PEMANFAATAN ALGORITMA GREEDY DALAM PEMBUATAN BOT PERMAINAN DIAMONDS

Tugas Besar

Oleh: Kelompok 3 (Not Understand)

Alliyah Salsabilla	123140014
Awi Septian Prasetyo	123140201
Muhammad Bimastiar	123140211

Dosen Pengampu: Winda Yulita, M.Cs.

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI SUMATERA

2025

DAFTAR ISI

BAB I
DESKRIPSI TUGAS

Diamonds merupakan suatu programming challenge yang mempertandingkan bot yang anda buat dengan bot dari para pemain lainnya. Setiap pemain akan memiliki sebuah bot dimana tujuan dari bot ini adalah mengumpulkan diamond sebanyak-banyaknya. Cara mengumpulkan diamond tersebut tidak akan sesederhana itu, tentunya akan terdapat berbagai rintangan yang akan membuat permainan ini menjadi lebih seru dan kompleks. Untuk memenangkan pertandingan, setiap pemain harus mengimplementasikan strategi tertentu pada masing-masing bot-nya.

Komponen-komponen dari permainan Diamonds antara lain:

1. Diamonds

Untuk memenangkan pertandingan, kita harus mengumpulkan diamond ini sebanyak-banyaknya dengan melewati/melangkahinya. Terdapat 2 jenis diamond yaitu diamond biru dan diamond merah. Diamond merah bernilai 2 poin, sedangkan yang biru bernilai 1 poin. Diamond akan di-regenerate secara berkala dan rasio antara diamond merah dan biru ini akan berubah setiap regeneration.

2. Red Button/Diamond Button

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

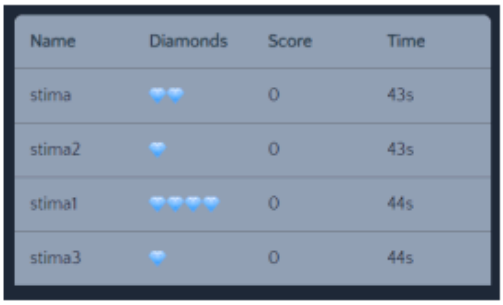
3. Teleporters

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

4. Bots and Bases

Pada game ini kita akan menggerakkan bot untuk mendapatkan diamond sebanyak banyaknya. Semua bot memiliki sebuah Base dimana Base ini akan digunakan untuk menyimpan diamond yang sedang dibawa. Apabila diamond disimpan ke base, score bot akan bertambah senilai diamond yang dibawa dan inventory (akan dijelaskan di bawah) bot menjadi kosong.

5. Inventory



Bot memiliki inventory yang berfungsi sebagai tempat penyimpanan sementara diamond yang telah diambil. Inventory ini memiliki kapasitas maksimum sehingga sewaktu waktu bisa penuh. Agar inventory ini tidak penuh, bot bisa menyimpan isi inventory ke base agar inventory bisa kosong kembali.

Permainan ini merupakan permainan berbasis web, sehingga setiap aksi yang dilakukan – mulai dari mendaftarkan bot hingga menjalankan aksi bot – akan memerlukan HTTP request terhadap API endpoint tertentu yang disediakan oleh backend. Berikut adalah urutan requests yang terjadi dari awal mula permainan.

1. Program bot akan mengecek apakah bot sudah terdaftar atau belum, dengan mengirimkan POST request terhadap endpoint /api/bots/recover dengan body berisi email dan password bot. Jika bot sudah terdaftar, maka backend akan memberikan response code 200 dengan body berisi id dari bot tersebut. Jika tidak, backend akan memberikan response code 404.
2. Jika bot belum terdaftar, maka program bot akan mengirimkan POST request terhadap endpoint /api/bots dengan body berisi email, name, password, dan team. Jika berhasil, maka backend akan memberikan response code 200 dengan body berisi id dari bot tersebut.
3. Ketika id bot sudah diketahui, bot dapat bergabung ke board dengan mengirimkan POST request terhadap endpoint /api/bots/{id}/join dengan body berisi board id yang diinginkan (preferredBoardId). Apabila bot berhasil bergabung, maka backend akan memberikan response code 200 dengan body berisi informasi dari board.
4. Program bot akan mengkalkulasikan move selanjutnya secara berkala berdasarkan kondisi board yang diketahui, dan mengirimkan POST request terhadap endpoint /api/bots/{id}/move dengan body berisi direction yang akan ditempuh selanjutnya (“NORTH”, “SOUTH”, “EAST”, atau “WEST”). Apabila berhasil, maka backend akan memberikan response code 200 dengan body berisi kondisi board setelah move tersebut. Langkah ini dilakukan terus-menerus hingga waktu bot habis. Jika waktu bot habis, bot secara otomatis akan dikeluarkan dari board.
5. Program frontend secara periodik juga akan mengirimkan GET request terhadap endpoint /api/boards/{id} untuk mendapatkan kondisi board terbaru, sehingga tampilan board pada frontend akan selalu ter-update.

BAB II
LANDASAN TEORI

2.1 Dasar Teori

Pendekatan greedy memiliki karakteristik khas, yaitu tidak mempertimbangkan keputusan yang telah diambil sebelumnya dan tidak melakukan peninjauan ulang (*backtracking*) terhadap solusi yang sedang dibangun. Sebaliknya, ia menyusun solusi secara bertahap berdasarkan keputusan lokal terbaik yang tersedia pada saat itu. Oleh karena itu, algoritma greedy umumnya lebih cepat dan efisien dalam hal waktu komputasi dibandingkan dengan metode lain seperti algoritma brute force atau pemrograman dinamis [1][2]. Namun, efisiensi tersebut datang dengan konsekuensi bahwa algoritma greedy tidak selalu menjamin menghasilkan solusi optimal, kecuali jika masalah yang diselesaikan memiliki sifat tertentu yang mendukung pendekatan ini.

Agar algoritma greedy dapat digunakan secara efektif dan menjamin solusi optimal, masalah yang dihadapi harus memiliki dua sifat utama, yaitu greedy choice property dan optimal substructure. *Greedy choice property* adalah sifat di mana pilihan terbaik secara lokal pada suatu langkah juga merupakan bagian dari solusi global yang optimal. Sedangkan *optimal substructure* berarti bahwa solusi optimal dari suatu masalah dapat dibangun dari solusi optimal submasalah-submasalahnya [1][2].

Dalam praktiknya, algoritma greedy telah terbukti sangat efektif untuk berbagai jenis masalah komputasi. Beberapa contoh klasik yang dapat diselesaikan dengan pendekatan greedy antara lain adalah masalah fractional knapsack, di mana barang dengan rasio nilai per berat tertinggi dipilih terlebih dahulu; masalah activity selection, di mana aktivitas dengan waktu selesai paling awal diprioritaskan; serta algoritma Prim dan Kruskal yang digunakan untuk menemukan *minimum spanning tree* dalam teori graf [1]. Selain itu, algoritma Huffman coding, yang digunakan dalam kompresi data, juga merupakan implementasi dari strategi greedy yang membangun pohon berdasarkan frekuensi karakter [2][3].

Walaupun demikian, penting untuk dicatat bahwa tidak semua masalah optimasi cocok diselesaikan dengan algoritma greedy. Pada kasus-kasus tertentu, pendekatan greedy dapat menghasilkan solusi yang tampak baik, tetapi sebenarnya bukan solusi optimal. Oleh karena itu, sebelum menerapkan algoritma greedy, perlu dilakukan analisis terlebih dahulu untuk

memastikan bahwa permasalahan memenuhi sifat *greedy choice property* dan *optimal substructure*. Jika tidak, maka pendekatan lain seperti dynamic programming atau backtracking mungkin lebih sesuai [2].

Secara keseluruhan, algoritma greedy merupakan pendekatan yang sederhana namun sangat kuat dalam menyelesaikan berbagai permasalahan komputasi yang kompleks, selama digunakan secara tepat pada jenis permasalahan yang sesuai. Keunggulan dalam efisiensi waktu dan kemudahan implementasi menjadikan algoritma ini banyak dipelajari dan digunakan dalam berbagai aplikasi, baik dalam dunia akademik maupun industri, seperti pengurutan tugas, penjadwalan, kompresi data, jaringan komputer, hingga pengembangan sistem cerdas [1][3].

2.2 Cara Kerja Program

ama permainan.Permainan “Diamonds” terdiri dari dua modul utama, yaitu Game Engine dan Bot Starter Pack. Game Engine berfungsi untuk menyimpan kode backend dan frontend dari permainan. Sementara itu, Bot Starter Pack berisi program untuk memanggil API yang disediakan oleh backend, program utama (main) permainan, serta yang paling penting, program logika bot.

Dalam mengimplementasikan bot, langkah pertama yang perlu dilakukan adalah membuat file baru dengan nama “Not_U” di dalam direktori “/game/logic”. Pada file tersebut, kita perlu membuat sebuah kelas yang mewarisi (inherit) dari kelas BaseLogic. Di dalam kelas ini, selain membuat constructor, kita juga harus mengimplementasikan method next_move, yang berfungsi untuk mengatur gerakan bot.

Setelah kelas selesai dibuat, langkah selanjutnya adalah mengimpor kelas tersebut ke dalam file main.py, lalu mendaftarkannya ke dalam dictionary bernama CONTROLLERS. Setelah itu, kita bisa mulai menuliskan logika bot pada method next_move. Method ini bisa dimodifikasi agar mengembalikan (return) gerakan yang ingin dilakukan bot dalam bentuk delta x dan delta y. Gerakan tersebut kemudian akan dikirim kembali ke program utama dan dijalankan dalam permainan.

Terdapat tiga komponen utama yang perlu dipahami untuk mengetahui cara kerja game ini, yaitu:

1. **Engine**, yaitu komponen yang bertugas untuk mengimplementasikan logika dan aturan dalam permainan.
2. **Runner**, yaitu komponen yang menjalankan suatu pertandingan (match) serta menghubungkan bot dengan engine.
3. **Bot**, yaitu komponen yang berperan sebagai pemain atau peserta dalam pertandingan.

Secara garis besar, alur kerja program game Etimo secara lokal adalah sebagai berikut:

1. Saat dijalankan, Runner akan meng-host sebuah match pada hostname tertentu. Untuk koneksi lokal, runner akan berjalan pada localhost:8082 dan menggunakan database lokal. Runner juga akan menampilkan tampilan papan permainan (game state).
2. Selanjutnya, Engine akan dijalankan dan melakukan koneksi ke runner melalui API gateway.
3. Bot akan mengirim request melalui API ke runner untuk setiap pergerakan yang dilakukan. Setiap URL pada API memiliki fungsi yang berbeda sesuai kebutuhan interaksi bot dengan permainan.
4. Permainan akan terus berjalan hingga waktu habis. Setelah pertandingan selesai, skor masing-masing pemain akan ditampilkan dalam tabel *Final Score* di sisi kanan layar.

2.3 Struktur Program

Dalam program *Diamonds* yang menjadi fokus tugas besar kali ini, terdapat beberapa komponen yang telah disediakan untuk mendukung jalannya permainan. Komponen-komponen tersebut antara lain:

1. Game Engine

Komponen ini terdiri dari file-file yang digunakan untuk menjalankan website serta mengatur konfigurasi Docker. Game engine juga berperan dalam menampilkan semua objek yang ada di dalam map permainan.

2. Bot Starter Pack

Komponen ini terdiri dari berbagai file yang ditulis menggunakan bahasa pemrograman Python. Pada starter pack ini, *source code* dibuat dengan menggunakan paradigma *Object-Oriented Programming (OOP)* yang berisi kelas-kelas untuk merepresentasikan objek-objek dalam permainan, seperti bot, diamonds, teleport, red button, base, map, dan lainnya. Selain itu, terdapat juga file API yang berfungsi untuk menghubungkan seluruh program dengan server. Untuk mengimplementasikan pergerakan bot, program ditulis melalui file Python yang terdapat pada Bot Starter Pack. Koneksi antara bot dengan server, beserta seluruh mekanisme pergerakannya, dilakukan melalui file main.py.

Pada folder game, terdapat file models.py yang berfungsi untuk merepresentasikan kelas-kelas dari objek yang ada dalam permainan. Selain itu, ada juga file util.py yang berisi fungsi-fungsi pendukung yang digunakan untuk menjalankan bot dan mengarahkan bot menuju tujuan tertentu di dalam permainan. Untuk mengembangkan logika permainan bot, dapat dibuat file baru di dalam folder logic. File ini akan berisi seluruh strategi yang diterapkan, termasuk strategi greedy, dengan memanfaatkan model dari models.py dan fungsi-fungsi dari util.py.

BAB III
APLIKASI STRATEGI GREEDY

3.1 Proses Mapping

Proses mapping merupakan langkah awal yang sangat krusial dalam merancang strategi pengumpulan diamond dalam permainan ini. Pada tahap ini, dilakukan identifikasi dan pemetaan secara detail terhadap elemen-elemen utama yang ada di dalam peta permainan, seperti lokasi diamond berbagai jenis, base (tempat pengembalian diamond), portal teleport, serta posisi dan pola pergerakan musuh (enemy bots). Pemetaan ini bukan sekadar menentukan titik koordinat, melainkan juga mengkaji hubungan antar elemen tersebut agar dapat dibuat gambaran yang lengkap dan akurat mengenai tata letak peta.

Posisi diamond dibedakan berdasarkan tipe dan nilai poinnya, terutama dengan prioritas pada diamond merah yang memiliki bobot nilai tertinggi dibanding diamond biasa. Pengelompokan diamond dalam cluster juga diperhitungkan, agar bot dapat merencanakan pengambilan diamond secara berturut-turut tanpa perlu berpindah lokasi terlalu jauh. Selanjutnya, posisi base yang menjadi tempat pengembalian diamond sangat diperhatikan. Karena bot harus mengembalikan diamond secara berkala agar poin terkumpul, jarak dari posisi diamond ke base menjadi faktor penting dalam perhitungan strategi. Semakin jauh jarak ke base, semakin besar risiko diamond hilang jika musuh menyerang saat bot dalam perjalanan.

Portal teleport ditandai sebagai titik strategis karena memiliki fungsi sebagai jalur shortcut untuk mempercepat pergerakan bot antar area peta yang cukup luas. Portal ini memungkinkan bot menghemat waktu perjalanan sehingga mampu mengambil diamond lebih banyak dalam waktu yang sama. Posisi dan pola pergerakan musuh juga dipetakan dengan tujuan untuk menghindari risiko benturan yang dapat menyebabkan bot kehilangan diamond. Pemetaan musuh dilakukan dengan memperhatikan radius ancaman dan kemungkinan jalur lintasan musuh yang sering dilalui, sehingga bot dapat merencanakan jalur alternatif yang aman.

Selain itu, untuk mendukung pengambilan keputusan, dibangun graph jarak antar objek yang menggunakan metode perhitungan jarak seperti Euclidean dan Manhattan. Graph ini berfungsi untuk menghitung estimasi langkah yang dibutuhkan bot untuk berpindah dari satu titik ke titik lainnya, baik menggunakan jalur biasa maupun melalui portal teleport.

3.2 Eksplorasi Alternatif Solusi Greedy

Strategi greedy merupakan pendekatan yang menitikberatkan pada pengambilan keputusan lokal terbaik di setiap langkah dengan tujuan mencapai solusi optimal secara keseluruhan secara bertahap. Dalam konteks pengumpulan diamond, berbagai pendekatan greedy dieksplorasi agar bot dapat beradaptasi dengan dinamika peta dan kondisi permainan.

Greedy by Nearest Diamond

Pendekatan ini mengarahkan bot untuk selalu memilih diamond yang paling dekat dari posisi saat ini, sehingga meminimalkan waktu tempuh antar pengambilan diamond. Keunggulan strategi ini adalah kesederhanaan dan kecepatan pengambilan keputusan yang tinggi, terutama efektif jika diamond tersebar merata dan tidak ada cluster besar. Namun, kekurangannya adalah bot bisa sering berpindah-pindah lokasi yang jauh dari base sehingga waktu untuk kembali ke base menjadi panjang, mengurangi efisiensi waktu pengumpulan poin.

Greedy by Best Cluster

Di sini, bot mengutamakan pengambilan diamond dalam sebuah cluster yang memiliki konsentrasi diamond tinggi. Dengan mengambil diamond secara berurutan dalam cluster tersebut, bot dapat memaksimalkan jumlah diamond yang dikumpulkan sebelum kembali ke base. Strategi ini sangat efektif saat cluster berada dekat dengan base atau jika bot mampu mengamankan cluster tersebut dari serangan musuh. Namun, jika cluster jauh dari base atau sudah mulai habis, bot harus berpindah cluster, yang menimbulkan waktu perjalanan tambahan dan potensi risiko.

Greedy by Weight (Bobot)

Pendekatan ini mengkombinasikan nilai poin diamond dengan jarak yang harus ditempuh untuk mengambilnya, sehingga bot memilih diamond berdasarkan rasio nilai terhadap jarak (point per step). Dengan metode ini, bot tidak hanya fokus pada diamond terdekat, tapi juga mempertimbangkan nilai poin yang diperoleh agar lebih efisien dalam pengumpulan poin. Meski demikian, jika bobot tinggi untuk diamond yang jauh, risiko kehilangan diamond saat perjalanan jauh meningkat akibat potensi serangan musuh.

Greedy by Enemy Bot Awareness

Strategi ini menambahkan dimensi kesadaran terhadap posisi musuh di sekitar diamond dan jalur bot. Bot diarahkan untuk menghindari area yang sudah dikontrol atau dikuasai oleh musuh, sehingga mengurangi risiko benturan yang bisa menyebabkan kehilangan diamond. Walau efektif mengurangi risiko, strategi ini seringkali memaksa bot mengambil jalur lebih panjang dan melewatkan diamond yang sebenarnya mudah diambil, yang berpengaruh pada jumlah poin maksimal yang dapat dikumpulkan.

Greedy dengan Perhitungan Rute Terpendek dan Pemanfaatan Portal Teleport

Menggunakan algoritma shortest path seperti Dijkstra, strategi ini menghitung rute tercepat antara posisi bot, diamond, dan base dengan memperhitungkan portal teleport sebagai jalur shortcut. Portal teleport digunakan secara selektif untuk mempercepat perjalanan sehingga bot dapat memaksimalkan waktu pengumpulan diamond. Kompleksitas perhitungan rute

meningkat, namun strategi ini memberikan keuntungan besar dalam efisiensi waktu perjalanan, terutama pada peta yang luas dan memiliki banyak portal.

3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy

Setiap alternatif strategi greedy memiliki karakteristik dan dampak yang berbeda terhadap performa bot:

Greedy by Nearest Diamond

Efisien dalam hal kecepatan pengambilan diamond lokal, strategi ini mudah diimplementasikan dan memiliki performa baik pada peta dengan distribusi diamond merata. Namun, seringkali mengabaikan faktor jarak ke base yang menyebabkan bot harus menempuh perjalanan jauh untuk mengembalikan diamond, sehingga efisiensi waktu secara keseluruhan berkurang.

Greedy by Best Cluster

Meningkatkan efisiensi dengan pengumpulan diamond berurutan dalam satu area, strategi ini cocok untuk memaksimalkan poin dalam waktu singkat. Namun, risiko perjalanan jauh dan cluster yang berkurang membuat bot harus melakukan penyesuaian strategi dan berpotensi menambah waktu perjalanan dan risiko serangan musuh.

Greedy by Weight

Memberikan keseimbangan antara nilai poin dan jarak tempuh, strategi ini mampu meningkatkan rasio poin per langkah. Akan tetapi, jika bobot terlalu mengutamakan diamond jauh, bot menjadi rentan terhadap risiko kehilangan diamond akibat serangan musuh selama perjalanan panjang ke base.

Greedy by Enemy Bot Awareness

Strategi ini berhasil mengurangi risiko kehilangan diamond dengan menghindari musuh, meningkatkan tingkat keberhasilan pengumpulan diamond. Kendalanya, bot harus mengambil rute memutar dan melewatkan diamond potensial, sehingga kecepatan dan jumlah poin maksimal yang dikumpulkan bisa berkurang.

Greedy dengan Rute Terpendek dan Portal Teleport

Memberikan peningkatan efisiensi dengan memanfaatkan portal sebagai jalur shortcut, mengurangi waktu tempuh signifikan. Kompleksitas perhitungan yang tinggi dan ketergantungan pada posisi portal membuat strategi ini menuntut algoritma cerdas dan optimalisasi, serta memerlukan sumber daya komputasi lebih besar.

3.4 Strategi Greedy yang Dipilih

Dalam menyusun strategi untuk bot dalam permainan ini, kami memilih untuk menggunakan pendekatan greedy karena kesesuaiannya dengan sifat permainan yang dinamis dan berbasis keputusan cepat. Tujuan utama dari bot adalah mengumpulkan sebanyak mungkin diamond dan menyimpannya ke dalam base secara efisien. Untuk mencapai hal tersebut, kami menggabungkan beberapa strategi greedy, yaitu Greedy by Distance, Greedy by Game Condition Priority, dan Greedy by Diamond's Point.

Pertama, strategi Greedy by Distance menjadi dasar utama dalam setiap pengambilan keputusan bot. Bot secara konsisten menghitung jarak antara posisinya dengan objek penting di sekitarnya seperti diamond, base, teleport, atau red button, dan selalu memilih objek dengan jarak terdekat sebagai tujuan selanjutnya. Namun, perhitungan jarak saja tidak cukup. Oleh karena itu, kami menerapkan strategi Greedy by Game Condition Priority, di mana bot mengatur urutan prioritasnya berdasarkan kondisi permainan. Misalnya, jika inventory sudah penuh, maka prioritas tertinggi adalah menuju base. Jika diamond masih banyak dan inventory belum penuh, maka diamond menjadi prioritas. Ketika tidak ada diamond di sekitar, barulah bot mempertimbangkan menuju red button untuk memunculkan kembali diamond.

Untuk meningkatkan efisiensi poin, kami juga menerapkan strategi Greedy by Diamond's Point, yaitu memilih diamond dengan nilai poin tertinggi terlebih dahulu, dalam hal ini, diamond merah akan lebih diprioritaskan dibanding diamond biasa jika perbedaannya signifikan dan jaraknya masih tergolong wajar. Selain itu, strategi kami juga secara tidak langsung membentuk pendekatan Greedy by Cluster. Bot tidak hanya mengejar satu diamond, tapi juga memperhitungkan apakah di sekitar diamond tersebut terdapat diamond lain, terutama yang posisinya lebih dekat ke base. Ini memungkinkan bot mengumpulkan beberapa diamond sekaligus dalam satu rute efisien, lalu segera kembali ke base.

Kami juga menyadari bahwa keberadaan bot musuh bisa memengaruhi keputusan. Maka dari itu, jika musuh berada dalam satu garis (baik row maupun column) dengan diamond dan tampak bergerak ke arah diamond tersebut, maka bot kami akan menghindari konflik dan memilih diamond lain, ini merupakan bentuk adaptasi terhadap lingkungan permainan yang kompetitif. Teleportasi juga menjadi bagian dari pertimbangan strategi greedy. Jika penggunaan teleport dapat mempercepat perjalanan menuju target, maka akan digunakan. Namun jika teleport justru membuat bot tersesat atau keluar dari area prioritas, maka akan dihindari.

Akhirnya, saat waktu permainan hampir habis (kurang dari 10 detik), strategi bot berubah. Bot akan langsung memprioritaskan kembali ke base jika membawa diamond, atau parkir di area aman jika inventory kosong, agar poin tidak hilang sia-sia. Ini merupakan penerapan prinsip greedy berdasarkan sisa waktu (time-aware greedy). Secara keseluruhan, strategi greedy yang kami gunakan tidak hanya mengejar hasil terbaik dalam satu langkah, tetapi juga mempertimbangkan kondisi permainan secara menyeluruh. Kombinasi dari berbagai pendekatan greedy ini bertujuan untuk membuat bot bergerak secara efisien, adaptif, dan kompetitif sepanjang permainan.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma Greedy

4.1.1 Pseudocode

Fungsi Utilitas

```
Fungsi count_steps(a, b):
    Kembalikan jumlah langkah Manhattan antara posisi a dan b

Fungsi coordinate_equals(x1, y1, x2, y2):
    Kembalikan true jika posisi (x1, y1) sama dengan (x2, y2)

Fungsi same_direction(pivot, a, b):
    Cek apakah posisi a dan b berada di kuadran yang sama
relatif terhadap pivot

Fungsi get_direction_alt(cx, cy, dx, dy):
    Hitung arah langkah dari (cx, cy) ke (dx, dy)
    Prioritaskan vertikal, jika tidak ada, pilih horizontal
```

Kelas Portals

```
Konstruktor(portals, player_pos):
    Simpan portal terdekat dan terjauh dari posisi pemain

Fungsi count_steps_by_portal(curr, target):
    Hitung jarak total dari curr ke target melalui dua portal

Fungsi is_closer_by_portal(curr, target):
    Return true jika jarak melalui portal lebih pendek daripada
jalur langsung
```

Kelas Player

```
Konstruktor(posisi, properti):
    Simpan posisi, jumlah diamond, waktu, base, dsb.

Fungsi is_inventory_full():
    Return true jika jumlah diamond = kapasitas maksimum
```

```
Fungsi set_target(objek):
    Set objek sebagai target

Fungsi set_target_position(posisi):
    Set posisi sebagai target langsung

Fungsi is_invalid_move(dx, dy, board):
    Return true jika hasil langkah keluar dari board

Fungsi avoid_obstacles(portals, is_avoiding, board):
    Hitung langkah menuju target
    Jika langkah berikutnya menuju portal dan bukan portal
target:
        Modifikasi arah untuk menghindari portal
    Simpan langkah berikutnya
```

Kelas Diamonds

```
Konstruktor(diamond_list, red_button, jumlah_held):
    Simpan daftar diamond yang layak dikumpulkan
    Simpan tombol merah

Fungsi filter_diamond(curr, enemy):
    Hapus diamond yang searah dengan musuh

Fungsi choose_diamond(player, portals):
    Untuk setiap diamond:
        Hitung jarak dari player (dan ke base jika inventory
penuh)
        Pilih diamond terbaik berdasarkan jarak dan poin
        Jika jalur portal lebih cepat, prioritaskan portal

Fungsi check_red_button(player, portals):
    Cek apakah tombol merah lebih layak jadi target
```

Kelas Enemies

```
Konstruktor(enemy_list, player_bot):
    Simpan semua musuh kecuali player

Fungsi check_nearby_enemy(diamonds, player, portals,
has_tackled):
    Temukan musuh terdekat
    Jika musuh dalam jarak tackle:
```

```
        Lakukan tackle
    Jika dalam jarak bahaya:
        Panggil fungsi untuk hindari musuh

Fungsi avoid_enemy(player, diamonds, portals):
    Filter diamond agar tidak searah musuh
    Pilih ulang diamond
    Cek tombol merah
```

Kelas GameState

```
Konstruktor(player_bot, board):
    Simpan state board dan bot

Fungsi initialize():
    Ambil objek penting dari board (diamond, portal, bot, red
button)
    Return instance Player, Diamonds, Portals, Enemies

Fungsi no_time_left(curr_pos, base_pos):
    Cek apakah sisa waktu cukup untuk kembali ke base
```

Kelas NotUnderstand (Main Logic)

```
Konstruktor():
    Inisialisasi flag untuk balik ke base, hindari portal,
tackle

Fungsi next_move(bot, board):
    Inisialisasi GameState
    Dapatkan Player, Diamonds, Portals, Enemies

    Update status jika player berada di base atau portal

    Jika inventory penuh atau waktu hampir habis:
        Set target ke base
        Jika lewat portal lebih cepat, set target portal

    Jika tidak:
        Pilih target diamond terbaik
        Cek apakah tombol merah lebih baik

    Jika masih ada waktu:
        Cek musuh terdekat, tackle jika mungkin
```

```
    Jika tidak sedang tackle:
        Jalankan logika menghindari portal

    Return langkah selanjutnya (dx, dy)
```

4.1.2 Penjelasan Alur Program

Berikut adalah implementasi algoritma Greedy yang dilakukan bot dalam bentuk python. Strategi diimplementasikan dalam file Not_U.py

Inisialisasi dan Pembacaan Papan Permainan

```
board = Board(game.board)
```

Membuat objek **Board** yang merepresentasikan papan permainan saat ini. Ini penting untuk memahami posisi elemen-elemen seperti diamond, musuh, dan portal.

Pembaruan Informasi Permainan

```
board.update(game.board)
player = game.player
diamonds.update(game.diamonds)
enemies.update(game.enemies)
```

Memperbarui informasi terkini dari papan permainan, posisi pemain, diamond, dan musuh. Ini memastikan bahwa keputusan yang diambil bot didasarkan pada data terbaru.

Pengecekan dan Penanganan Tombol Merah (Red Button)

```
diamonds.check_red_button(player, portals)
```

Memeriksa apakah ada diamond yang terkunci dan membutuhkan aktivasi tombol merah. Jika ya, bot akan memprioritaskan untuk menekan tombol tersebut sebelum mengambil diamond.

Pengecekan Musuh Terdekat dan Penentuan Aksi

```
enemies.check_nearby_enemy(diamonds, player, portals,
self.tackle)
self.tackle = enemies.try_tackle
```

Menilai apakah ada musuh di sekitar yang perlu dihindari atau diserang. Keputusan ini akan mempengaruhi strategi pergerakan bot selanjutnya.

Pemilihan Diamond Berdasarkan Strategi Greedy

```
diamonds.choose_diamond(player, portals)
```

Menggunakan strategi greedy untuk memilih diamond yang paling menguntungkan berdasarkan jarak dan nilai. Ini adalah inti dari strategi pengumpulan diamond oleh bot.

Penghindaran Portal Jika Tidak Menyerang

```
if not enemies.try_tackle:
    player.avoid_obstacles(portals, self.is_avoiding_portal,
board)
    self.is_avoiding_portal = player.is_avoiding_portal
```

Jika bot tidak sedang dalam mode menyerang, ia akan menghindari portal untuk mencegah pergerakan yang tidak diinginkan atau terjebak.

Penentuan Langkah Selanjutnya

```
return player.next_move
```

Mengembalikan langkah yang telah ditentukan oleh bot berdasarkan semua pertimbangan sebelumnya. Ini adalah aksi yang akan dilakukan pada giliran saat ini.

4.2 Struktur Data yang Digunakan

Struktur data program yang kami buat untuk bot permainan Diamonds kami adalah sebagai berikut :

```
├── doc
│   └── Not_Understand.pdf
└── src
    ├── __pycache__
    │   └── decode.cpython-311.pyc
    ├── game
    │   ├── __pycache__
    │   │   ├── __init__.cpython-311.pyc
    │   │   ├── api.cpython-311.pyc
    │   │   ├── board_handler.cpython-311.pyc
    │   │   ├── bot_handler.cpython-311.pyc
    │   │   ├── models.cpython-311.pyc
    │   │   └── util.cpython-311.pyc
    │   ├── logic
    │   │   ├── __pycache__
    │   │   │   ├── __init__.cpython-311.pyc
    │   │   │   ├── base.cpython-311.pyc
    │   │   │   ├── Not_U.cpython-311.pyc
    │   │   │   └── random.cpython-311.pyc
    │   │   ├── __init__.py
    │   │   ├── base.py
    │   │   ├── Not_U.py
    │   │   └── random.py
```

```
│   ├── __init__.py
│   ├── api.py
│   ├── board_handler.py
│   ├── bot_handler.py
│   ├── models.py
│   └── util.py
├── gitignore
├── decode.py
├── main.py
├── README.md
├── requirements.txt
├── run-bots.bat
└── run-bots.sh
```

Seperti yang bisa dilihat pada struktur program diatas, terdapat 2 folder utama yang dijadikan modul dalam program, meliputi game dan logic. **Folder game** merupakan modul utama yang mengatur jalannya permainan. Di dalamnya terdapat file seperti board_handler.py untuk membaca dan memproses papan permainan, bot_handler.py untuk menjalankan logika bot, models.py untuk mendefinisikan elemen-elemen game seperti posisi atau objek, dan util.py yang berisi fungsi bantu umum. File api.py digunakan untuk menangani komunikasi dengan sistem eksternal atau API game. **Folder logic** (di dalam game) berisi berbagai strategi bot yang dapat digunakan. File Not_U.py merupakan implementasi strategi utama berbasis pendekatan **greedy**, yang dirancang untuk memaksimalkan pengambilan diamond dengan memperhatikan posisi objek game. Selain itu, terdapat file base.py sebagai kerangka dasar strategi dan random.py sebagai strategi sederhana berbasis gerakan acak.

4.3 Pengujian Program

4.3.1 Skenario Pengujian

Setup Awal Game dan Objek

Siapkan papan permainan (Board) dengan ukuran tertentu, misal 10x10 grid.

Tempatkan: Pemain (bot) di posisi awal (misal: (1,1)), dengan properti tertentu (inventory size, diamonds held, waktu tersisa). Berlian di beberapa posisi dengan nilai poin berbeda (1 atau lebih). Dua portal teleport di posisi yang berbeda (misal: (2,2) dan (7,7)). Musuh (bots lain) di posisi yang bervariasi, termasuk dekat dan jauh dari pemain. Base di posisi tetap (misal: (0,0)). Tombol merah (red button) pada posisi tertentu.

Pengujian Pergerakan Dasar

Pastikan bot dapat menentukan langkah yang benar untuk bergerak menuju target yang dipilih (berlian, base, portal). Contoh: Bot diberi target berlian, periksa next_move

mengarahkan bot ke langkah terdekat menuju berlian. Bot diberi target base, periksa next_move mengarahkan bot pulang ke base.

Pengujian Pemilihan Target Berlian

Berikan bot beberapa berlian dengan poin berbeda dan posisi berbeda. Pastikan bot memilih berlian dengan jarak terdekat dan nilai poin terbaik sesuai prioritas (contoh: prioritas berlian bernilai 1 jika inventory belum penuh, prioritas berlian dekat jika inventory penuh). Periksa jika bot memilih berlian yang tidak satu arah dengan musuh (filter diamond).

Pengujian Logika Portal

Tempatkan portal di posisi yang memungkinkan bot memilih lewat portal lebih cepat menuju target (berlian atau base). Pastikan bot memutuskan lewat portal jika jalur tersebut memang lebih efisien daripada langsung. Pastikan bot menandai dirinya sebagai "inside portal" saat berada di posisi portal.

Pengujian Inventory dan Waktu

Simulasikan kondisi inventory penuh (misal 4 berlian dipegang). Simulasikan waktu tersisa mendekati habis. Pastikan bot mengubah target menjadi base dan berusaha kembali ke base secepat mungkin. Pastikan bot menggunakan portal jika itu mempercepat pulang ke base.

Pengujian Interaksi Musuh

Tempatkan musuh di posisi yang berjarak 2 atau 3 langkah dari bot. Jika jarak 2, pastikan bot mencoba "tackle" (bergerak mendekati musuh). Jika jarak 3, pastikan bot menghindari musuh dengan memilih berlian lain yang aman. Pastikan bot menghindari berlian yang satu arah dengan musuh.

Pengujian Logika Hindari Portal

Pastikan saat bot bergerak ke target bukan portal, dan langkah berikutnya mengarah ke portal, bot melakukan penghindaran dengan mengubah langkah. Pastikan penghindaran gerak portal diaktifkan dan dinonaktifkan sesuai kondisi.

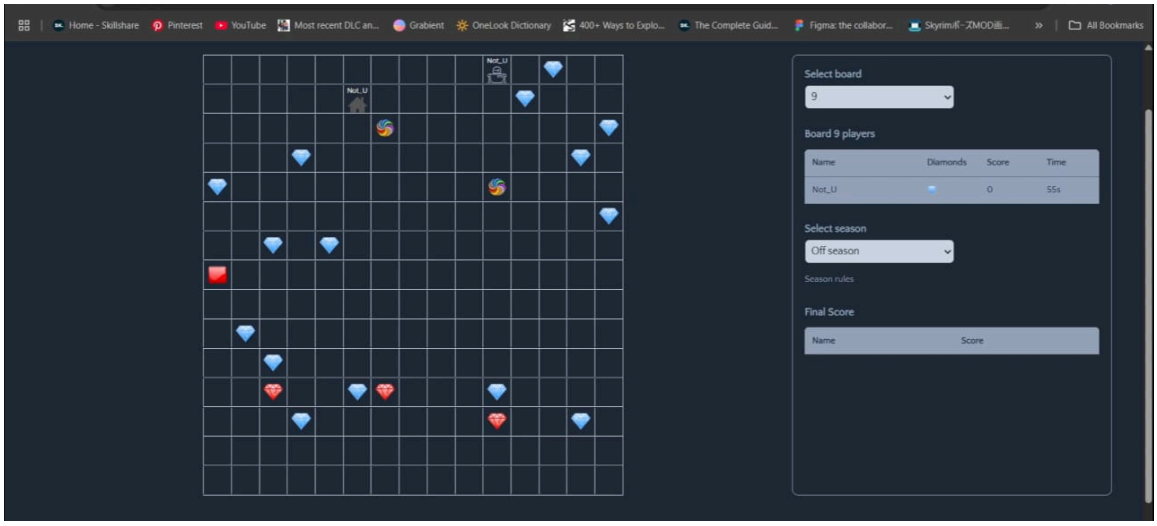
Pengujian Kondisi Tidak Ada Berlian

Jika tidak ada berlian tersisa di papan, bot harus langsung kembali ke base.

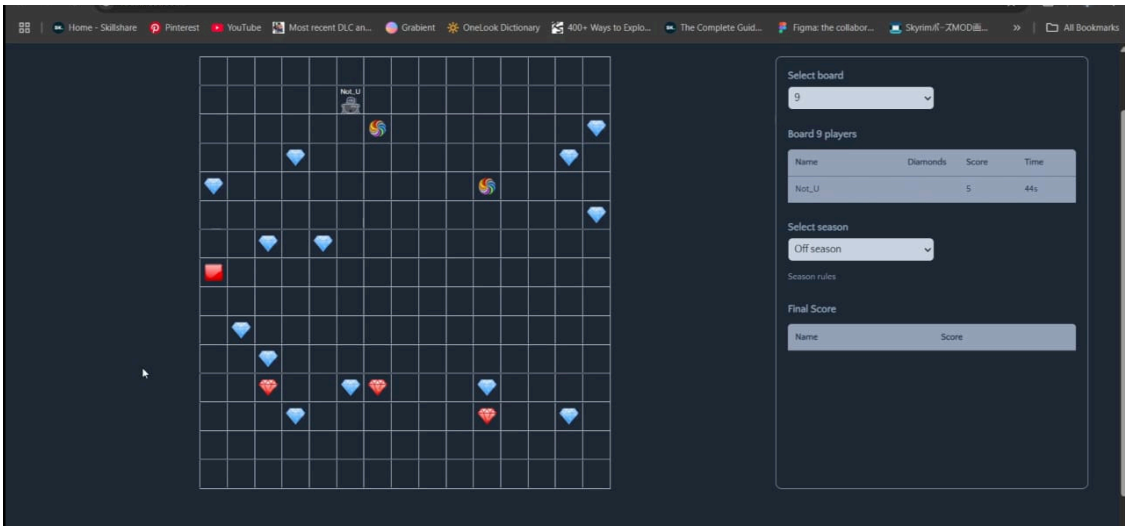
Pengujian Konsistensi State

Pastikan variabel status back_to_base, is_avoiding_portal, dan tackle berubah dengan benar sesuai kondisi permainan. Jalankan beberapa langkah berturut-turut, cek apakah state berubah sesuai perubahan posisi dan kondisi.

4.3.2 Hasil Pengujian dan Analisis



Pada hasil analisis dan pengujian oleh Bot Not Understand atau Not_U yang kami lakukan sebanyak 15 kali pengujian. Rata-rata selama percobaan Bot melakukan algoritma greedy untuk mendahulukan pengambilan diamond biru.



Selanjutnya bot akan kembali ke Bases untuk mengumpulkan diamond blue. Diamond red diambil jika letaknya dekat dengan diamond blue. Diamond button diambil jika diamond blue sudah sedikit dan letaknya jauh.

Selama 15 kali percobaan didapatkan data sebagai berikut:

Percobaan ke-	**Hasil Pengujian**
1	13
2	12
3	10
4	12
5	13
6	10
7	10
8	12
9	13
10	12
11	13
12	14
13	13
14	12
15	10
Rata-rata	11,93

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dokumen ini membahas pemanfaatan algoritma greedy dalam pembuatan bot untuk permainan *Diamonds*, sebuah game berbasis web di mana bot ditugaskan mengumpulkan diamond sebanyak mungkin. Strategi greedy dipilih karena kemampuannya mengambil keputusan cepat berdasarkan kondisi lokal terbaik tanpa perlu meninjau ulang langkah sebelumnya. Dalam implementasinya, tim menggabungkan berbagai pendekatan greedy seperti pemilihan berdasarkan jarak terdekat, nilai diamond tertinggi, kondisi permainan (misalnya inventory penuh atau waktu hampir habis), serta kesadaran terhadap posisi musuh dan penggunaan portal teleport. Bot yang dikembangkan berhasil menunjukkan performa stabil dalam 15 kali pengujian, dengan rata-rata skor 11,93 poin. Meskipun bot cenderung memprioritaskan diamond biru yang lebih mudah diakses, strategi adaptif juga diterapkan untuk memaksimalkan efisiensi dan keamanan pengumpulan diamond. Kesimpulannya, pendekatan greedy terbukti efektif dalam konteks permainan ini, namun pengembangan lebih lanjut disarankan untuk menyempurnakan logika pengambilan keputusan dan penanganan situasi dinamis yang lebih kompleks.

5.2 Saran

Saran untuk pengembangan kelompok kami adalah agar lebih memperdalam perencanaan logika algoritma bot, khususnya dalam mengoptimalkan strategi pengambilan target dan penghindaran musuh. Perencanaan pengembangan program dapat ditingkatkan supaya proses pembuatan bot lebih efektif. Kami juga perlu lebih fokus pada pengujian fungsi-fungsi utama seperti manajemen portal dan pemilihan berlian. Selain itu, meningkatkan sesi brainstorming terkait penanganan situasi dinamis dalam permainan akan membantu menemukan solusi yang lebih efektif dan inovatif selama proses pengerjaan.

LAMPIRAN

A. Repository Github

https://github.com/12-211-Muhamma-Bimastiar/Tubes-STIMA-Not_Understand.git

B. Video Penjelasan

https://drive.google.com/file/d/1K9ClCMDTwJcApJuiMwWJnTUhM2D-GSmq/view?usp=drivesdk

DAFTAR PUSTAKA

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.

[2] Kleinberg, J., & Tardos, É. (2006). *Algorithm Design*. Pearson Education.

[3] Dasgupta, S., Papadimitriou, C., & Vazirani, U. (2006). *Algorithms*. McGraw-Hill.