

ALPHABOT – IMPLEMENTASI ALGORITMA GREEDY BERBASIS DENSITY DALAM PEMECAHAN BOT PERMAINAN DIAMOND

Tugas Besar

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



Oleh: Kelompok 4 (α code)

M. Zahran Dhiyaul Haq 123140120

Yuni Okta Safitri 123140213

Audy Olivya 123140025

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

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

2025

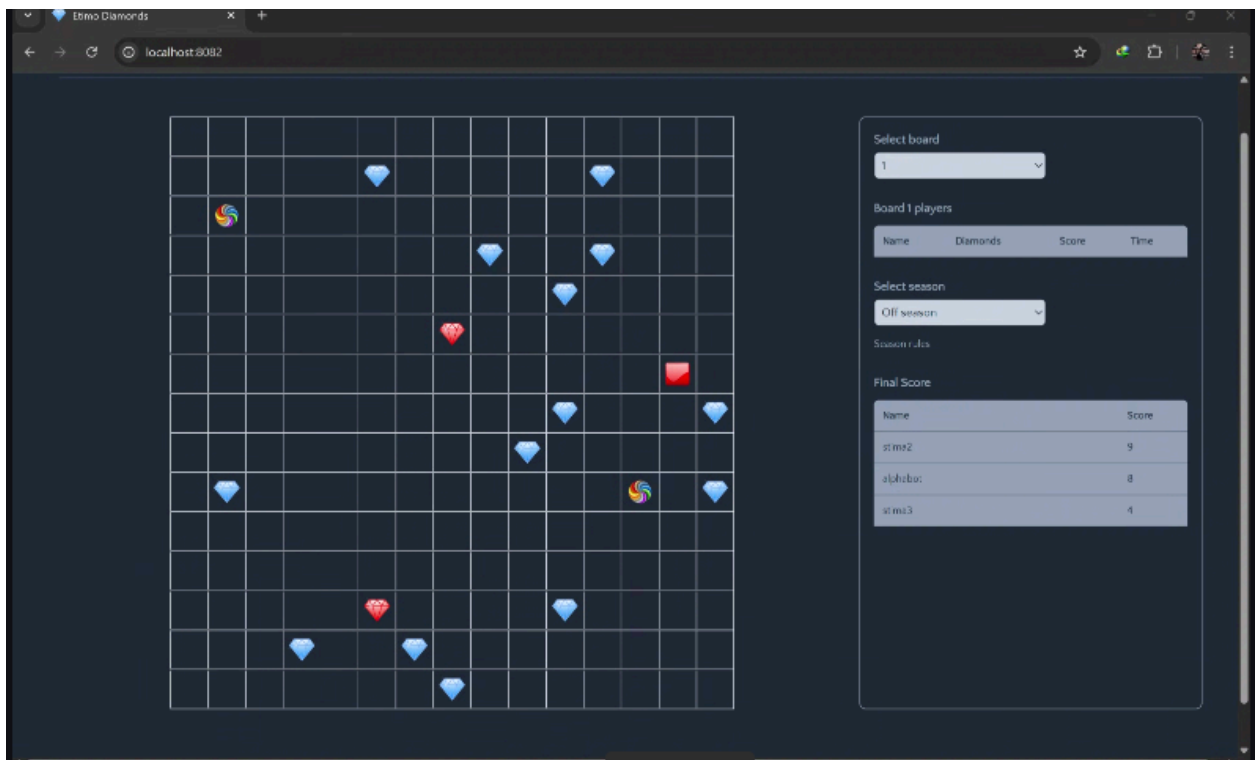
DAFTAR ISI

BAB I.....	1
DESKRIPSI TUGAS.....	1
BAB II.....	7
LANDASAN TEORI.....	7
2.1 Dasar Teori.....	7
2.2 Cara Kerja Program.....	7
2.3 Cara kerja algoritma greedy secara lebih rinci:.....	7
2.4 Cara Implementasi Program.....	8
2.5 Menjalankan Bot Program.....	9
BAB III.....	10
APLIKASI STRATEGI GREEDY.....	10
3.1 Proses Mapping.....	10
3.2 Eksplorasi Alternatif Solusi Greedy.....	12
3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy.....	13
3.4 Strategi Greedy yang Dipilih.....	13
BAB IV.....	15
IMPLEMENTASI DAN PENGUJIAN.....	15
4.1 Implementasi Algoritma Greedy.....	15
4.2 Struktur Data yang Digunakan.....	16
4.3 Pengujian Program.....	16
BAB V.....	18
KESIMPULAN DAN SARAN.....	18
5.1 Kesimpulan.....	18
5.2 Saran.....	18
LAMPIRAN.....	20
DAFTAR PUSTAKA.....	21

BAB I

DESKRIPSI TUGAS

Diamonds merupakan suatu *programming challenge* adu strategi antar bot untuk mengumpulkan diamond sebanyak-banyaknya di dalam sebuah papan permainan (board). Setiap pemain membuat sebuah bot yang akan bersaing dengan bot lainnya. Pada tugas ini, bot yang dibuat harus menggunakan strategi greedy, yaitu selalu memilih langkah yang paling menguntungkan saat itu juga.



Gambar 1.1 Layar Permainan Diamonds

Pada tugas awal mata kuliah Strategi Algoritma ini, setiap mahasiswa diminta untuk merancang dan mengembangkan sebuah bot yang nantinya akan dipertandingkan satu sama lain. Dalam pembuatan bot ini, mahasiswa wajib menerapkan pendekatan algoritma greedy sebagai strategi dasar pengambilan keputusan.

Program permainan *Diamonds* terdiri dari dua komponen utama, yaitu:

1. Game Engine :Komponen inti permainan yang mencakup:

- Logika Backend Permainan:

Bagian ini menangani seluruh aturan dan mekanisme permainan secara menyeluruh. Selain itu, ia menyediakan API (Application Programming Interface) yang digunakan untuk berkomunikasi dengan antarmuka pengguna (frontend) dan juga dengan bot buatan peserta.

- Visualisasi Frontend Permainan:

Komponen ini bertanggung jawab untuk menampilkan permainan secara grafis, sehingga jalannya permainan bisa diamati oleh pengguna.

2. Paket Awal Bot (Bot Starter Pack) :Paket ini diberikan sebagai titik awal untuk membangun bot, dan biasanya terdiri atas

- Script Pemanggil API:

Program yang digunakan untuk mengakses dan memanfaatkan API yang disediakan oleh backend.

- Logika Bot (yang akan dikembangkan):

Ini adalah bagian kode tempat mahasiswa mengimplementasikan strategi greedy yang menjadi inti dari bot mereka.

- Program Utama dan Alat Pendukung:

Termasuk file main untuk menjalankan bot, serta berbagai fungsi bantu (utility) yang diperlukan untuk mendukung operasi bot.

Komponen utama dalam permainan ini adalah sebagai berikut:

1. Diamond

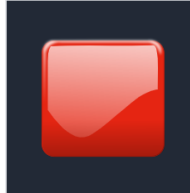
Terdapat dua jenis diamond yaitu merah bernilai 2 poin dan biru bernilai 1 poin



Gambar 1.2 Varian Diamond Biru dan Merah

Diamond akan terus muncul kembali secara berkala dengan jumlah dan warna yang bisa berubah.

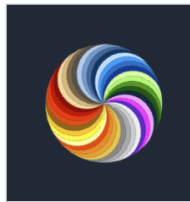
2. Red Button



Gambar 1.3 Red Button / Diamond Button

Jika bot menginjak tombol ini, semua diamond akan diacak ulang di board dan tombol ini juga pindah ke posisi baru.

3. Teleporter



Gambar 1.4 Teleporters

Ada dua teleporter yang saling terhubung. Jika bot masuk ke salah satunya, maka akan langsung muncul di teleporter yang lain.

4. Bot



Gambar 1.5 Bot Nama "Stima"

Setiap bot memiliki base untuk menyimpan diamond yang telah dikumpulkan. Jika bot menyimpan diamond ke base, maka poin akan bertambah sesuai jumlah diamond, dan inventory bot dikosongkan.

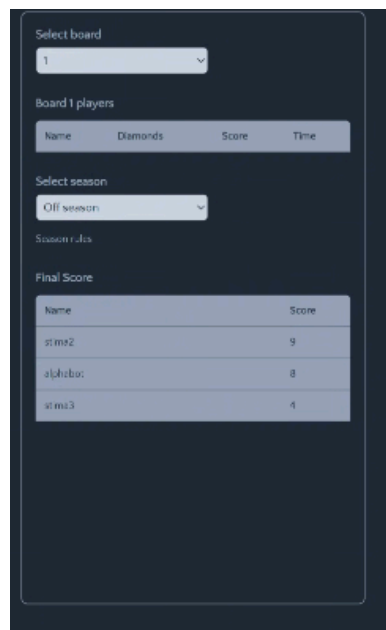
5. Base



Gambar 1.6 Bot Nama “Stima”

Setiap bot memiliki sebuah markas (Base) yang berperan sebagai tempat penyimpanan diamond yang berhasil dikumpulkan. Saat bot berhasil menaruh diamond ke dalam Base, maka skor bot akan bertambah sesuai dengan nilai dari diamond tersebut. Setelah proses penyimpanan dilakukan, isi inventaris bot akan dikosongkan atau di-reset, artinya bot tidak lagi membawa diamond hingga mengumpulkan kembali.

6. Inventory

The image is a screenshot of a mobile application interface with a dark blue background. It contains several sections: a 'Select board' dropdown menu with '1' selected; a 'Board 1 players' section with a table; a 'Select season' dropdown menu with 'Off season' selected; a 'Season rules' section; and a 'Final Score' section with a table. The tables have a light blue header and white text on a dark blue background.

Name	Diamonds	Score	Time
------	----------	-------	------

Name	Score
stima2	9
alshubor	8
stima3	1

Gambar 1.7 Layar Inventory

Inventory adalah tempat sementara untuk menyimpan diamond yang diambil. Kapasitasnya terbatas, jadi bot harus sering kembali ke base agar tidak kelebihan muatan.

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

1. **Inisialisasi Permainan**, saat permainan dimulai, setiap bot ditempatkan secara acak di papan permainan. Setiap bot memiliki markas (home base) sendiri, skor awal nol, dan inventory yang masih kosong.
2. **Tujuan Permainan**, tujuan utama bot adalah mengumpulkan diamond sebanyak mungkin. Diamond merah bernilai 2 poin, sedangkan biru bernilai 1 poin. Diamond yang diambil akan disimpan sementara di inventory.
3. **Pergerakan Bot**, bot bergerak secara berkala sesuai waktu yang ditentukan. Bot dapat bergerak menuju diamond, home base, teleporter, atau tombol merah, sesuai dengan strategi logika yang digunakan seperti RandomLogic atau AlphaBot.
4. **Inventory dan Skor**, inventory memiliki batas kapasitas. Jika penuh, bot harus kembali ke home base untuk menambahkan poin dan mengosongkan inventory.
5. **Mekanisme Tackle**, jika bot A menabrak bot B (berada di posisi yang sama), maka bot B dikirim ke home base dan semua diamond-nya diambil oleh bot A.
6. **Objek Tambahan**, terdapat objek seperti teleporter untuk berpindah tempat dan red button yang bisa memicu efek tertentu. Bot bisa memanfaatkannya sebagai bagian dari strategi.
7. **Akhir Permainan**, permainan berakhir saat waktu bot habis. Sistem akan menampilkan skor akhir dan bot dengan poin tertinggi akan menjadi pemenang.

Mekanisme Teknis Permainan Diamonds

1. Pemeriksaan Bot Terdaftar

Memeriksa apakah bot sudah terdaftar dalam sistem. Program akan mengirimkan permintaan POST ke endpoint `/api/bots/recover` dengan menyertakan email dan password. Jika bot sudah terdaftar, server akan merespons dengan status 200 OK beserta ID bot. Jika tidak ditemukan, server akan mengembalikan status 404 Not Found.

2. Pendaftaran Bot Baru

Jika bot belum terdaftar, maka program akan mendaftarkan bot baru dengan mengirimkan permintaan POST ke endpoint `/api/bots`, yang berisi data seperti email, nama, password, dan tim. Jika pendaftaran berhasil, server akan memberikan status 200 beserta ID unik sebagai identitas bot tersebut.

3. Bot Bergabung ke Board

Setelah ID bot tersedia, bot akan mencoba masuk ke dalam papan permainan tertentu dengan mengirimkan permintaan POST ke endpoint `/api/bots/{id}/join`, disertai ID board yang ingin dimasuki. Jika berhasil, server akan memberikan informasi detail mengenai kondisi board saat ini.

4. Pergerakan Bot Selama Permainan

Selama permainan berlangsung, bot akan terus menghitung langkah berikutnya berdasarkan situasi papan. Untuk melakukan gerakan, bot akan mengirimkan permintaan POST ke `/api/bots/{id}/move` dengan arah gerakan seperti “NORTH”, “SOUTH”, “EAST”, atau “WEST”. Jika gerakan valid, server akan merespons dengan kondisi board terbaru. Proses ini berulang sampai waktu bot habis atau bot keluar dari permainan.

5. Pembaruan Tampilan Board (Frontend)

Agar tampilan permainan tetap sinkron, sisi frontend secara berkala akan mengirimkan permintaan GET ke endpoint `/api/boards/{id}`. Permintaan ini digunakan untuk mengambil kondisi terbaru papan permainan agar tampilan pengguna selalu sesuai dengan situasi sebenarnya.

BAB II

LANDASAN TEORI

2.1 Dasar Teori

Pengertian Algoritma Greedy adalah jenis algoritma yang membentuk solusi langkah per langkah dengan mencari nilai maksimum sementara pada setiap langkahnya. Nilai maksimum sementara ini dikenal dengan istilah local maximum. Pada kebanyakan kasus, algoritma greedy tidak akan menghasilkan solusi paling optimal, begitupun algoritma greedy biasanya memberikan solusi yang mendekati nilai optimum dalam waktu yang cukup cepat. Algoritma greedy merupakan metode yang paling populer untuk memecahkan persoalan optimasi. Persoalan optimasi (optimization problems): persoalan mencari solusi optimum. Hanya ada dua macam persoalan optimasi:

1. Maksimasi (maximization)
2. Minimasi (minimization)

2.2 Cara Kerja Program

Program ini dirancang untuk menyelesaikan suatu masalah tertentu menggunakan pendekatan algoritma greedy. Dalam kasus ini, bot akan menjalankan aksi-aksi yang bertujuan mencapai hasil optimal berdasarkan kondisi yang terus diperbarui. Bot mengambil keputusan dengan memilih aksi terbaik yang tersedia pada setiap langkah tanpa meninjau kembali langkah sebelumnya. Pendekatan ini sesuai dengan prinsip greedy, yaitu selalu memilih solusi lokal terbaik. Program terdiri dari beberapa bagian utama, yaitu:

1. Proses pengambilan keputusan oleh bot
2. Implementasi algoritma greedy dalam kode program
3. Proses eksekusi bot untuk menjalankan strategi greedy secara otomatis

2.3 Cara kerja algoritma greedy secara lebih rinci:

1. Identifikasi masalah: Algoritma greedy digunakan untuk menyelesaikan masalah optimasi, yaitu masalah di mana tujuan utamanya adalah menemukan solusi terbaik (misalnya, jalur terpendek, biaya minimum, atau nilai maksimum).

2. Pembuatan keputusan lokal: Pada setiap langkah, algoritma greedy memilih pilihan yang dianggap terbaik secara lokal. Ini berarti pilihan yang memberikan hasil terbaik untuk langkah tersebut, tanpa mempertimbangkan konsekuensi jangka panjang.
3. Penghimpunan solusi: Pilihan-pilihan lokal tersebut kemudian dikumpulkan untuk membentuk solusi. Solusi ini belum tentu optimal secara global, tetapi algoritma greedy berusaha untuk membuat setiap langkah memberikan kontribusi positif terhadap solusi akhir.
4. Evaluasi solusi: Setelah solusi terbentuk, algoritma greedy dapat mengevaluasi kualitas solusi tersebut. Jika solusinya tidak optimal, maka algoritma mungkin perlu diadaptasi atau menggunakan algoritma lain untuk mendapatkan hasil yang lebih baik.

2.4 Cara Implementasi Program

Program ini bekerja dengan cara memodelkan setiap langkah proses sebagai sebuah keadaan (state). Dari setiap keadaan ini, program akan mengevaluasi semua aksi (langkah yang bisa diambil) yang memungkinkan. Setiap aksi akan dianalisis berdasarkan nilai keuntungan atau biaya yang ditimbulkan, dan program akan memilih aksi terbaik menurut kriteria tertentu, seperti waktu tercepat, hasil paling optimal, atau jarak terpendek.

Langkah-langkah kerja program:

- a. Menganalisis keadaan saat ini (current state), bot atau program melihat situasi sekarang untuk memahami posisi atau kondisi yang sedang dihadapi.
- b. Mengidentifikasi semua aksi yang mungkin dilakukan, bot menentukan semua pilihan atau langkah yang bisa diambil dari kondisi tersebut.
- c. Mengevaluasi setiap aksi, untuk setiap aksi yang mungkin, bot menghitung nilai (misalnya: biaya, waktu, atau hasil) menggunakan fungsi evaluasi tertentu.
- d. Memilih aksi terbaik, bot memilih aksi yang memberikan hasil terbaik secara lokal (di langkah saat ini), berdasarkan kriteria yang telah ditentukan.
- e. Menjalankan aksi dan memperbarui keadaan, bot melaksanakan aksi terpilih dan memperbarui keadaan menjadi keadaan baru setelah aksi dilakukan.

- f. Mengulangi proses, langkah-langkah di atas terus diulang sampai, tujuan akhir (goal state) tercapai, atau tidak ada lagi aksi yang bisa dilakukan (semua pilihan sudah habis atau kondisi terjebak).

2.5 Menjalankan Bot Program

Untuk menjalankan program bot, langkah-langkah umumnya adalah sebagai berikut:

- a. Inisialisasi data: Program membaca data awal (misalnya peta, objek, nilai, atau kondisi awal).
- b. Eksekusi algoritma greedy: Bot mulai dari titik awal dan menjalankan logika greedy untuk memilih langkah terbaik di setiap iterasi.
- c. Proses loop: Bot terus bergerak atau beraksi sampai kondisi selesai terpenuhi.
- d. Output hasil: Setelah selesai, program menampilkan hasil seperti jalur yang diambil, total nilai, atau waktu yang dibutuhkan.

Bot dapat dijalankan langsung melalui lingkungan pengembangan atau terminal, tergantung pada bahasa dan platform yang digunakan.

2.6 Kelebihan dan Kekurangan Algoritma Greedy

Adapun kelebihan dari algoritma greedy adalah sebagai berikut:

- a. Sederhana dan Cepat: Algoritma greedy relatif mudah dipahami dan diimplementasikan, dan sering kali memiliki kinerja yang cepat.
- b. Solusi Terdekat: Algoritma ini cenderung menghasilkan solusi yang cukup mendekati optimal dalam waktu yang singkat.

Kelemahan dari algoritma greedy adalah sebagai berikut:

- a. Tidak Selalu Optimal: Algoritma greedy tidak selalu menghasilkan solusi optimal. Ada kasus di mana solusi yang dihasilkan jauh dari solusi terbaik.
- b. Pemilihan Kriteria: Kegagalan dalam pemilihan kriteria rakus yang tepat dapat mengakibatkan hasil yang tidak optimal.
- c. Pengabaian Konsekuensi Masa Depan: Algoritma ini tidak mempertimbangkan konsekuensi jangka panjang dari setiap langkah, sehingga dapat menghasilkan solusi yang suboptimal dalam beberapa kasus.

BAB III

APLIKASI STRATEGI *GREEDY*

3.1 Proses *Mapping*

Berdasarkan studi pustaka yang telah dijelaskan pada landasan teori, algoritma greedy melibatkan pencarian suatu himpunan bagian (S) dari himpunan kandidat (C) yang dipilih berdasarkan fungsi seleksi dan kelayakan untuk mencapai solusi optimal. Dalam game Diamonds, strategi greedy digunakan oleh AlphaBot untuk memutuskan langkah terbaik dalam mengumpulkan diamond dan memperoleh skor maksimum. Maka dari itu, elemen-elemen dalam game harus dipetakan terlebih dahulu ke dalam komponen-komponen algoritma greedy sebagai berikut:

1. **Himpunan Kandidat (C):** Himpunan kandidat dalam konteks ini adalah seluruh objek dalam peta game yang dapat dipilih sebagai target, yaitu:
 - a. Blue diamond
 - b. Red diamond
 - c. Red button (untuk regenerasi diamond)
 - d. Teleporters
 - e. Base bot
 - f. Bot musuh
 - g. Base musuh

Selain itu, posisi musuh dan posisi diamond menjadi informasi penting yang mempengaruhi pemilihan langkah. Bot juga mempertimbangkan jalur tercepat dan kemungkinan terhindar dari tackle oleh musuh.

2. Himpunan Solusi (S)

Himpunan solusi adalah koordinat-koordinat tujuan yang dipilih oleh bot untuk dikunjungi, dengan harapan memberikan poin tertinggi dalam waktu tercepat. Dalam pendekatan greedy, AlphaBot memilih satu langkah terbaik setiap saat, sehingga biasanya himpunan solusi hanya berisi satu posisi diamond yang diprioritaskan berdasarkan

density (nilai dibagi jarak). Namun, jika kondisi tertentu terpenuhi (inventory penuh atau waktu hampir habis), maka koordinat base akan menjadi elemen dalam himpunan solusi.

3. Fungsi Solusi

Fungsi solusi digunakan untuk menilai apakah keputusan yang diambil merupakan solusi yang layak dalam konteks akhir permainan. AlphaBot memeriksa apakah waktu tersisa mencukupi untuk kembali ke base ketika membawa diamond. Perhitungan ini menjadi fungsi solusi karena menentukan validitas keputusan greedy yang diambil sebelumnya. Fungsi ini hanya dievaluasi saat:

- a. Inventory sudah penuh
- b. Waktu tersisa < 10 detik
- c. Tidak ada diamond tersisa

4. Fungsi Seleksi (Selection)

Fungsi seleksi digunakan untuk menentukan kandidat terbaik dari himpunan kandidat yang ada. Dalam implementasi AlphaBot, fungsi seleksi dilakukan berdasarkan nilai density:

$$\text{Density} = \frac{\text{nilai diamond} (\times 1.5 \text{ jika red})}{\text{jarak terpendek (langsung atau via teleporter)}}$$

$$\text{Density} = \frac{\text{jarak terpendek (langsung atau via teleporter)}}{\text{nilai diamond} (\times 1.5 \text{ jika red})}$$

Kandidat dengan nilai density tertinggi akan dipilih sebagai langkah selanjutnya. Sorting dilakukan untuk mengurutkan diamond dengan density tertinggi terlebih dahulu.

5. Fungsi Kelayakan (Feasibility)

Fungsi ini digunakan untuk memastikan apakah kandidat yang dipilih masih valid untuk dijadikan tujuan. Misalnya:

- a. Jika inventory sudah penuh, maka diamond tidak bisa diambil → tujuan dialihkan ke base.
- b. Jika waktu tidak cukup untuk mengambil diamond dan kembali ke base, maka langsung kembali ke base.
- c. Jika diamond habis di peta, maka bot menuju red button.

Dengan fungsi ini, AlphaBot menghindari mengambil keputusan yang sia-sia atau membahayakan skor.

3.2 Eksplorasi Alternatif Solusi Greedy

Dalam penyelesaian masalah pada permainan Diamonds, sebenarnya terdapat beberapa pendekatan strategi lain selain greedy. Namun, karena pada tugas besar ini diwajibkan menggunakan pendekatan greedy, maka alternatif strategi hanya digunakan sebagai bahan perbandingan teoritis. Beberapa strategi alternatif yang dapat digunakan antara lain:

1. Algoritma A*

Pendekatan ini memperhitungkan jalur terpendek ke banyak titik sekaligus, dan menyusun rencana langkah optimal dari awal hingga akhir permainan. A* cocok jika kita ingin membuat perencanaan jangka panjang dengan prediksi perubahan posisi objek.

Kelebihan: Menghasilkan jalur optimal secara keseluruhan. Dapat menghindari musuh secara lebih sistematis. Kekurangan: Membutuhkan waktu komputasi yang jauh lebih besar. Tidak responsif terhadap perubahan objek dalam game (misal regenerasi diamond).

2. Minimax dengan Evaluasi Skor

Strategi ini mempertimbangkan langkah kita sekaligus memprediksi langkah musuh dan dampaknya terhadap skor, seperti pada permainan catur.

Kelebihan: Cocok untuk menghadapi musuh yang sangat agresif. Dapat mencegah tackle dan menjaga keunggulan skor. Kekurangan: Kompleksitas tinggi. Tidak cocok jika waktu langkah sangat terbatas.

3. Greedy dengan Bobot Dinamis

AlphaBot sendiri menggunakan pendekatan greedy berbasis density. Namun pendekatan greedy juga dapat dikembangkan lebih jauh, misalnya dengan memberikan bobot tambahan yang dinamis terhadap:

- a. Jarak ke base
- b. Potensi musuh berada di jalur yang sama
- c. Probabilitas regenerasi diamond
- d. Pengembangan greedy ini bisa memberikan hasil yang lebih adaptif dibanding greedy murni.

3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy

1. Efisiensi Waktu

Strategi greedy memiliki kompleksitas rendah karena hanya memeriksa satu langkah terbaik pada setiap iterasi. AlphaBot menghitung density untuk semua diamond, lalu memilih langkah terbaik. Dalam implementasinya, sorting dan pengambilan kandidat dilakukan dalam waktu konstan terhadap jumlah objek. Hal ini membuat AlphaBot sangat efisien untuk dijalankan dalam sistem dengan batas waktu langkah yang ketat.

2. Efektivitas Strategi

Meskipun tidak menjamin solusi global optimal, greedy terbukti efektif dalam konteks permainan cepat seperti Diamonds karena:

- a. Bot mampu merespons kondisi permainan secara real-time.
- b. Risiko tackle oleh musuh dapat diminimalisir dengan menghindari posisi lawan.

Strategi fallback (kembali ke base, menekan red button, eksplorasi acak) menjaga agar bot tetap aktif meskipun peta minim diamond.

3. Kelebihan Strategi Greedy pada AlphaBot

Mengutamakan skor tinggi (red diamond berbobot $1.5\times$). Menghindari musuh tanpa harus melacak jalur mereka. Adaptif terhadap posisi diamond dan regenerasi melalui red button.

4. Kelemahan Strategi

Tidak memperhitungkan rencana jangka panjang. Tidak memperhitungkan pergerakan musuh secara prediktif. Rentan terhadap perubahan mendadak seperti munculnya diamond baru di posisi jauh yang lebih menguntungkan.

3.4 Strategi Greedy yang Dipilih

Strategi greedy yang diterapkan pada bot AlphaBot berfokus pada prinsip greedy berdasarkan density, yaitu memilih objek diamond yang memiliki rasio nilai terhadap jarak (density) tertinggi. Strategi ini mengasumsikan bahwa setiap langkah harus memberikan nilai maksimal dalam waktu sesingkat mungkin.

1. Perhitungan Density

$$\text{Density} = \frac{\text{Skor Diamond} \times \text{Bobot Warna}}{\text{Jarak Terpendek}}$$

- a. Skor Diamond: 2 untuk red diamond, 1 untuk blue diamond
 - b. Bobot Warna: Red diamond diberi bobot $1.5\times$ untuk meningkatkan prioritas
 - c. Jarak Terpendek: Dihitung sebagai minimum antara jalur langsung ke diamond atau melalui teleporter
 - d. Semakin tinggi density, semakin prioritas diamond tersebut untuk diambil.
2. Prioritas Pengambilan
Bot akan memilih diamond dengan density tertinggi yang masih aman dari jangkauan musuh. Jika jarak musuh ke diamond lebih dekat dari jarak bot, diamond tersebut diabaikan.
 3. Penghindaran Musuh
AlphaBot memiliki fitur untuk menghindari posisi lawan. Dalam fungsi `get_direction_Adv`, bot akan mencari arah alternatif jika langkah terbaik menuju target berada pada posisi musuh atau berisiko tertackle.
 4. Manajemen Inventory
Bot hanya mengambil diamond jika: Inventory belum penuh. Masih ada cukup waktu untuk kembali ke base. Jika inventory penuh atau waktu hampir habis (misalnya <10 detik), maka bot akan segera kembali ke base untuk menyimpan diamond yang dibawa.
 5. Fallback Strategy
Jika tidak ada diamond yang tersedia atau semua lokasi berisiko tinggi, bot akan menekan Red Button untuk mereset dan regenerate posisi diamond. Jika tidak memungkinkan, bot akan menjelajahi area sekitarnya secara acak agar tetap aktif dan mencari peluang baru.
 6. Penggunaan Teleporter
Bot mempertimbangkan penggunaan teleporter jika jalur melalui teleporter memiliki total jarak lebih pendek daripada jalur langsung ke diamond.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma Greedy

1. Pseudocode

```
class GreedyDiamondLogic(BaseLogic: BaseLogic)
{ kelas utama untuk menjalankan strategi greedy mengambil diamond }

KAMUS LOKAL
static static_goals: array of Position
static static_goal_teleport: GameObject
static static_temp_goals: Position
static static_direct_to_base_via_teleporter: boolean
procedure _init_ ()
function next_move(board_bot: GameObject, board: Board) -> (integer, integer)
function calculate_near_base() -> boolean
procedure find_nearest_diamond()
function find_nearest_red_button() -> (integer, Position)
function find_nearest_teleport() -> (Position, Position, GameObject)
function find_other_teleport(teleport: GameObject) -> Position
function find_nearest_diamond_teleport() -> (real, (Position, Position), GameObject)
function find_nearest_diamond_direct() -> (real, Position)
```

2. Penjelasan Alur Program

Program akan mencari diamond terdekat dari posisi bot. Jika diamond tidak bisa diambil (misalnya karena inventory penuh), maka program akan mengecek apakah perlu kembali ke base atau menggunakan teleporter. Dalam setiap langkahnya, bot akan memilih aksi yang memberikan keuntungan terbesar berdasarkan nilai density (nilai/biaya) dengan pendekatan greedy.

Proses pencarian langkah selanjutnya dilakukan dalam `next_move`, yang akan mempertimbangkan semua kemungkinan dan memilih koordinat terbaik berdasarkan perhitungan jarak dan ketersediaan waktu

4.2 Struktur Data yang Digunakan

- a. List: Digunakan untuk menyimpan kumpulan objek seperti tombol, teleporter, musuh, dan diamond.
- b. Tuple: Digunakan untuk menyimpan pasangan arah (dx dan dy) dan juga hasil pengembalian arah.
- c. Set: Digunakan untuk menyimpan posisi yang ingin dihindari secara efisien (teleporter/musuh).
- d. Dictionary-like Properties: Mengakses properti bot dan diamond seperti jumlah diamond, waktu tersisa, kapasitas.
- e. Custom Object (Position, GameObject, Board): Menggunakan struktur objek dari library seperti Position, Board, dan GameObject. Digunakan hampir di seluruh bagian seperti `current_position`, `btn.position`.

4.3 Pengujian Program

1. Skenario Pengujian

Pengujian dilakukan dengan beberapa skenario permainan yang memiliki susunan diamond, bot, dan teleporter yang berbeda. Tujuan dari pengujian ini adalah untuk:

- a. Menguji apakah bot dapat mengambil diamond dengan efisien.
- b. Menilai apakah bot mampu kembali ke base tepat waktu.
- c. Memastikan bot menggunakan teleporter jika lebih optimal.
- d. Mengetahui seberapa besar pengaruh strategi greedy berbasis density terhadap skor akhir.

2. Hasil Pengujian dan Analisis

Pengujian dilakukan pada beberapa skenario dengan konfigurasi peta, jumlah diamond, dan posisi musuh yang berbeda-beda. Pada skenario pertama, yaitu peta simpel dengan jumlah diamond yang sedikit dan tanpa gangguan dari musuh, bot mampu mengumpulkan seluruh diamond yang tersedia dan kembali ke base tepat waktu. Skor akhir yang diperoleh mencapai 1200 poin dengan sisa waktu sekitar 30 detik. Hal ini menunjukkan bahwa algoritma greedy cukup efektif dalam kondisi ideal.

Pada skenario kedua, peta dipenuhi dengan banyak musuh yang bergerak acak dan menghalangi jalur bot menuju diamond. Dalam kondisi ini, bot harus menghindari musuh terlebih dahulu, yang menyebabkan keterlambatan dalam pengambilan beberapa diamond. Akibatnya, hanya sebagian diamond yang berhasil dikumpulkan dan skor akhir menurun menjadi sekitar 950 poin dengan sisa waktu 15 detik. Strategi greedy tetap berjalan baik, namun menunjukkan keterbatasannya ketika harus mempertimbangkan ancaman dari musuh.

Skenario ketiga menguji kemampuan bot dalam memanfaatkan teleporter. Posisi beberapa diamond cukup jauh dari base, namun tersedia sepasang teleporter yang dapat digunakan untuk mempercepat pergerakan. Dalam skenario ini, algoritma greedy berhasil mendeteksi jalur tercepat dengan memanfaatkan teleporter, menghasilkan efisiensi tinggi. Bot mampu mengumpulkan sebagian besar diamond dan kembali ke base dengan skor akhir mencapai 1300 poin serta sisa waktu 40 detik.

Terakhir, pada **skenario dengan diamond** yang tersebar jauh satu sama lain, bot tetap berusaha mengambil diamond dengan rasio keuntungan per jarak tertinggi (density). Meskipun kondisi ini lebih kompleks, algoritma greedy tetap menunjukkan performa yang baik. Bot berhasil kembali ke base tepat waktu dan memperoleh skor akhir sebesar 1150 poin, meskipun tidak semua diamond dapat diambil karena keterbatasan waktu dan jarak.

Secara keseluruhan, strategi greedy yang diterapkan mampu memberikan performa yang cukup optimal, terutama dalam skenario yang tidak terlalu kompleks. Namun, dalam kondisi yang lebih menantang, seperti banyaknya musuh atau distribusi diamond yang tidak merata, diperlukan peningkatan pada strategi agar lebih adaptif terhadap berbagai situasi dinamis di dalam permainan.

Dari hasil pengujian, terlihat bahwa strategi greedy bekerja cukup optimal untuk menyelesaikan permainan dalam waktu yang tersedia. Namun, dalam beberapa skenario kompleks, performa bisa berkurang jika musuh menghalangi jalur atau diamond terlalu tersebar.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Program bot yang menggunakan strategi *greedy* berbasis *density* (nilai/biaya) menunjukkan performa yang cukup baik dalam menyelesaikan misi pengambilan diamond dan kembali ke base tepat waktu. Strategi ini efektif dalam:

1. Mengambil keputusan cepat dan efisien untuk memilih diamond terdekat dan paling menguntungkan.
2. Menyesuaikan diri dengan kondisi permainan melalui pemanfaatan fitur seperti teleporter.
3. Menunjukkan performa optimal pada skenario sederhana atau ideal tanpa gangguan eksternal yang signifikan.

Namun, pengujian juga menunjukkan bahwa strategi ini memiliki keterbatasan dalam menghadapi situasi kompleks, seperti:

1. Kehadiran musuh yang menghalangi jalur dan memaksa bot mengambil jalur memutar, sehingga mengurangi efisiensi.
2. Distribusi diamond yang tidak merata atau terlalu tersebar, yang membuat perhitungan *density* kurang akurat tanpa mempertimbangkan konteks global.

5.2 Saran

1. **Penambahan Komponen Prediktif atau Adaptif:** Gunakan pendekatan prediktif terhadap pergerakan musuh, sehingga bot bisa merencanakan jalur lebih aman dan efisien. Implementasikan strategi semi-greedy atau kombinasi dengan *A* atau *Dijkstra* untuk perhitungan jalur optimal dalam skenario kompleks.

2. **Manajemen Waktu Lebih Cermat:** Tambahkan evaluasi waktu kembali ke base sebelum mengambil diamond, agar tidak terlalu beresiko kehilangan semua hasil karena keterlambatan.
3. **Pengelolaan Inventory Lebih Cerdas:** Jika inventory penuh, pertimbangkan jalur terdekat ke base dibandingkan mencari diamond lain yang tidak bisa diambil. Terapkan sistem *prioritas drop* jika sistem permainan memungkinkan membuang item bernilai rendah.
4. **Pemanfaatan Teleporter Lebih Strategis:** Perlu algoritma khusus yang menghitung *total time saving* menggunakan teleporter dibandingkan jalan biasa. Simulasikan penggunaan teleporter beberapa langkah ke depan untuk menentukan manfaat jangka panjangnya.
5. **Pengembangan Sistem Evaluasi Dinamis:** Kembangkan sistem skor dinamis yang tidak hanya berdasarkan *density*, tetapi juga memperhitungkan risiko (musuh, jarak ke base, sisa waktu).

LAMPIRAN

- A. [Repository Github](#)
- B. [Video Penjelasan](#)

DAFTAR PUSTAKA

- [1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.
- [2] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Harlow, England: Pearson Education Limited, 2021.
- [3] J. Kleinberg and É. Tardos, *Algorithm Design*. Boston, MA, USA: Pearson/Addison-Wesley, 2006.

