LAPORAN TUGAS BESAR I

IF2211 STRATEGI ALGORITMA

PEMANFAATAN ALGORITMA GREEDY DALAM PEMBUATAN BOT PERMAINAN DIAMONDS

Disusun untuk memenuhi tugas mata kuliah Strategi Algoritma pada Semester 2 (dua) Tahun Akademik 2023/2024.

Oleh Kelompok FreeDiamonds:

Novelya Putri Ramadhani	13522096
Diana Tri Handayani	13522104
Lidya Rahmatul Fitri	10023485
Rafii Ahmad Fahreza	10023570

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

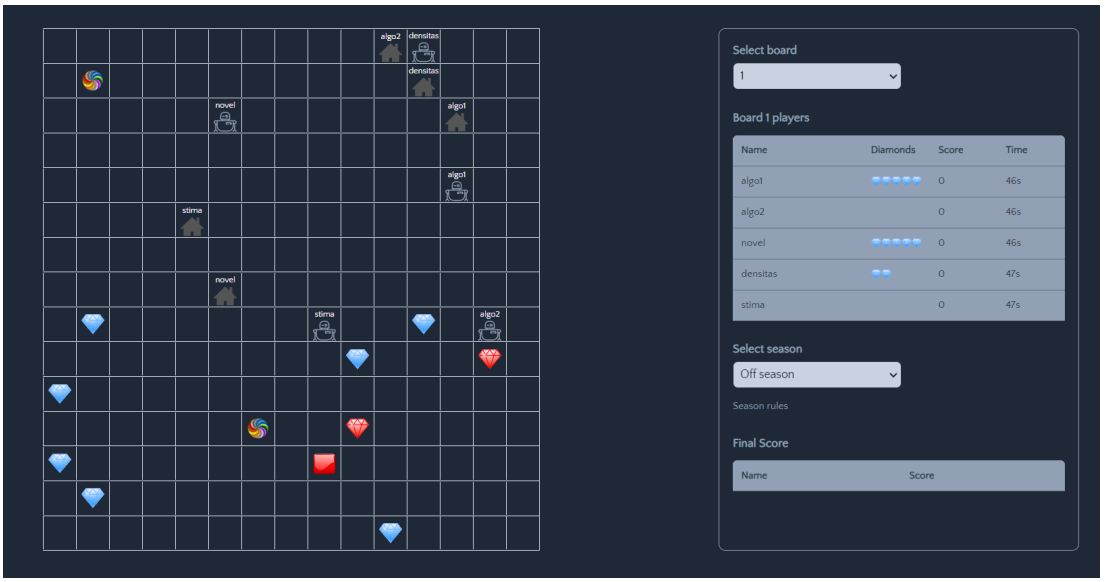
2024

DAFTAR ISI

DAFTAR ISI 1
BAB I 2
DESKRIPSI TUGAS 2
BAB II 3
LANDASAN TEORI 3
2.1 Algoritma Greedy 3
2.2 Game Engine 4
2.3 Deskripsi Permainan 6
2.4 Cara Kerja Program 7
BAB III 9
APLIKASI STRATEGI GREEDY 9
3.1 Proses *Mapping* Persoalan 9
3.2 Eksplorasi Alternatif Solusi 10
3.3 Analisis Efisiensi dan Efektivitas Solusi 13
3.4 Strategi Greedy Terpilih 16
BAB IV 19
IMPLEMENTASI DAN PENGUJIAN 19
4.1 Implementasi Algoritma Greedy 19
4.2 Struktur Data 20
4.3 Analisis Desain Solusi 21
BAB V 24
KESIMPULAN dan SARAN 24
5.1 Kesimpulan 24
5.2 Saran 24
LAMPIRAN 26
DAFTAR PUSTAKA 27

BAB I
DESKRIPSI TUGAS

Diamonds merupakan suatu *programming challenge* yang mempertandingkan beberapa bot sekaligus. 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. Untuk deskripsi permainan secara lengkap terdapat pada bab dua.



Gambar 1. Visualisasi Game Diamonds

Dari permainan tersebut terdapat berbagai strategi yang ada untuk mendapatkan skor diamond sebanyak-banyaknya. Salah satu strategi algoritma yang dapat diimplementasikan adalah algoritma Greedy. Kali ini, kami mencoba mengeksplorasi algoritma Greedy yang dibuat dalam bahasa pemrograman python sebagai bahan pembelajaran dalam pemahaman algoritma Greedy.

BAB II
TEORI SINGKAT

Dalam Tugas Besar I ini terdapat penyelesaian yang harus dicapai, yaitu solusi algoritma greedy dalam permainan diamonds.

2.1 Algoritma Greedy

Algoritma Greedy adalah strategi penyelesaian masalah yang berfokus pada optimasi lokal di setiap langkah, beroperasi di bawah prinsip bahwa keputusan terbaik yang tampaknya saat ini. Dengan harapan ketika diambil, akan membawa kita ke solusi terbaik secara keseluruhan. Ada dua karakteristik utama dari algoritma ini. Pertama, algoritma ini membangun solusi secara bertahap, dengan solusi akhir yang didapatkan melalui penggabungan pilihan-pilihan optimal lokal pada setiap langkah. Ini berarti bahwa setiap keputusan yang diambil selama proses ini berkontribusi langsung terhadap solusi akhir. Kedua, algoritma ini berfokus pada keuntungan jangka pendek. Ini berarti bahwa keputusan diambil berdasarkan keuntungan yang dapat diperoleh saat ini, tanpa mempertimbangkan dampak jangka panjang yang mungkin timbul dari keputusan tersebut. Meskipun ini dapat menghasilkan solusi yang efisien dalam beberapa kasus, penting untuk diingat bahwa pendekatan ini mungkin tidak selalu menghasilkan solusi terbaik dalam konteks yang lebih luas atau dalam situasi di mana dampak jangka panjang dari keputusan harus dipertimbangkan.

Algoritma Greedy memiliki beberapa kelebihan yang membuatnya menjadi pilihan yang menarik dalam berbagai situasi. Pertama, algoritma ini mudah dipahami dan diimplementasikan, yang berarti bahwa bahkan mereka yang baru mengenal pemrograman dapat mulai menggunakannya dengan relatif cepat. Kedua, algoritma ini seringkali menghasilkan solusi yang mendekati optimal, terutama untuk jenis masalah tertentu. Ketiga, algoritma ini cepat dan efisien dalam hal penggunaan resource komputasi,

yang membuatnya menjadi pilihan yang baik untuk situasi di mana kecepatan dan efisiensi adalah prioritas.

Namun, algoritma Greedy juga memiliki beberapa kekurangan. Meskipun algoritma ini dapat menghasilkan solusi yang baik dalam banyak kasus, ia tidak selalu menjamin solusi optimal secara keseluruhan. Selain itu, algoritma ini bisa terjebak dalam optimum lokal, yang berarti bahwa solusi terbaik secara keseluruhan mungkin tidak tercapai.

Untuk memberikan gambaran tentang bagaimana algoritma Greedy dapat digunakan dalam praktek, berikut adalah beberapa contoh penerapannya. Pertama, algoritma ini dapat digunakan untuk menukar uang dengan pecahan terkecil, memastikan bahwa jumlah pecahan yang digunakan sekecil mungkin. Kedua, algoritma ini dapat digunakan dalam penjadwalan tugas, di mana tujuannya adalah untuk menyelesaikan sebanyak mungkin tugas dalam waktu yang tersedia. Ketiga, algoritma Dijkstra, yang merupakan contoh dari algoritma Greedy, dapat digunakan untuk mencari rute terpendek dalam graf. Terakhir, algoritma Prim, yang juga merupakan contoh dari algoritma Greedy, dapat digunakan untuk menemukan minimum spanning tree dalam graf.

2.2 Game Engine

Game engine adalah perangkat lunak canggih yang berfungsi sebagai kerangka kerja fundamental dalam proses pengembangan video game. Dengan menyediakan berbagai tools dan fungsionalitas yang esensial, game engine memungkinkan pembuat game untuk membangun dan merancang game dengan lebih efisien dan efektif.

Salah satu komponen utama dari game engine adalah graphics engine, yang bertanggung jawab atas aspek visual dalam game. Ia mengelola rendering objek, animasi, dan efek visual, menciptakan tampilan yang menarik dan imersif. Selain itu, ada juga physics engine yang mensimulasikan aspek fisika dalam game, seperti gravitasi, benturan, dan gerakan, memberikan realisme dan kedalaman pada game play.

Sound engine juga merupakan bagian integral dari game engine, mengelola semua aspek audio dalam game, termasuk suara latar, efek suara, dan musik, yang semuanya berkontribusi pada pengalaman audio yang kaya dan mendalam. Selain itu, game engine juga mengelola input pemain dari berbagai perangkat seperti keyboard, mouse, dan gamepad, serta mengalokasikan dan mengelola memori yang digunakan game secara efisien.

Fitur networking mendukung permainan online dan multiplayer, memungkinkan pemain untuk berinteraksi dan berkompetisi satu sama lain dalam lingkungan virtual. Dengan semua fungsionalitas ini, game engine mempercepat proses pengembangan game, menyediakan fungsionalitas dasar yang dibutuhkan game, dan memungkinkan developer untuk fokus pada aspek kreatif game, seperti cerita, karakter, dan desain level.

Ada dua jenis utama game engine yang dapat digunakan dalam pengembangan game, yaitu 2D game engine dan 3D game engine. 2D game engine adalah pilihan yang ideal untuk game dua dimensi seperti platformer, puzzle, dan side-scrolling games, sedangkan 3D game engine lebih cocok untuk game tiga dimensi seperti FPS, RPG, dan action-adventure games.

Beberapa contoh dari game engine yang populer di antara developer adalah Unity, Unreal Engine, Godot Engine, dan GameMaker Studio. Masing-masing dari game engine ini memiliki kelebihan dan kekurangan sendiri, dan pilihan yang tepat tergantung pada berbagai faktor.

Pemilihan game engine harus mempertimbangkan beberapa aspek penting. Pertama, jenis game yang ingin dikembangkan. Beberapa game engine lebih cocok untuk jenis game tertentu dibandingkan dengan yang lain. Kedua, kebutuhan developer. Ini bisa mencakup kemampuan untuk memprogram, fitur yang dibutuhkan, dan sebagainya. Ketiga, platform target. Beberapa game engine lebih baik dalam mendukung platform tertentu, seperti PC, konsol, atau mobile. Oleh karena itu, penting untuk

memilih game engine yang paling sesuai dengan kebutuhan dan tujuan Anda dalam pengembangan game.

2.3 Deskripsi Permainan

Diamonds adalah permainan strategi yang mengharuskan pemain untuk mengumpulkan sebanyak mungkin berlian dan mencegah pemain lain mengambilnya. Setiap pemain akan diberikan giliran untuk bergerak di papan permainan yang terdiri dari kotak-kotak yang berisi berlian. Pemain dapat memilih untuk bergerak ke kotak yang berisi berlian atau memblokir akses pemain lain ke kotak tersebut.

Tujuan utama permainan ini adalah untuk mengumpulkan sebanyak mungkin berlian. Pemain dengan jumlah berlian terbanyak akan menjadi pemenangnya. Namun, pemain juga harus berhati-hati karena pemain lain dapat mencuri berlian yang sudah dikumpulkan.

Strategi yang baik dalam permainan Diamonds adalah menggunakan pendekatan Greedy. Pemain harus mempertimbangkan dengan baik setiap langkah yang diambil untuk memaksimalkan jumlah berlian yang dikumpulkan dan meminimalkan risiko berlian dicuri oleh pemain lain. Permainan Diamonds menawarkan tantangan yang menarik dalam merencanakan strategi dan mengambil keputusan yang tepat.

Berikut ini adalah cara kerja 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.

2.4 Cara Kerja Program

Untuk menjalankan game, terlebih dahulu harus memenuhi requirement yang harus di-install. Requirement tersebut meliputi Node.js, Docker desktop, dan Yarn. Setelah menginstal semua requirement, langkah selanjutnya adalah melakukan instalasi dan konfigurasi awal. Pertama, download source code game engine dari release yang tersedia dan ekstrak file zip tersebut. Kemudian, masuk ke folder hasil ekstraksi dan buka terminal. Selanjutnya, masuk ke root directory dari project dan jalankan perintah untuk menginstal dependencies menggunakan Yarn. Setelah itu, setup default environment variable dengan menjalankan script yang sesuai dengan sistem operasi yang digunakan. Selanjutnya, setup local database dengan menjalankan perintah yang disediakan. Setelah semua langkah tersebut selesai, build program dengan menjalankan perintah "npm run build" dan jalankan program dengan perintah "npm run start". Jika berhasil, program akan berjalan dan dapat diakses melalui localhost menggunakan browser.

Untuk menjalankan bot, juga perlu memenuhi requirement yang harus di-install. Requirement tersebut meliputi Python. Setelah menginstal Python, langkah selanjutnya adalah mengunduh source code bot dari release yang tersedia dan ekstrak file zip tersebut. Kemudian, masuk ke folder hasil ekstraksi dan buka terminal. Selanjutnya, masuk ke root directory dari project dan jalankan perintah untuk menginstal dependencies menggunakan pip. Setelah itu, jalankan bot dengan perintah yang sesuai dengan jumlah bot yang ingin dijalankan. Untuk menjalankan satu bot, jalankan perintah "python main.py --logic --email= --name= --password= --team etimo". Untuk menjalankan beberapa bot sekaligus, jalankan perintah yang disediakan. Dalam menjalankan bot, dapat melakukan penyesuaian pada file yang berisi logic bot yang ingin digunakan. Setelah menjalankan bot, dapat melihat bot yang telah bergabung melalui frontend. Saat permainan selesai, final score akan muncul pada sisi kanan bawah.

Untuk mengimplementasikan bot, buatlah folder baru pada direktori /game/logic dan buat kelas yang meng-inherit kelas BaseLogic. Implementasikan constructor dan method next_move pada kelas tersebut. Setelah itu, import kelas yang telah dibuat pada file main.py dan daftarkan pada dictionary CONTROLLERS. Jalankan program dengan perintah yang sesuai dengan nama bot yang telah terdaftar pada CONTROLLERS.

Ada beberapa konfigurasi permainan yang dapat diubah, seperti jumlah bot yang bermain dan konfigurasi board. Untuk mengubah konfigurasi permainan, dapat melakukan perubahan pada database menggunakan perintah SQL. Misalnya, untuk mengubah session length menjadi 100 detik, jalankan perintah SQL yang sesuai. Perubahan konfigurasi tersebut tidak akan digunakan pada kompetisi, namun dapat digunakan untuk bereksperimen dengan bot.

BAB III
ANALISIS PEMECAHAN MASALAH

3.1. Mapping Persoalan FreeDiamonds ke Elemen Algoritma Greedy

Himpunan Kandidat	Himpunan semua *diamond* di *board*
Himpunan Solusi	*Diamond-diamond* yang terpilih untuk menjadi tujuan bot pada tiap langkah
Fungsi Solusi	Mengarahkan bot menuju ke *diamond* terpilih
Fungsi Seleksi (selection function)	Pilih *diamond* yang memiliki densitas terbesar
Fungsi Kelayakan (feasible)	Memeriksa apakah *diamond* yang akan dipilih apabila poinnya dijumlahkan dengan poin semua *diamond* di himpunan tidak melebihi ukuran *inventory*
Fungsi Obyektif	Total poin *diamond* yang didapatkan maksimal dan jumlah langkah yang ditempuh minimum

Pada permainan Diamonds komponen-komponen di dalamnya dapat diklasifikan ke dalam elemen-elemen dari Algoritma Greedy. Himpunan kandidat pada persoalan Diamonds ini berupa komponen-komponen yang ada pada permainan diamonds, seperti diamond biru, diamond merah, bot-bot lain yang memiliki nilai. Kemudian red button dan teleporters yang

dapat mengubah keadaan dengan mengacak posisi semua komponen permainan dan juga memungkinan untuk mendekatkan bot dengan diamond biru dan merah. Serta home base yang dapat digunakan untuk mengamankan nilai yang diambil dan sedang berada di *inventory*. Himpunan solusinya yaitu komponen-komponen permainan yang dituju. Mengarahkan bot menuju ke komponen permainan merupakan fungsi solusi dari permasalahan diamonds. Sedangkan Mengarahkan bot menuju ke komponen permainan dengan keuntungan terbanyak terbanyak berdasarkan poin/jarak adalah fungsi seleksinya. Kemudian fungsi kelayakannya adalah memvalidasi objek dan memeriksa apakah keuntungan yang didapat dari komponen yang baru dipilih jika dijumlahkan tidak lebih dari lima. Serta fungsi objektif nya adalah memperoleh berlian yang didapatkan dengan maksimal dengan langkah yang minimum.

3.2. Eksplorasi Alternatif Solusi

a. Alternatif 1: Berdasarkan Jarak

Alternatif algoritma pertama yang dapat digunakan yaitu Algoritma *Greedy by distance* yang serupa dengan. *Greedy by distance* merupakan pendekatan algoritma greedy yang memprioritaskan jarak antara pemain dengan objek yang terpendek/terdekat. Bot akan menuju ke objek yang memiliki jarak terdekat dan memakan objek tersebut. Kemudian bot akan menuju ke objek lain dengan jarak terdekat.

```
fungsi find_nearest_diamond(bot_position,
list_diamonds, prefer_low_value = False):
    Inisialisasi jarak terpendek dengan nilai tak
terhingga
    Atur variabel nearest_diamond menjadi null

    Iterasi melalui setiap diamond dalam list_diamonds,
    untuk setiap diamond dalam list_diamonds:
        Hitung jarak antara diamond dan posisi bot
```

```
        jarak = absolut(diamond.position.x -
bot_position.x) + absolut(diamond.position.y -
bot_position.y)

        Jika prefer_low_value True, lewati diamond yang
bukan memiliki nilai 1
        Jika prefer_low_value = true dan
diamond.properties.points != 1:
            lanjutkan iterasi ke diamond berikutnya

        Perbarui nearest_diamond jika ditemukan diamond
yang lebih dekat
        jika jarak < nearest_diamond:
            jarak_terpendek = jarak
            nearest_diamond = diamond

    kembalikan nearest_diamond
```

b. Alternatif 2: Berdasarkan Poin.

Alternatif algoritma selanjutnya yaitu Algoritma *Greedy by point*. Memprioritaskan objek-objek dengan poin terbesar merupakan konsep dari algoritma *Greedy by point* ini. Bot akan menuju ke objek dengan poin terbesar yaitu *red diamond*. Selanjutnya bot akan menuju ke objek lain yang memiliki poin terbesar lagi.

```
Fungsi find_greatest_diamond menerima bot_position,
list_diamonds, prefer_low_value (default ke False):
     Atur shortest_distance ke infinity
     Atur greatest_diamond ke None
     Cek apakah ada diamond dengan poin 2 di
list_diamonds

     Untuk setiap diamond di list_diamonds:
```

```
     Hitung jarak dari bot_position ke posisi diamond

     Jika prefer_low_value adalah True:
          Jika poin diamond adalah 1 dan jarak lebih
pendek dari shortest_distance:
               Atur shortest_distance ke jarak saat
ini
               Atur greatest_diamond ke diamond saat
ini
     Jika tidak:
          Jika ada diamond dengan poin 2:
               Jika jarak lebih pendek dari
shortest_distance dan poin diamond adalah 2:
                    Atur shortest_distance ke jarak
saat ini
                    Atur greatest_diamond ke diamond
saat ini
          Jika tidak:
               Jika jarak lebih pendek dari
shortest_distance dan poin diamond adalah 1:
                    Atur shortest_distance ke jarak
saat ini
                    Atur greatest_diamond ke diamond
saat ini

     Kembalikan greatest_diamond
```

c. Alternatif 3: Berdasarkan Perbandingan Poin Per Jarak (Densitas).

Alternatif algoritma terakhir yaitu Algoritma *Greedy by density*. Algoritma ini akan memprioritaskan objek-objek yang memiliki densitas (poin/jarak) terbesar. Setiap objek yang memiliki poin akan dihitung densitasnya, kemudian bot akan menuju ke objek dengan densitas terbesar. Pseudocode untuk alternatif ini ditampilkan pada empat.

3.3. Analisis Efisiensi dan Efektivitas Solusi

Setelah dilakukan eksplorasi algoritma Greedy yang dapat dilakukan untuk mendapatkan diamond sebanyak-banyaknya, berikut analisis efisiensi dan efektivitas dari ketiga alternatif solusi;

a. Berdasarkan Kompleksitas Algoritma

1. Alternatif 1: berdasarkan jarak

Algoritma ini memiliki fungsi dasar find_nearest_diamond yang didalamnya terdapat perulangan pada setiap diamond yang ada pada list. List diamond sudah disediakan dari sisi *backend* permainan. Sehingga dengan N sebagai banyak diamond pada papan permainan, kompleksitas algoritma ini adalah T(N) dan O(N).

2. Alternatif 2: berdasarkan poin.

Algoritma ini memiliki fungsi dasar find_greatest_diamond yang didalamnya menggunakan fungsi any() dari python untuk mendapatkan list semua diamond yang berpoin 2. Setelah itu, dilakukan perulangan untuk menentukan jarak terdekat dari list diamond berpoin 2 tersebut untuk mendapatkan posisi diamond yang akan dituju. Sehingga dengan N adalah banyak diamond pada papan permainan dan dengan kemungkinan terburuk yaitu setiap diamond berpoin 2, kompleksitas algoritma ini adalah T(2N) atau O(N).

3. Alternatif 3: berdasarkan perbandingan poin per jarak (densitas).

Algoritma ini memiliki fungsi dasar highest_density yang didalamnya terdapat perulangan untuk mengecek densitas (poin/jarak) dari setiap diamond. Sehingga dengan N adalah banyak diamond pada papan permainan, kompleksitas algoritma ini adalah T(N) atau O(N).

b. Berdasarkan Pertandingan

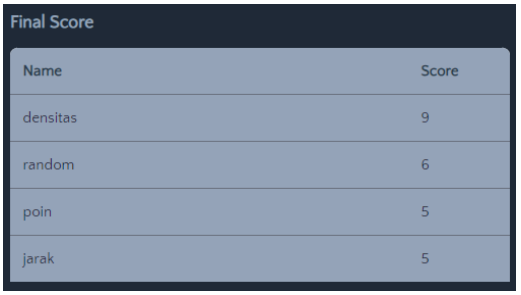
Dilakukan pertandingan sebanyak 5 kali antar alternatif yang ada dan didapatkan hasil sebagai berikut:

Jenis Algoritma	Pertandingan 1	Pertandingan 2	Pertandingan 3	Pertandingan 4	Pertandingan 5	Total

Jarak	9	5	10	10	10	44
Poin	10	5	5	13	10	43
Densitas	10	9	10	13	14	56
Random	0	6	0	1	0	7

Bukti dokumentasi pertandingan:

Gambar 2. Hasil Pertandingan 1



Gambar 3. Hasil Pertandingan 2

Gambar 4. Hasil Pertandingan 3

Gambar 5. Hasil Pertandingan 4

Gambar 6. Hasil Pertandingan 5

c. Pembahasan Analisis

Berdasarkan hasil pengambilan sampel dari lima pertandingan, didapatkan algoritma Greedy berdasarkan densitas atau perbandingan poin per jarak mendapatkan total poin yang terbanyak dibandingkan algoritma lainnya. Selain itu, meskipun alternatif ketiga tidak selalu mendapatkan peringkat pertama tetapi secara konsistensi, alternatif ketiga cukup stabil dalam memperoleh poin.

Algoritma Greedy berdasarkan jarak memiliki kelemahan ketika terdapat dua diamond dengan jarak yang sama tetapi memiliki poin yang berbeda. Dalam kasus tersebut, Greedy berdasarkan jarak memungkinkan memilih yang poinnya yang lebih sedikit, akibat dari tidak memperhatikan poin dan hanya berdasarkan posisi terdekat yang pertama kali ditemukan. Sedangkan algoritma Greedy berdasarkan poin memiliki kelemahan ketika diamond yang berpoin dua memiliki jarak yang sangat jauh dibandingkan diamond berpoin satu. Hal tersebut tidak efisien secara waktu dan terdapat kemungkinan terdapat dua diamond berpoin satu dengan jarak yang lebih dekat.

Algoritma Greedy berdasarkan densitas dapat mengatasi masalah-masalah Greedy berdasarkan jarak atau poin dengan lebih rasional. Secara teori pula Greedy berdasarkan densitas lebih baik dibandingkan Greedy yang hanya berdasarkan poin atau jarak. Karena pada dasarnya densitas mengkombinasikan kerakusan antara poin terbesar dan jarak terdekat. Sehingga bisa disebut pula keputusan dalam Greedy densitas lebih melibatkan banyak variabel dalam pengambilan keputusan.

Namun, algoritma Greedy berdasarkan densitas ini tidak dapat dipastikan akan selalu menjadi yang terbaik, terlihat pula dalam beberapa pertandingan memiliki jumlah skor yang sama dengan alternatif lain. Hal tersebut terjadi karena masih banyak variabel atau aturan lain yang tidak dapat ditangani oleh algoritma Greedy dasar. Seperti adanya aturan *tackle* yang tidak dapat diperkirakan secara pasti dalam Greedy yang bersifat hanya memikirkan keputusan terbaik saat itu juga. Penempatan posisi awal bot yang dapat menentukan seberapa cepat dalam mendapatkan diamond,

pada permainan ini bersifat acak sehingga keadilan antar bot setiap pertandingan dapat dibilang tidak konsisten.

3.4. Strategi Greedy Terpilih

Berdasarkan hasil analisis efisiensi dan efektifitas dari berbagai alternatif algoritma solusi yang ada, kami memutuskan untuk menggunakan algoritma alternatif ketiga yaitu berdasarkan perbandingan poin/jarak (densitas). Tetapi kami juga mengembangkan beberapa strategi tambahan baik dengan mengkombinasikan algoritma Greedy lain maupun dengan memaksimalkan beberapa fitur yang ada di permainan. Fitur yang dimaksud ialah seperti adanya teleport, *red button,* ataupun kemampuan *tackle* lawan.

Berikut adalah alur algoritma yang telah dibuat dengan memaksimalkan semua algoritma dan fitur:

1. Cek berapa diamond yang telah diperoleh bot pemain, jika telah memenuhi inventory maka bot akan langsung bergerak ke base. Karena dengan inventory penuh, bot pemain tidak dapat melakukan penambahan diamond lagi.
2. Jika terdapat bot lawan yang berada tepat disebelahnya, gerak bot lawan lebih lambat dan bot lawan membawa sejumlah diamond serta masih terdapat ruang inventory di bot pemain, maka dilakukan *tackle* terhadap lawan.
3. Jika waktu permainan yang tersisa dalam satuan detik sama dengan jarak perjalanan bot pemain ke base dan sedang membawa diamond, maka bot pemain akan langsung bergerak ke base untuk segera mengamankan diamond yang telah didapatkan.
4. Selanjutnya dilakukan pencarian letak diamond dengan densitas tertinggi. Dalam pencarian tersebut, sekaligus dilakukan pengecekan untuk jarak jika menggunakan teleport.
5. Setelah didapatkan posisi diamond tertinggi dilakukan pengecekan jumlah diamond yang tersisa di papan

permainan. Jika yang tersisa kurang dari sama dengan 7 akan dicari letak *red button* yang dapat men-*generate* ulang diamond. Jika letak *red button* lebih dekat daripada letak posisi diamond, bot pemain akan memilih untuk bergerak menuju *red button*.

6. Jika diamond yang sedang dituju memiliki poin dua, sedangkan ruang inventory yang tersisa hanya satu, maka akan dilakukan pengubahan tujuan ke diamond yang berpoin satu.
7. Bot pemain akan menuju posisi tujuan berdasarkan hasil semua pengecekan. Dilakukan pemanfaatan teleport jika memungkinkan untuk mendapatkan jarak perjalanan yang lebih dekat.
8. Jika pergerakan yang dihasilkan tidak valid akan dilakukan pergerakan random dengan fungsi random dari python.

Algoritma Greedy dilakukan pada langkah ke-4 yaitu mencari diamond dengan densitas tertinggi. Algoritma tersebut dilakukan setelah pengecekan berbagai kejadian yang dapat menyebabkan bot pemain tidak dapat bersifat 'greedy' yaitu seperti inventory penuh atau justru menyebabkan bot pemain dapat bersifat lebih 'greedy' yaitu seperti dengan *tackle* lawan. Algoritma Greedy berdasarkan pertahanan (*by defense*) juga dilakukan ketika waktu permainan yang tersisa sedikit yaitu dengan segera kembali ke base.

Pemanfaatan fitur teleport dilakukan pada langkah ke-4 ketika pencarian diamond dengan densitas tertinggi. Selain itu juga dilakukan pada langkah ke-6, ketika sebelum bot pemain benar-benar bergerak ke posisi tujuan seperti base dan *red button* agar mendapatkan jarak perjalanan yang efektif. Sedangkan fitur *red button* dimanfaatkan ketika jumlah diamond pada papan permainan tersisa sedikit dan jarak posisi menuju *red button* lebih dekat dibandingkan posisi menuju ke diamond. Sehingga pemanfaatan

fitur *red button* ini secara tidak langsung juga menerapkan algoritma Geedy berdasarkan jarak.

BAB IV
IMPLEMENTASI DAN UJI COBA

4.1. Implementasi Algoritma Greedy pada Bot FreeDiamonds dalam pseudocode

```
Fungsi highest_density menerima bot_position, board,
list_diamonds, prefer_low_value (default ke False):
      Atur highest_density ke 0
      Atur highest_density_diamond ke None
      Atur highest_density_position ke None
      Atur teleporters ke daftar semua objek game
teleporter di papan

      Jika prefer_low_value adalah True:
      Cari diamond terdekat (dengan poin 1) ke
bot_position
      Kembalikan posisi diamond terdekat

      Jika tidak:
      Untuk setiap diamond di list_diamonds:
            Hitung jarak dari bot_position ke posisi
diamond
            Hitung jarak dari bot_position ke posisi
diamond jika menggunakan teleporter

            Jika jarak teleporter lebih kecil dari jarak
langsung:
                  Atur jarak ke jarak teleporter

            Dapatkan poin dari diamond
            Jika jarak bukan 0:
                  Hitung density sebagai poin dibagi
jarak
```

```
                Jika densitas lebih besar dari
highest_density yang ditemukan sejauh ini, atau jika
density sama tetapi poin lebih tinggi:
                    Atur highest_density ke density
saat ini
                    Atur highest_density_diamond ke
diamond saat ini
                    Atur highest_density_position ke
posisi diamond saat ini

     Kembalikan posisi diamond dengan density tertinggi
```

4.2. Struktur Data

Struktur Data	Deskripsi
GameObject	Objek dalam game yang mewakili entitas pada papan, seperti bot, diamond, teleporter, dan lainnya. Setiap GameObject memiliki posisi dan properti tertentu
Board	Mewakili keadaan papan permainan saat ini, termasuk ukuran papan dan posisi semua GameObject
Position	Struktur data yang menyimpan koordinat posisi objek pada papan (x, y)
Tuple	Digunakan untuk menyimpan koleksi item yang tidak dapat berubah. Pada program, tuple digunakan untuk menyimpan arah pergerakan sebagai pasangan nilai x dan y
List	Digunakan untuk menyimpan koleksi item yang dapat berubah. Pada program, list digunakan untuk menyimpan berbagai objek permainan dan memungkinkan operasi seperti iterasi, penambahan, dan penghapusan elemen

4.3. Analisis Desain Solusi

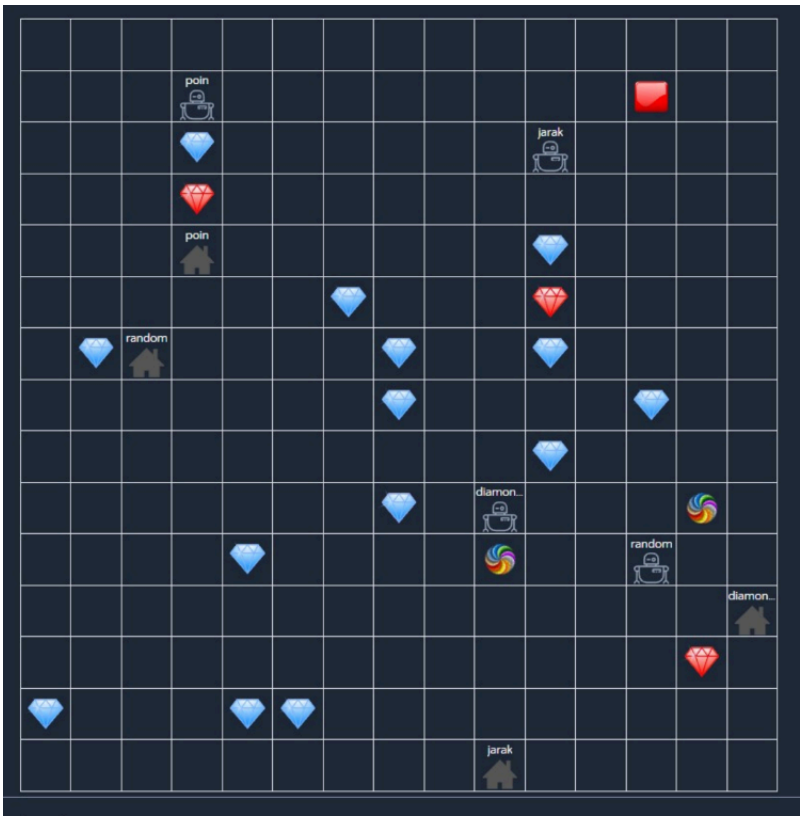
Pada bot yang kami buat, kami mengimplementasikan algoritma Greedy berdasarkan densitas terbesar. Densitas dihitung dengan membagi nilai poin dari *diamond* dengan jarak dari posisi bot ke *diamond*. Kami juga mendesain agar Greedy berdasarkan poin terbesar menjadi prioritas kedua, yaitu pada kasus dua *diamond* mempunyai densitas yang sama, bot akan memilih untuk mendapatkan *diamond* dengan poin yang lebih besar. Selain itu, ketika *inventory* hanya bisa memuat 1 poin lagi, kami menerapkan Greedy berdasarkan jarak terdekat dengan membuat bot berjalan ke *diamond* dengan poin 1 terdekat.

Pemanfaatan berbagai fitur yang ada dalam permainan juga kami lakukan untuk meningkatkan tingkat 'greedy'. Dengan memanfaatkan teleport bot permainan dapat meminimumkan langkah dalam mengambil *diamond*. Kemudian dengan pengecekan posisi bot lawan untuk melakukan *tackle* sehingga dapat menambah poin dengan cepat. Serta *red button* untuk memperbanyak kesempatan mengambil kembali *diamond*. Langkah-langkah secara implementatif telah kami jelaskan dalam bab 3 pada sub bab strategi Greedy terpilih.

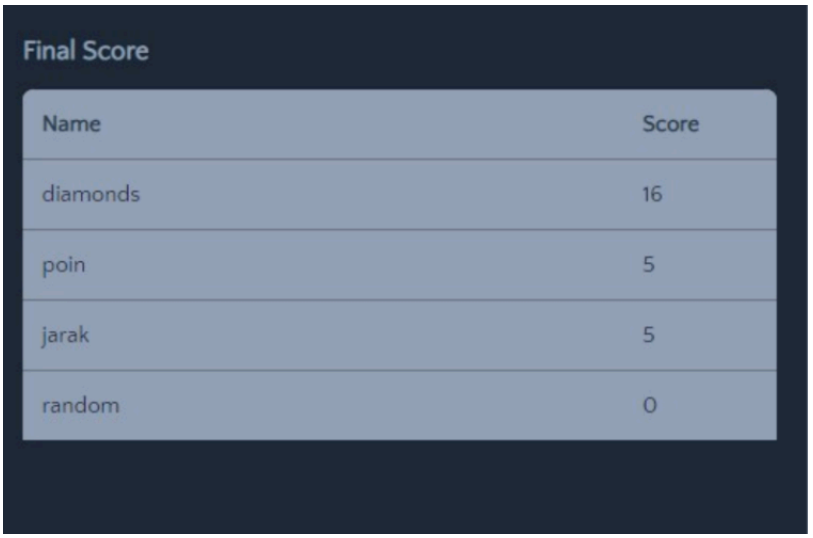
Selama melakukan pengujian, kami mendapatkan beberapa kasus yang menyebabkan solusi Greedy optimal dan tidak optimal. Dalam skenario di mana *diamond* tersebar rata di seluruh papan, pendekatan berdasarkan densitas terbesar cenderung efektif karena memaksimalkan nilai poin yang diperoleh relatif terhadap jarak yang ditempuh. Namun, ketika terdapat kasus *diamond* yang berkelompok, jika sekelompok *diamond* berada sedikit lebih jauh dibandingkan dengan satu *diamond* terdekat, strategi Greedy berdasarkan densitas lebih memilih untuk mengambil satu diamond terdekat (densitas lebih besar) daripada mengumpulkan sekelompok diamond dalam satu perjalanan, yang seharusnya lebih menguntungkan.

Pada pengujian di bawah, bot yang kami implementasikan, yaitu bot "diamonds", memiliki *score* yang cukup tinggi dan berbeda jauh dengan bot

uji lainnya. Dapat dilihat bahwa letak *diamond* pada *board* memiliki persebaran yang rata.



Gambar 4.1 *Board* memiliki persebaran *diamond* yang rata

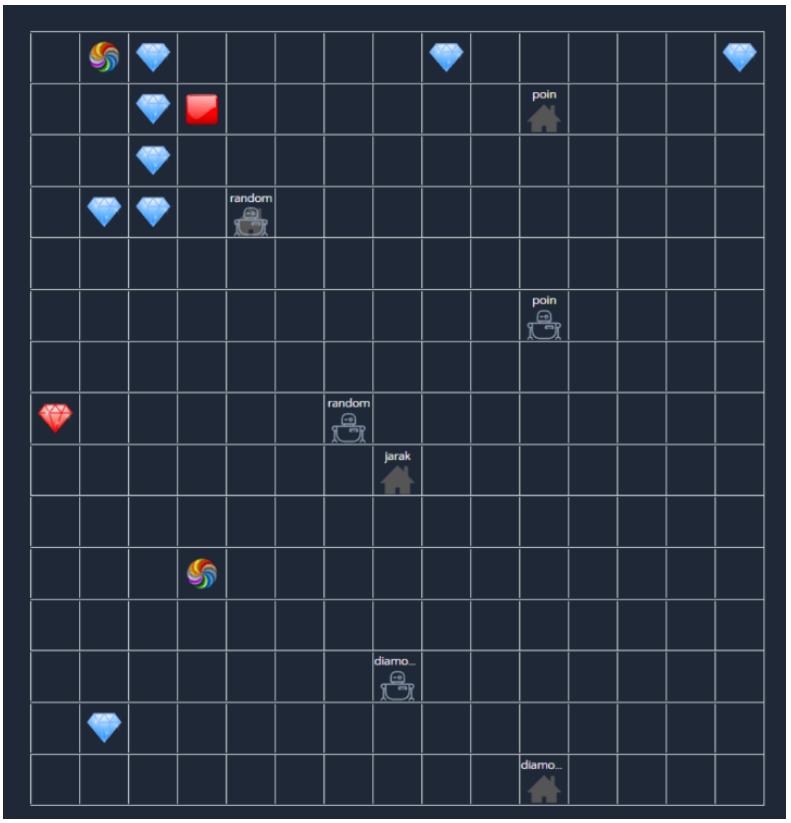


Final Score

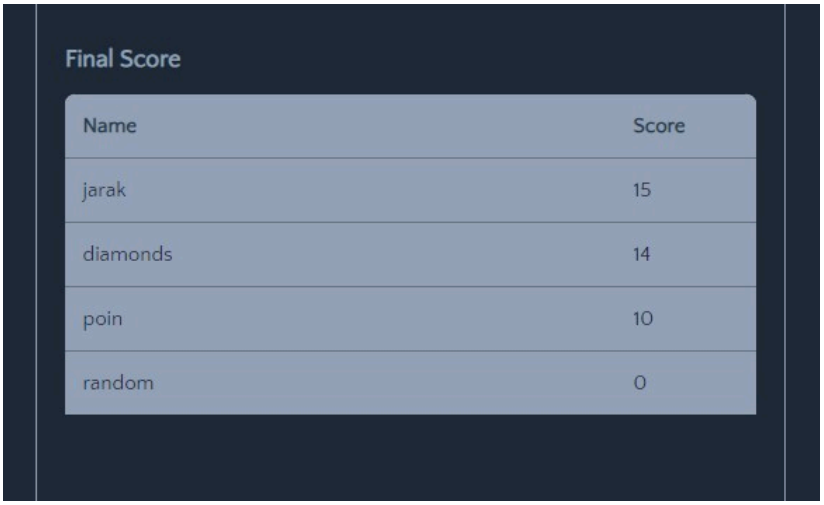
Name	Score
diamonds	16
poin	5
jarak	5
random	0

Gambar 4.2 Bot “diamonds” memiliki *score* yang tinggi

Pada kasus pengujian berikut, bot “diamonds” berada jauh dari kerumunan *diamond*. Sedangkan, bot “jarak” berada di dekat kerumunan tersebut. Dapat dilihat hasil *score* akhirnya, bot “jarak” mengalahkan bot “diamonds”.



Gambar 4.3 Skenario ketika bot berada jauh dari kerumunan *diamond*



Final Score

Name	Score
jarak	15
diamonds	14
poin	10
random	0

Gambar 4.4 Bot yang mengimplementasikan jarak menang dari bot

BAB V
KESIMPULAN, SARAN, DAN REFLEKSI

5.1 Kesimpulan

Terdapat berbagai macam strategi algoritma, salah satunya adalah algoritma Greedy. Salah satu contoh pemanfaatan algoritma Greedy adalah untuk menyelesaikan permainan *Diamonds* ini. Dalam permainan ini pemain dituntut untuk melakukan 'kerakusan' dalam mendapatkan diamond sebanyak-banyaknya, sehingga algoritma Greedy sangat cocok untuk mengatasinya.

Melalui pembuatan bot ini, kami semakin memahami tentang bagaimana algoritma Greedy dibuat dan dapat bekerja. Kami telah mengeksplorasi hingga menemukan berbagai alternatif algoritma Greedy untuk permainan *Diamonds* ini. Lalu, setelah melakukan analisis perbandingan efisiensi dan efektifitas algoritma, kami menyimpulkan algoritma greedy berdasarkan densitas adalah yang paling optimal dibandingkan alternatif lainnya.

5.2 Saran

Kami menyadari bahwa bot ini tidak selalu menghasilkan solusi optimal dan mungkin terdapat algoritma Greedy yang lebih baik dari yang kami implementasikan. Oleh karena itu, kami sangat menghargai kritik dan saran dari semua pihak. Kami juga mengusulkan agar pada saat melakukan *setup game engine*, prosesnya dapat disederhanakan agar lebih mudah dipahami oleh kami dan langkah-langkahnya dipastikan dapat berjalan bagi semua orang. Dengan demikian, kami bisa mengerjakan lebih optimal lagi tanpa harus menghabiskan waktu menyelesaikan *error* saat *setup game engine.*

LAMPIRAN

Link github: https://github.com/novelxv/Tubes1_FreeDiamonds

Link video: https://youtu.be/pLJuYmBtVQw?si=hqPK6_CcU8XT7U5X

DAFTAR PUSTAKA

Greedy Algorithm in Algorithm Design and Analysis. Jaro Education, https://www.jaroeducation.com/blog/what-is-greedy-algorithm-in-algorithm-design-and-analysis/.

Greedy Algorithms. freeCodeCamp, https://www.freecodecamp.org/news/greedy-algorithms/.

Greedy Algorithms. GeeksforGeeks, https://www.geeksforgeeks.org/greedy-algorithms/.

IF2211 Strategi Algoritma - Semester II Tahun 2023/2024. (n.d.). Informatika. Retrieved February 9, 2024, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/stima23-24.htm

Spesifikasi Tugas Besar 1 IF2211 Strategi Algoritma Pemanfaatan Algoritma Greedy dalam pembuatan bot permainan Diamonds. (2021). Bandung: Institut Teknologi Bandung.