IMPLEMENTASI GREEDY ALGORITHM WITH DYNAMIC PRIORITIZATION 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 6 (KRL)

Keira Lakeisha Fachra Fuady	122140142
Refi Ikhsanti	123140126
Louis Hutabarat	123140052

Dosen Pengampu: Winda Yulita, M.Cs.

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

DAFTAR ISI

BAB I
DESKRIPSI TUGAS

Tugas besar ini mengajak mahasiswa untuk mengembangkan sebuah bot permainan bernama Diamonds, di mana setiap pemain akan bersaing untuk mengumpulkan diamond sebanyak mungkin di dalam sebuah papan permainan (board) yang kompleks dan dinamis. Mahasiswa diminta untuk mengimplementasikan algoritma greedy sebagai strategi utama bot dalam mengambil keputusan dan mengoptimalkan jumlah diamond yang berhasil dikumpulkan.

1.1 Latar Belakang

Menjelaskan pentingnya penyusunan strategi algoritma dalam pembuatan bot otomatis, khususnya dalam konteks permainan Diamonds, serta penerapan algoritma greedy untuk mencapai tujuan maksimal.

1.2 Tujuan Tugas Besar

Tugas besar ini memiliki beberapa tujuan utama, yaitu:

- Mengimplementasikan algoritma greedy dalam bentuk bot otomatis yang mampu bersaing dalam permainan Diamonds.
- Melatih mahasiswa untuk merancang solusi strategis dan efisien dalam konteks permainan berbasis grid dan pergerakan dinamis.
- Mengembangkan kemampuan analisis dalam memilih dan menyesuaikan strategi greedy yang sesuai dengan kondisi permainan.
- Mendorong mahasiswa untuk memahami keterkaitan antara teori algoritma dan penerapannya dalam sistem berbasis API serta permainan berbasis web.
- Menumbuhkan semangat kolaborasi dalam pengembangan program melalui kerja kelompok dan kompetisi antarbot.

1.3 Ruang Lingkup Tugas

Ruang lingkup tugas besar ini meliputi:

- Pembuatan bot menggunakan bahasa Python berbasis starter pack yang telah disediakan.
- Implementasi strategi greedy sebagai inti dari logika pengambilan keputusan bot.
- Pengujian dan evaluasi performa bot dalam permainan Diamonds yang disimulasikan pada game engine resmi.
- Penulisan laporan lengkap yang mencakup teori, desain strategi, implementasi, dan hasil pengujian.
- (Opsional) Pembuatan video presentasi berisi penjelasan strategi dan simulasi permainan.

Mahasiswa tidak diwajibkan untuk memahami detail teknis dari game engine, karena seluruh API dan infrastruktur teknis telah disediakan. Fokus utama adalah pada logika algoritmik dan strategi greedy yang dirancang.

1.4 Spesifikasi Tugas

Berikut adalah spesifikasi tugas besar:

- Setiap kelompok (2–3 orang) harus mengembangkan bot yang mengimplementasikan algoritma greedy untuk mengumpulkan diamond sebanyak mungkin dalam permainan Diamonds.
- Strategi greedy harus dijelaskan secara eksplisit dalam laporan dan diuji kecocokannya saat demo.
- Permainan dijalankan pada game engine berbasis web yang menyediakan API endpoint untuk setiap aksi (move, join, register, dll).
- Komponen permainan meliputi: diamond biru (1 poin), diamond merah (2 poin), red button, teleporter, base, dan inventory.
- Bot harus mampu:
  - Mengelola inventory secara efisien.
  - Menghindari tackle dari bot lawan.
  - Memanfaatkan fitur-fitur interaktif dengan cerdas.
- Kode program harus disusun modular dengan komentar yang jelas.
- Bot akan dikompetisikan dengan bot dari kelompok lain dan dinilai berdasarkan efektivitas strategi dan performa saat pertandingan.

BAB II
LANDASAN TEORI

2.1 Dasar Teori

Algoritma greedy adalah paradigma algoritma yang membuat pilihan optimal lokal pada setiap langkah dengan harapan menemukan solusi optimal global[1]. Algoritma ini bekerja dengan prinsip ”greedy choice property”, yaitu pilihan yang dibuat pada setiap langkah adalah pilihan terbaik yang tersedia saat itu tanpa mempertimbangkan konsekuensi di masa depan. Karakteristik utama algoritma greedy meliputi: (1) membuat keputusan berdasarkan informasi yang tersedia saat ini, (2) tidak pernah mengubah keputusan yang sudah dibuat sebelumnya, dan (3) berharap bahwa serangkaian pilihan optimal lokal akan menghasilkan solusi optimal global.

Algoritma greedy umumnya digunakan untuk masalah optimasi seperti shortest path, minimum spanning tree, dan activity selection problem. Meskipun tidak selalu menghasilkan solusi optimal untuk semua masalah, algoritma greedy sangat efisien dalam hal kompleksitas waktu dan sering memberikan solusi yang cukup baik untuk banyak masalah praktis [2].

2.2 Cara Kerja Program

Bot KRLBot yang dikembangkan mengimplementasikan algoritma greedy untuk membuat keputusan strategis dalam permainan pengumpulan diamond. Bot ini bekerja dengan mengevaluasi situasi saat ini di papan permainan dan memilih aksi terbaik berdasarkan sistem prioritas yang telah ditentukan. Proses pengambilan keputusan dilakukan setiap turn dengan memperhatikan:

1. **Posisi bot dan jumlah diamond yang dibawa**.
2. **Kondisi lingkungan**, seperti posisi diamond, tombol merah, dan teleporter.
3. **Kehadiran musuh** di sekitar (untuk tackle atau penghindaran).
4. **Jarak ke base** untuk menyimpan diamond.

Bot kemudian memberikan perintah arah gerak yang akan dilakukan berdasarkan hasil evaluasi tersebut.

2.2.1 Cara Implementasi Program

Bot KRLBot yang dikembangkan mengimplementasikan algoritma greedy untuk membuat keputusan strategis dalam permainan pengumpulan diamond. Dikembangkan menggunakan bahasa **Python**, berbasis pada framework game yang telah disediakan (game engine). Implementasi utamanya mengacu pada strategi greedy dengan langkah-langkah sebagai berikut:

1. **Evaluasi Kondisi Saat Ini:**
   (a) Menggunakan fungsi `manhattan_distance()` untuk menghitung jarak ke objek seperti diamond, base, musuh, dan tombol merah.
   (b) Memeriksa jumlah diamond yang dimiliki melalui `board_bot.properties.diamonds` serta apakah musuh berada di sekitar menggunakan fungsi `find_enemy_in_range_to_tackle()`.
2. **Penghitungan Bobot Prioritas:**
   (a) **Weight Return Base:** Meningkat seiring jumlah diamond yang dimiliki (2 diamond = bobot 3, 3 diamond = bobot 6, 4+ diamond = bobot 10) dihitung dalam fungsi `compute_priority_weights()`.
   (b) **Weight Attack:** Berdasarkan kedekatan dengan lawan yang memiliki diamond (bobot 5-jarak untuk lawan dalam radius 2) juga dihitung dalam fungsi `compute_priority_weights()`.
   (c) **Weight Collect:** Berdasarkan jumlah diamond tersedia di papan (6+ diamond = bobot 8, 3-5 diamond = bobot 5, $<$ 3 diamond = bobot 2) dihitung dalam fungsi `compute_priority_weights()`.
   (d) **Weight Button:** Diberikan bobot 7 jika tombol lebih dekat dari diamond terdekat dan diamond di papan $<$ 4, juga dihitung dalam fungsi `compute_priority_weights()`.
3. **Pemilihan Aksi Greedy:** Bot mengurutkan prioritas berdasarkan bobot tertinggi menggunakan fungsi `find_best_target()` dan memilih aksi pertama yang dapat dilakukan, menerapkan prinsip greedy choice.
4. **Evaluasi Teleporter:** Jika tidak ada aksi prioritas yang tersedia, bot mengevaluasi penggunaan teleporter dengan membandingkan jarak

total melalui teleporter versus jalur langsung menggunakan fungsi `evaluate_teleporters()`.

2.2.2 Menjalankan Bot Program

Untuk menjalankan bot program, langkah pertama adalah memastikan bahwa semua kebutuhan library dan dependensi telah terinstal. Beberapa library yang diperlukan untuk menjalankan bot ini termasuk :

- `random` (untuk operasi angka acak)
- `typing` (terutama `Optional` untuk tipe opsional)
- `game` (library internal yang berisi modul `logic.base` dan `models` untuk logika dan model objek permainan)
- Modul `util` internal yang menyediakan fungsi `get_direction` untuk menentukan arah gerakan)

Serta pengaturan API yang diperlukan untuk berinteraksi dengan platform yang relevan. Berikut adalah tahapan-tahapan untuk menjalankan bot:

1. Pastikan API key sudah dikonfigurasi dengan benar.
2. Lakukan instalasi library yang dibutuhkan menggunakan pip, misalnya: `pip install library-name`.
3. Jalankan bot dengan command tertentu atau melalui script Python yang sudah dipersiapkan.
4. Pastikan bot berfungsi dengan memverifikasi status atau melakukan testing sederhana.

2.2.3 Fitur Tambahan atau Aspek Lain

Bot KRLBot memiliki beberapa fitur tambahan dan aspek khusus yang meningkatkan performanya:

1. **Sistem Deteksi dan Penghindaran Musuh:** Bot dilengkapi dengan fungsi `find_enemy_in_range_to_tackle()` dan `avoid_adjacent_enemies()` yang memungkinkan bot untuk mendeteksi keberadaan musuh di sekitar dan memilih jalur aman untuk menghindari tackle yang merugikan.

2. **Evaluasi Teleporter Cerdas:** Fitur `evaluate_teleporters()` memungkinkan bot untuk mengevaluasi efisiensi penggunaan teleporter dengan membandingkan jarak total melalui teleporter versus jalur langsung, termasuk pertimbangan keamanan ketika membawa banyak diamond.
3. **Sistem Prioritas Dinamis:** Bot menggunakan sistem pembobotan yang dapat beradaptasi dengan kondisi permainan yang berubah, memberikan fleksibilitas dalam pengambilan keputusan sesuai dengan situasi yang dihadapi, melalui fungsi `compute_priority_weights()`.
4. **Strategi Tackle Agresif:** Bot dapat secara otomatis mendeteksi dan menyerang lawan yang berada dalam jangkauan satu langkah jika lawan tersebut membawa diamond menggunakan fungsi `find_enemy_in_range_to_tackle()`, memberikan keuntungan taktis dalam permainan.
5. **Optimasi Performa:** Penggunaan jarak Manhattan untuk semua kalkulasi jarak dalam fungsi `manhattan_distance()` memberikan efisiensi komputasi yang baik, memungkinkan bot untuk membuat keputusan dengan cepat tanpa overhead yang berlebihan.
6. **Handling Edge Cases:** Bot dilengkapi dengan validasi posisi dan penanganan situasi khusus seperti ketika tidak ada target yang tersedia, bot akan default kembali ke base position sebagai strategi fallback dalam fungsi `find_best_target()`.

BAB III
APLIKASI STRATEGI *GREEDY*

3.1 Proses *Mapping*

Proses mapping dimulai dari pembacaan objek-objek pada papan permainan (`Board`) yang terdiri dari diamond, bot lawan, base, tombol merah, dan teleporter. Mapping ini dilakukan untuk memahami kondisi lingkungan dan mendasari pergitungan prioritas greedy. Berikut langkah-langkah mapping yang dilakukan oleh KRLBot:

1. **Membaca Posisi Bot dan Base:** Mengakses posisi bot pemain dan base-nya dari properti `board_bot`.
2. **Deteksi Objek Papan:** Mengambil data dari `board.game_bjects` untuk mendeteksi:
   - Diamond (type: `DiamondGameObject`)
   - Tombol merah (type: `DiamondButtonGameObject`)
   - Teleporter (type: `TeleporterGameObject`)
3. **Deteksi Musuh:** Mengambil posisi bot lawan dari `board.bots` dan mengevaluasi jumlah diamond yang dibawa.
4. **Perhitungan Jarak:** Menggunakan jarak Manhattan (`manhattan_distance`) untuk semua evaluasi terhadap objek penting.

3.2 Eksplorasi Alternatif Solusi Greedy

Alternatif strategi greedy yang dipertimbangkan dalam pengembangan bot:

1. **Risk-Aware Greedy:** Mempertimbangkan risiko seperti banyaknya musuh di sekitar sebelum menyerang atau mengambil diamond.
2. **Predictive Scoring:** Mengevaluasi skor masa depan dari beberapa langkah ke depan, bukan hanya langkah saat ini.
3. **Cluster-Based Targeting:** Mengincar area dengan banyak diamond sekaligus, bukan satu per satu.
4. **Blocking Strategy:** Memblokir rute musuh untuk mengganggu strategi mereka.

5. **Decay-Based Value:** Semakin jauh jarak, semakin kecil nilai target (value decay), membuat bot lebih realistis memilih target.

3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy

Tabel 3.1: Perbandingan Alternatif Solusi Greedy

Strategi	Kecepatan Eksekusi	Efektivitas Skor	Kompleksitas
Risk-Aware Greedy	Sedang	Sedang	Sedang
Predictive Scoring	Lambat	Tinggi	Tinggi
Cluster-Based Targetinng	Sedang	Sedang	Sedang
Blocking Strategy	Sedang	Sedang	Sedang
Decay-Based Value	Sedang	Sedang	Sedang

3.3.1 Strategi Greedy yang Dipilih

Strategi utama yang dipilih adalah **Greedy with Dynamic Prioritization**, yaitu bot membuat keputusan berdasarkan bobot prioritas yang dihitung berdasarkan:

- Jumlah diamond yang dibawa: prioritas kembali ke base.
- Jarak ke musuh yang membawa diamond: prioritas tackle.
- Banyaknya diamond di papan: prioritas pengambilan diamond.
- Jarak ke tombol merah: prioritas tekan tombol jika diamond sedikit.

Dengan pendekatan ini, bot lebih adaptif terhadap situasi dan dapat menyesuaikan trategi secara dinamis.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma *Greedy*

4.1.1 Pseudocode

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

```
function GreedyDecision(board_bot, board):

  if enemy_can_be_tackled_immediately:
    return move_to(position_of_enemy_to_tackle)

  goal_position = find_best_objective_target(board_bot, board)

  if an_enemy_is_adjacent AND bot_is_not_currently_tackling_that_enemy:
    available_safe_moves = get_list_of_safe_directions_away_from_enemies()
    if available_safe_moves_is_not_empty:
      return
      move_in_randomly_selected_safe_direction_from(available_safe_moves)

  if goal_position_is_defined_and_valid:
    return move_toward(goal_position)
  else:
    return idle_move
```

4.1.2 Penjelasan Alur Program

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

1. **Evaluasi Alur Program:** Jika ada musuh dalam radius tackle (1 kotak) dan membawa diamond, langsung serang.
2. **Hitung Bobot Prioritas:** Fungsi `compute_priority_weights` menghitung

bobot untuk:

- Return ke base (jika diamond ≥ 2)
- Attack (jika musuh dekat)
- Collect (tergantung jumlah diamond di papan)
- Button (jika lebih dekat dari diamond dan diamond < 4)

3. **Urutkan dan Eksekusi Prioritas:** Mengurutkan aksi berdasarkan bobot tertinggi dan mengambil aksi pertama yang mungkin dilakukan.
4. **Evaluasi Teleporter:** Jika tidak ada aksi optimal, evaluasi apakah menggunakan teleporter lebih hemat langkah.
5. **Penghindaran Musuh:** Jika musuh dekat dan tidak bisa tackle, gunakan `get_safe_directions` untuk berpindah ke posisi aman.

4.2 Struktur Data yang Digunakan

Untuk mendukung implementasi algoritma greedy, digunakan beberapa struktur data berikut:

1. **List**: Menyimpan daftar diamond, bot, dan objek lainnya di papan.
2. **Dict**: Menyimpan bobot prioritas (`weights`).
3. **Boolean**: Validasi kondisi aman/tidak, musuh dekat/tidak.
4. **Tuple**: Representasi arah gerakan (`dx, dy`) dan posisi (`x,y`).
5. **Optional[Position]**: Menyimpan target tujuan bot, bisa `none` jika tidak ada target yang cocok.

Struktur data ini memungkinkan program berjalan secara efisien dalam memproses data berdasarkan strategi greedy.

4.3 Pengujian Program

4.3.1 Skenario Pengujian

Pengujian dilakukan untuk membandingkan performa bot KRLBot (menggunakan strategi greedy with dynamic prioritization) melawan bot RandomLogic (strategi acak), dalam dua skenario waktu berbeda: 60 detik dan 120 detik. Setiap skenario diuji sebanyak 5 kali, dengan fokus pada akumulasi skor (jumlah diamond yang berhasil

dikumpulkan oleh masing-masing bot).

Tabel 4.1: Perbandingan jumlah diamond pada waktu 60s dan 120s

Sesi	Jumlah Diamond (60s)		Jumlah Diamond (120s)	
	KRLBot	RandomLogic	KRLBot	RandomLogic
1	13	0	13	4
2	9	0	19	0
3	16	0	25	5
4	6	4	21	1
5	10	2	19	2

4.3.2 Hasil Pengujian dan Analisis

Berdasarkan pengujian yang dilakukan, berikut analisis performa dari kedua bot:

1. **Perbandingan rata-rata skor**:
   - Pada durasi 60 detik, KRLBot memperoleh skor rata-rata 10.8, jauh lebih tinggi dibandingkan RandomLogic yang hanya 1.2 (sekitar 9 kali lebih besar).
   - Pada durasi 120 detik, KRLBot meningkat menjadi 19.4, sedangkan RandommLogic naik sedikit menjadi 2.4 (sekitar 8 kali lebih besar).
2. **Konsistenti**: Skor KRLBot relatif stabil walau ada naik turun, contohnya di 60 detik paling rendah di 6 dan tertinggi di 16 diamond. Untuk 120 detik, skor KRLBot meningkat dan paling rendah 13 dengan tertinggi di 25 diamond.
3. **Pengaruh waktu terhadap skor**: KRLBot memanfaatkan waktu tambahan dengan baik, skor rata-rata meningkat dari 10.8 ke 19.4 saat durasi dilipatgandakan. RandomLogic hanya naik dari 1.6 ke 2.4, artinya strategi acak tidak berpengaruh banyak walaupun waktu bermain lebih lama.
4. **Kesimpulan**: Strategi greedy yang dipakai KRLBot (greedy with dynamic prioritization) terbukti lebih unggul dan efisien dibandingkan strategi yang dipakai RandomLogic (strategi acak), terutama dapat dilihat dari skor yang jauh lebih tinggi dan kemampuan beradaptasi dengan waktu bermain yang lebih panjang.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan implementasi dan pengujian yang telah dilakukan, bot KRLBot yang menerapkan algoritma Greedy with Dynamic Prioritization menunjukkan performa yang jauh lebih unggul dibandingkan dengan bot RandomLogic yang menggunakan strategi acak. Pada pengujian dengan durasi 60 detik, KRLBot berhasil mengumpulkan skor rata-rata 10.8 diamond, sementara RandomLogic hanya mencapai 1.2 diamond. Keunggulan ini semakin signifikan pada durasi 120 detik, di mana KRLBot mencapai skor rata-rata 19.4, berbanding 2.4 untuk RandomLogic. Hasil ini mengindikasikan bahwa KRLBot sekitar 8 hingga 9 kali lebih efektif dalam mengumpulkan diamond.

Strategi prioritas dinamis yang diimplementasikan, dengan memperhitungkan jumlah diamond yang dibawa, jarak ke musuh yang membawa diamond, ketersediaan diamond di papan, dan kedekatan dengan tombol merah, terbukti efektif dalam beradaptasi dengan berbagai kondisi permainan. KRLBot juga menunjukkan kemampuan untuk memanfaatkan waktu tambahan secara efisien, terlihat dari peningkatan skor yang signifikan saat durasi permainan diperpanjang. Dengan demikian, dapat disimpulkan bahwa algoritma Greedy with Dynamic Prioritization merupakan pendekatan yang solid dan efisien untuk pemecahan masalah dalam permainan Diamonds ini.

5.2 Saran

Berdasarkan hasil analisis dan pengujuian pada Subbab 4.3, berikut beberapa saran yang dapat dipertimbangkan untuk pengembangan bot atau Tugas Besar selanjutnya:

- **Adaptasi Bobot Prioritas yang Lebih Dinamis**: Sistem prioritas saat ini menggunakan bobot yang telah ditentukan. Untuk pengembangan lebih lanjut, bisa dipertimbangkan mekanisme dimana bot dapat menyesuaikan bobot-bobot ini secara dinamis berdasarkan fase permainan (awal, tengah, akhir) atau berdasarkan perilaku bot lawan yang paling sering ditemui.

- **Penyempurnaan Logika Penghindaran Musuh**: Mensikipun bot sudah memiliki mekanisme penghindaran musuh (`get_safe_directions`) dan deteksi musuh untuk tackle (`find_enemy_in_range_to_tackle`), strategi penghindaran bisa lebih ditingkatkan. Misal dengan memprediksi pergerakan musuh beberapa langkah kedepan atau memepertimbangkan rute kabur yang paling optimal, tidak hanya sekedar mencari kotak aman terdekat secara acak dari pilihan yang ada.
- **Optimasi Penggunaan Teleporter Lebih Lanjur**: Evaluasi teleporter saat ini sudah mempertimbankan efisiensi jarak dan keamanan saat membawa banyak diamond. Namun, bisa di eskplorasi penggunaan teleporter untuk tujuan strategis lain, seperti memancing lawan.

DAFTAR PUSTAKA

[1] Steven S. Skiena. *The Algorithm Design Manual*. Springer, 2008.

[2] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, et al. *Introduction to Algorithms*. MIT Press, 2009.

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 KRL**.

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 Demo KRL**.