POST request terhadap endpoint `/api/bots/recover` dengan body berisi email dan password bot. Jika bot sudah terdaftar, maka backend akan memberikan response code 200 dengan body berisi id dari bot tersebut. Jika tidak, backend akan memberikan response code 404.
2. Jika bot belum terdaftar, maka program bot akan mengirimkan POST request terhadap endpoint `/api/bots` dengan body berisi email, name, password, dan team. Jika berhasil,

maka backend akan memberikan response code 200 dengan body berisi id dari bot tersebut.

3. Ketika id bot sudah diketahui, bot dapat bergabung ke board dengan mengirimkan POST request terhadap endpoint `/api/bots/{id}/join` dengan body berisi board id yang diinginkan (preferredBoardId). Apabila bot berhasil bergabung, maka backend akan memberikan response code 200 dengan body berisi informasi dari board.
4. Program bot akan mengkalkulasikan move selanjutnya secara berkala berdasarkan kondisi board yang diketahui, dan mengirimkan POST request terhadap endpoint `/api/bots/{id}/move` dengan body berisi direction yang akan ditempuh selanjutnya (“NORTH”, “SOUTH”, “EAST”, atau “WEST”). Apabila berhasil, maka backend akan memberikan response code 200 dengan body berisi kondisi board setelah move tersebut. Langkah ini dilakukan terus-menerus hingga waktu bot habis. Jika waktu bot habis, bot secara otomatis akan dikeluarkan dari board.
5. Program frontend secara periodik juga akan mengirimkan GET request terhadap endpoint `/api/boards/{id}` untuk mendapatkan kondisi board terbaru, sehingga tampilan board pada frontend akan selalu ter-update

BAB III

APLIKASI STRATEGI GREEDY

3. 1 Mapping

Proses mapping dalam persoalan diamonds menjadi elemen-elemen Algoritma Greedy meliputi himpunan kandidat, himpunan solusi, fungsi solusi, fungsi seleksi, fungsi kelayakan, dan fungsi objektif. Penjelasan lebih lanjut mengenai proses ini tertera pada subbab 3.2.

3. 2 Eksplorasi Alternatif Solusi Greedy

Dalam pengujian permainan ini, kami mencoba beberapa jenis algoritma greedy. Algoritma-algoritma tersebut ada yang berdasarkan jarak, jenis diamond, kemungkinan menyerang, ataupun kemungkinan menghindar.

3.2.1 Greedy by Distance

Greedy by distance adalah pendekatan algoritma greedy dengan memprioritaskan jarak bot dengan suatu objek lain, seperti diamond, portal, dan base. Jarak ini meliputi $x + y$ (total langkah bot). Konsep “greedy by distance” dalam kode ini mencari kemungkinan jarak terdekat terhadap suatu objek dengan kondisi tertentu. Contohnya bila jarak bot dengan diamond adalah yang terdekat, maka *goal position* bot adalah diamond tersebut. Contoh lainnya adalah ketika jarak bot ke portal awal ditambah dengan jarak portal akhir ke diamond adalah yang terdekat, maka *goal position* bot adalah portal awal. Selanjutnya, apabila jarak bot ke base adalah yang terdekat maka *goal position* bot adalah base. Begitu pula ketika jarak bot dengan *red button* lebih dekat maka *goal position* bot adalah *red button*.

a. Mapping Elemen Greedy

- 1) Himpunan kandidat: Semua aksi yang tersedia.
- 2) Himpunan solusi: Aksi yang terpilih.
- 3) Fungsi solusi: Memeriksa apakah aksi yang dipilih menghasilkan jarak terkecil.
- 4) Fungsi seleksi: Pilih aksi yang menghasilkan jarak terkecil.

- 5) Fungsi kelayakan: Memeriksa apakah aksi yang dipilih memenuhi kriteria jarak terkecil antara bot dan objek yang menjadi prioritas, seperti diamond, portal, base, atau red button.
- 6) Fungsi objektif: Mencari output jarak yang minimum.

b. Analisis Efisiensi Solusi

Dalam pengaplikasian greedy by distance, untuk mencari target objek dengan jarak terdekat, diperlukan kompleksitas sorting $O(n)$ dengan n adalah banyaknya objek yang terdapat pada board. Lalu untuk pencarian arah gerak bot yang optimal menggunakan algoritma sorting dengan kompleksitas $O(n)$ dengan n adalah banyaknya kandidat arah gerak bot. Sehingga, kompleksitas algoritma strategi ini adalah $O(n)$.

c. Analisis Efektivitas Solusi

Greedy by distance berasal dari adanya elemen-elemen pada papan yang dapat membantu bot memenangkan permainan. Namun pendekatan algoritma ini bukanlah solusi yang paling efektif dikarenakan terdapat faktor-faktor seperti jenis diamond, portal yang digunakan, dan kemungkinan kandidat arah gerak yang dapat mempengaruhi bot untuk bergerak dengan arah yang tidak diharapkan.

Strategi ini efektif jika :

- Terdapat banyak diamond tersebar dengan posisi yang berdekatan dengan bot
- Terdapat portal yang terletak berdekatan dengan bot yang bisa memperpendek jarak antara bot dengan diamond yang dekat dengan portal
- Tidak terjadi perubahan tiba-tiba terhadap papan terutama saat setelah bot sampai pada ujung portal lain

Strategi ini tidak efektif jika :

- Posisi diamond merah dan biru dengan jarak yang sama atau pada saat diamond merah berjarak kurang dari dua kali jarak bot dengan diamond biru
- *Red button* ditekan
- Bot musuh menggunakan strategi serang
- Jarak semua objek jauh dari bot

3.2.2 Greedy by Diamond Type

Greedy by diamond type adalah pendekatan algoritma greedy dengan memprioritaskan diamond yang berwarna merah dibandingkan dengan diamond yang berwarna biru. Konsep “greedy by diamond type” dalam kode ini mencari kemungkinan bot untuk mencapai diamond merah terlebih dahulu dibandingkan diamond biru. Walaupun jarak bot ke diamond merah lebih jauh daripada diamond biru, bot akan tetap mengejar diamond merah tersebut dibandingkan diamond biru yang dekat.

a. Mapping Elemen Greedy

- 1) Himpunan kandidat: Semua aksi yang tersedia.
- 2) Himpunan solusi: Aksi yang terpilih.
- 3) Fungsi solusi: Memeriksa apakah aksi yang dipilih mengutamakan diamond merah (dengan poin 2) daripada diamond biru (dengan poin 1).
- 4) Fungsi seleksi: Pilih aksi yang mengutamakan diamond merah (dengan poin 2) daripada diamond biru (dengan poin 1).
- 5) Fungsi kelayakan: Memeriksa apakah aksi yang dipilih memenuhi kriteria prioritas diamond merah daripada diamond biru.
- 6) Fungsi objektif: Mencari output aksi yang mengutamakan *goal position* diamond merah daripada diamond biru.

b. Analisis Efisiensi Solusi

Dalam greedy by diamond type, target posisi atau *goal position* ialah diamond dengan poin terbesar, diperlukan kompleksitas sorting $O(n)$ dengan n adalah banyaknya diamond yang terdapat pada board. Lalu untuk pencarian arah gerak bot yang optimal menggunakan algoritma sorting dengan kompleksitas $O(n)$ dengan n adalah banyaknya kandidat arah gerak bot. Sehingga, kompleksitas algoritma strategi ini adalah $O(n)$.

c. Analisis Efektivitas Solusi

Greedy by diamond type berasal dari penetapan poin pada diamond biru dan merah. Diamond merah memiliki poin dua sedangkan diamond biru memiliki poin satu. Tentu semakin banyak poin yang dikumpulkan, semakin besar pula

kemungkinan bot menang. Akan tetapi, pendekatan algoritma ini bukan solusi yang paling efektif dikarenakan pada solusi ini tidak memperhatikan jarak antara bot dengan diamond sehingga dapat memilih langkah yang kurang efektif.

Strategi ini efektif jika :

- Terdapat banyak diamond tersebar dengan posisi yang berdekatan dengan bot
- Terdapat diamond merah yang jaraknya lebih dekat dengan bot dibandingkan jarak diamond biru dengan bot
- Tidak terjadi perubahan tiba-tiba terhadap papan terutama saat setelah bot sampai pada ujung portal lain

Strategi ini tidak efektif jika :

- Posisi diamond merah dan biru dengan jarak yang sama atau pada saat diamond merah berjarak lebih dari dua kali jarak bot dengan diamond biru
- *Red button* ditekan
- Bot musuh menggunakan strategi serang

3.2.3 Greedy by Attack

Greedy by attack dilakukan dengan tanpa mempertimbangkan jarak diamond dan jenisnya. Pada greedy ini, bot memiliki otak menyerang bot lain saja sehingga *goal position* bot ini tergantung pada bot musuh terdekatnya. Dengan kata lain, greedy ini fokus untuk mengejar musuh dengan tujuan mencuri diamond yang dibawa oleh musuh. Hanya saja, pendekatan ini bukan solusi yang paling efektif. Bot lain menjadi faktor utama dalam kemenangan bot yang dapat menyebabkan bot tidak menghasilkan hasil yang paling optimal (tidak memperbesar kesempatan menang bot).

a. Mapping Elemen Greedy

- 1) Himpunan kandidat: Semua aksi yang tersedia.
- 2) Himpunan solusi: Aksi yang terpilih.
- 3) Fungsi solusi: Memeriksa apakah aksi yang dipilih membuat bot mengejar bot musuh terdekat.
- 4) Fungsi seleksi: Pilih aksi yang akan menyerang bot lain dengan jarak antar bot paling kecil.
- 5) Fungsi kelayakan: Memeriksa apakah aksi yang dipilih memenuhi persyaratan dasar untuk mengejar dan menyerang bot musuh terdekat.

6) Fungsi objektif: Mencari output aksi yang mencari jarak dengan bot musuh yang paling minimum.

b. Analisis Efisiensi Solusi

Dalam greedy by attack, *goal position* bot ialah bot musuh dengan jarak terdekat. Diperlukan kompleksitas sorting $O(n)$ dengan n adalah banyaknya bot musuh yang terdapat pada board. Lalu untuk pencarian arah gerak bot yang optimal menggunakan algoritma sorting dengan kompleksitas $O(n)$ dengan n adalah banyaknya kandidat arah gerak bot. Sehingga, kompleksitas algoritma strategi ini adalah $O(n)$.

c. Analisis Efektivitas Solusi

Greedy by attack berasal dari keberadaan bot-bot musuh dan adanya fitur mencuri diamond musuh akibat hasil tabrakan antarbot. Hasil pencurian diamond milik bot musuh menjadi satu-satunya sumber bot untuk mendapatkan diamond. Semakin banyak poin yang dikumpulkan, semakin besar pula kemungkinan bot menang. Akan tetapi, pendekatan algoritma ini bukan solusi yang paling efektif dikarenakan pada solusi ini tidak berfokus untuk mengumpulkan diamond yang ada pada board melainkan hanya bergantung pada fitur mencuri diamond yang dimiliki oleh bot musuh.

Strategi ini efektif jika :

- Bot-bot musuh berada di sekitar bot
- Bot musuh sedang membawa lebih dari 0 diamond

Strategi ini tidak efektif jika :

- Posisi bot musuh berada jauh dari bot
- Saat banyak diamond di sekitarnya namun bot fokus untuk mengejar bot musuh
- Bot musuh sedang tidak membawa diamond
- Bot musuh menggunakan strategi serang

3.2.4 Greedy by Avoidant

Konsep "greedy by avoidant" dalam kode ini menggambarkan strategi bot untuk memilih berlian dengan hati-hati, mengutamakan berlian merah yang memiliki nilai

poin lebih tinggi, sambil tetap menghindari risiko dan interaksi dengan bot lain. Bot mencoba mengoptimalkan skornya dengan mengambil berlian terbaik yang tersedia, namun juga berusaha untuk tidak terlalu berisiko atau terjebak dalam situasi yang berbahaya. Strategi ini bertujuan untuk memungkinkan bot tetap dalam posisi menguntungkan dalam permainan sambil memaksimalkan skor dan mengurangi risiko yang tidak perlu.

a. Mapping Elemen Greedy

- 1) Himpunan kandidat : Semua aksi yang tersedia.
- 2) Himpunan solusi : Aksi yang terpilih.
- 3) Fungsi solusi : Memeriksa apakah aksi yang dipilih memberi langkah yang dapat membuat bot mengambil diamond tanpa gangguan bot lain
- 4) Fungsi seleksi : Pilih aksi yang memberikan langkah yang aman dari gangguan bot lain
- 5) Fungsi kelayakan : Memeriksa apakah aksi yang dipilih memberikan langkah yang aman bagi bot untuk mengambil diamond tanpa gangguan dari bot lain.
- 6) Fungsi objektif : Mencari output dengan langkah selanjutnya merupakan langkah yang tidak diganggu bot lain

b. Analisis Efisiensi Solusi

Dalam greedy by avoidant, *goal position* bot ialah bot musuh dengan jarak terdekat. Diperlukan kompleksitas sorting $O(n)$ dengan n adalah banyaknya bot musuh yang terdapat pada board. Lalu untuk pencarian arah gerak bot yang optimal menggunakan algoritma sorting dengan kompleksitas $O(n)$ dengan n adalah banyaknya kandidat arah gerak bot. Sehingga, kompleksitas algoritma strategi ini adalah $O(n)$.

c. Analisis Efektivitas Solusi

Greedy by avoidant berasal dari keberadaan bot-bot musuh dan adanya fitur mencuri diamond musuh akibat hasil tabrakan antar bot. Pencurian dari musuh dapat merugikan bot dalam tujuan mengumpulkan diamond sebanyak-banyaknya. Akan tetapi, pendekatan algoritma ini bukan solusi yang paling efektif dikarenakan solusi ini tidak berfokus utama dalam mengumpulkan diamond yang ada pada

board melainkan berfokus apakah langkahnya menuju diamond sekitarnya aman dan jauh dari kemungkinan penyerangan dari bot lain.

Strategi ini efektif jika :

- Bot-bot musuh berada di sekitar bot tetapi tidak memiliki langkah yang sama dengan bot
- Bot membawa diamond lebih dari tiga.

Strategi ini tidak efektif jika :

- Posisi-posisi bot musuh berada dekat dari bot
- Saat banyak diamond di sekitarnya namun bot fokus untuk menghindar bot musuh
- Bot musuh menggunakan strategi serang

3. 3. Strategi Greedy yang Dipilih

Berdasarkan solusi-solusi yang telah dipaparkan pada subbab 3.2, terdapat kelebihan dan kekurangan dari masing-masing solusi dikarenakan pada setiap round, game memiliki kondisi yang tidak menentu. Maka dari itu, kami memilih algoritma yang dapat membuat bot mempertimbangkan pendekatan greedy yang sesuai dengan kondisi aktual saat itu dan menentukan aksi apa yang akan dipilihnya.

Kami memutuskan untuk menggunakan gabungan algoritma *greedy by distance* dan *greedy by diamond type* pada program bot kami dengan alasan bahwa kedua algoritma ini yang paling efektif dan efisien jika digabung. Urutan prioritas dalam alur pemrograman bot kami adalah sebagai berikut:

- 1) Kembali ke base ketika jarak bot ke base lebih dekat daripada jarak bot ke diamond dan jumlah diamond yang dia bawa lebih dari sama dengan 3
- 2) Mengincar *red button* saat di sekitar bot tidak ada diamond (diamond berada jauh dari bot) dan jarak *red button* ke bot lebih dekat
- 3) Mencari diamond biru dengan jarak terdekat (baik melalui portal atau tidak) dari bot
- 4) Mencari diamond biru dengan jarak terdekat (baik melalui portal atau tidak) dari bot

Poin-poin tersebut dibuat berdasarkan solusi-solusi algoritma greedy pada subbab 3.2. Poin 1 dan 2 adalah kombinasi dari *greedy by distance* dan *greedy by diamond type*. Poin 4 dan 5 merepresentasikan *greedy by distance*. Sedangkan poin 7 merupakan alternatif agar

bot menyimpan diamond terakhir yang dia bawa sebelum waktu permainan habis dengan masih merepresentasikan *greedy by distance*.

Prioritas tertinggi yang akan dipertimbangkan oleh bot adalah mengambil diamond dengan jarak terdekat (entah itu biru ataupun merah). Namun pada tahap ini juga ada kondisi untuk memprioritaskan diamond merah daripada biru. Prioritas kedua masih mengenai jarak bot dengan objek lain. Pada prioritas ini, bot akan menghitung total jarak terdekat dari bot ke portal awal ditambah dengan jarak terdekat portal akhir ke diamond. Sama seperti prioritas sebelumnya, tahap ini juga ada kondisi untuk memprioritaskan diamond merah daripada biru.

Kedua prioritas di atas memanfaatkan kombinasi antara *greedy by distance* dan *greedy by diamond type*. Namun prioritas selanjutnya hanya memanfaatkan *greedy by distance*. Bot akan membandingkan jarak antara dia ke base dengan jarak antara dia ke diamond. Apabila jarak bot ke base lebih dekat, maka bot akan kembali ke base untuk menyimpan diamond yang sedang ia bawa dengan syarat jumlah diamond yang dibawa lebih dari sama dengan 3.

Prioritas selanjutnya juga masih memanfaatkan *greedy by distance*. Bot akan mempertimbangkan jarak antara dia ke diamond dengan jarak antara dia ke *red button*. Apabila jarak bot ke *red button* lebih dekat daripada ke diamond, maka bot akan mengincar *red button* untuk men-*shuffle* objek-objek pada board sehingga ada kemungkinan ia akan berada di area yang dekat dengan diamond.

Prioritas terakhir adalah saat sisa waktu permainan - 1 kurang dari sama dengan jarak bot ke base. Pada kondisi ini, bot akan diarahkan untuk kembali ke base dalam rangka menyimpan diamond terakhir yang ia bawa. Tujuannya agar diamond yang telah dikumpulkan ini tidak sia-sia. Aksi yang menjadi default adalah poin 1 dan 2 karena apabila tidak ada kondisi-kondisi spesial lainnya, bot akan terus mencari diamond yang terdekat.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma Greedy

Pada kelas OmegaLogic

4.1.1 Prosedur `__init__`

Prosedur untuk menginisialisasi seluruh atribut

```
BEGIN
    Function __init__():
        self.goal_position ← None
        self.portal ← None, None
        self.red_diamonds ← []
        self.blue_diamonds ← []
        self.distance_to_Base ← 0
        self.distance_to_redButton ← 0
        self.redButton_position ← []
END
```

4.1.2 Fungsi `getDistance`

Fungsi ini mengembalikan jarak antara dua posisi, baik itu bot maupun objek

```
BEGIN
    Function getDistance(A: Position, B: Position):
        Return (abs(B.x - A.x) + abs(B.y - A.y))
END
```

4.1.3 Fungsi `isInDomain`

Fungsi ini mengembalikan boolean untuk mengecek apakah suatu objek berada pada domain X dan Y

```
BEGIN
    Function isInDomain(X: Position, Y: Position, A: Position):
        Return (X.x ≥ A.x ≥ Y.x or X.x ≤ A.x ≤ Y.x) and (X.y ≥ A.y ≥ Y.y or X.y ≤ A.y ≤ Y.y) and (X.x ≠ A.x and X.y ≠ A.y) and (Y.x ≠ A.x and Y.y ≠ A.y)
END
```

4.1.4 Fungsi `getPortal`

Fungsi ini bertujuan untuk mendapatkan posisi portal awal yang terdekat dari bot dan posisi portal kedua yang terjauh dari bot

```

BEGIN
    Function getPortal(board_bot: GameObject, board: Board):
        portal_positions ← [d.position for d in board.game_objects
if d.type = 'TeleportGameObject']
        bot_position ← board_bot.position
        if (self.getDistance(portal_positions[0], bot_position) <
self.getDistance(portal_positions[1], bot_position)) then
            nearest_portal ← portal_positions[0]
            second_portal ← portal_positions[1]
        else
            nearest_portal ← portal_positions[1]
            second_portal ← portal_positions[0]
        Return nearest_portal, second_portal
END

```

4.1.5 Fungsi logicPortal

Fungsi ini memilih jarak antara bot dengan suatu diamond melalui portal/tidak

```

BEGIN
    Function logicPortal(diamonds_position: list[Position],
board_bot: GameObject):
        distance_from_bot ← self.getDistance(diamonds_position[-1],
board_bot.position)
        distance_with_portal ← self.getDistance(self.portal[0],
board_bot.position) + self.getDistance(self.portal[1],
diamonds_position[-1])
        if distance_with_portal < distance_from_bot then
            result ← True
            temp_final_distance ← distance_with_portal
        else
            result ← False
            temp_final_distance ← distance_from_bot
        end if

        Return temp_final_distance, result
END

```

4.1.6 Fungsi logicDiamond

Fungsi ini mencari jarak antara bot dengan diamond

```

BEGIN
    Function logicDiamond(diamonds_position: list[Position],
board_bot: GameObject):
        distance_from_bot ← self.getDistance(diamonds_position[-1],
board_bot.position)
        Return distance_from_bot
END

```

4.1.7 Prosedur greedyNearestDiamond

Prosedur ini bertujuan untuk mencari diamond terdekat dengan bot menggunakan algoritma greedy by distance dan greedy by diamond type. Bot akan mencari diamond merah terdekat terlebih dahulu.

```
BEGIN
    Function greedyNearestDiamond(blue_diamonds_position:
list[Position], red_diamonds_position: list[Position], board_bot:
GameObject, board: Board)
        goal_blue ← None
        goal_red ← None
        leng_blue ← length(blue_diamonds_position)
        leng_red ← length(red_diamonds_position)

        if leng_blue ≠ 0 then
            final_distance_blue ←
self.getDistance(blue_diamonds_position[-1], board_bot.position)
            goal_blue ← blue_diamonds_position[-1]

            if leng_red ≠ 0 then
                final_distance_red ←
self.getDistance(red_diamonds_position[-1], board_bot.position)
                goal_red ← red_diamonds_position[-1]

                while leng_blue ≠ 0 or leng_red ≠ 0 do
                    if leng_blue ≠ 0 then
                        temp_blue ←
self.logicDiamond(blue_diamonds_position, board_bot)
                        if temp_blue < final_distance_blue then
                            final_distance_blue ← temp_blue
                            goal_blue ← blue_diamonds_position[-1]
                            blue_diamonds_position.pop()
                            leng_blue -= 1

                        if leng_red ≠ 0 then
                            temp_red ← self.logicDiamond(red_diamonds_position,
board_bot)
                            if temp_red < final_distance_red then
                                final_distance_red ← temp_red
                                goal_red ← red_diamonds_position[-1]
                                red_diamonds_position.pop()
                                leng_red -= 1

                    if goal_blue ≠ None and goal_red ≠ None then
                        if final_distance_red ≤ final_distance_blue and
board_bot.properties.diamonds ≠ 4 then
                            final_distance_diamond ← final_distance_red
```

```

        self.goal_position ← goal_red
    else
        final_distance_diamond ← final_distance_blue
        self.goal_position ← goal_blue

    elif goal_blue == None then
        final_distance_diamond ← final_distance_red
        self.goal_position ← goal_red

    else
        final_distance_diamond ← final_distance_blue
        self.goal_position ← goal_blue

        if self.distance_to_base ≤ final_distance_diamond and
board_bot.position ≠ board_bot.properties.base and
board_bot.properties.diamonds ≥ 3 then
            if board_bot.position in self.portal then
                self.goal_position ← board_bot.properties.base
            else
                self.logicBasePortal(board_bot, board)

        elif self.distance_to_redButton ≤ final_distance_diamond
then
            if board_bot.position in self.portal then
                self.goal_position ← self.redButton_position[0]
            else
                self.logicRedButtonPortal(board_bot, board)
END

```

4.1.8 Prosedur logicBasePortal

Prosedur ini memilih jarak terdekat dari bot ke base melalui portal/tidak.

```

BEGIN
    Function logicBasePortal(board_bot: GameObject, board: Board)
        bot_position ← board_bot.position
        base ← board_bot.properties.base
        if (self.getDistance(bot_position, self.portal[0]) +
self.getDistance(self.portal[1], base) ≤
self.getDistance(bot_position, base)) then
            self.goal_position ← self.portal[0]
        else
            self.goal_position ← base
END

```

4.1.9 Prosedur logicRedButtonPortal

Prosedur ini bertujuan agar bot memilih jarak terdekat dari bot ke red button melalui portal/tidak

```

BEGIN
    Function logicRedButtonPortal(board_bot: GameObject, board:
Board)
        bot_position ← board_bot.position
        redButton ← self.redButton_position[0]
        if (self.getDistance(bot_position, self.portal[0]) +
self.getDistance(self.portal[1], redButton) ≤
self.getDistance(bot_position, redButton)) then
            self.goal_position ← self.portal[0]
        else
            self.goal_position ← redButton
END

```

4.1.10 Prosedur greedyDiamondPortal

Prosedur ini bertujuan untuk mencari diamond terdekat dengan bot melalui portal dengan algoritma greedy. Sama seperti 4.1.7, namun bedanya disini adalah apabila diamond tersebut dapat diraih lebih dekat dengan melalui portal, maka bot akan menuju ke portal.

```

BEGIN
    Function greedyDiamondPortal(blue_diamonds_position:
list[Position], red_diamonds_position: list[Position], board_bot:
GameObject, board: Board):
        goal_position_blue ← None
        goal_position_red ← None
        leng_blue ← length(blue_diamonds_position)
        leng_red ← length(red_diamonds_position)

        if leng_blue ≠ 0 then
            goal_position_blue ← blue_diamonds_position[-1]
            final_distance_blue ←
self.getDistance(blue_diamonds_position[-1], board_bot.position)

            if leng_red ≠ 0 then
                goal_position_red ← red_diamonds_position[-1]
                final_distance_red ←
self.getDistance(red_diamonds_position[-1], board_bot.position)

                while leng_blue ≠ 0 or leng_red ≠ 0 do
                    if leng_blue ≠ 0 then
                        temp_position_blue ←
self.logicPortal(blue_diamonds_position, board_bot)
                        temp_final_distance_blue ← temp_position_blue[0]
                        if temp_final_distance_blue < final_distance_blue
then
                            final_distance_blue ←
temp_final_distance_blue

```

```

        if temp_position_blue[1] then
            goal_position_blue ← self.portal[0]
        else
            goal_position_blue ←
blue_diamonds_position[-1]

            blue_diamonds_position.pop()
leng_blue -= 1

        if leng_red ≠ 0 then
            temp_position_red ←
self.logicPortal(red_diamonds_position, board_bot)
            temp_final_distance_red ← temp_position_red[0]
            if temp_final_distance_red < final_distance_red
then
                final_distance_red ← temp_final_distance_red
                if temp_position_red[1] then
                    goal_position_red ← self.portal[0]
                else
                    goal_position_red ←
red_diamonds_position[-1]

                    red_diamonds_position.pop()
leng_red -= 1

        if goal_position_blue ≠ None and goal_position_red ≠ None
then
            if final_distance_red ≤ final_distance_blue and
board_bot.properties.diamonds ≠ 4 then
                final_distance_diamond ← final_distance_red
                self.goal_position ← goal_position_red
            else
                final_distance_diamond ← final_distance_blue
                self.goal_position ← goal_position_blue

            elif goal_position_blue == None then
                final_distance_diamond ← final_distance_red
                self.goal_position ← goal_position_red

            else
                final_distance_diamond ← final_distance_blue
                self.goal_position ← goal_position_blue

            if self.distance_to_base ≤ final_distance_diamond and
board_bot.position ≠ board_bot.properties.base and
board_bot.properties.diamonds ≥ 3 then
                if board_bot.position in self.portal then
                    self.goal_position ← board_bot.properties.base
                else

```

```

        self.logicBasePortal(board_bot, board)

    elif self.distance_to_redButton ≤ final_distance_diamond
then
    if board_bot.position in self.portal then
        self.goal_position ← self.redButton_position[0]
    else
        self.logicRedButtonPortal(board_bot, board)
END

```

4.1.11 Fungsi diamondInArea

Fungsi ini mengecek apakah ada diamond di sekitar bot (dalam radius tertentu).

```

BEGIN
    Function diamondInArea(position: Position, radius: int):
        blue ← [b for b in self.blue_diamonds if ((position.x - radius) ≤ b.x ≤ (position.x + radius)) and ((position.y - radius) ≤ b.y ≤ (position.y + radius))]
        red ← [r for r in self.red_diamonds if (position.x - radius) ≤ r.x ≤ (position.x + radius) and (position.y - radius) ≤ r.y ≤ (position.y + radius)]
        Return length(blue) + length(red)
END

```

4.1.12 Prosedur next_move

Prosedur ini bertujuan untuk mencari langkah selanjutnya yang akan diambil oleh bot dengan mengimplementasikan fungsi-fungsi yang telah dibuat.

```

BEGIN
    Function next_move(board_bot: GameObject, board: Board):
        props ← board_bot.properties
        base ← board_bot.properties.base
        self.blue_diamonds ← [diamond.position for diamond in board.game_objects if diamond.type = 'DiamondGameObject' if diamond.properties.points = 1]
        self.red_diamonds ← [diamond.position for diamond in board.game_objects if diamond.type = 'DiamondGameObject' if diamond.properties.points = 2]
        self.portal ← self.getPortal(board_bot, board)

        self.distance_to_base ←
self.getDistance(board_bot.position, board_bot.properties.base)
        distance_to_base_portal ←
self.getDistance(board_bot.position, self.portal[0]) +
self.getDistance(self.portal[1], base)
        if distance_to_base_portal ≤ self.distance_to_base then
            self.distance_to_base ← distance_to_base_portal

```

```

        self.isBasePortal ← True

        self.redButton_position ← [x.position for x in
board.game_objects if x.type = "DiamondButtonGameObject"]
        self.distance_to_redButton ←
self.getDistance(self.redButton_position[0], board_bot.position)
        distance_to_redButton_portal ←
self.getDistance(board_bot.position, self.portal[0]) +
self.getDistance(self.portal[1], self.redButton_position[0])

        if distance_to_redButton_portal ≤
self.distance_to_redButton then
            self.distance_to_redButton ←
distance_to_redButton_portal
            self.isRedButtonPortal ← True

        if props.diamonds = 5 or
(board_bot.properties.milliseconds_left/1000 - 1 ≤
self.distance_to_base) or (props.diamonds = 4 and
len(self.blue_diamonds) = 0) then
            if (board_bot.position in self.portal) then
                self.goal_position ← base
            else
                self.logicBasePortal(board_bot, board)

        else
            if (board_bot.position in self.portal) then
                self.greedyNearestDiamond(self.blue_diamonds,
self.red_diamonds, board_bot, board)
            else
                self.greedyDiamondPortal(self.blue_diamonds,
self.red_diamonds, board_bot, board)

        current_position ← board_bot.position

        delta_x, delta_y ← get_direction(
            current_position.x,
            current_position.y,
            self.goal_position.x,
            self.goal_position.y
        )
        print(board_bot.properties.milliseconds_left)
        return delta_x, delta_y
END

```

4.2 Penjelasan Struktur Data

Bot yang digunakan dalam permainan Etimo Diamond dikembangkan dalam bahasa Python. Program bot memiliki struktur data sebagai berikut:

```
└── game          # Program
    └── logic      # Folder untuk menyimpan algoritma
        ├── __init__.py
        ├── omegaBot.py
        ├── __init__.py
        ├── api.py
        ├── board_handler.py
        ├── bot_handler.py
        ├── models.py
        └── util.py
    ├── .gitignore
    ├── decode.py
    ├── main.py
    ├── requirements.txt
    └── run-bots.bat
    └── run-bots.sh
```

1. Game

Folder game berisi data-data game dan logika pergerakan bot.

2. Logic

Folder logic berisi logika pergerakan bot dan memiliki satu file penting yaitu omegaBot.py yang berisi semua fungsi-fungsi dalam menentukan pergerakan bot selanjutnya.

3. api.py

File ini mengambil data dari game engine dan mengirim data yang dihasilkan oleh logic ke game engine.

4. Board_handler.py

File ini berisi kelas BoardHandler yang mengambil data mengenai Board dari api.

5. Bot_handler.py

File ini berisi kelas BotHandler yang mengambil data mengenai Bot dari api.

6. models.py

File ini berisi kelas-kelas yang mendefinisikan game seperti: game_object(bot, diamond, portal, redButton), board.

7. util.py

File ini berisi fungsi-fungsi yang diperlukan pada folder logic.

8. decode.py

File ini berfungsi untuk mengubah penulisan dari Camel ke Snake.

9. main.py

File ini menerima data hasil dari file logic dan menghubungkannya dengan game-engine.

10. run-bots.bat

File ini berguna untuk menjalankan beberapa bot sekaligus pada OS Windows.

11. run-bots.sh

File ini berguna untuk menjalankan beberapa bot sekaligus pada OS Linux/macOS.

4.3 Analisis dan Pengujian

Kami melakukan analisis dan pengujian dengan cara menjalankan permainan dengan memasukkan bot kami dengan logic yang telah kami buat dan *reference* bot yang telah kami kembangkan menggunakan *greedy by distance*. Setelah kami menjalankan permainan, kami melakukan pengamatan terhadap aksi-aksi yang bot kami lakukan menggunakan *visualizer* yang tersedia.

Berikut adalah analisis yang kami lakukan terhadap suatu pertandingan bot kami yaitu dengan nama “stima3”.

1. Ketika sudah membawa 5 diamond maka bot akan kembali ke base

The screenshot shows a 10x10 grid game board. On the right side, there are four player bases labeled 'stima' through 'stims3'. Each base has a house icon and a chest icon. Blue diamond icons are scattered across the board, with concentrations near the bases and in the center. A red diamond icon is located near the bottom-left base. Two colorful swirl icons are at the bottom center. The top right corner contains a 'Select board' dropdown set to '1', a 'Board 1 players' table, a 'Select season' dropdown set to 'Off season', and a 'Final Score' table.

Name	Diamonds	Score	Time
stima	5	0	27s
stims1	5	5	27s
stims2	5	5	28s
stims3	3	3	29s

Select board: 1

Board 1 players

Name	Diamonds	Score	Time
stima	5	0	27s
stims1	5	5	27s
stims2	5	5	28s
stims3	3	3	29s

Select season: Off season

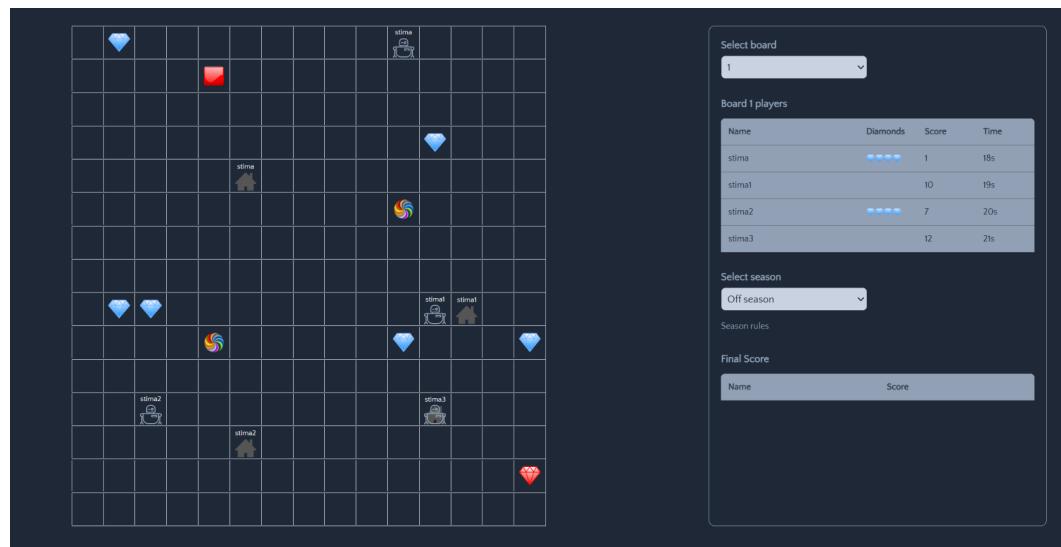
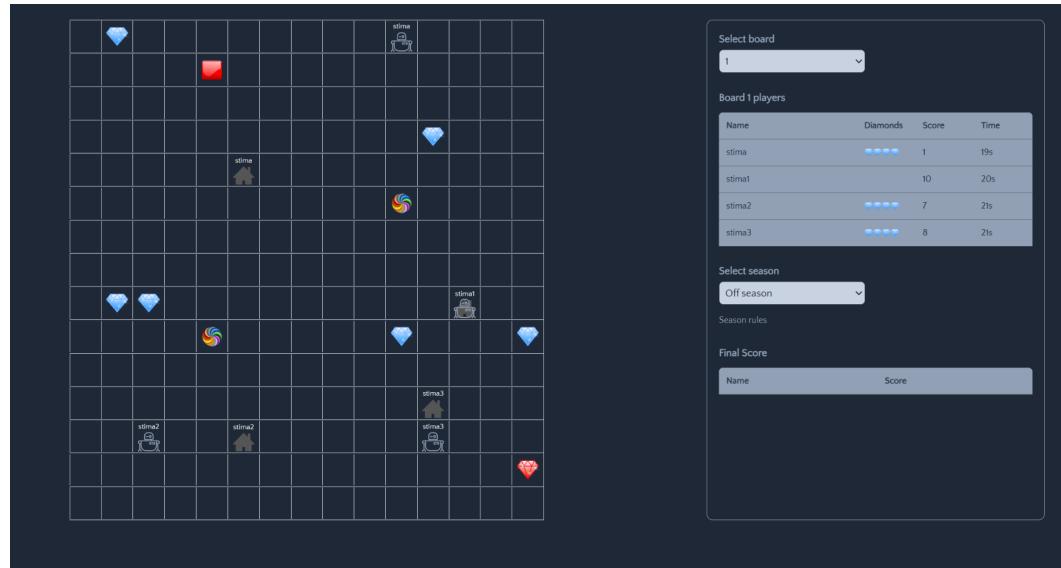
Final Score

Name	Score

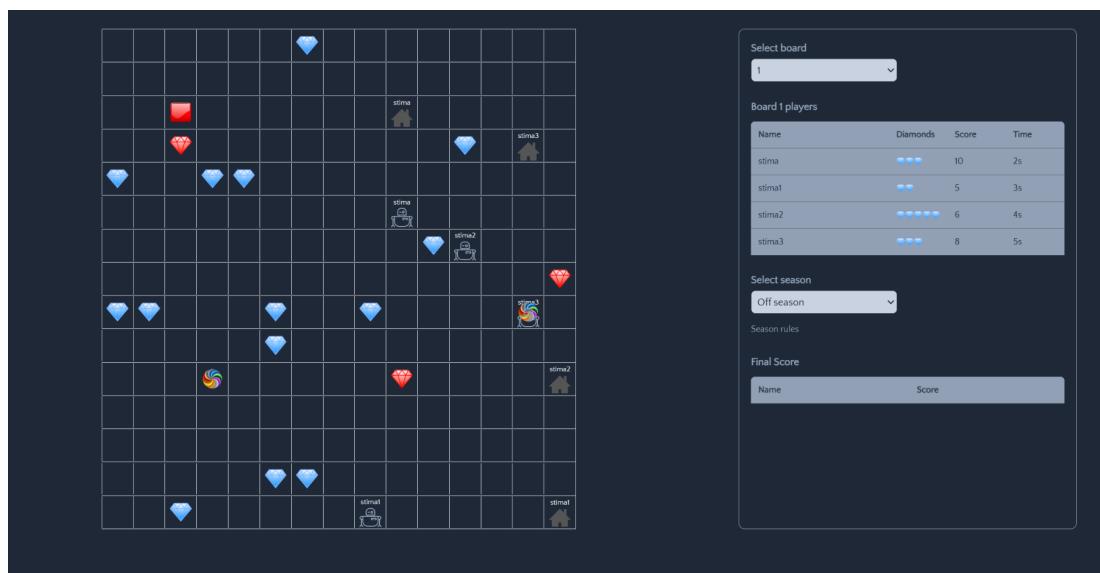
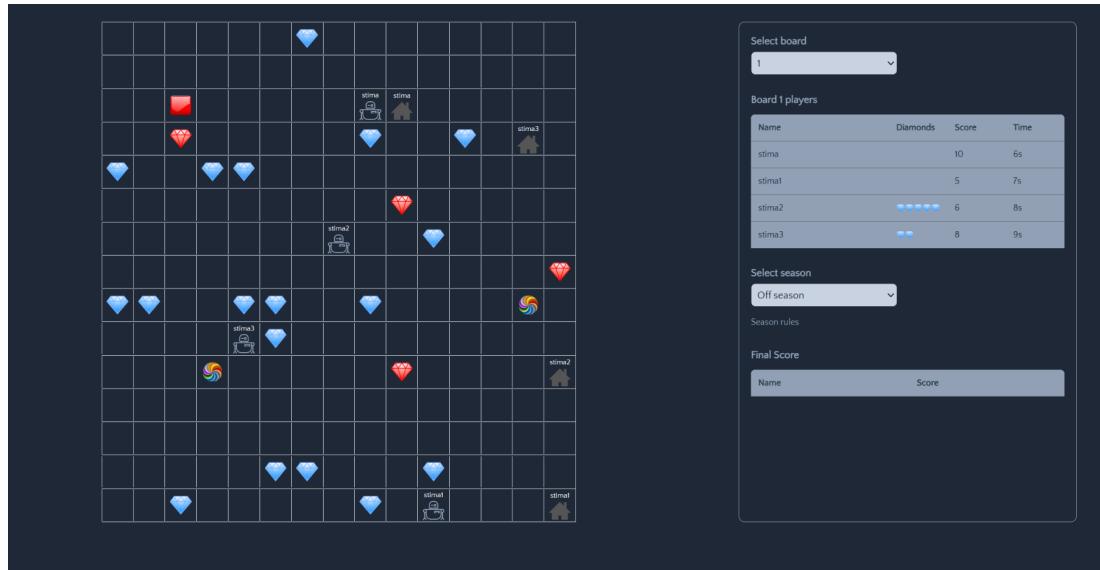
This screenshot is nearly identical to the first one, showing the same board state and player positions. The main difference is in the 'Final Score' table, which now includes a row for 'stims3' with a score of 8. The other player scores remain the same: stima (0), stims1 (5), and stims2 (5).

Name	Score
stima	0
stims1	5
stims2	5
stims3	8

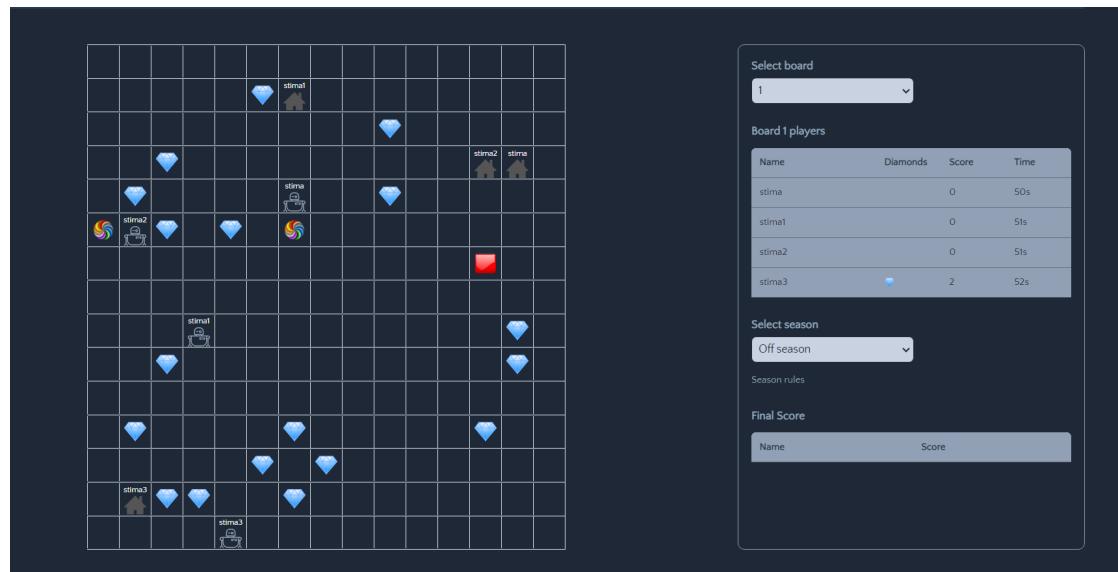
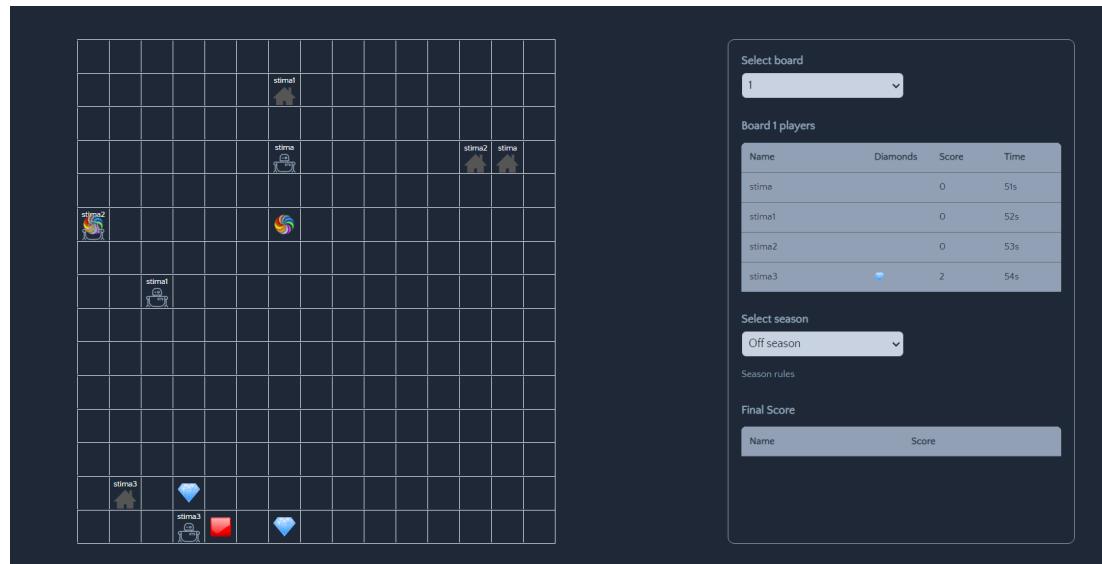
2. Ketika sudah membawa lebih dari/sama dengan 3 diamond dan jarak bot terhadap base lebih dekat daripada jarak bot terhadap diamond terdekat maka bot akan kembali ke base.



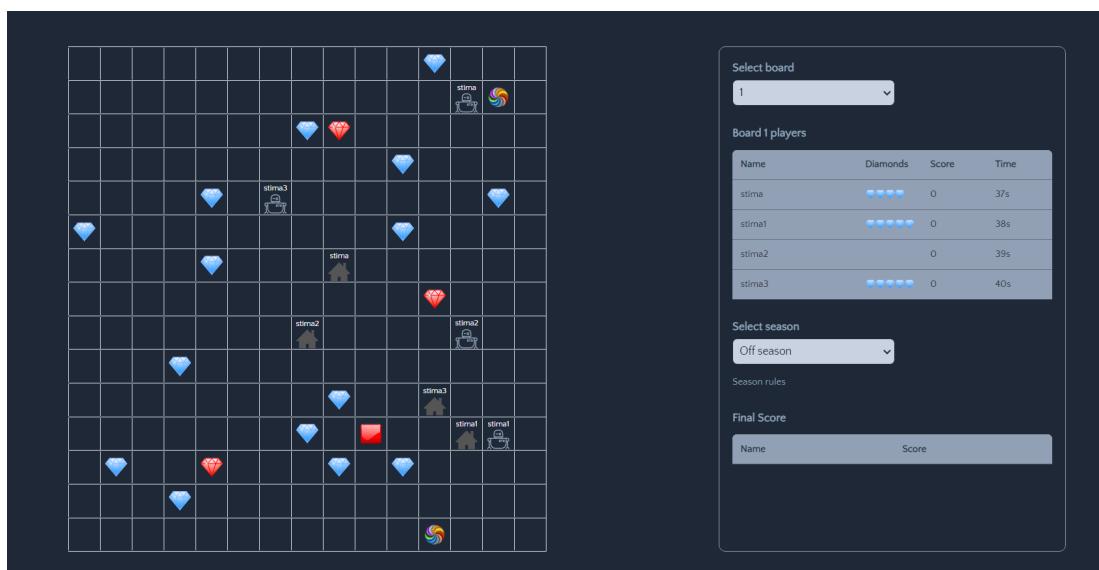
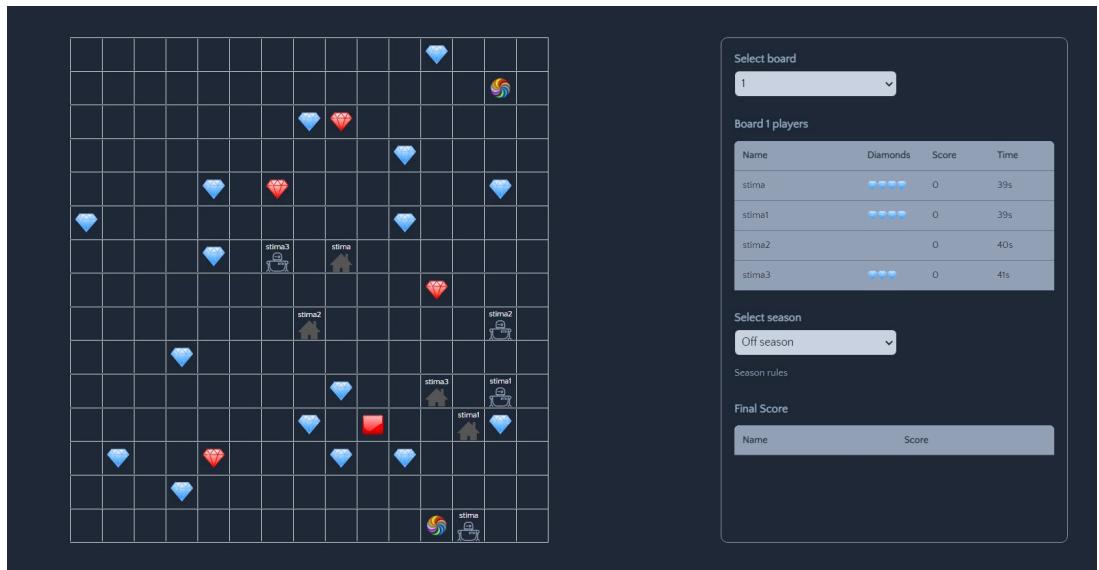
3. Ketika jarak bot terhadap diamond terdekat yang melalui portal lebih dekat daripada jarak bot terhadap diamond terdekat tanpa melalui portal maka bot akan memasuki portal.



4. Ketika jarak bot terhadap *red button* lebih dekat daripada jarak bot terhadap objek-objek lainnya (diamond, base dan/atau portal) maka bot akan menuju *red button*.



5. Ketika jarak bot terhadap diamond merah berada lebih dekat/sama dengan diamond biru maka bot akan mengejar diamond merah terlebih dahulu.



BAB V

KESIMPULAN DAN SARAN

5. 1 Kesimpulan

Melalui Tugas Besar I Strategi Algoritma ini, kami telah berhasil membuat logika bot untuk permainan Etimo Diamond dengan pendekatan algoritma *greedy*. Program final yang kami buat merupakan hasil kombinasi dari dua buah solusi *greedy* yang tersedia dan sudah kami uji untuk menghasilkan strategi terbaik yaitu *greedy by distance and diamond type*.

5.2 Saran

Saran untuk kelompok Omega untuk kedepannya adalah sebagai berikut,

1. Mengoptimalkan struktur pembagian tugas serta meningkatkan kerjasama tim untuk mengoptimalkan hasil program.
2. Perluasan pengembangan algoritma untuk meningkatkan efisiensi dalam mencapai goal program.
3. Peningkatan dalam penulisan komentar dan dokumentasi untuk mempermudah kerja sama tim dan *maintenance* kode program.

LAMPIRAN

- Tautan *repository* Github:

https://github.com/wigaandini/Tubes1_omega.git

- Tautan video penjelasan logika bot:

<https://youtu.be/xtaVg9WamMs?si=vlEcWDkgcGVoOr8N>

DAFTAR PUSTAKA

- [1] Rinaldi, M. (n.d.). *Algoritma Greedy (2021) Bag1*. Diakses 04 Maret 2024 melalui
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf)
- [2] Rinaldi, M. (n.d.). *Algoritma Greedy (2021) Bag2*. Diakses 04 Maret 2024 melalui
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf)
- [3] Rinaldi, M. (n.d.). *Algoritma Greedy (2021) Bag3*. Diakses 04 Maret 2024 melalui
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-Bag3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag3.pdf)
- [4] Gregory, J. (2018). *Game Engine Architecture, Third Edition*. CRC Press LLC.