IMPLEMENTASI ALGORITMA GREEDY DALAM PEMECAHAN BOT PERMAINAN DIAMOND

Tugas Besar

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

Oleh: Kelompok 2 (GaChoAn)

Garis Rayya Rabbani	123140018
Choirunnisa Syawaldina	123140136
Anisah Octa Rohila	123140137

Dosen Pengampu: Winda Yulita, M.Cs.

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SUMATERA
LAMPUNG SELATAN
2025

DAFTAR ISI

DAFTAR TABEL

BAB I
DESKRIPSI TUGAS

Permainan Diamonds merupakan sebuah simulasi digital di mana agen (bot) diinstruksikan untuk mengumpulkan diamond sebanyak mungkin demi memperoleh skor tertinggi. Setiap bot bergerak pada papan permainan yang berisi diamond dengan dua jenis nilai, yaitu diamond merah dan biru, serta berbagai elemen tambahan seperti red button, teleporter, base, dan inventory. Diamond dapat dikumpulkan dengan cara melangkahi posisinya, namun penyimpanan hanya bersifat sementara di inventory, sehingga bot perlu kembali ke base untuk mengamankan poin. Selama permainan berlangsung, posisi elemen-elemen ini akan berubah secara dinamis, menuntut strategi yang efisien dalam pengambilan keputusan.

Untuk menyelesaikan permasalahan tersebut, digunakan pendekatan algoritma Greedy sebagai inti dari logika bot. Algoritma ini memungkinkan bot untuk menentukan langkah terbaik berdasarkan kondisi papan secara lokal pada setiap waktu. Implementasi dilakukan dalam bahasa Python dengan menggunakan starter pack dan game engine yang telah disediakan. Bot dikembangkan dengan strategi yang mengedepankan efektivitas perolehan skor, optimalisasi penggunaan inventory, serta penghindaran interaksi yang merugikan dengan bot lawan. Proses pengembangan ini turut disertai dokumentasi lengkap mengenai pemetaan persoalan ke dalam komponen algoritma Greedy, eksplorasi strategi alternatif, serta evaluasi kinerja berdasarkan hasil pengujian dan pengamatan selama simulasi permainan berlangsung.

1.1 Latar Belakang

Perancangan strategi algoritmik dalam penyelesaian masalah nyata merupakan salah satu aspek penting dalam bidang informatika. Salah satu pendekatan yang sering digunakan untuk menghasilkan solusi efisien secara waktu adalah algoritma Greedy. Pendekatan ini bekerja dengan cara mengambil keputusan terbaik pada setiap langkah secara lokal, dengan harapan bahwa keputusan tersebut akan menghasilkan solusi global yang optimal atau mendekati optimal. Untuk menguji pemahaman dan kemampuan dalam mengimplementasikan strategi ini, digunakan sebuah simulasi

permainan interaktif yang kompleks, yaitu Diamonds.

Permainan Diamonds menyediakan lingkungan dinamis yang menantang, di mana setiap agen (bot) harus mengumpulkan diamond dengan efisien sambil mempertimbangkan berbagai keterbatasan dan rintangan. Keputusan yang diambil oleh bot harus mempertimbangkan nilai diamond, kapasitas inventory, posisi base, dan keberadaan bot lawan. Permainan ini juga memuat elemen acak seperti red button dan teleporter yang menambah kompleksitas jalannya permainan. Dengan demikian, pengembangan bot dalam konteks ini menjadi sarana yang tepat untuk menerapkan dan mengevaluasi efektivitas algoritma Greedy secara langsung melalui simulasi kompetitif yang menyerupai skenario dunia nyata.

1.2 Tujuan Tugas Besar

Tugas ini bertujuan untuk:

- Memahami dan mengaplikasikan konsep algoritma Greedy pada konteks permainan.
- Merancang strategi pengambilan keputusan otomatis berbasis kondisi lingkungan.
- Mengembangkan program Python yang mampu bersaing dengan bot lain dalam permainan Diamonds.
- Mengevaluasi dan menguji efektivitas strategi melalui pertandingan dan analisis performa.

1.3 Ruang Lingkup Tugas

Ruang lingkup tugas ini mencakup perancangan dan implementasi strategi algoritma Greedy untuk mengendalikan pergerakan sebuah bot dalam permainan Diamonds. Fokus utama terletak pada pengembangan logika pengambilan keputusan bot agar dapat mengumpulkan diamond secara efisien dengan mempertimbangkan berbagai elemen permainan, seperti nilai diamond, keterbatasan kapasitas inventory, posisi base, serta interaksi dengan bot lawan.

Tugas ini dibatasi pada aspek logika dan strategi pergerakan bot yang

diimplementasikan dalam bahasa Python, tanpa melakukan modifikasi terhadap game engine atau antarmuka visualisasi yang telah disediakan. Proses implementasi dilakukan menggunakan starter pack bot resmi dan diuji dalam lingkungan permainan yang telah ditentukan. Selain itu, strategi yang dirancang harus dapat dipetakan secara jelas ke dalam komponen-komponen algoritma Greedy dan didukung oleh dokumentasi serta analisis performa berdasarkan hasil pengujian yang sistematis.

1.4 Spesifikasi Tugas

Tugas ini berfokus pada pengembangan sebuah program bot yang mampu bersaing dalam permainan Diamonds dengan menerapkan strategi algoritma Greedy. Bot yang dikembangkan harus mampu bergerak secara mandiri, mengambil keputusan secara optimal berdasarkan kondisi permainan, serta menghindari situasi yang merugikan. Untuk memastikan kesesuaian antara strategi dan implementasi, terdapat sejumlah ketentuan teknis dan non-teknis yang harus dipenuhi, yaitu:

1. Bot harus diimplementasikan menggunakan bahasa pemrograman Python, dan dapat dijalankan dengan baik menggunakan game engine yang telah disediakan.
2. Strategi utama yang digunakan dalam pengambilan keputusan bot adalah algoritma Greedy. Strategi ini harus dirancang secara mandiri dan disesuaikan dengan objektif permainan, yaitu memperoleh diamond sebanyak-banyaknya.
3. Implementasi logika bot dilakukan dengan menggunakan *starter pack* yang telah disediakan. Modifikasi difokuskan pada bagian pemrosesan keputusan bot, sedangkan struktur utama *game engine* dan antarmuka tidak diubah.
4. Bot harus dapat berinteraksi dengan *backend* permainan melalui API sesuai dokumentasi, dan dikendalikan secara otomatis tanpa intervensi pengguna selama pertandingan berlangsung.
5. Setiap strategi yang diterapkan harus dijelaskan secara eksplisit dalam laporan, disertai dengan pemetaan elemen permainan ke dalam komponen algoritma Greedy dan analisis efektivitas strategi yang digunakan.

BAB II
LANDASAN TEORI

2.1 Dasar Teori

Algoritma Greedy merupakan salah satu pendekatan dalam pemecahan masalah optimasi yang bekerja dengan cara mengambil keputusan terbaik pada setiap langkah secara lokal, dengan harapan solusi tersebut akan menghasilkan hasil yang optimal secara keseluruhan. Pendekatan ini tidak meninjau seluruh kemungkinan solusi, melainkan secara iteratif memilih opsi yang dianggap paling menjanjikan pada saat itu berdasarkan suatu fungsi seleksi.

Salah satu ciri khas dari algoritma ini adalah tidak adanya mekanisme untuk melakukan koreksi atau peninjauan ulang terhadap keputusan yang telah diambil sebelumnya. Oleh karena itu, agar solusi akhir yang diperoleh benar-benar optimal, masalah yang dipecahkan harus memenuhi dua properti utama, yaitu greedy choice property dan optimal substructure. Greedy choice property menyatakan bahwa solusi optimal global dapat dibangun dengan memilih solusi optimal lokal pada setiap langkah. Sedangkan optimal substructure berarti bahwa solusi optimal dari suatu masalah dapat dibentuk dari solusi optimal dari submasalah-submasalahnya [1].

2.2 Cara Kerja Program

Program bot dalam permainan Diamonds dirancang untuk mengambil keputusan secara otomatis berdasarkan kondisi permainan yang sedang berlangsung. Proses pengambilan keputusan dilakukan secara berulang (iteratif) selama waktu permainan berjalan, dengan tujuan utama mengumpulkan diamond sebanyak mungkin dan menyimpannya ke base untuk memperoleh poin. Bot beroperasi dengan memanfaatkan API yang disediakan oleh game engine, serta memproses informasi papan permainan (board) untuk menentukan langkah berikutnya berdasarkan strategi yang diimplementasikan dengan algoritma Greedy.

Secara garis besar, alur kerja bot terdiri atas beberapa tahap utama, yaitu:

1. Inisialisasi bot dan pendaftaran ke dalam permainan.

2. Pembacaan kondisi terkini papan permainan melalui API.
3. Pemrosesan kondisi papan untuk menentukan aksi terbaik menggunakan algoritma Greedy.
4. Pengiriman aksi (gerakan) ke server.
5. Pembaruan kondisi dan pengulangan proses hingga waktu habis.

2.2.1 Cara Implementasi Program

Cara kerja program ini adalah dengan mengimplementasikan **algoritma Greedy** pada bot permainan *Diamonds*, sesuai dengan ketentuan tugas besar yang diberikan. Program bot dibuat dalam bahasa Python dengan memanfaatkan *starter pack* yang sudah disediakan. Algoritma Greedy dijalankan dengan logika sederhana: pada setiap langkah, bot akan memproses informasi papan permainan yang diperoleh dari *API backend*, lalu menentukan aksi yang memberikan keuntungan terbesar pada saat itu (misalnya mengambil *diamond* bernilai tertinggi atau mendekati *base* jika *inventory* penuh). Setiap iterasi, bot akan:

1. Membaca kondisi terkini papan permainan melalui API (informasi posisi *diamond*, *base*, *red button*, *teleporter*, dan bot lain).
2. Menilai semua kemungkinan langkah (*utara*, *selatan*, *timur*, *barat*) dan menghitung mana yang paling sesuai dengan kriteria Greedy yang telah dirumuskan (seperti nilai *diamond* terdekat).
3. Memilih langkah yang dianggap paling menguntungkan (aksi yang memaksimalkan poin atau meminimalkan risiko *tackle*).
4. Mengirimkan aksi (arah gerak) tersebut ke server menggunakan API.
5. Menerima kondisi terbaru dari server dan mengulangi proses ini hingga waktu permainan habis.

Dalam implementasinya, program memanfaatkan struktur data yang efisien, seperti *list* dan *dictionary*, untuk menyimpan kondisi board, status *inventory*, dan informasi bot. Dengan demikian, pengambilan keputusan dapat dilakukan dengan cepat dan akurat pada setiap langkah permainan.

2.2.2 Menjalankan Bot Program

Program bot dijalankan menggunakan bahasa Python, memanfaatkan *starter pack* dan *game engine* yang sudah disediakan. Sebelum menjalankan bot, pastikan semua dependensi sudah terinstal, dan bot sudah dikonfigurasi pada file konfigurasi yang sesuai (misalnya pengaturan email, password, dan nama bot). Langkah-langkah umum menjalankan program bot:

- Program bot dijalankan menggunakan bahasa Python, dengan memanfaatkan *starter pack* dan *game engine* yang sudah disediakan.
- Semua *dependencies* diinstal sesuai yang tercantum pada file `requirements.txt`.
- Bot dijalankan menggunakan perintah Python pada file utama `main.py` dengan menambahkan nama bot dan logic nya pada CONTROLLERS.
- Lalu untuk menjalankan bot nya dapat memanggil logic bot, email, nama, password, dan tim sesuai keinginan user. Jika ingin menjalankan lebih dari 1 bot sekaligus dapat memanggil langsung melalui jumlah terminal yang sama dengan jumlah bot yg ingin dijalankan (contoh 3 bot membutuhkan 3 terminal) atau menjalankan bot menggunakan `run-bots.bat` dengan memanggil langsung seluruh bot yang ingin dijalankan.
- Setelah dijalankan, program bot akan secara otomatis melakukan:
  1. Autentikasi ke server.
  2. Bergabung ke permainan (*board*).
  3. Memulai loop pengambilan keputusan hingga waktu permainan habis.
- Visualisasi permainan dapat dipantau melalui antarmuka *frontend web* yang sudah disediakan.
- Konfigurasi logic bot disimpan pada direktori `/game/logic` dan didaftarkan pada file `main.py`.
- Dengan demikian, bot akan terus bergerak berdasarkan strategi Greedy yang telah diimplementasikan, dan permainan dapat dilihat secara real-time.

Pastikan juga semua file konfigurasi (misalnya `config.json`) sudah disesuaikan agar bot dapat dijalankan dengan lancar.

2.2.3 Fitur Tambahan atau Aspek Lain

Selama proses pengembangan bot, terdapat beberapa fitur tambahan dan pengaturan teknis yang digunakan untuk memaksimalkan kinerja bot serta memudahkan proses debugging, simulasi, dan pengujian strategi. Fitur-fitur ini tidak bersifat wajib, namun sangat berguna dalam meningkatkan kualitas implementasi.

1. **Versi Bot dan Eksperimen Strategi**

   Direktori `game/logic` berisi beberapa versi logic bot seperti `GachoanBot.py` sebagai bot yang utama dan terpilih, `WawanBot.py` sebagai bot yang menjadi pertimbangan terkuat, `GachoanLevel8.py` sebagai bot yang juga menjadi pertimbangan, dan bot lainnya yang dikembangkan menggunakan logika yang berbeda-beda. Tiap bot menerapkan pendekatan Greedy yang berbeda untuk dibandingkan dan diuji, seperti mengejar diamond terdekat, mempertimbangkan nilai diamond, atau menghindari lawan.

2. **Script Eksekusi Paralel**

   File `run-bots.bat` dan `run-bots.sh` memungkinkan eksekusi beberapa bot secara paralel melalui terminal. Hal ini digunakan untuk menguji kompetisi antar bot dalam satu atau lebih sesi permainan guna mengevaluasi efektivitas strategi masing-masing.

3. **Modularisasi Fungsi dan Struktur Data**

   File seperti `models.py` menyimpan struktur data utama (seperti posisi, objek permainan, dan papan permainan), sedangkan `util.py` berisi fungsi-fungsi bantu seperti `get_direction()`, `clamp()`, dan `position_equals()` yang menyederhanakan logika pergerakan bot. Modularisasi ini membantu pemisahan antara logika inti dengan fungsi utilitas, sehingga program lebih mudah dibaca dan dirawat.

4. **Dokumentasi dan Konfigurasi**

   Tersedia `README.md` yang menjelaskan cara instalasi dan penggunaan bot.

BAB III
APLIKASI STRATEGI *GREEDY*

3.1 Proses *Mapping*

Untuk mengimplementasikan algoritma **Greedy** pada permainan **Diamonds**, tugas ini dapat dimapping ke dalam elemen-elemen berikut:

1. Himpunan Kandidat

   Himpunan kandidat pada permainan ini adalah semua kemungkinan arah gerak yang dapat dilakukan oleh bot pada setiap langkah permainan. Bot harus memilih satu dari beberapa arah gerak yang tersedia, yang akan menjadi langkahnya pada iterasi tersebut.

   - **NORTH** (utara)
   - **SOUTH** (selatan)
   - **EAST** (timur)
   - **WEST** (barat)

   Arah-arah gerak ini kemudian akan dipertimbangkan untuk dipilih oleh fungsi seleksi yang ada pada bot.

2. Himpunan Solusi

   Himpunan solusi merupakan rangkaian arah gerakan yang diambil bot selama permainan berlangsung. Himpunan ini terbentuk dari hasil pemilihan arah gerak terbaik pada setiap iterasi hingga permainan berakhir. Sehingga, solusi akhir adalah urutan langkah yang menjadi strategi Greedy bot.

3. Fungsi Seleksi

   Fungsi seleksi adalah bagian terpenting dalam algoritma Greedy. Pada kode bot ini, fungsi seleksi berupa logika prioritas yang menentukan langkah mana yang memberikan keuntungan maksimal. Fungsi ini mempertimbangkan beberapa kriteria dan kondisi permainan, seperti:

   - *Greedy by Escape / Return*: kembali ke base jika inventory penuh atau musuh dekat dan diamond banyak.
   - *Last Dash Diamond Grab*: jika waktu kritis, cari diamond yang bisa diambil

lalu segera kembali.

- *Greedy by Tackle*: jika musuh kaya berada di sebelah, tackle langsung.
- *Greedy by Inventory Full*: jika inventory penuh, langsung ke base.
- *Greedy by Red Button*: jika red button lebih menguntungkan, tekan.
- *Greedy by Proactive Tackle*: jika musuh kaya dekat, dekati untuk tackle.
- *Greedy by Diamond Collection*: jika tidak ada musuh kaya, cari diamond terdekat.
- *Default*: kembali ke base jika tidak ada strategi lain.

4. Fungsi Kelayakan

   Fungsi kelayakan memastikan bahwa setiap langkah yang dipilih bot adalah valid dan aman untuk dilakukan. Bot hanya akan memilih langkah yang:

   - Langkah tidak keluar papan permainan.
   - Mempertimbangkan inventory penuh.
   - Mempertimbangkan waktu yang tersisa.
   - Memastikan diamond muat dalam inventory.
   - Memastikan jalur menggunakan *teleporter* jika lebih efisien.

5. Fungsi Objektif

   Fungsi objektif dari algoritma Greedy ini adalah memaksimalkan poin total yang diperoleh bot di akhir permainan. Fungsi ini secara tidak langsung muncul pada kode bot, dengan:

   $$\max(\text{jumlah diamond merah} \times 2 + \text{jumlah diamond biru} \times 1)$$

   Bot memprioritaskan diamond merah (2 poin), tetapi jika terlalu jauh dibandingkan diamond biru (1 poin), bot memilih yang terdekat. Bot juga memanfaatkan tackle atau teleporter untuk memaksimalkan poin sambil menghindari risiko.

3.2 Eksplorasi Alternatif Solusi Greedy

Dalam mengimplementasikan algoritma Greedy pada permainan **Diamonds**, terdapat beberapa alternatif solusi yang dapat dieksplorasi untuk menentukan strategi pemilihan langkah yang optimal. Alternatif solusi ini masing-masing mempertimbangkan kriteria lokal yang berbeda, yang pada akhirnya mempengaruhi

total poin yang diperoleh. Berikut adalah beberapa alternatif solusi Greedy yang dapat dipertimbangkan:

1. Greedy Berdasarkan Jarak Terdekat ke Diamond (Tanpa Prioritas)

   Alternatif ini memilih diamond terdekat, tanpa memperhatikan apakah diamond tersebut merah (2 poin) atau biru (1 poin). Bot hanya memprioritaskan langkah tercepat untuk mengambil diamond apa pun yang berada pada jarak terdekat.

2. Greedy Prioritas Diamond Merah

   Alternatif ini selalu memprioritaskan diamond merah (karena nilai 2 poinnya) meskipun jaraknya sedikit lebih jauh dari diamond biru. Jika diamond merah tidak ada atau terlalu jauh, maka diamond biru akan diambil sebagai alternatif.

3. Greedy dengan Pemanfaatan Teleporter

   Alternatif ini menggunakan logika teleportasi: jika menggunakan teleporter dapat mengurangi jarak ke diamond secara signifikan, maka bot akan bergerak ke teleporter terlebih dahulu. Hal ini memungkinkan bot untuk mengambil diamond yang secara langsung tidak terjangkau dalam beberapa langkah.

4. Greedy Berbasis Inventaris

   Bot mempertimbangkan kapasitas inventory dan akan memutuskan untuk kembali ke base jika inventory hampir penuh, daripada terus mengambil diamond di sekitar. Alternatif ini mengutamakan penyimpanan poin daripada risiko kehilangan (misalnya akibat tackle musuh).

5. Greedy Berbasis Waktu Tersisa

   Pada alternatif ini, waktu tersisa menjadi faktor utama. Jika waktu hampir habis, bot akan memprioritaskan langkah untuk kembali ke base meskipun ada diamond dekat yang bisa diambil, guna memastikan poin yang sudah dikumpulkan dapat disimpan.

6. Greedy Berbasis Tackle Musuh

   Bot akan lebih agresif jika menemukan musuh yang membawa banyak diamond dan berada dekat. Alternatif ini mencoba untuk mencuri diamond dari musuh (tackle) demi meningkatkan poin secara langsung, meskipun jaraknya sedikit lebih jauh dari diamond biasa.

7. Greedy Menggabungkan Beberapa Kriteria (Hybrid)

Alternatif ini adalah strategi hybrid yang mengombinasikan beberapa kriteria di atas: jarak terdekat, nilai diamond, kapasitas inventory, waktu tersisa, dan potensi tackle. Dengan demikian, bot selalu memilih langkah terbaik dari perspektif Greedy lokal, tetapi memperhitungkan banyak faktor sekaligus.

Dengan mengeksplorasi berbagai alternatif solusi Greedy ini, bot dapat disesuaikan untuk menghadapi berbagai kondisi permainan dan lawan yang berbeda. Strategi akhir yang diimplementasikan akan dipilih berdasarkan efektivitasnya dalam kompetisi.

3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy

Setelah mengeksplorasi berbagai alternatif solusi Greedy, berikut adalah analisis terkait kecepatan eksekusi, akurasi solusi, dan kompleksitas masing-masing alternatif.

Tabel 3.1: Perbandingan Alternatif Solusi Greedy

Alternatif Strategi	**Kecepatan Eksekusi**	**Akurasi Solusi**	**Kompleksitas**
Greedy Jarak Terdekat (tanpa prioritas)	Cepat	Sedang	Rendah
Greedy Prioritas Diamond Merah	Sedang	Tinggi	Sedang
Greedy dengan Teleporter	Sedang	Tinggi	Tinggi
Greedy Berbasis Inventaris	Cepat	Sedang	Sedang
Greedy Berbasis Waktu Tersisa	Sedang	Tinggi	Sedang
Greedy Berbasis Tackle Musuh	Sedang	Tinggi	Tinggi
Greedy Hybrid (Multi-kriteria)	Lambat	Tinggi	Tinggi

Dari tabel di atas, terlihat bahwa alternatif “Greedy Hybrid (Multi-kriteria)” memiliki akurasi solusi yang tinggi, tetapi kecepatan eksekusi lebih lambat karena mempertimbangkan banyak faktor. Sementara itu, alternatif “Greedy Jarak Terdekat” memiliki kecepatan eksekusi cepat, tetapi akurasi solusi sedang karena tidak mempertimbangkan nilai diamond atau kondisi waktu dan musuh.

3.3.1 Strategi Greedy yang Dipilih

Strategi Greedy yang diimplementasikan pada bot ini adalah trategi hybrid multi-kriteria yang memprioritaskan langkah dengan keuntungan terbesar di setiap iterasi, namun tetap mempertimbangkan beberapa faktor penting secara hierarkis [2].

Secara garis besar, urutan prioritas strategi adalah:

1. Kembali ke *base* jika *inventory* penuh atau waktu hampir habis (*Greedy by Return/Escape*).
2. Jika musuh dekat dan inventory penuh, kembali ke base untuk menghindari *tackle* (Greedy by Escape).
3. Jika waktu sangat kritis, cari *diamond* yang dekat dan memungkinkan untuk langsung kembali ke base (*Last Dash Diamond Grab*).
4. Jika musuh kaya (banyak *diamond*) berada dekat, *tackle* musuh untuk mencuri poin (*Greedy by Tackle*).
5. Jika tidak ada kondisi darurat, cari *diamond* merah atau biru terdekat (*Greedy by Diamond Collection*).
6. Manfaatkan *red button* atau *teleporter* jika memberikan jalur yang lebih efisien.
7. Jika tidak ada langkah prioritas lain, kembali ke *base* untuk menyimpan poin yang sudah dikumpulkan.

Strategi hybrid ini dipilih karena mampu menyesuaikan diri dengan kondisi permainan yang dinamis, memanfaatkan peluang di sekitar (*diamond*, musuh, *red button*, *teleporter*), dan tetap fokus pada objektif utama yaitu memaksimalkan poin di akhir permainan.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma *Greedy*

Bot GachoanBot dirancang dengan pendekatan algoritma greedy, yang pada setiap langkahnya selalu memilih aksi lokal terbaik berdasarkan kondisi terkini di papan permainan (board). Tujuan utama dari algoritma ini adalah mengumpulkan poin sebanyak-banyaknya dalam batasan waktu dan kapasitas inventory, serta meminimalkan risiko kehilangan diamond akibat tackle musuh. Strategi greedy dalam GachoanBot dijalankan secara bertingkat, dari prioritas tertinggi ke prioritas rendah. Setiap prioritas dievaluasi satu per satu, dan aksi akan langsung dipilih jika strategi tersebut valid. Urutan strategi:

1. Melarikan diri ke base jika membawa banyak diamond dan musuh mendekat.
2. Kembali ke base jika waktu hampir habis dan membawa diamond.
3. Ambil 1 diamond cepat lalu pulang jika waktu sangat kritis.
4. Tackle musuh di sebelah jika memungkinkan.
5. Kembali ke base jika inventory penuh.
6. Menekan tombol merah jika jumlah diamond di papan sedikit.
7. Dekati musuh untuk tackle proaktif.
8. Ambil diamond terdekat berdasarkan nilai dan jarak efektif.
9. Default kembali ke base jika tidak ada aksi lain.

4.1.1 Pseudocode

Berikut adalah pseudocode dari algoritma *Greedy* yang digunakan dalam GachoanBot:

```
CLASS GachoanBot EXTENDS BaseLogic:

    INIT():
        Inisialisasi self.goal = None
```

```
FUNCTION distance(pos_a, pos_b):
    RETURN Manhattan distance antara pos_a dan pos_b

FUNCTION get_teleporters(board):
    RETURN list objek teleport di board

FUNCTION distance_with_teleporter(start, end, board):
    IF < 2 teleporters:
        RETURN jarak langsung
    ELSE:
        Hitung jarak via teleportasi (dua kemungkinan)
        RETURN jarak minimum antara langsung vs teleport

FUNCTION get_best_teleport_or_target(bot_pos, target, board):
    IF < 2 teleporters:
        RETURN target
    ELSE:
        Bandingkan jarak via dua kombinasi teleport
        RETURN teleport entry terdekat jika lebih efisien,
        else target

FUNCTION get_closest_diamond(bot, board, red_only=False,
blue_only=False):
    Untuk setiap diamond:
        - Cek apakah sesuai warna dan kapasitas cukup
        - Hitung jarak efektif via teleport
    RETURN diamond dengan jarak efektif terpendek

FUNCTION get_red_button(board):
    RETURN objek tombol merah jika ada
```

```
FUNCTION get_game_status_info(bot, board):
    Ambil skor bot & skor tertinggi lawan
    Deteksi apakah lawan sedang dalam kondisi siap score besar
    RETURN dict dengan informasi:
        - my_score
        - highest_opponent_score
        - am_i_leading
        - lead_margin
        - opponent_primed_for_big_score

FUNCTION next_move(bot, board):
    Ambil status bot (posisi, diamond, base, waktu, dll)
    Dapatkan status permainan (via get_game_status_info)
    Tentukan current_turn_goal_pos = None

    # PRIORITAS STRATEGIS:

    1. Jika bawa >= 3 diamond dan musuh dekat (<=2):
        Pulang lewat teleport terbaik
    2. Jika unggul tipis, waktu menipis (tapi belum genting),
    dan bawa diamond:
        Pulang untuk amankan skor (secure_points_time_factor)
    3. Jika waktu sangat kritis dan ada diamond:
        Pulang segera
    4. Jika belum bawa diamond dan waktu sangat sedikit
    (last dash):
        Cari diamond sangat dekat dan cukup waktu untuk
        ambil dan pulang
    # STRATEGI UTAMA (JIKA BELUM ADA GOAL):
    5. Tackle Langsung:
        Jika lawan dekat dan bawa banyak diamond,
```

```
    dan bot tidak sedang bawa terlalu banyak
6. Jika inventory penuh:
    Pulang lewat teleport
7. Gunakan Tombol Merah:
    - Jika diamond langka dan inventory tidak penuh
    - Jika lawan siap score besar
    - Jika tertinggal dan tombol dekat
8. Tackle Proaktif:
    Jika lawan sedikit lebih jauh (2 unit) dan bawa
    banyak diamond
9. Cari Diamond:
    Ambil red atau blue diamond terdekat yang dapat
    diangkut
10. Default:
    Pulang

# PENYESUAIAN AKHIR:
Jika dekat dengan base dan bawa diamond, pastikan pulang
RETURN arah (delta_x, delta_y) menuju goal
```

4.1.2 Penjelasan Alur Program

GachoanBot ini mengimplementasikan algoritma greedy untuk mencari solusi optimal. Berikut adalah langkah-langkah dari alur program:

1. Fungsi `next_move()` memulai dengan mengambil informasi penting seperti posisi bot (`pos`), posisi base (`base`), jumlah diamond yang dibawa (`diamonds`), kapasitas maksimal inventory (`diamonds_carried_max`), dan waktu tersisa (`time_left`). Data ini menjadi dasar evaluasi strategi pada giliran tersebut.
2. Jika bot sedang membawa minimal 3 diamond dan terdapat musuh dalam jarak dua petak, maka bot akan segera kembali ke base untuk menghindari risiko tackle. Jika waktu tersisa hampir habis dan bot membawa diamond, bot juga

akan langsung kembali ke base untuk menyetorkan poin. Dalam kondisi kritis, ketika bot tidak membawa diamond dan waktu hampir habis, bot akan mencari diamond terdekat yang memungkinkan untuk diambil dan disetor kembali ke base sebelum waktu habis (strategi ‘last dash”).

3. Apabila ada musuh yang berada di petak sebelah dan membawa sedikitnya 2 diamond, maka bot akan melakukan tackle. Jika inventory bot penuh, maka tanpa pertimbangan lain bot akan kembali ke base untuk menyetorkan diamond.
4. Bila jumlah diamond di papan sangat sedikit, bot akan mempertimbangkan menekan tombol merah untuk meregenerasi diamond. Keputusan ini dipertimbangkan berdasarkan jarak ke tombol merah, jumlah diamond yang tersisa, dan kondisi inventory bot.
5. Jika inventory belum penuh dan ada musuh yang membawa banyak diamond dalam jarak dua petak, bot akan mendekat untuk membuka peluang tackle pada giliran berikutnya.
6. Bot akan mencari diamond terdekat dengan mempertimbangkan jarak efektif menggunakan teleporter. Jika tersedia, diamond merah akan diprioritaskan karena nilainya lebih tinggi, namun diamond biru dapat dipilih jika jaraknya jauh lebih dekat.
7. Jika tidak ada strategi yang bisa dijalankan, bot akan bergerak kembali ke base sebagai tindakan default.
8. Jika bot berada di sebelah base dan sedang membawa diamond, maka bot akan menyetorkan diamond tersebut ke base, meskipun sebelumnya memiliki tujuan berbeda.
9. Setelah tujuan (goal) dipilih, bot akan menghitung arah gerakan menggunakan fungsi `get_direction()` dan mengirimkan nilai `delta_x` dan `delta_y` ke game engine untuk dieksekusi.

Langkah-langkah ini memastikan bahwa solusi yang diperoleh sesuai dengan prinsip greedy, yaitu mengambil keputusan terbaik pada setiap langkah.

4.2 Struktur Data yang Digunakan

Berikut adalah struktur data utama yang digunakan dalam implementasi bot `GachoanBot`:

Tabel 4.2: Struktur Data yang Digunakan dalam Bot GachoanBot

Struktur Data	**Tipe / Jenis**	**Fungsi**
`Position`	Objek (class)	Menyimpan koordinat posisi (x, y) di papan permainan
`GameObject`	Objek (class)	Mewakili objek seperti bot, diamond, teleporter, dan tombol merah
`Board`	Objek (class)	Mewakili kondisi permainan saat ini, termasuk isi board dan waktu tersisa
`List`	Tipe koleksi	Menyimpan kumpulan objek, seperti daftar diamond atau bot
`Optional`	Tipe opsional	Menyatakan bahwa suatu nilai bisa bernilai objek atau `None`
`Tuple[int, int]`	Tipe pasangan nilai	Menyimpan arah gerakan bot dalam bentuk perubahan posisi (dx, dy)
`get_direction`	Fungsi utilitas	Mengubah posisi asal dan tujuan menjadi arah gerak (dx, dy)

4.3 Pengujian Program

Pengujian dilakukan untuk mengevaluasi efektivitas strategi greedy yang diimplementasikan pada bot GachoanBot dalam berbagai kondisi permainan.

4.3.1 Skenario Pengujian

Pengujian dilakukan untuk membandingkan performa dua metode algoritma greedy berbeda yang sudah kelompok kami buat, yaitu bot yang terpilih yaitu GachoanBot dengan bot lain yang kami buat dan tidak terpilih dengan nama GACHOANLEVEL8 dan WawanBot. Berikut adalah hasil pengujian yang dilakukan sebanyak 10 pertandingan dan selama 60 detik per pertandingan kedua bot ini saling berkompetisi dengan hasil pengujian sebagai berikut.

Tabel 4.3: Perbandingan Strategi antara tiga bot

Aspek Strategi	**GACHOANLEVEL8**	**GachoanBot**	**WawanBot**
Winrate (%)	10%	70%	20%
Avg. Diamond per Game	7	13	11
Escape jika bawa banyak diamond & musuh dekat	Ya, $\geq$3 diamond dan musuh $\leq$ 2 langkah	Ya, evaluasi awal turn	Ya, evaluasi prioritas tertinggi
Kembali ke base saat waktu kritis dan bawa diamond	Ya, berdasarkan `time_left` dan jarak ke base	Ya, dengan buffer dan teleporter	Ya, dengan buffer 4 langkah
Last Dash Diamond Grab (waktu kurang dari 10)	Tidak ada	Ada, evaluasi diamond cepat + pulang	Ada, diamond $\leq$ 2 langkah + pulang
Tackle musuh kaya (langsung)	Ya, jika musuh $\geq$2 diamond dan bot miskin	Ya, plus evaluasi skor bot vs musuh	Ya, dengan syarat jumlah diamond
Tackle proaktif (jarak 2)	Ya, jika bawa sedikit diamond	Ya, dengan kontrol agresivitas berbasis skor	Ya, jika inventory masih cukup kosong
Kembali ke base jika inventory penuh	Ya, langsung pulang	Ya, evaluasi awal di strategi	Ya, sederhana dan efisien
Red Button Usage (regenerasi diamond)	Ya, jika diamond < threshold	Ya, dengan tambahan logika disrupsi musuh	Ya, sederhana, dengan evaluasi jarak
Red Button for Disruption	Tidak ada	Ya, jika musuh siap mencetak skor besar	Tidak ada
Diamond Collection (merah > biru, jarak efisien)	Ya, berdasarkan jarak dan kapasitas	Ya	Ya, efisien dan modular
Gunakan Teleporter	Ya, di banyak fungsi jarak dan navigasi	Ya, semua evaluasi pakai TP	Ya, digunakan eksplisit di semua rute utama
Opportunistic Base Return (di sebelah base + bawa diamond)	Ya	Ya	Ya
Skor Lawan Dipertimbangkan	Tidak	Ya	Tidak
Secure Critical Points (unggul tipis + waktu cukup)	Tidak	Ya	Tidak
Modularisasi Strategi	Sudah cukup rapi	Sangat rapi dan scalable	Cukup baik, logika eksplisit
Level Strategi	Menengah	Tinggi (turnamen/advance)	Menengah-Tinggi

4.3.2 Hasil Pengujian dan Analisis

Berdasarkan hasil pengujian, berikut analisis dari masing-masing aspek strategi:

1. **Winrate:** GachoanBot memiliki winrate tertinggi, menunjukkan efektivitas strategi keseluruhan dalam memenangkan permainan.
2. **Rata-rata Diamond:** GachoanBot mengumpulkan rata-rata diamond terbanyak per game, mencerminkan efisiensi pergerakan dan pengambilan keputusan.
3. **Strategi Escape:** Ketiga bot memiliki strategi melarikan diri saat terancam, namun GACHOANLEVEL8 paling konservatif dengan kondisi lebih ketat.
4. **Pulang Saat Waktu Kritis:** Semua bot mampu kembali ke base saat waktu genting, dengan GachoanBot dan WawanBot lebih fleksibel menggunakan buffer langkah.
5. **Last Dash Grab:** GachoanBot dan WawanBot mampu memanfaatkan waktu akhir secara agresif, sedangkan GACHOANLEVEL8 tidak memiliki strategi ini.
6. **Tackle Musuh Kaya:** Semua bot memiliki kemampuan tackle, namun GachoanBot mengevaluasi skor musuh dan kondisi lebih kompleks.
7. **Tackle Proaktif:** GachoanBot paling adaptif dalam tackle berdasarkan skor dan kondisi, membuatnya unggul dalam duel taktis.
8. **Pulang Saat Inventory Penuh:** Ketiga bot langsung pulang saat inventory penuh, menunjukkan efisiensi manajemen beban.
9. **Red Button - Regenerasi Diamond:** Semua bot memakai red button untuk regenerasi, namun GachoanBot menambahkan logika disrupsi musuh.
10. **Red Button - Disruption:** GachoanBot satu-satunya yang memanfaatkan red button untuk menghambat musuh mencetak skor.
11. **Pengambilan Diamond:** Semua bot mempertimbangkan efisiensi jarak dan nilai diamond, namun GachoanBot lebih modular dan fleksibel.
12. **Teleporter:** Ketiga bot menggunakan teleporter, tapi GachoanBot paling konsisten dalam semua evaluasi navigasi.
13. **Opportunistic Return:** Semua bot memanfaatkan peluang pulang cepat saat berada di sebelah base sambil membawa diamond.
14. **Pertimbangan Skor Lawan:** Hanya GachoanBot yang memperhitungkan skor

musuh, menunjukkan strategi kompetitif tingkat lanjut.

15. **Secure Critical Points:** GachoanBot satu-satunya yang memiliki strategi menjaga posisi penting saat unggul dan waktu cukup.
16. **Modularisasi Strategi:** Strategi GachoanBot paling modular, memungkinkan pengembangan dan pemeliharaan jangka panjang.
17. **Level Strategi:** GachoanBot berada di level strategi tertinggi dan cocok untuk skenario kompetitif atau turnamen.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dalam tugas besar ini, telah berhasil diimplementasikan algoritma Greedy dalam bentuk bot permainan yang mampu mengambil keputusan secara otomatis dalam permainan *Diamond Game*. Bot utama, yaitu **GachoanBot**, dirancang dengan pendekatan Greedy hybrid yang mempertimbangkan berbagai faktor lokal seperti nilai diamond, kapasitas inventory, waktu tersisa, serta posisi musuh dan teleporter. Bot ini secara bertahap mengevaluasi strategi dari prioritas tertinggi hingga terendah, sehingga mampu menyesuaikan diri dengan kondisi permainan yang dinamis dan kompetitif.

Berdasarkan hasil pengujian terhadap tiga bot berbeda, GachoanBot terbukti memiliki performa tertinggi dengan winrate 70% dan rata-rata diamond per game sebanyak 13, mengungguli GACHOANLEVEL8 dan WawanBot. Strategi-strategi kunci seperti *last dash diamond grab*, *tackle musuh kaya*, *red button for disruption*, serta pertimbangan skor lawan menjadikan GachoanBot unggul dalam skenario kompetitif. Modularitas struktur dan logika juga memudahkan pengembangan dan pemeliharaan strategi di masa depan.

Dapat disimpulkan bahwa pendekatan algoritma Greedy, ketika dipadukan dengan desain logika multi-kriteria dan evaluasi lokal yang cermat, mampu memberikan solusi yang tidak hanya efisien tetapi juga adaptif terhadap dinamika permainan. Hal ini menunjukkan bahwa Greedy bukan sekadar pendekatan sederhana, melainkan dapat dimanfaatkan secara strategis untuk menghasilkan bot cerdas yang kompetitif dan optimal di berbagai kondisi permainan.

5.2 Saran

Untuk pengembangan lebih lanjut, disarankan agar strategi yang telah ada dapat diperluas dengan pendekatan berbasis pembelajaran mesin seperti *reinforcement learning*, guna meningkatkan adaptivitas dan kemampuan bot dalam menghadapi pola permainan lawan yang beragam. Selain itu, eksperimen lanjutan terhadap berbagai peta

permainan dan konfigurasi musuh yang lebih kompleks dapat memperkuat validitas generalisasi strategi yang dikembangkan dalam tugas besar ini.

Berdasarkan hasil pengujian pada Subbab 4.3, terlihat bahwa bot GachoanBot memiliki tingkat kemenangan yang tinggi dan rata-rata pengumpulan diamond yang lebih banyak dibandingkan bot lain. Hal ini menunjukkan efektivitas penerapan strategi hybrid multi-kriteria dalam menghadapi kondisi permainan yang dinamis. Sebagai saran pengembangan lebih lanjut, disarankan untuk melakukan eksplorasi mendalam terhadap kombinasi strategi lain yang lebih adaptif, khususnya dalam pemanfaatan red button dan teleporter. Selain itu, mempertimbangkan kondisi papan yang berubah cepat serta potensi interaksi dengan bot lawan menjadi aspek penting yang dapat meningkatkan keunggulan kompetitif GachoanBot di skenario permainan yang lebih kompleks. references

LAMPIRAN

A Repository GitHub

Hasil dari program bot permainan dapat diakses melalui repository GitHub yang berisi kode dan dokumentasi. Anda dapat mengunduh dan memeriksa versi terbaru dari program pada tautan berikut: GitHub Repository.

B Video Penjelasan

Penjelasan tambahan mengenai implementasi dan hasil dari program bot permainan dapat ditemukan dalam video yang diunggah ke Google Drive. Tautan untuk menonton video tersebut adalah: Video Penjelasan di Google Drive.

DAFTAR PUSTAKA

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, et al. *Introduction to Algorithms*. 3rd. Cambridge, Massachusetts: The MIT Press, 2009.

[2] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. Pearson, 2010.