

LAPORAN TUGAS BESAR STRATEGI ALGORITMA IF2211

PENERAPAN ALGORITMA GREEDY DALAM PEMBUATAN LOGIC UNTUK BOT PERMAINAN DIAMONDS



Disusun oleh Kelompok MahaSantuy:

123140054	-	Marcel Kevin Togap Siagian
123140060	-	Nadia Anatashiva
123140090	-	Andika Rahman Pratama

DAFTAR ISI

DAFTAR ISI.....	ii
DAFTAR GAMBAR.....	iii
DAFTAR TABEL.....	iv
BAB I.....	1
DESKRIPSI TUGAS.....	1
BAB II.....	7
LANDASAN TEORI.....	7
A. Algoritma Greedy.....	7
B. Cara Kerja Program.....	8
C. Implementasi Greedy Dalam Program.....	10
BAB III.....	13
APLIKASI STRATEGI ALGORITMA.....	13
A. Mapping Persoalan Pada Permainan Diamonds.....	13
B. Eksplorasi Alternatif Solusi Greedy.....	14
C. Solusi Greedy Pilihan.....	19
BAB IV.....	20
IMPLEMENTASI DAN PENGUJIAN.....	20
A. Penjelasan Algoritma.....	20
B. Penjelasan Struktur Data.....	24
C. Analisis Solusi Pengujian.....	26
BAB V.....	28
Kesimpulan.....	28
Saran.....	29
LAMPIRAN.....	30
DAFTAR PUSTAKA.....	31

DAFTAR GAMBAR

Gambar 1.1 Layar Permainan Diamond	2
Gambar 1.2 Varian Diamond Biru	3
Gambar 1.3 Varian Diamond Merah	3
Gambar 1.4 Red Button / Diamond Button	3
Gambar 1.5 Teleporters	4
Gambar 1.6 Bot dengan name “stima”	4
Gambar 1.7 Base dengan name “stima”	4
Gambar 1.8 Layar Inventory	5
Gambar 2.1 Struktur Direktori Program	10
Gambar 2.2 Base Logic bot	11

DAFTAR TABEL

Tabel 1.1 Greedy by base_time	14
Tabel 1.2 Greedy by diamonds near base	15
Tabel 1.3 Greedy by fallback base	16
Tabel 1.4 Greedy by nearest blue gem	17
Tabel 1.5 Greedy by nearest red gem	18
Tabel 1.6 Greedy by pursue enemies	18
Tabel 1.7 Greedy by switch merah	19
Tabel 2.1 Penjelasan Struktur Data	24
Tabel 2.2 Penjelasan Struktur Data	26
Tabel 2.3 Analisis Pengujian	26
Tabel 2.4 Analisis Pengujian	27

BAB I

DESKRIPSI TUGAS

Permainan Diamonds adalah sebuah game strategi di mana pemain mengendalikan sebuah bot untuk mengumpulkan diamond sebanyak-banyaknya dan memperoleh skor tertinggi. Dalam permainan ini, terdapat dua jenis diamond, yaitu diamond biru yang bernilai 1 poin dan diamond merah yang bernilai 2 poin. Diamond-diamond ini tersebar di seluruh board dan akan terus di-regenerasi secara berkala dengan posisi serta rasio jenis yang berubah-ubah, sehingga pemain harus terus menyesuaikan strategi pengambilan diamond sepanjang permainan.

Setiap bot memiliki sebuah inventory, yaitu tempat penyimpanan sementara diamond yang telah dikumpulkan. Namun, inventory ini memiliki kapasitas terbatas, sehingga jika penuh, bot tidak bisa mengumpulkan diamond lagi. Oleh karena itu, bot harus sesekali kembali ke base miliknya untuk menyetorkan isi inventory. Saat bot mencapai base, semua diamond yang dibawa akan dikonversi menjadi skor, dan inventory bot akan dikosongkan kembali agar bisa digunakan untuk pengumpulan berikutnya. Posisi awal bot dan base ditentukan secara acak di awal permainan.

Permainan ini juga dilengkapi dengan beberapa elemen tambahan yang bisa dimanfaatkan secara strategis. Salah satunya adalah red button. Ketika bot menginjak atau melewati red button, maka seluruh diamond di board akan di-reset dan digenerate ulang pada posisi acak, termasuk posisi red button itu sendiri. Selain itu, terdapat juga teleporter, yakni dua buah titik pada board yang saling terhubung—jika bot memasuki salah satu teleporter, maka ia akan langsung berpindah ke teleporter yang lain, memungkinkan pergerakan instan ke bagian lain dari peta.

Interaksi antar bot juga menjadi bagian penting dalam strategi permainan. Jika sebuah bot menempati posisi yang sama dengan bot lawan (baik disengaja maupun tidak), maka bot lawan akan dianggap tertackle, dikembalikan ke base miliknya, dan seluruh diamond yang ada di inventory-nya akan hilang dan berpindah ke inventory bot yang men-tackle. Ini menjadikan positioning dan penghindaran terhadap bot lawan sebagai aspek krusial dalam menyusun strategi.

Setiap bot diberi waktu gerak yang sama dalam setiap giliran, dan permainan akan berlangsung hingga waktu habis. Setelah waktu berakhir, seluruh skor akan dihitung berdasarkan jumlah diamond yang berhasil disetorkan ke base, dan hasil akhir ditampilkan dalam tabel Final Score. Tujuan utama dari permainan ini adalah mengoptimalkan

pengumpulan dan penyeteroran diamond sembari memanfaatkan fitur-fitur seperti teleporter dan red button, serta menghindari konfrontasi yang merugikan dengan bot lawan.

Untuk memenangkan kompetisi, peserta harus menerapkan strategi khusus pada bot masing-masing, seperti :



Gambar 1.1 Layar Permainan Diamond

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

Program permainan Diamonds terdiri atas:

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

Komponen-komponen dari permainan Diamonds antara lain:

1. Diamonds

Untuk meraih kemenangan dalam pertandingan, pemain harus mengumpulkan sejumlah diamond dengan cara melewati atau mengambilnya. Ada dua varian diamond dalam permainan: diamond merah dan biru. Setiap diamond merah memiliki nilai dua poin, sementara diamond biru bernilai satu poin. Diamond akan terus muncul kembali secara periodik, dan proporsi antara diamond merah dan biru akan berubah-ubah setiap kali terjadi regenerasi.



Gambar 1.2 Varian Diamond Biru



Gambar 1.3 Varian Diamond Merah

2. Red Button / Diamond Button

Saat tombol merah dilewati atau dilangkahi, semua diamond, diamond merah maupun diamond biru, akan muncul kembali di papan permainan dengan posisi yang acak. Lokasi tombol merah itu sendiri juga akan berubah ke posisi acak setelah tombol tersebut diaktifkan.



Gambar 1.4 Red Button / Diamond Button

3. Teleporters

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



Gambar 1.5 Teleporters

4. Bots

Pada game ini kita akan menggerakkan bot untuk mendapatkan diamond sebanyak banyaknya. Bot ini bisa men-tackle bot lainnya.



Gambar 1.6 Bot dengan name “stima”

5. Bases

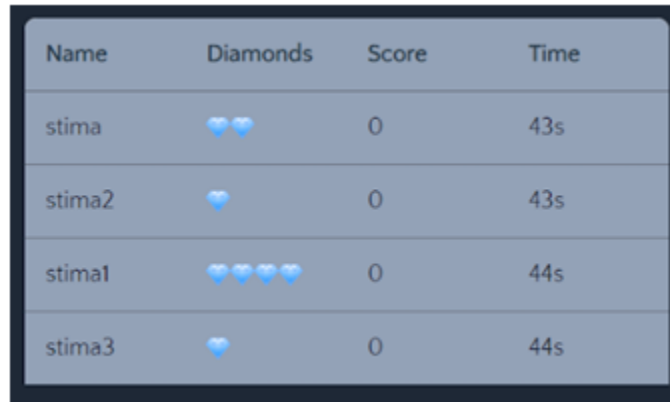
Setiap bot dilengkapi dengan sebuah Base, yang berfungsi sebagai tempat untuk menampung diamond yang dikumpulkan. Ketika diamond disimpan di Base, skor dari bot tersebut akan meningkat sesuai dengan nilai diamond yang disimpan, dan inventaris bot (yang akan diuraikan lebih lanjut) akan dikosongkan.



Gambar 1.7 Base dengan name “stima”

6. Inventory

Bot dilengkapi dengan sebuah inventaris, yang berperan sebagai lokasi penyimpanan sementara untuk diamond yang telah dikumpulkan. Inventaris ini dibatasi oleh sebuah kapasitas maksimal, yang artinya dapat terisi penuh. Untuk menghindari keadaan penuh tersebut, bot dapat mentransfer isi inventaris mereka ke Base, sehingga memungkinkan inventaris tersebut untuk dikosongkan kembali.



Name	Diamonds	Score	Time
stima	2	0	43s
stima2	1	0	43s
stima1	4	0	44s
stima3	1	0	44s

Gambar 1.8 Layar Inventory

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.

Dalam tugas besar ini, mahasiswa diminta untuk bekerja dalam kelompok minimal dua orang dan maksimal tiga orang, dengan lintas kelas dan lintas kampus diperbolehkan. Tujuan dari tugas ini adalah untuk mengimplementasikan algoritma Greedy pada bot permainan Diamonds dengan strategi yang dirancang untuk memenangkan permainan. Setiap kelompok diminta untuk mengembangkan strategi Greedy terbaik yang berkaitan dengan fungsi objektif permainan, yaitu mengumpulkan diamond sebanyak mungkin dan mencegah diamond tersebut diambil oleh bot lain.

Strategi Greedy yang diterapkan oleh setiap kelompok harus dijelaskan secara eksplisit dalam laporan, dan kode program yang sesuai dengan strategi tersebut harus disertakan. Mahasiswa dilarang menggunakan kode program yang diunduh dari internet; setiap kelompok diharapkan untuk membuat program mereka sendiri dengan menggunakan kreativitas mereka. Program harus dapat dijalankan pada game engine yang ditentukan dan dapat bersaing dengan bot dari kelompok lain.

Setiap implementasi strategi Greedy harus dijelaskan dengan komentar yang jelas dalam kode program. Selain itu, terdapat bonus poin untuk kelompok yang membuat video tentang aplikasi Greedy pada bot dan simulasinya dalam permainan, dengan menampilkan wajah dari setiap anggota kelompok. Asistensi tugas besar bersifat opsional, namun jika diperlukan, mahasiswa dapat meminta bimbingan dari asisten yang dipilih.

Kelompok yang telah membuat bot akan berkompetisi dengan kelompok lain dalam kompetisi yang disaksikan oleh seluruh peserta kuliah. Hadiah menarik akan diberikan kepada kelompok yang memenangkan kompetisi. Informasi terkait pendataan kelompok, asistensi, demo, dan pengumpulan laporan telah disediakan melalui tautan yang telah disediakan.

BAB II

LANDASAN TEORI

A. Algoritma Greedy

Algoritma greedy adalah suatu pendekatan pemecahan masalah yang bekerja dengan membuat pilihan terbaik secara langsung pada setiap langkah berdasarkan kondisi saat itu. Ia tidak memperhitungkan konsekuensi jangka panjang dari keputusan yang diambil, melainkan hanya fokus pada manfaat instan yang tampak paling menguntungkan. Pendekatan ini cenderung sederhana dan cepat karena tidak perlu memeriksa semua kemungkinan atau kombinasi solusi, sehingga sangat efisien untuk masalah berskala besar dengan batasan waktu komputasi. Keputusan yang diambil bersifat tidak dapat diubah, dan proses berlanjut secara bertahap hingga solusi terbentuk dengan memenuhi dua syarat, yaitu :

1. Greedy Choice Property atau Solusi optimal global dapat dibangun dari pilihan lokal yang optimal.
2. Optimal Substructure atau Solusi optimal dari masalah mengandung solusi optimal dari sub masalahnya.

Meskipun demikian, algoritma greedy tidak selalu menjamin tercapainya solusi yang optimal secara keseluruhan. Dalam beberapa persoalan, pilihan terbaik di setiap langkah belum tentu membentuk keseluruhan solusi terbaik. Misalnya, dalam kasus Travelling Salesman Problem (TSP), strategi greedy bisa memilih kota terdekat pada setiap langkah, namun keputusan tersebut tidak selalu menghasilkan lintasan dengan total jarak minimum. Karena itu, greedy seringkali dipakai sebagai pendekatan heuristik atau pendekatan hampiran (approximation) untuk mendapatkan solusi cepat yang cukup baik, terutama bila pencarian solusi optimal membutuhkan waktu dan sumber daya yang besar. Dalam proses perancangannya, algoritma greedy melibatkan sejumlah komponen utama yang saling berkaitan, yaitu :

1. Himpunan kandidat (C), yaitu kumpulan elemen yang mungkin dipilih sebagai bagian dari solusi.
2. Himpunan solusi (S), yaitu himpunan kandidat yang telah dipilih sejauh ini.

3. Sebuah fungsi solusi diperlukan untuk memeriksa apakah solusi yang terbentuk telah memenuhi kriteria penyelesaian.
4. Pilihan kandidat ditentukan oleh fungsi seleksi, yang merupakan aturan greedy yang bersifat heuristik—umumnya berdasarkan nilai tertinggi, durasi terpendek, jarak terdekat, atau rasio tertentu.
5. Selain itu, setiap elemen yang dipilih harus melewati fungsi kelayakan, yang memastikan bahwa penambahan kandidat tersebut masih menghasilkan solusi yang valid.
6. Dan, fungsi objektif digunakan untuk menentukan ukuran keberhasilan solusi, apakah berupa nilai maksimum atau minimum dari parameter yang dioptimalkan.

B. Cara Kerja Program

Diamonds merupakan sebuah tantangan pemrograman berbasis permainan strategi, di mana setiap peserta diminta untuk mengembangkan sebuah bot otomatis yang bertujuan untuk mengumpulkan diamond sebanyak mungkin di dalam sebuah papan permainan digital. Permainan ini tidak sekadar melibatkan pengumpulan item, tetapi juga mencakup berbagai mekanisme interaktif dan rintangan kompleks, seperti:

- **Teleporter**, untuk memindahkan bot ke lokasi lain secara instan.
- **Red Button**, untuk mengatur ulang posisi diamond di papan.
- **Tackle System**, supaya bot dapat menabrak bot lain untuk mencuri diamond yang dimiliki lawan.

Seluruh elemen ini menghadirkan tantangan menarik yang menuntut penerapan strategi dan pengambilan keputusan yang cerdas secara real-time. 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

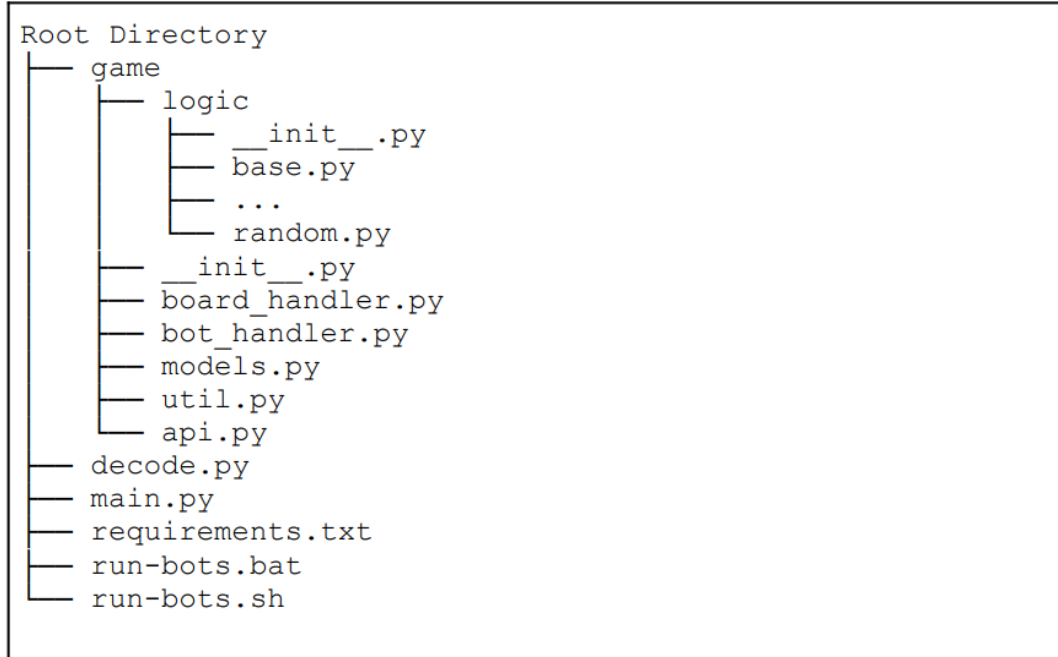
team. Jika berhasil, maka backend akan memberikan response code 200 dengan body berisi id dari bot tersebut.

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.

C. Implementasi Greedy Dalam Program

1. Struktur Direktori Program

Direktori dari program adalah *root\game\logic*



Gambar 2.1 Struktur Direktori Program

2. Implementasi Algoritma Greedy Bot

Membuat file dari logic bot terlebih dahulu di dalam direktori, misalnya *root\game\logic\nama_bot.py*. Kemudian akan melakukan inheritance kelas *BaseLogic*, kemudian mengimplementasikan Algoritma Greedy.

```

from typing import Optional, List, Tuple
from game.logic.base import BaseLogic
from game.models import GameObject, Board, Position

class MahaSantuyLogic(BaseLogic):
    def __init__(self):
        self.move_vectors = [(1, 0), (0, 1), (-1, 0), (0, -1)]
        self.target_spot: Optional[Position] = None
        self.pursuit_count = 0
        self.using_portal = False

```

Gambar 2.2 Base Logic bot

3. Menjalankan Bot

Program bot dapat dijalankan dengan menggunakan terminal ataupun cmd. Untuk menjalankan satu bot dengan logic yang bernama MahaSatuyLogic maka dapat melakukan perintah seperti di bawah ini.

```
python main.py --logic Mahasantuy --email=mahasantuy@email.com --name=mahasantuy
--password=123456 --team etimo
```

Untuk menjalankan beberapa bot sekaligus , perlu membuat file run-bots.bat pada Windows atau [run-bots.sh](#) pada Linux dan MacOS. File tersebut akan berisi perintah-perintah sebagai berikut.

```

@echo off
start cmd /c "python main.py --logic Killer --email=killer@email.com --name=killer
--password=123456 --team etimo"
start cmd /c "python main.py --logic Mohosantoy --email=mohosantoy@email.com
--name=mohosantoy --password=123456 --team etimo"
start cmd /c "python main.py --logic Mehesantuy --email=tercepat@email.com
--name=tercepat --password=123456 --team etimo"
start cmd /c "python main.py --logic Mahasantuy --email=mahasantuy@email.com
--name=mahasantuy --password=123456 --team etimo"

```

Kemudian dapat menjalankan file nya dengan memberikan perintah:

- Untuk Windows

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

-Untuk Linux / MacOS

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


BAB III

APLIKASI STRATEGI ALGORITMA

A. Mapping Persoalan Pada Permainan Diamonds

Pada pembuatan bot permainan diamonds, algoritma greedy dibuat dengan pertama-tama melakukan mapping dari elemen pada permainan diamonds kepada elemen pada algoritma greedy:

1. Himpunan Kandidat (C)

Kumpulan kandidat pada permasalahan ini mencakup koordinat-koordinat dari berbagai objek dalam permainan. Objek-objek tersebut meliputi blue diamond, red diamond, red button, teleporter, base bot, bot musuh, serta base musuh. Selain itu, terdapat elemen penting lain yang perlu diperhatikan dalam permainan, yaitu jarak antara bot dan titik tujuan.

2. Himpunan Solusi (S)

Himpunan solusi terdiri dari koordinat-koordinat yang memberikan keuntungan maksimal saat dikunjungi oleh bot. Dengan menggunakan pendekatan greedy, himpunan ini dapat berupa satu koordinat saja atau sejumlah koordinat yang perlu dilewati secara berurutan.

3. Fungsi solusi

Fungsi solusi berperan untuk mengevaluasi apakah kandidat yang dipilih sudah menghasilkan solusi yang valid. Tujuannya adalah untuk menghitung apakah ada cukup yang tersedia dan masih mencukupi untuk mencapai suatu koordinat di dalam peta dan kembali ke base.

4. Fungsi seleksi (selection function)

Fungsi seleksi berperan dalam menentukan kandidat mana yang dipilih, sesuai dengan strategi greedy yang digunakan. Karena setiap pendekatan greedy memiliki cara seleksi yang unik, pembahasan lebih rinci mengenai fungsi ini akan disampaikan pada bagian alternatif solusi.

5. Fungsi kelayakan (feasible)

Fungsi kelayakan berfungsi untuk memastikan bahwa kandidat yang dipilih memang dapat dimasukkan ke dalam solusi. Dalam permainan *Diamonds*, fungsi ini mengecek apakah suatu objek di dalam game masih bisa dijadikan target berikutnya. Sebagai contoh, jika bot ingin mengambil red diamond tetapi sudah membawa empat diamond di inventarisnya, maka red diamond tersebut tidak bisa

diambil. Fungsi kelayakan akan mencegah bot bergerak ke arah objek tersebut, karena jika dipaksakan, bisa terjadi bug seperti bot yang terus berputar di sekitarnya atau bahkan melakukan pergerakan yang tidak valid.

6. Fungsi objektif

Fungsi objektif ini berfungsi untuk memilih kandidat terbaik dari sekumpulan pilihan yang ada, dengan fokus pada mendapatkan keuntungan terbesar atau meminimalkan sesuatu yang kurang diinginkan. Dalam konteks ini, fungsi ini membantu menentukan kandidat mana yang paling menguntungkan untuk dijadikan solusi. Jadi, intinya fungsi ini dipakai supaya hasil yang didapat dari kandidat-kandidat tersebut bisa maksimal.

B. Eksplorasi Alternatif Solusi Greedy

1. Greedy by base_time

Strategi greedy ini fokus pada pemilihan aksi yang mengutamakan jarak terdekat menuju base sebagai prioritas utama. Tujuannya adalah memastikan bot atau bot bisa segera kembali ke base sebelum waktu habis atau kondisi berubah, sehingga mengamankan poin, sumber daya, atau keselamatan. Biasanya digunakan dalam situasi di mana waktu menjadi faktor kritis dan bot harus memastikan hasil kerja tidak hilang.

Elemen Greedy	Deskripsi
Himpunan Kandidat	Posisi base sebagai target utama
Himpunan Solusi	Gerakan bot menuju base untuk mengamankan hasil sebelum waktu habis
Fungsi Solusi	Solusi tercapai saat bot sampai di base sebelum waktu habis.
Fungsi Seleksi	Kandidat base diprioritaskan dengan skor sangat tinggi
Fungsi Kelayakan	Hanya aktif saat waktu hampir habis, atau saat harus kembali base
Fungsi Objektif	Minimalisasi jarak ke base supaya bisa sampai tepat waktu

Tabel 1.1 Greedy by base_time

Analisis :

- Efisiensi

Sangat tinggi, karena meminimalkan risiko kehabisan waktu di lapangan dengan memperpendek jarak pulang. Sistem ini cenderung hemat waktu dan memperkecil potensi kerugian total.

- Efektivitas

Cukup tinggi dalam skenario time-critical, tetapi rendah dalam memaksimalkan nilai (score) karena bisa mengabaikan sumber daya bernilai tinggi yang lebih jauh.

2. Greedy by diamonds near base

Dalam strategi ini, prioritas utama adalah status inventori bot. Ketika kapasitas inventory sudah penuh, bot diprioritaskan untuk kembali ke base untuk mengosongkan inventori agar dapat terus mengumpulkan sumber daya. Fokus utama strategi ini adalah efisiensi pengumpulan dengan meminimalkan waktu kehilangan peluang akibat inventori penuh, sehingga bot tidak kehabisan ruang untuk menyimpan hasil kerja.

Elemen Greedy	Deskripsi
Himpunan Kandidat	Posisi base sebagai target kembali jika inventori permata penuh
Himpunan Solusi	Gerakan bot kembali ke base untuk menurunkan inventori
Fungsi Solusi	Solusi tercapai saat inventori bot kosong setelah kembali ke base
Fungsi Seleksi	Kandidat base dengan diaktifkan jika inventori penuh
Fungsi Kelayakan	Kondisi inventori sudah mencapai batas tertentu (dengan default 5 permata)
Fungsi Objektif	Memaksimalkan keuntungan dengan cepat mengamankan permata

Tabel 1.2 Greedy by diamonds near base

Analisis :

- **Efisiensi:** Sangat baik untuk menghindari aksi sia-sia. Membantu menjaga kontinuitas misi dan meminimalkan waktu terbuang karena overcapacity.
- **Efektivitas:** Moderat hingga tinggi, tergantung pada bagaimana sistem menghitung trade-off antara perjalanan dan reward. Bisa tidak efektif jika agen pulang terlalu cepat dan belum mengisi inventori secara optimal.

3. Greedy by fallback base

Strategi ini menggunakan base sebagai titik aman terakhir atau fallback ketika tidak ada target yang layak atau kondisi lingkungan tidak memungkinkan untuk mengambil risiko. bot akan selalu punya opsi kembali ke base sebagai langkah defensif untuk menghindari kerugian atau ancaman, sehingga memberikan mekanisme keamanan dan kontinuitas dalam pengambilan keputusan secara greedy.

Elemen Greedy	Deskripsi
Himpunan Kandidat	Posisi base sebagai target jika kondisi darurat terjadi (misal tidak ada kandidat lain valid)
Himpunan Solusi	Gerakan kembali ke base sebagai pilihan terakhir
Fungsi Solusi	Solusi aman dengan kembali ke base
Fungsi Seleksi	Kandidat base dengan prioritas sedang
Fungsi Kelayakan	Jika tidak ada kandidat lain yang feasible.
Fungsi Objektif	Meminimalisasi risiko kehilangan poin dengan kembali ke base.

Tabel 1.3 Greedy by fallback base

Analisis :

- **Efisiensi:** Lebih rendah dibanding strategi lain karena sering menghindari risiko, tapi justru mengurangi kejadian kegagalan fatal.
- **Efektivitas:** Tinggi di lingkungan berisiko tinggi atau tidak pasti. Cocok untuk mengamankan progress, tetapi tidak ideal dalam sistem yang mengejar skor maksimal dengan agresif.

4. Greedy by nearest blue gem

Strategi ini mengarahkan bot untuk selalu memilih permata biru yang paling dekat sebagai target pengumpulan. Permata biru biasanya memiliki nilai yang stabil dan mudah dijangkau, sehingga strategi ini berfokus pada efisiensi waktu dan pengumpulan poin

secara konsisten tanpa mempertimbangkan risiko besar atau nilai tinggi yang lebih jauh.

Elemen Greedy	Deskripsi
Himpunan Kandidat	Posisi permata biru terdekat
Himpunan Solusi	Gerakan menuju permata biru untuk mengumpulkan poin
Fungsi Solusi	Permata biru diambil dan menambah inventori
Fungsi Seleksi	Kandidat permata biru dipilih berdasarkan skor
Fungsi Kelayakan	Ada permata biru yang bisa dicapai
Fungsi Objektif	Memaksimalkan jumlah permata biru yang dikumpulkan dengan jarak minimum

Tabel 1.4 Greedy by nearest blue gem

Analisis :

- **Efisiensi:** Tinggi, karena minim waktu tempuh per tindakan. Sangat cocok untuk sistem yang menghargai frekuensi over value.
- **Efektivitas:** Cukup rendah dalam konteks jangka panjang, karena reward dari blue gem biasanya lebih kecil. Agen bisa menghabiskan banyak energi untuk hasil yang totalnya tidak optimal.

5. Greedy by nearest red gem

Berbeda dengan blue gem, strategi ini mengutamakan pengambilan permata merah terdekat yang biasanya memiliki nilai lebih tinggi. Meskipun permata merah bisa berada lebih jauh atau di lokasi yang lebih berisiko, strategi ini mengutamakan potensi keuntungan besar secara langsung dengan memilih target yang menjanjikan poin tinggi dalam jarak yang relatif dekat.

Elemen Greedy	Deskripsi
Himpunan Kandidat	Posisi permata merah terdekat
Himpunan Solusi	Gerakan ke permata merah untuk mengumpulkan poin lebih tinggi
Fungsi Seleksi	Kandidat permata merah dengan skor lebih tinggi

Fungsi Kelayakan	Ada permata merah yang valid dan bisa dicapai
Fungsi Objektif	Memaksimalkan keuntungan dengan prioritas permata merah

Tabel 1.5 Greedy by nearest red gem

Analisis :

- **Efisiensi:** Sedang hingga rendah tergantung posisi permata merah. Perjalanan jauh bisa menyita banyak waktu dan energi.
- **Efektivitas:** Tinggi jika berhasil, karena reward besar dapat memberikan nilai yang signifikan untuk sedikit aksi. Namun bisa sangat buruk jika gagal mencapai target atau terlalu banyak waktu terbuang.

6. Greedy by pursue enemies

Strategi ini memfokuskan pada pengejaran musuh sebagai prioritas utama. Tujuannya adalah mengganggu pergerakan lawan, mengalahkan musuh, atau merebut sumber daya yang sedang dikuasai musuh. Ini merupakan strategi agresif yang menggunakan pendekatan serangan sebagai metode untuk mendapatkan keuntungan taktis dalam permainan.

Elemen Greedy	Deskripsi
Himpunan Kandidat	Posisi musuh yang bisa dikejar
Himpunan Solusi	Gerakan menuju musuh untuk men-tackle
Fungsi Seleksi	Kandidat musuh yang memungkinkan
Fungsi Kelayakan	Musuh ada dan bisa dikejar
Fungsi Objektif	Memaksimalkan keuntungan dengan men-tackle musuh

Tabel 1.6 Greedy by pursue enemies

Analisis :

- **Efisiensi:** Rendah hingga sedang; memburu musuh sering kali membuat agen keluar jalur utama dan menyita waktu tanpa jaminan hasil.
- **Efektivitas:** Kontekstual. Bisa sangat efektif untuk menurunkan performa lawan atau merebut sumber daya mereka, tetapi juga bisa merugikan jika agen terlalu agresif dan tidak memiliki reward yang jelas.

7. Greedy by switch merah

Strategi ini memilih untuk mengaktifkan switch merah tertentu yang dapat memicu perubahan kondisi strategis di area permainan, seperti membuka jalan, menonaktifkan perangkat, atau memberikan keuntungan khusus lainnya. Fokusnya adalah menggunakan elemen lingkungan sebagai alat untuk memperkuat posisi dan memaksimalkan potensi pengumpulan poin secara greedy.

Elemen Greedy	Deskripsi
Himpunan Kandidat	Posisi switch merah yang bisa ditekan
Fungsi Solusi	Kandidat switch merah dengan skor
Fungsi Seleksi	Ada switch merah yang valid dan bisa dicapai
Fungsi Kelayakan	Kondisi inventori sudah mencapai batas tertentu (dengan default 5 permata)
Fungsi Objektif	Memaksimalkan keuntungan dengan mengaktifkan switch yang menguntungkan

Tabel 1.7 Greedy by switch merah

Analisis :

- **Efisiensi:** Situasional; bisa sangat efisien jika switch membuka akses ke reward besar, atau sebaliknya menjadi pemborosan jika tidak memberi efek langsung.
- **Efektivitas:** Tinggi jika digunakan dalam strategi makro. Mengontrol peta atau mengubah jalur bisa memberi keuntungan besar jangka panjang. Namun membutuhkan pemahaman konteks dan waktu yang tepat.

C. Solusi Greedy Pilihan

Setelah dilakukan serangkaian uji coba, ditemukan bahwa bot mahasantuy merupakan bot dengan strategi paling efisien. Hal ini didasarkan pada hasil pengujian bot Mahasantuy mendapatkan skor terbanyak lebih sering daripada strategi bot lainnya dengan format 1 v 1 v 1 v 1 v 1 v 1 v 1 v 1 v 1. Berdasarkan pengamatan kami, bot mahasantuy mampu unggul dibanding algoritma lainnya, sebab mahasantuy unggul dalam memilih berlian berdasarkan poin ataupun jarak dari beberapa berlian sekaligus. Salah satunya, bot akan memprioritaskan berlian dalam radius 3 petak dari base, dan menghindari pengejaran diamond jauh yang memakan waktu.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

A. Penjelasan Algoritma

```
CLASS MahaSantuyLogic
  INITIALIZE:
    move_vectors ← [(1,0), (0,1), (-1,0), (0,-1)]
    target_spot ← NULL
    pursuit_count ← 0
    using_portal ← FALSE

  FUNCTION nearby_gems(bot, board):
    RETURN list of diamond positions where:
      diamond is within 3 tiles of bot's base (Manhattan distance)

  FUNCTION is_near_home(bot):
    RETURN TRUE if bot is within 3 tiles of its base, ELSE FALSE

  FUNCTION nearest_gem_to_base(bot, gems):
    RETURN gem position from gems closest to bot (Manhattan
distance)

  FUNCTION has_gems_near_base(bot, board):
    RETURN TRUE if any diamond is within 2 tiles of bot's base, ELSE
FALSE

  FUNCTION nearest_blue_gem(bot, board):
    IF no blue diamonds (1 point) exist:
      RETURN NULL
    RETURN position of nearest blue diamond to bot

  FUNCTION blue_gem_distance(bot, board):
    closest ← nearest_blue_gem(bot, board)
    IF closest is NULL:
      RETURN 999
    RETURN Manhattan distance from bot to closest

  FUNCTION nearest_red_gem(bot, board):
    IF no red diamonds (2 points) exist:
      RETURN NULL
    RETURN position of nearest red diamond to bot

  FUNCTION red_gem_distance(bot, board):
    closest ← nearest_red_gem(bot, board)
    IF closest is NULL:
      RETURN 999
    RETURN Manhattan distance from bot to closest

  FUNCTION home_distance(bot):
```



```

RETURN Manhattan distance from bot to its base

FUNCTION get_density(diamond, bot_pos):
    dist ← Manhattan distance from bot_pos to diamond position
    IF dist = 0:
        RETURN infinity
    RETURN diamond.points / dist

FUNCTION needed_steps(start, dest):
    RETURN Manhattan distance between start and dest

FUNCTION best_density_target(bot, board):
    IF no diamonds:
        RETURN NULL
    max_density ← 0
    best_target ← bot.base
    FOR each diamond in board.diamonds:
        density ← get_density(diamond, bot.position)
        IF density > max_density:
            max_density ← density
            best_target ← diamond.position
    IF teleporters exist (at least 2):
        marco, polo ← two closest teleporters to bot
        target_tele ← closer of marco or polo
        exit_tele ← other teleporter
        dist_to_tele ← Manhattan distance to target_tele
        FOR each diamond in board.diamonds:
            tele_density ← diamond.points / (dist_to_tele + distance
from exit_tele to diamond)
            IF tele_density > max_density:
                max_density ← tele_density
                best_target ← target_tele.position
    RETURN best_target

FUNCTION enemy_distance(bot, enemy):
    RETURN (x, y) distance from bot to enemy

FUNCTION find_target_enemies(bot, board):
    enemies ← empty list
    FOR each enemy in board.bots:
        IF enemy is not bot AND enemy is not at its base AND
        enemy has ≥3 diamonds AND more diamonds than bot:
            ADD enemy to enemies
    RETURN enemies

FUNCTION pursue_enemies(bot, board):
    IF bot is within 4 tiles of base AND pursuit_count ≤ 5:
        FOR each enemy in find_target_enemies(bot, board):
            dist ← enemy_distance(bot, enemy)
            IF dist = (0,0):
                target_spot ← bot.base
                RETURN FALSE
            IF |dist.x| ≤ 3 AND |dist.y| ≤ 3:
                target_spot ← enemy.position
                RETURN TRUE
    target_spot ← NULL

```

```

    pursuit_count ← 0
    using_portal ← FALSE
    RETURN FALSE

FUNCTION locate_red_switch(board):
    FOR each item in board.game_objects:
        IF item is DiamondButtonGameObject:
            RETURN item
    RETURN NULL

FUNCTION prefer_red_switch(bot, board):
    IF nearest_blue_gem exists:
        red_switch ← locate_red_switch(board)
        IF red_switch exists AND red_switch_distance <
blue_gem_distance:
            RETURN TRUE
    RETURN FALSE

FUNCTION red_switch_distance(bot, board):
    red_switch ← locate_red_switch(board)
    IF red_switch is NULL:
        RETURN 999
    RETURN Manhattan distance from bot to red_switch

FUNCTION find_all_portals(bot, board):
    portals ← list of TeleportGameObject in board
    SORT portals by Manhattan distance from bot
    RETURN portals

FUNCTION use_portal_to_base(bot, board):
    IF fewer than 2 portals:
        RETURN
    marco, polo ← two closest portals
    dist_to_base_marco ← distance from marco to base
    dist_to_base_polo ← distance from polo to base
    IF dist_to_base_marco = dist_to_base_polo:
        RETURN
    dist_to_bot ← distance from bot to marco
    IF dist_to_base_marco + dist_to_bot < home_distance(bot):
        using_portal ← TRUE
        target_spot ← marco.position

FUNCTION compute_path(current_x, current_y, dest_x, dest_y):
    delta_x ← |dest_x - current_x|
    delta_y ← |dest_y - current_y|
    x ← 1 if dest_x > current_x, ELSE -1
    y ← 1 if dest_y > current_y, ELSE -1
    IF delta_x ≥ delta_y:
        RETURN (x, 0)
    ELSE:
        RETURN (0, y)

FUNCTION next_move(bot, board):
    candidates ← empty list
    IF home_distance(bot) ≥ bot.milliseconds_left:
        ADD (bot.base, 999, "base_time") to candidates

```

```

ELSE IF (home_distance(bot) = 2 AND bot.diamonds > 2) OR
(home_distance(bot) = 1 AND bot.diamonds > 0) OR
bot.diamonds = 5:
    ADD (bot.base, 999, "base_inventory") to candidates
ELSE IF bot.diamonds ≥ 3:
    IF has_gems_near_base(bot, board):
        target ← nearest_gem_to_base(bot, nearby_gems(bot,
board))
        distance ← needed_steps(bot.position, target)
        ADD (target, 1/distance, "near_base") to candidates
    ELSE IF nearest_blue_gem OR nearest_red_gem exists:
        IF bot.diamonds = 3 AND red_gem_distance ≤ 3:
            target ← nearest_red_gem
            distance ← red_gem_distance
            ADD (target, 2/distance, "red_gem") to candidates
        ELSE IF blue_gem_distance ≤ 3:
            target ← nearest_blue_gem
            distance ← blue_gem_distance
            ADD (target, 1/distance, "blue_gem") to candidates
        ELSE:
            ADD (bot.base, 999, "base_fallback") to candidates
    ELSE:
        ADD (bot.base, 999, "base_fallback") to candidates
ELSE:
    IF (has_gems_near_base AND is_near_home) OR
(has_gems_near_base AND ≥3 gems near base):
        target ← nearest_gem_to_base(bot, nearby_gems(bot,
board))
        distance ← needed_steps(bot.position, target)
        ADD (target, 1/distance, "near_base") to candidates
    IF pursue_enemies(bot, board):
        pursuit_count ← pursuit_count + 1
        ADD (target_spot, 2, "enemy") to candidates
    IF prefer_red_switch(bot, board):
        red_switch ← locate_red_switch(board)
        distance ← red_switch_distance(bot, board)
        ADD (red_switch.position, 1.5/distance, "red_switch") to
candidates
    IF nearest_red_gem exists:
        red_distance ← red_gem_distance(bot, board)
        ADD (nearest_red_gem, 2/red_distance, "red_gem") to
candidates
    IF nearest_blue_gem exists:
        blue_distance ← blue_gem_distance(bot, board)
        ADD (nearest_blue_gem, 1/blue_distance, "blue_gem") to
candidates
    density_target ← best_density_target(bot, board)
    IF density_target:
        FOR each diamond in board.diamonds:
            IF diamond.position = density_target:
                density ← get_density(diamond, bot.position)
                ADD (density_target, density, "density") to
candidates
                BREAK
        ELSE IF density_target is teleporter:
            marco, polo ← two closest portals

```

```

        target_tele ← closer of marco or polo
        exit_tele ← other portal
        FOR each diamond in board.diamonds:
            tele_density ← diamond.points / (distance to
target_tele + distance from exit_tele to diamond)
            IF tele_density > 0:
                ADD (density_target, tele_density,
"density_tele") to candidates
            IF candidates is empty:
                ADD (bot.base, 999, "base_fallback") to candidates
            best_candidate ← max(candidates, prioritize "base_time",
"base_inventory", "base_fallback", then highest score)
            target_spot ← best_candidate.position
            IF target_spot = bot.base AND NOT using_portal:
                use_portal_to_base(bot, board)
            RETURN compute_path(bot.position.x, bot.position.y,
target_spot.x, target_spot.y)
ENDCLASS

```

B. Penjelasan Struktur Data

Class MahaSantuyLogic memiliki atribut dan method sebagai berikut:

No.	Atribut	Tipe Data	Penjelasan
1.	move_vectors	List[Tuple(integer, integer)]	Daftar vektor gerakan untuk menentukan arah bot
2.	target_spot	Optional[Position]	Posisi target yang akan dituju
4.	pursuit_count	integer	Untuk jumlah langkah pengejaran musuh
5.	using_portal	boolean	Penanda penggunaan portal

Tabel 2.1 Penjelasan Struktur Data

No.	Fungsi	Penjelasan
1.	__init__	Konstruktor kelas
2.	nearby_gems	Mengembalikan daftar posisi berlian dalam

		radius 3 petak dari base
3.	is_near_home	Memeriksa apakah bot berada dalam radius 3 petak dari base
4.	nearest_gem_to_base	Menemukan posisi diamond biru terdekat dari bot
5.	has_gems_near_base	Memeriksa apakah ada berlian dalam radius 2 petak dari base
6.	nearest_blue_gem	Menemukan posisi diamond biru terdekat dari bot
7.	blue_gem_distance	Menghitung jarak Manhattan ke diamond biru terdekat
8.	nearest_red_gem	Menemukan posisi diamond merah terdekat ke bot
9.	red_gem_distance	Menghitung jarak Manhattan ke diamond merah terdekat
10.	home_distance	Menghitung jarak Manhattan dari bot ke base
11.	get_density	Menghitung (poin/jarak) diamond
12.	needed_steps	Menghitung jarak Manhattan antara dua posisi
13.	best_density_target	Menemukan target terbaik berdasarkan density
14.	enemy_distance	Menghitung jarak (x, y) dari bot ke bot musuh
15.	find_target_enemies	Menemukan musuh dengan ≥ 3 diamond dan lebih banyak diamond dari bot
16.	pursue_enemies	Mengejar musuh dalam radius 3 petak jika bot dekat base
17.	locate_red_switch	Menemukan posisi tombol merah
18.	prefer_red_switch	Memeriksa apakah tombol merah lebih dekat dari diamond
19.	red_switch_distance	Menghitung jarak Manhattan ke tombol

		merah
20.	find_all_portals	Menemukan lokasi dari semua teleporter
21.	use_portal_to_base	Memeriksa apakah teleporter lebih cepat untuk kembali ke base
22.	compute_path	Menghitung arah gerakan berikutnya dengan pola zigzag menuju target
23.	next_move	Menentukan langkah berikutnya berdasarkan strategi greedy

Tabel 2.2 Penjelasan Struktur Data

C. Analisis Solusi Pengujian

Pengujian terhadap algoritma dilakukan dengan dua skema. Skema pertama melakukan pengujian dengan melakukan tes keoptimalan bot dalam mengambil diamond dalam waktu 60 detik tanpa musuh. Berikut adalah hasil pengujian yang telah dilakukan.

Percobaan	Diamond yang diperoleh
1	10
2	13
3	12
4	11
5	17
Rata-rata	12.6

Tabel 2.3 Analisis Pengujian

Berdasarkan hasil tersebut, dapat disimpulkan bahwa bot bekerja dengan cukup optimal dalam mengumpulkan diamond. Saat pengujian, bot juga berhasil melewati tes tanpa ada bug sama sekali, bot juga mampu memanfaatkan teleport dan tombol merah dengan baik.

Skema kedua melakukan pengujian melawan bot lain. Pengujian ini akan dilakukan dengan cara memainkan bot bersamaan dengan 3 bot lain dengan nama mohosantoy, tercepat, dan killer. Berikut adalah hasil pengujian yang dilakukan.

Percobaan	Diamond yang diperoleh			
	mahasantuy	mohosantoy	tercepat	killer
1	14	11	5	5
2	14	8	0	6
3	12	5	5	2
4	10	10	6	7
5	15	10	8	8
Rata-rata	13	8.8	4.8	5.6

Tabel 2.4 Analisis Pengujian

Berdasarkan hasil yang didapatkan, bot berjalan dengan sangat optimal dan berhasil mengambil diamond paling banyak dengan rata-rata nya 13 diamond.

BAB V

Kesimpulan

MahaSantuy adalah bot yang kami buat dengan menggunakan pendekatan Multi-Strategy yang memungkinkan bot untuk beradaptasi secara cepat dalam setiap perubahan kondisi permainan, seperti lokasi permata, musuh, teleporter, dan waktu. Pengujian yang kami lakukan menunjukkan kesimpulan :

1. Fleksibilitas Yang Tinggi

Dengan menggabungkan berbagai strategi greedy seperti densitas, base_time, inventory, hingga red_gem, bot menjadi lebih adaptif. Bot tidak lagi terpaku pada satu jenis keputusan, melainkan dapat memilih strategi yang paling menguntungkan berdasarkan kondisi saat itu.

2. Efektivitas Terbaik pada Kombinasi Strategis

Strategi seperti **greedy by density** memberikan hasil paling stabil secara skor dan efisiensi. Namun, dalam kondisi tertentu seperti waktu hampir habis atau inventory penuh, strategi seperti **base_time** dan **base_inventory** memberikan *fallback* yang kuat untuk menjaga keberhasilan pulang.

3. Kelemahan Masih Ada pada Risiko & Ketidakpastian

Strategi seperti **greedy by red_gem** dan **enemies** cenderung meningkatkan resiko kegagalan, terutama tanpa logika prediktif terhadap musuh atau reward yang belum pasti. Selain itu, saat terjadi *edge case* seperti tackle berturut-turut, error logic masih muncul dan menurunkan stabilitas.

4. Efisiensi Bot Stabil di Mode Single Player dan Multiplayer

Saat tidak ada musuh, Mahasantuy dapat fokus pada pengumpulan permata, terbukti dengan skor rata-rata yang tinggi. Dan saat ada musuh, bot dapat memilih pilihan yang paling efisien antara mengambil diamonds, teleporter atau melakukan tackle.

Saran

Jika waktu pengembangan mencukupi, masih terdapat sejumlah peningkatan yang dapat diimplementasikan untuk meningkatkan performa bot dalam menyelesaikan permainan :

- 1. Evaluasi Teleporter untuk Semua Target:** Selain untuk kembali ke base, periksa apakah teleporter dapat mengurangi jarak ke berlian tertentu (baik biru maupun merah). Misalnya, hitung jarak total (bot ke teleporter + teleporter ke berlian) dan bandingkan dengan jarak langsung ke berlian.
- 2. Fallback untuk Peta Kosong:** Jika tidak ada berlian (board.diamonds kosong), pastikan bot langsung kembali ke base atau menekan tombol merah (jika ada).
- 3. Kombinasi dengan Berlian:** Jika bot mengejar musuh, periksa apakah ada berlian di jalur pengejaran. Jika ada, bot bisa mengambil berlian sekaligus mengejar, meningkatkan efisiensi.

LAMPIRAN

1. Link GitHub : https://github.com/Arcel-S/Tubes1_MahaSantuy
2. Link Youtube : bit.ly/Tubes1_MahaSantuy

DAFTAR PUSTAKA

- Munir, Rinaldi. 2024. Algoritma Greedy (Bagian 1). [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). Diakses pada 28 Mei 2025.
- Munir, Rinaldi. 2024. Algoritma Greedy (Bagian 2). [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). Diakses pada 28 Mei 2025.
- Munir, Rinaldi. 2024. Algoritma Greedy (Bagian 3). [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-\(2022\)-Bag3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf). Diakses pada 28 Mei 2025.