PEMANFAATAN ALGORITMA GREEDY DALAM APLIKASI PERMAINAN BOT DIAMOND

Tugas Besar

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

Oleh: Kelompok Valerian:

Ahmad Ali Mukti	123140155
Falent Antonius Panjaitan	123140124
Nayla Devina Febrianti	123140061

Dosen Pengampu: Imam Ekowicaksono, S.Si., M.Si.

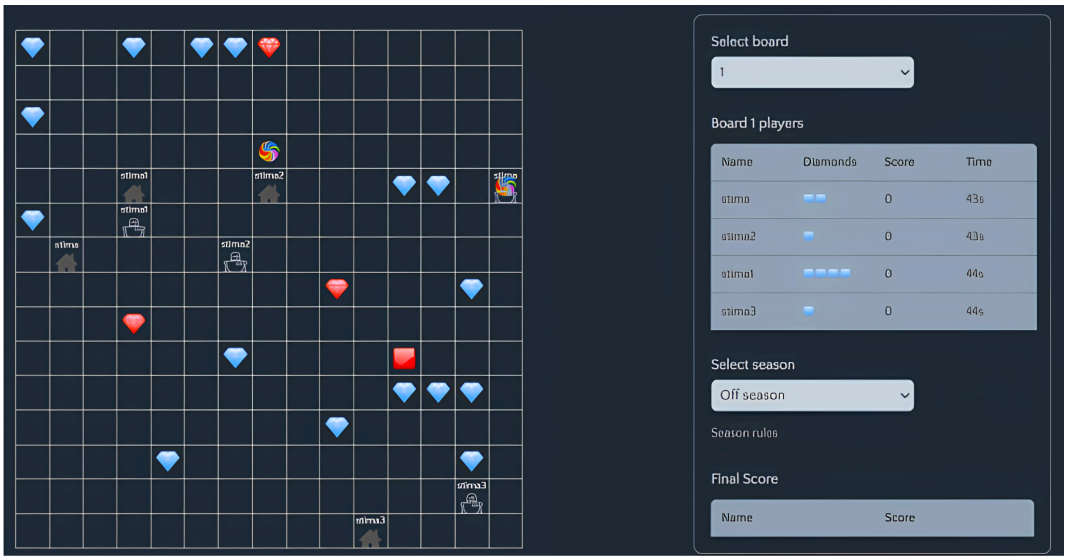
PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SUMATERA

2025

DAFTAR ISI

BAB I
DESKRIPSI TUGAS

Diamonds merupakan suatu *programming challange* dimana bot yang kita kembangkan akan bersaing dengan bot milik pemain lain. Setiap peserta akan membuat sebuah bot dengan tujuan utama untuk mengumpulkan sebanyak mungkin diamond. Namun, proses pengumpulan diamond ini tidak akan mudah, karena akan ada berbagai rintangan yang membuat permainan menjadi lebih menarik dan menantang. Untuk bisa memenangkan kompetisi ini, setiap pemain perlu merancang dan menerapkan strategi yang efektif pada bot mereka masing-masing. Berikut ini detail lebih lanjut mengenai aturan permainan dan cara kerja permainan.



Gambar 1. *Permainan Diamonds*

Komponen - komponen dari permainan Diamonds :

- Diamonds

  Untuk memenangkan dan pertandingan, kita harus mengumpulkan diamond sebanyak-banyaknya. Ada 2 jenis diamond, yaitu diamond biru yang bernilai 1 poin dan diamond merah bernilai 2 poin.
- Red Button/Diamond Button

  Ketika red button ini dilewati/dilangkahi, semua diamond akan di generate kembali dengan posisi acak. Posisi red button ini juga akan berubah secara acak jika dilewati/dilangkahi.
- Teleporters

  Ada 2 teleporter yang saling terhubung, jika bot melewatinya maka bot akan berpindah ke teleporter lainnya.
- Bots & Bases

  Kita akan menggerakan bot untuk mendapatkan diamond sebanyak banyaknya. Setiap bot memiliki base yang digunakan untuk menyimpan diamond yang sedang dibawa. Ketika diamond disimpan ke base, maka score bot akan bertambah senilai diamond yang dikumpulkan dan inventory bot menjadi kosong.
- Inventory

  Bot memiliki inventory yang berfungsi untuk tempat penyimpanan diamond sementara yang sedang dibawa. Inventory memiliki kapasitas maksimum sehingga sewaktu waktu bisa penuh. Bot bisa menyimpan isi inventory ke base supaya inventory tidak penuh.

Berikut cara kerja dari 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 menimpa posisi bot B, bot B akan dikirim ke home base dan semua diamond pada inventory bot B akan hilang, diambil masuk ke inventory bot A (istilahnya tackle).
7. Selain itu, terdapat beberapa fitur tambahan seperti teleporter dan red button yang dapat digunakan apabila anda menuju posisi objek tersebut.
8. Apabila waktu seluruh bot telah berakhir, maka permainan berakhir. Score masing-masing pemain akan ditampilkan pada tabel Final Score di sisi kanan layar.

BAB II
LANDASAN TEORI

2.1 Dasar Teori

Algoritma greedy merupakan salah satu pendekatan penyelesaian masalah yang dilakukan dengan cara mengambil keputusan terbaik pada setiap langkah secara lokal dengan harapan keputusan tersebut menghasilkan solusi global yang optimal. Pendekatan ini bersifat myopic (berpandangan pendek) karena tidak mempertimbangkan konsekuensi jangka panjang dari keputusan yang diambil saat ini.[1]

Ciri khas utama algoritma greedy adalah proses pemilihan solusi dilakukan secara bertahap, di mana pada setiap tahap algoritma:

- Memilih elemen terbaik dari himpunan kandidat (selection function),
- Memastikan elemen tersebut layak untuk dijadikan bagian solusi (feasibility function),
- Memasukkan elemen tersebut ke dalam himpunan solusi sementara,
- Melakukan pengecekan apakah solusi telah lengkap (solution function),
- Berusaha memaksimalkan atau meminimalkan fungsi objektif tergantung konteks permasalahan.

Meskipun tidak selalu memberikan solusi optimal secara keseluruhan, algoritma greedy sangat efisien untuk masalah-masalah tertentu yang memiliki sifat greedy-choice property dan optimal substructure, seperti algoritma Dijkstra, Prim, dan Huffman coding.[2]

2.2 Cara Kerja Program

Permainan Diamonds merupakan simulasi strategi berbasis bot, di mana setiap peserta mengembangkan algoritma untuk mengendalikan bot dengan tujuan mengumpulkan diamond sebanyak mungkin demi meraih skor tertinggi. Bot harus mampu beradaptasi terhadap kondisi dinamis peta permainan, seperti munculnya diamond secara acak, keberadaan bot lawan, serta elemen-elemen khusus seperti red button dan teleporter.

Secara umum, program dirancang dengan mengimplementasikan algoritma greedy yang dikombinasikan dengan pengambilan keputusan berbasis heuristik. Bot akan melakukan evaluasi terhadap keadaan sekitar dan menentukan aksi terbaik secara iteratif berdasarkan strategi yang ditetapkan.

2.2.1 Cara Implementasi Program

Program dibangun menggunakan bahasa Python dan dikendalikan oleh modul strategi yang menentukan arah gerak bot. Strategi greedy yang diimplementasikan memungkinkan bot untuk memilih jalur atau target diamond yang dinilai paling menguntungkan berdasarkan berbagai pertimbangan, seperti:

- Nilai diamond.
- Jarak menuju target.
- Kapasitas inventory.
- Sisa waktu permainan.
- Posisi lawan dan peluang tackle.

Pemilihan strategi dapat berupa:

- Greedy berdasarkan nilai diamond (Greedy by Value).
- Greedy berdasarkan jarak (Greedy by Distance).
- Kombinasi keduanya (Value/Distance Ratio).
- Strategi khusus seperti kembali ke base secara terjadwal (Strategic Return) atau penggunaan tombol reset.

2.2.2 Mekanisme Eksekusi Bot

Pada setiap giliran, bot akan melakukan :

1. Mengamati kondisi sekitar dengan memindai elemen penting pada peta.
2. Mengevaluasi kondisi internal (inventory, waktu tersisa, skor sementara).
3. Menjalankan fungsi seleksi untuk menentukan langkah selanjutnya.
4. Mengambil keputusan terbaik: mengejar diamond, kembali ke base, menekan reset, atau menghindari bot lawan.
5. Memperbarui status setelah langkah dijalankan.

2.2.3 Komponen Pendukung

Program ini juga dilengkapi dengan sejumlah komponen tambahan untuk mendukung efisiensi dan fleksibilitas, antara lain:

- Fungsi penelusuran jalur (pathfinding), seperti BFS atau A*, untuk mencari rute optimal ke target.
- Fungsi heuristik untuk mengevaluasi nilai dari setiap aksi yang memungkinkan.
- Sistem pengambilan keputusan berprioritas yang menetapkan aksi berdasarkan urutan kepentingan (misalnya: kembali ke base lebih penting daripada mengambil diamond jika inventory hampir penuh atau waktu hampir habis).

BAB III
APLIKASI STRATEGI *GREEDY*

3.1 Proses Mapping

Secara umum, permainan Diamonds merupakan permainan untuk mengumpulkan diamond yang merupakan representasi poin sebanyak-banyaknya. Bot yang sudah mengumpulkan diamond pada inventory akan mendapatkan poin dengan kembali ke base dan menyimpan diamond tersebut, sehingga inventory kembali kosong dan bot mencari lagi diamond.

Setiap strategi greedy yang digunakan memiliki fokus utama dan pendekatan langkah yang berbeda-beda, dan dapat dideskripsikan sebagai berikut,

- Himpunan kandidat, C : berisi kandidat yang akan dipilih pada setiap Langkah.
- Himpunan solusi, S : berisi kandidat yang sudah dipilih.
- Fungsi solusi: menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi.
- Fungsi seleksi (selection function): memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik.
- Fungsi kelayakan (feasible): memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak).
- Fungsi obyektif : memaksimumkan atau meminimumkan.

3.2 Eksplorasi Alternatif Solusi Greedy

Dalam menganalisis penyelesaian persoalan diamond ini. Kami telah mengidentifikasi beberapa alternatif solusi greedy yang dapat dipertimbangkan untuk meningkatkan performa bot dalam permainan diamond ini. Berikut beberapa alternatif solusi yang kami dapatkan :

3.2.1. Greedy by Value

Greedy by value Strategi greedy ini memilih jalur dengan total poin diamond terbesar, tanpa memperhitungkan jarak yang ditempuh. Kelebihannya, bot dapat mencapai total poin

diamond yang tinggi dalam waktu singkat dan mudah diimplementasikan. Kekurangannya, bot mungkin menempuh jarak yang jauh, menghabiskan waktu dan energi, serta berisiko ditabrak (di-tackle) oleh bot lain.

Mapping Elemen Greedy by Value :

- Himpunan Kandidat (C):
    - Semua diamond yang dapat dijangkau pada board.
    - Setiap langkah (atas, bawah, kiri, kanan) yang mungkin diambil bot.
- Himpunan Solusi (S):
    - Urutan langkah yang diambil bot.
    - Kumpulan diamond yang berhasil dikumpulkan di inventory.
- Fungsi Solusi:
    - Bot telah kembali ke base dengan diamond di inventory, atau waktu permainan habis.
    - Nilai total diamond yang dikumpulkan di base.
- Fungsi Seleksi (Heuristic):
    - Pilih langkah yang menuju ke diamond dengan nilai tertinggi yang dapat dijangkau, atau ke kumpulan diamond terdekat yang memiliki total nilai tertinggi. Abaikan jarak jika ada diamond bernilai sangat tinggi. Jika inventory penuh, kandidatnya adalah langkah menuju base.
- Fungsi Kelayakan (Feasible):
    - Langkah yang dipilih valid (tidak keluar board, tidak menabrak tembok jika ada).
    - Bot memiliki kapasitas inventory untuk mengambil diamond.
    - Waktu permainan masih tersisa.
- Fungsi Objektif:
    - Memaksimalkan total skor (nilai diamond yang berhasil disimpan di base)

3.2.2 Greedy by Distance

Greedy by distance Strategi greedy ini memilih jalur dengan jarak terpendek, tanpa memperhitungkan total poin diamond yang diperoleh. Kelebihannya, bot menghemat waktu dengan menempuh jarak pendek dan memiliki risiko tertabrak bot lain yang lebih rendah. Kekurangannya, bot mungkin mendapatkan total poin diamond yang rendah dan melewatkan diamond bernilai tinggi.

Mapping Elemen by Distance :

- Himpunan Kandidat (C):
    - Semua diamond yang dapat dijangkau pada board.
    - Setiap langkah (atas, bawah, kiri, kanan) yang mungkin diambil bot.

- Himpunan Solusi (S):
    - Urutan langkah yang diambil bot.
    - Kumpulan diamond yang berhasil dikumpulkan di inventory.
- Fungsi Solusi:
    - Bot telah kembali ke base dengan diamond di inventory, atau waktu permainan habis.
    - Nilai total diamond yang dikumpulkan di base.
- Fungsi Seleksi (Heuristic):
    - Pilih langkah yang menuju ke diamond terdekat (jarak Euclidean terpendek), tanpa mempedulikan nilai diamond tersebut. Jika inventory penuh, kandidatnya adalah langkah menuju base (juga dengan jarak terpendek).
- Fungsi Kelayakan (Feasible):
    - Langkah yang dipilih valid.
    - Bot memiliki kapasitas inventory.
    - Waktu permainan masih tersisa.
- Fungsi Objektif:
    - Memaksimalkan total skor dengan asumsi pengumpulan cepat akan lebih efisien. (Secara lokal, meminimalkan jarak ke diamond berikutnya).

3.2.3 Greedy by Value Combination Distance

Greedy by value combination distance Strategi greedy ini menggabungkan kedua pendekatan sebelumnya dengan memilih jalur yang memiliki kombinasi terbaik antara total poin diamond dan jarak yang ditempuh, tetapi tidak mempertimbangkan pergerakan bot lain. Kelebihannya, bot mencapai keseimbangan antara total poin diamond dan jarak yang ditempuh, serta lebih adaptif terhadap situasi berbeda. Kekurangannya, implementasi algoritma ini lebih kompleks dan penentuan bobot untuk total poin diamond, jarak yang ditempuh dapat menjadi rumit, dan risiko ditabrak oleh bot lain lebih tinggi.

Mapping Elemen Greedy by Distance :

- Himpunan Kandidat (C):
    - Semua diamond yang dapat dijangkau.
    - Setiap langkah yang mungkin diambil.
- Himpunan Solusi (S):
    - Urutan langkah yang diambil.
    - Kumpulan diamond yang dikumpulkan.
- Fungsi Solusi:
    - Bot kembali ke base atau waktu habis.
    - Nilai total diamond di base.

- Fungsi Seleksi (Heuristic):
    - Pilih langkah menuju diamond yang memaksimalkan rasio Nilai_Diamond / Jarak_ke_Diamond (atau fungsi kombinasi lain, misal w1*Nilai - w2*Jarak). Jika inventory penuh, pilih langkah menuju base.
- Fungsi Kelayakan (Feasible):
    - Langkah valid.
    - Inventory belum penuh (kecuali tujuan adalah base).
    - Waktu tersisa.
- Fungsi Objektif:
    - Memaksimalkan total skor.

3.2.4 Greedy by Tackle

Greedy by tackle Strategi greedy ini hanya berfokus untuk mengejar bot lain untuk mendapatkan poin (tackle) memiliki keuntungan dan kekurangan. Keuntungannya, bot mendapatkan poin tambahan, menghambat bot lain, dan memberikan keuntungan strategis. Kekurangannya, bot mengabaikan diamond, berisiko tertabrak, dan kurang efisien apabila terdapat banyak bot dalam satu board

Mapping Elemen Greedy by Tackle :

- Himpunan Kandidat (C):
    - Semua bot lawan yang dapat dijangkau.
    - (Sebagai fallback) Diamond terdekat/bernilai.
    - Setiap langkah yang mungkin diambil.
- Himpunan Solusi (S):
    - Urutan langkah yang diambil.
    - Kumpulan diamond yang dikumpulkan (dari map atau hasil tackle).
- Fungsi Solusi:
    - Bot kembali ke base atau waktu habis.
    - Nilai total diamond di base.
- Fungsi Seleksi (Heuristic):
    - Pilih langkah menuju bot lawan yang paling "menguntungkan" untuk di-tackle (misalnya, bot lawan yang membawa banyak diamond dan berada dalam jarak dekat). Jika tidak ada target tackle yang baik, bisa beralih ke pengumpulan diamond biasa. Jika inventory penuh (dari tackle), pilih langkah menuju base.
- Fungsi Kelayakan (Feasible):
    - Langkah valid.
    - Bot target dapat di-tackle.
    - Waktu tersisa.
    -

- Fungsi Objektif:
    - Memaksimalkan total skor (termasuk diamond dari tackle).

3.2.5 Greedy by Reset Map Button

Strategi greedy ini berfokus pada penggunaan tombol "reset map" untuk mendapatkan konfigurasi diamond yang lebih menguntungkan atau untuk mengganggu progres bot lain.

Mapping Elemen Greedy by Reset Map Button:

- Himpunan Kandidat (C):
    - Langkah menuju tombol "reset map".
    - (Sebagai fallback jika reset tidak diinginkan/tidak mungkin) Diamond terdekat/bernilai.
    - (Sebagai fallback jika inventory penuh) Langkah menuju base.
    - Setiap langkah yang mungkin diambil.
- Himpunan Solusi (S):
    - Urutan langkah yang diambil.
    - Kumpulan diamond yang dikumpulkan.
    - Frekuensi penggunaan tombol reset.
- Fungsi Solusi:
    - Bot kembali ke base atau waktu habis.
    - Nilai total diamond di base.
- Fungsi Seleksi (Heuristic):
    - Pilih langkah menuju tombol "reset map" jika kondisi map saat ini dinilai kurang menguntungkan (misalnya, sedikit diamond, diamond jauh, banyak diamond terkumpul di area lawan yang kuat, atau diamond terkumpul di area yang sulit dijangkau bot kita).
    - Pertimbangkan juga jika bot memiliki sedikit diamond dan butuh "gebrakan baru".
    - Jika menekan tombol reset tidak menjadi prioritas (misalnya map cukup baik, atau baru saja di-reset), beralih ke pengumpulan diamond biasa.
    - Jika inventory penuh, pilih langkah menuju base.
- Fungsi Kelayakan (Feasible):
    - Langkah valid.
    - Tombol reset map dapat dijangkau dan diaktifkan (misalnya, tidak ada cooldown, tidak ada penalti yang terlalu besar).
    - Waktu tersisa cukup untuk mengambil manfaat dari map baru setelah reset.

- Fungsi Objektif:
    - Memaksimalkan total skor, dengan asumsi bahwa penggunaan reset map pada waktu yang tepat akan menghasilkan perolehan diamond yang lebih tinggi secara keseluruhan dibandingkan tidak mereset.

3.2.6 Greedy by Strategic Time-Constrained Return

Strategi greedy ini berfokus pada pengumpulan diamond sebanyak mungkin, namun dengan perhitungan cermat untuk kembali ke base sebelum waktu habis atau sebelum inventory penuh, sehingga memaksimalkan diamond yang berhasil diamankan..

Mapping Elemen Greedy by Strategic Time-Constrained Return:

- Himpunan Kandidat (C):
    - Diamond terdekat/bernilai yang bisa diambil dan masih memungkinkan kembali ke base tepat waktu.
    - Langkah menuju base.
    - Setiap langkah yang mungkin diambil.
- Himpunan Solusi (S):
    - Urutan langkah yang diambil.
    - Kumpulan diamond yang dikumpulkan dan berhasil dibawa ke base.
- Fungsi Solusi:
    - Bot kembali ke base atau waktu habis.
    - Nilai total diamond di base.
- Fungsi Seleksi (Heuristic):
    - Prioritaskan pengumpulan diamond terdekat/bernilai.
    - Secara kontinu hitung waktu_tempuh_ke_base.
    - Jika u_permainan <= waktu_tempuh_ke_base + margin_keamanan_waktu, maka prioritas utama adalah kembali ke base.
    - Jika kapasitas_inventory - inventory_saat_ini <= sisa_waktthreshold_minimum_pengambilan (misalnya, hanya cukup untuk 1 diamond kecil lagi sebelum penuh), dan diamond terdekat tidak cukup bernilai untuk perjalanan pulang-pergi, maka prioritas adalah kembali ke base.
    - Jika tidak ada kondisi di atas yang terpenuhi, pilih langkah menuju diamond yang paling menguntungkan (nilai/jarak) yang masih memungkinkan kembali ke base setelah mengambilnya.
- Fungsi Kelayakan (Feasible):
    - Langkah valid.
    - Waktu tersisa cukup untuk mengambil diamond (jika targetnya diamond) DAN kembali ke base.
    - Inventory tidak penuh (kecuali targetnya base).

- Fungsi Objektif:
    - Memaksimalkan total diamond yang berhasil diamankan di base sebelum waktu permainan habis.

3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy

Dalam menganalisis berbagai pendekatan greedy untuk bot pengumpul diamond, ditemukan perbandingan efisiensi komputasi dan efektivitas strategi dalam memaksimalkan skor. Berikut analisisnya

3.3.1 Greedy by Value

- Efisiensi:
    - Kompleksitas: O(D) per langkah, di mana D adalah jumlah diamond yang terlihat/dapat dijangkau. Perlu mengevaluasi nilai semua kandidat diamond.
    - Perhitungan: Sederhana, hanya membandingkan nilai diamond.
    - Optimal untuk: Papan di mana nilai diamond sangat bervariasi dan beberapa diamond memiliki nilai yang sangat tinggi dibandingkan yang lain.
- Efektivitas:
    - Potensi: Dapat menghasilkan skor tinggi jika diamond bernilai tinggi mudah dijangkau.
    - Kelemahan: Sangat tidak efisien dalam hal pergerakan. Mengabaikan jarak, waktu tempuh, dan energi yang mungkin terbuang. Risiko tinggi untuk di-tackle karena perjalanan jauh atau kehabisan waktu sebelum sampai base.
    - Cocok untuk: Skenario di mana ada diamond "jackpot" yang nilainya jauh melebihi yang lain, dan bot memiliki keunggulan kecepatan atau energi, atau pada awal permainan ketika risiko di-tackle rendah.

3.2.2. Greedy by Distance

- Efisiensi:
    - Kompleksitas: O(D) per langkah untuk menghitung jarak ke semua diamond yang terlihat.
    - Perhitungan: Perhitungan jarak (misalnya, Manhattan atau Euclidean) ke setiap diamond.
    - Optimal untuk: Papan dengan banyak diamond yang tersebar merata.
- Efektivitas:
    - Potensi: Cepat mengumpulkan diamond dalam jumlah banyak (meskipun nilainya mungkin kecil), menghemat waktu dan energi. Risiko di-tackle lebih rendah karena pergerakan cepat ke target terdekat.

- Kelemahan: Cenderung mengumpulkan diamond bernilai rendah jika itu yang terdekat, berpotensi melewatkan diamond bernilai tinggi yang sedikit lebih jauh. Skor akhir mungkin tidak maksimal.
- Cocok untuk: Awal permainan untuk mengisi inventory dengan cepat, atau pada map dengan diamond yang nilainya tidak terlalu beragam. Efektif jika kecepatan pengumpulan lebih penting daripada nilai per diamond.

3.3.3 Greedy by Value Combination Distance

- Efisiensi:
  - Kompleksitas: O(D) per langkah, karena perlu menghitung rasio (atau fungsi kombinasi) untuk setiap diamond.
  - Perhitungan: Melibatkan pembagian (Nilai/Jarak) atau operasi aritmatika lain (w1*Nilai* - *w2*Jarak) untuk setiap diamond. Sedikit lebih kompleks dari sekadar nilai atau jarak saja, tapi masih sangat cepat.
  - Optimal untuk: Kebanyakan situasi karena menyeimbangkan dua faktor krusial.
- Efektivitas:
  - Potensi: Strategi yang umumnya paling seimbang dan seringkali paling efektif. Memaksimalkan "nilai per unit usaha/jarak".
  - Kelemahan: Penentuan bobot (jika menggunakan w1*Nilai - w2*Jarak) bisa jadi tricky dan memerlukan tuning. Masih mengabaikan faktor lain seperti bot lawan atau waktu permainan secara eksplisit (kecuali ditambahkan).
  - Cocok untuk: Sebagai strategi dasar yang solid di hampir semua fase permainan dan kondisi map.

3.3.4 Greedy by Tackle

- Efisiensi:
  - Kompleksitas: O(B) per langkah untuk mengevaluasi semua bot lawan (B) sebagai target potensial. Jika fallback ke diamond, maka O(D) juga berperan.
  - Perhitungan: Memerlukan evaluasi jarak ke bot lawan, dan idealnya, estimasi diamond yang dibawa lawan.
  - Optimal untuk: Papan dengan jumlah bot lawan yang tidak terlalu banyak.
- Efektivitas:
  - Potensi: Keuntungan ganda: mendapatkan poin dari tackle dan mengurangi poin lawan. Memberikan keuntungan strategis dengan menghambat progres lawan.
  - Kelemahan: Sangat berisiko; bisa gagal tackle, di-tackle balik, atau lawan tidak membawa banyak diamond. Mengabaikan pengumpulan diamond dari map secara langsung saat fokus mengejar. Kurang efisien jika banyak bot atau target sulit ditangkap.

- Cocok untuk: Mid-late game ketika bot lain sudah mengumpulkan diamond. Saat bot kita memiliki keunggulan (misalnya, lebih cepat atau inventory lawan hampir penuh). Situasi di mana mengganggu lawan lebih penting daripada mengumpulkan diamond dari map.

3.3.5 Greedy by Reset Map Button

- Efisiensi:
  - Kompleksitas: O(1) untuk keputusan menarget tombol (karena targetnya tetap), ditambah O(D) jika fallback ke diamond.
  - Perhitungan: Evaluasi kondisi map saat ini (heuristik sederhana, misal jumlah diamond rendah atau terkonsentrasi di area lawan). Perhitungan jarak ke tombol reset.
  - Optimal untuk: Situasi di mana keputusan reset relatif jarang dilakukan.
- Efektivitas:
  - Potensi: Dapat "mengocok ulang" permainan jika map saat ini sangat tidak menguntungkan atau sudah habis. Mengganggu strategi lawan yang sudah terencana.
  - Kelemahan: Tidak ada jaminan map baru akan lebih baik. Mengorbankan waktu/energi untuk menekan tombol. Bisa jadi tindakan sia-sia jika map saat ini sebenarnya cukup baik. Memberikan kesempatan yang sama bagi semua bot.
  - Cocok untuk: Situasi putus asa ketika map saat ini sangat buruk, atau sebagai strategi kejutan untuk mengganggu lawan. Berguna jika bot memiliki sedikit diamond dan perlu "gebrakan baru".

3.3.6 Greedy by Strategic Time-Constrained Return

- Efisiensi:
  - Kompleksitas: O(D) untuk pemilihan diamond, ditambah perhitungan konstan untuk cek waktu dan kapasitas.
  - Perhitungan: Perlu menghitung jarak/waktu ke diamond dan jarak/waktu kembali ke base. Perbandingan dengan sisa waktu permainan dan kapasitas inventory.
  - Optimal untuk: Semua situasi, karena overhead komputasi untuk pengecekan waktu relatif kecil.
- Efektivitas:
  - Potensi: Sangat penting untuk memastikan diamond yang dikumpulkan berhasil diamankan. Mencegah kehilangan poin karena waktu habis atau inventory penuh saat jauh dari base.
  - Kelemahan: Bisa terlalu konservatif jika margin keamanan waktu terlalu besar, menyebabkan bot kembali terlalu dini dan melewatkan kesempatan. Membutuhkan estimasi waktu tempuh yang akurat.

- Cocok untuk: Terutama penting di mid-late game atau kapan pun inventory mendekati penuh. Merupakan komponen krusial yang sebaiknya diintegrasikan dengan strategi pengumpulan diamond lainnya.

3.4 Strategi Greedy yang Dipilih

Strategi utama yang diterapkan dalam bot ini mengadopsi pendekatan greedy berbasis kombinasi nilai dan jarak (value/distance ratio) untuk pengumpulan diamond, yang dipadukan secara krusial dengan mekanisme kembali ke markas secara strategis. Bot secara cerdas memilih diamond yang paling menguntungkan berdasarkan rasio poin per jarak efektif, dan secara proaktif memutuskan untuk kembali ke markas guna mengamankan poin. Implementasi ini menggabungkan beberapa lapisan keputusan strategis:

1. Prioritas Pengamanan Skor (Implementasi Greedy by Strategic Time-Constrained Return): Bot akan memprioritaskan kembali ke markas jika salah satu kondisi berikut terpenuhi:
   - Sisa waktu permainan di bawah ambang batas kritis (URGENCY_TIMER_LIMIT) dan bot membawa diamond.
   - Inventaris diamond sudah penuh.
   - Bot telah mengumpulkan sejumlah diamond tertentu (CASH_OUT_THRESHOLD) dan tidak ada diamond lain yang mudah dijangkau di sekitarnya (NEARBY_RADIUS).
2. Optimasi Pemilihan Target Diamond (Implementasi Greedy by Value Combination Distance): Jika tidak ada kondisi mendesak untuk kembali ke markas, bot akan mengevaluasi semua diamond yang tersedia dan memilih target yang memaksimalkan rasio nilai_poin / jarak_efektif_ke_diamond.
3. Optimalisasi Rute dengan Teleportasi: Perhitungan "jarak efektif" (baik ke diamond maupun ke markas) secara otomatis mempertimbangkan penggunaan portal teleportasi jika hal tersebut menghasilkan rute yang lebih pendek dibandingkan perjalanan langsung.

Dari segi efektivitas, strategi ini mengutamakan dalam beberapa aspek seperti:

1. Manajemen Risiko dan Pengamanan Poin: Dengan adanya URGENCY_TIMER_LIMIT, pengecekan kapasitas inventaris, dan CASH_OUT_THRESHOLD, bot secara signifikan

mengurangi risiko kehilangan diamond karena waktu habis atau karena terus mengumpulkan tanpa sempat kembali ke markas.

2. Efisiensi Akuisisi Diamond: Penggunaan rasio nilai/jarak memastikan bot tidak hanya mengejar diamond bernilai tinggi yang sangat jauh atau diamond terdekat yang nilainya sangat kecil, melainkan mencari keseimbangan optimal untuk memaksimalkan perolehan poin per unit usaha/pergerakan.
3. Optimalisasi Gerakan Lanjutan: Integrasi perhitungan rute via teleportasi memungkinkan bot untuk mengambil jalur terpendek secara dinamis, menghemat waktu dan energi, yang pada akhirnya dapat meningkatkan jumlah diamond yang dikumpulkan dan diamankan dalam satu permainan.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma Greedy

4.1.1 Pseudocode

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

class Bot_logic(BaseLogic):

    def __init__(self):
        self.URGENCY_TIMER_LIMIT = 15000
        self.CASH_OUT_THRESHOLD = 3
        self.NEARBY_RADIUS = 5
        self.gem_locations: List[GameObject] = []
        self.other_bots: List[GameObject] = []
        self.portals: List[GameObject] = []

    def scan_board_state(self, game_state: Board):
        """Mengkategorikan semua objek game di papan untuk akses
yang lebih mudah."""
        self.gem_locations.clear()
        self.other_bots.clear()
        self.portals.clear()

        for item in game_state.game_objects:
            if item.type == "DiamondGameObject":
                self.gem_locations.append(item)
            elif item.type == "BotGameObject":
                self.other_bots.append(item)
            elif item.type == "TeleportGameObject":
                self.portals.append(item)
```

```
    def calculate_dist(self, p1: Position, p2: Position) -> float:
        return ((p1.x - p2.x)**2 + (p1.y - p2.y)**2)**0.5

    def check_local_gem_presence(self, my_location: Position,
search_radius: int) -> bool:

        for gem in self.gem_locations:
            delta_x = abs(gem.position.x - my_location.x)
            delta_y = abs(gem.position.y - my_location.y)

            if delta_x <= search_radius and delta_y <=
search_radius:
                return True
        return False

    def determine_optimal_route(self, start_pos: Position, end_pos:
Position) -> Tuple[float, Position]:
        direct_route_dist = self.calculate_dist(start_pos, end_pos)

        if len(self.portals) >= 2:
            entry_portal = min(self.portals, key=lambda p:
self.calculate_dist(start_pos, p.position))
            exit_portal = min(self.portals, key=lambda p:
self.calculate_dist(end_pos, p.position))

            portal_route_dist = self.calculate_dist(start_pos,
entry_portal.position) + self.calculate_dist(exit_portal.position,
end_pos)

            if portal_route_dist < direct_route_dist:
                return portal_route_dist, entry_portal.position

        return direct_route_dist, end_pos

    def next_move(self, my_robot: GameObject, game_state: Board) ->
Tuple[int, int]:
```

```
        self.scan_board_state(game_state)
        robot_stats = my_robot.properties
        my_location = my_robot.position

        target_destination = robot_stats.base
        is_returning_home = False

        time_left = robot_stats.milliseconds_left or float('inf')
        if time_left < self.URGENCY_TIMER_LIMIT and
robot_stats.diamonds > 0:
            is_returning_home = True

        elif robot_stats.diamonds >= robot_stats.inventory_size:
            is_returning_home = True

        elif robot_stats.diamonds >= self.CASH_OUT_THRESHOLD and not
self.check_local_gem_presence(my_location, self.NEARBY_RADIUS):
            is_returning_home = True

        if is_returning_home:
            _, target_destination =
self.determine_optimal_route(my_location, robot_stats.base)
        else:
            optimal_gem_location = None
            max_value_ratio = -1.0 # Inisialisasi dengan nilai
negatif
            inventory_room = robot_stats.inventory_size -
robot_stats.diamonds

            for gem in self.gem_locations:
                gem_points = gem.properties.points or 1
                if gem_points <= inventory_room:
                    cost_to_reach, _ =
self.determine_optimal_route(my_location, gem.position)
                    cost_to_reach = max(cost_to_reach, 0.01)
                    value_per_distance = gem_points / cost_to_reach
```

```
                    if value_per_distance > max_value_ratio:
                        max_value_ratio = value_per_distance
                        optimal_gem_location = gem.position


            if optimal_gem_location:
                _, target_destination =
self.determine_optimal_route(my_location, optimal_gem_location)


        if target_destination == my_location:
            return random.choice([(1, 0), (-1, 0), (0, 1), (0, -1)])

        move_x, move_y = 0, 0
        if target_destination.x > my_location.x:
            move_x = 1
        elif target_destination.x < my_location.x:
            move_x = -1

        if target_destination.y > my_location.y:
            move_y = 1
        elif target_destination.y < my_location.y:
            move_y = -1

        if move_x != 0 and move_y != 0:
            if abs(target_destination.x - my_location.x) >
abs(target_destination.y - my_location.y):
                move_y = 0
            else:
                move_x = 0

        return move_x, move_y
```

Strategi yang digunakan menggabungkan dua prinsip utama:

- Greedy berdasarkan rasio nilai dan jarak: Bot akan memilih diamond yang memberikan nilai terbaik relatif terhadap jarak tempuh.
- Greedy berbasis waktu dan kapasitas (strategic return): Bot akan secara otomatis kembali ke base jika inventory penuh atau waktu hampir habis untuk mengamankan poin.

4.1.2 Penjelasan Alur Program

Secara garis besar, alur kerja program terbagi menjadi beberapa tahapan:

1. Inisialisasi: Bot ditempatkan secara acak di papan permainan dengan kondisi awal (skor dan inventory) nol.
2. Siklus permainan:
    - Bot mengamati lingkungan sekitar, seperti posisi diamond, posisi lawan, letak base, serta elemen khusus seperti teleporter dan red button.
    - Berdasarkan evaluasi kondisi (inventory, sisa waktu, dan posisi), bot menentukan prioritas aksi.
    - Bot memilih tindakan: mengejar diamond, kembali ke base, menekan tombol reset, atau melakukan tackle ke bot lawan.
    - Langkah dipilih secara strategis, kemudian bot memperbarui kondisinya sesuai aksi yang diambil.
3. Penyelesaian: Permainan berakhir saat waktu habis, dan skor dihitung berdasarkan diamond yang berhasil dikumpulkan dan disimpan di base.

4.2 Struktur Data yang Digunakan

Program Bot Etimo Diamonds menggunakan pendekatan pemrograman berorientasi objek (OOP) untuk merepresentasikan berbagai entitas dan logika dalam permainan. Struktur data utama dalam program ini terdiri dari dua objek sentral, yaitu GameObject dan Board, yang bekerja sama untuk menyusun dunia permainan dan pengambilan keputusan oleh bot. Selain itu, beberapa kelas tambahan digunakan untuk mendukung pengelolaan posisi, properti, serta logika permainan.

1. Objek GameObject

GameObject merupakan representasi dari semua elemen dalam game. Objek ini menjadi dasar dari berbagai jenis entitas seperti:

- DiamondGameObject: merepresentasikan diamond yang dapat diambil.
- BotGameObject: bot pemain yang dapat bergerak dan mengumpulkan diamond.

- TeleportGameObject: objek teleportasi yang memungkinkan perpindahan posisi.

Properti dalam GameObject

Setiap GameObject memiliki atribut utama, yaitu:

- type: jenis objek (misalnya "diamond", "bot", atau "teleport").
- id: identitas unik dari setiap objek.
- position: koordinat (x, y) di papan permainan.
- properties: atribut tambahan dalam bentuk objek Properties, yang dapat mencakup:
    - diamonds: jumlah diamond yang dibawa bot.
    - inventory_size: kapasitas maksimal diamond yang bisa dibawa.
    - points: nilai poin dari diamond.

2. Objek Board

Objek Board berfungsi sebagai peta permainan, yang menyimpan semua objek permainan dalam bentuk list: game_objects.

Fungsi Utama Board

- Mengelompokkan objek berdasarkan kategori (diamonds, bots, teleports) untuk memudahkan proses pencarian dan pengambilan keputusan.
- Mengelola logika permainan seperti:
    - Menghitung jarak antar objek (menggunakan algoritma seperti Manhattan distance).
    - Mendeteksi musuh di sekitar bot.
    - Menentukan strategi gerakan, misalnya menuju diamond terdekat, teleportasi, atau kembali ke base saat inventory penuh.
    - Penghindaran dan pertempuran jika ada bot musuh di sekitar.

4.3 Pengujian Program

4.3.1 Skenario Pengujian

Pengujian dilakukan dalam beberapa kondisi yang mewakili kompleksitas permainan, antara lain:

1. Peta sederhana: Diamond tersebar secara merata dan tidak ada bot lawan.
2. Peta kompleks: Adanya hambatan serta konsentrasi diamond di area tertentu.
3. Simulasi kompetitif: Diuji melawan beberapa bot lawan dengan strategi berbeda.
4. Kondisi kritis waktu: Uji performa saat waktu hampir habis.
5. Penggunaan Red Button: Menilai efektivitas penggunaan fitur reset peta.

4.3.1 Hasil Pengujian dan Analisis

Berdasarkan hasil pengujian, didapatkan bahwa strategi greedy berbasis rasio nilai terhadap jarak dan mekanisme kembali ke base secara strategis menghasilkan performa yang konsisten dan optimal dalam berbagai skenario. Beberapa temuan utama meliputi:

- Efisiensi Skor: Bot mampu menghindari kerugian akibat kehabisan waktu berkat perhitungan waktu kembali ke base.
- Adaptivitas terhadap lawan: Dalam skenario dengan bot lawan, strategi ini tetap unggul karena mengutamakan keamanan skor.
- Efektivitas tombol reset: Terbukti bermanfaat dalam mengubah dinamika permainan saat kondisi peta kurang menguntungkan.
- Optimalisasi jalur: Integrasi penggunaan teleporter mampu memangkas jarak secara signifikan, meningkatkan efisiensi gerakan.

Secara keseluruhan, pendekatan yang digunakan memberikan keseimbangan antara efisiensi waktu, nilai diamond yang diperoleh, dan strategi pengambilan risiko yang terukur.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan dari tugas besar IF2211 Strategi Algoritma ini kami berhasil menciptakan bot permainan Diamonds dengan mengimplementasikan beberapa strategi greedy yang memiliki prioritasnya masing masing. Berdasarkan strategi yang kami pilih, strategi ini memungkinkan bot untuk bertindak secara cerdas dalam pengumpulan diamond sekaligus mengamankan hasil dengan mempertimbangkan waktu dan jarak efisien. Sehingga bot ini dapat mengumpulkan diamond sebanyak banyaknya.

5.2 Saran

Strategi greedy yang kami terapkan sudah cukup efektif dalam mengoptimalkan rasio nilai terhadap jarak dan pengambilan keputusan kembali ke markas secara strategis. Ada beberapa saran untuk pengembangan bot ini :

1. Penerapan optimalisasi penggunaan teleportasi
   Penggunaan teleportasi sebaiknya mempertimbangkan faktor lain seperti posisi musuh dan waktu permainan. Hal ini dapat membantu bot dalam menghindari bentrokan yang tidak menguntungkan dan memanfaatkan peluang secara lebih efektif.
2. Integrasi Prediksi Pergerakan Musuh
   Meskipun strategi greedy yang digunakan sudah mempertimbangkan rasio nilai terhadap jarak, bot dapat ditingkatkan dengan menambahkan kemampuan untuk memprediksi pergerakan musuh. Dengan demikian, bot dapat menghindari area yang berisiko tinggi atau bersaing secara lebih efektif dalam mengumpulkan diamond

LAMPIRAN

A. Repository Github : https://github.com/Kyura01/Tubes1_Valerian

DAFTAR PUSTAKA

[1] T. H. Cormen et al., Introduction to Algorithms, 3rd ed., MIT Press, 2009.

[2] S. Dasgupta et al., Algorithms, McGraw-Hill, 2006.