LAPORAN TUGAS BESAR

IF2211 STRATEGI ALGORITMA

PEMANFAATAN ALGORITMA GREEDY DALAM PEMBUATAN BOT PERMAINAN DIAMONDS



Dipersiapkan oleh:

Diwan Ramadhani Dwi Putra 123140116

M. Gymnastiar Syahputra 123140135

Jordy Anugrah Akbar 123140141

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK INDUSTRI

INSTITUT TEKNOLOGI SUMATERA

2025

Contents

[BAB II 7](#_7ctoyo917nvu)

[LANDASAN TEORI 7](#_f7ot7x3baih8)

[BAB III 8](#_6edhle137wap)

[APLIKASI STRATEGI GREEDY 8](#_hnpn49rl7ona)

[BAB IV 9](#_u3szo5h14ieh)

[IMPLEMENTASI DAN PENGUJIAN 9](#_93yg0lcf5kdk)

[BAB V 10](#_dgu900asvcj7)

[KESIMPULAN DAN SARAN 10](#_q6ey0vhywz4e)

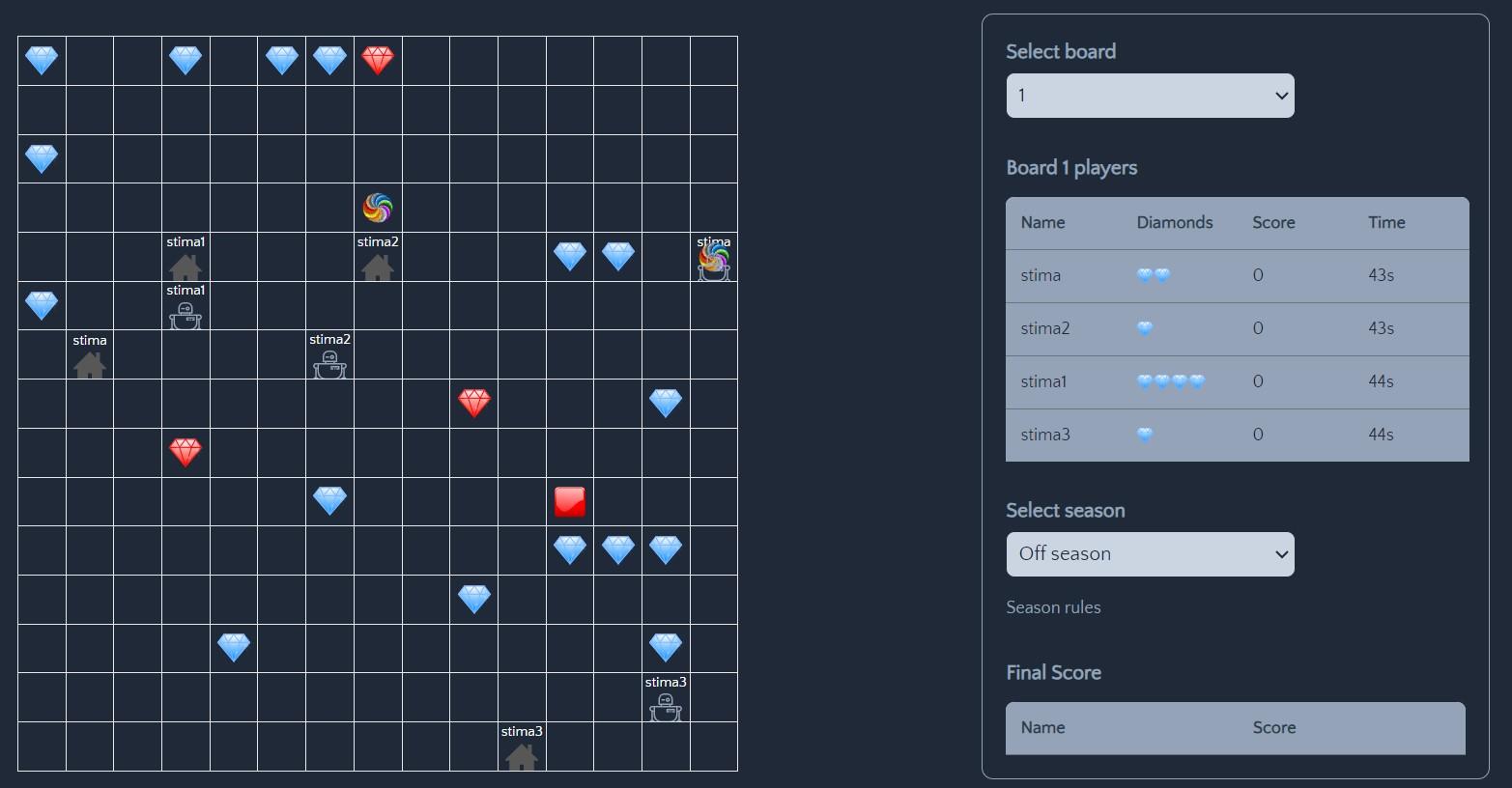
[LAMPIRAN 11](#_muvds5ngx0mn)

[DAFTAR PUSTAKA 12](#_po0tapmbxtqf)

**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. Penjelasan lebih lanjut mengenai aturan permainan akan dijelaskan di bawah.



Pada tugas pertama Strategi Algoritma ini, mahasiswa diminta untuk membuat sebuah bot yang nantinya akan dipertandingkan satu sama lain. Tentunya mahasiswa harus menggunakan **strategi *greedy*** dalam membuat bot ini.

Program permainan Diamonds terdiri atas:

1. *Game engine,* yang secara umum berisi:
   1. Kode *backend* permainan, yang berisi *logic* permainan secara keseluruhan serta API yang disediakan untuk berkomunikasi dengan *frontend* dan program bot
   2. Kode *frontend* permainan, yang berfungsi untuk memvisualisasikan permainan
2. *Bot starter pack,* yang secara umum berisi:
   1. Program untuk memanggil API yang tersedia pada *backend*
   2. Program *bot logic* (bagian ini yang akan kalian implementasikan dengan algoritma *greedy* untuk bot kelompok kalian)
   3. Program utama (*main*) dan utilitas lainnya

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*.

1. **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.

1. **Teleporters**



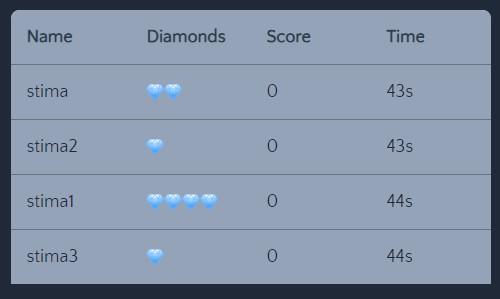
Terdapat 2 *teleporter* yang saling terhubung satu sama lain. Jika bot melewati sebuah *teleporter* maka bot akan berpindah menuju posisi *teleporter* yang lain.

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.

Untuk mengetahui *flow* dari game ini, 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.

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**

**A.** **Algortima Greedy**

Algoritma greedy adalah algoritma yang memecahkan persoalan secara langkah per langkah (step by step) sedemikian sehingga, pada setiap langkah:

1. Mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsip“take what you can get now!”).

2. Berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Dalam menjalankan algoritma greedy ada beberapa elemen yang diperlukan agar proses pemecahan masalah secara greedy lebih mudah untuk dilakukan. Elemen- elemen tersebut adalah sebagai berikut.

1. Himpunan kandidat (C), berisi kandidat yang akan dipilih pada setiap Langkah. Contohnya adalah simpul atau sisi di dalam graf, job, task, koin, benda, dan karakter.

2. Himpunan solusi (S), berisi kandidat yang sudah dipilih untuk menjadi solusi.

3. Fungsi solusi yang digunakan menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi atau belum.

4. Fungsi seleksi (selection function) untuk memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik yang tentunya berbeda untuk setiap masalah yang dihadapi.

5. Fungsi kelayakan (feasible) untuk memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi atau tidak.

6. Fungsi objektif untuk memaksimumkan atau meminimumkan kandidat yang dipilih.

Setelah semua elemen- elemen tersebut lengkap, maka proses pencarian solusi akan bisa dilakukan. Algoritma greedy melibatkan pencarian sebuah himpunan bagian (S) dari himpunan kandidat (C) yang dalam hal ini, himpunan S harus memenuhi beberapa kriteria 7 yang ditentukan, yaitu S menyatakan suatu solusi dan S di optimisasi oleh fungsi objektif. Algoritma greedy akan menghasilkan beberapa solusi optimum lokal dan dari semua solusi lokal tersebut akan dipilih solusi terbaik yang menjadi solusi optimum global.

Akan tetapi, solusi optimum global yang dihasilkan algoritma greedy belum tentu merupakan solusi terbaik karena sangat mungkin solusi tersebut merupakan solusi sub-optimum atau pseudo-optimum. Ada dua alasan kenapa hal tersebut bisa terjadi, yaitu sebagai berikut:

1. Algoritma greedy tidak beroperasi secara menyeluruh terhadap semua kemungkinan solusi yang ada (sebagaimana pada metode exhaustive search).

2. Terdapat beberapa fungsi seleksi yang berbeda, sehingga perlu memilih fungsi yang tepat jika algoritma diinginkan untuk menghasilkan solusi optimal.

Sebagai kesimpulan, algoritma greedy adalah metode penyelesaian masalah secara bertahap dengan memilih solusi terbaik pada setiap langkah tanpa mempertimbangkan konsekuensi jangka panjang, dengan harapan solusi lokal yang dipilih akan menghasilkan solusi global yang optimal. Untuk menjalankan algoritma ini secara efektif, diperlukan beberapa elemen penting seperti himpunan kandidat, himpunan solusi, fungsi seleksi, fungsi kelayakan, dan fungsi objektif. Meskipun efisien, algoritma greedy tidak selalu menjamin solusi optimal secara global karena tidak mengevaluasi semua kemungkinan solusi dan sangat bergantung pada strategi seleksi yang digunakan. Oleh karena itu, pemilihan fungsi seleksi yang tepat sangat penting untuk mendekati hasil yang optimal.

Dalam penerapannya, ada cukup banyak permasalahan yang bisa diselesaikan dengan pendekatan greedy, yaitu sebagai berikut:

1. Persoalan penukaran uang (coin exchange problem)

2. Persoalan memilih aktivitas (activity selection problem)

3. Minimisasi waktu di dalam system

4. Persoalan knapsack (knapsack problem)

5. Penjadwalan job dengan tenggat waktu (job schedulling with deadlines)

6. Pohon merentang minimum (minimum spanning tree)

7. Lintasan terpendek (shortest path)

8. Kode Huffman (Huffman code)

9. Pecahan Mesir (Egyptian fraction)

# **BAB III**

# **APLIKASI STRATEGI GREEDY**

1. **Mapping Persoalan Diamonds**

Berdasarkan studi pustaka yang telah dilakukan pada landasan teori, dikatakan algoritma greedy melibatkan pencarian sebuah himpunan bagian (S) dari himpunan kandidat (C) yang dalam hal ini, himpunan S harus memenuhi beberapa kriteria yang ditentukan, yaitu S menyatakan suatu solusi dan S di optimisasi oleh fungsi objektif. Oleh karena itu, untuk bisa menerapkan algoritma greedy dalam permasalahan ini, maka perlu diidentifikasi terlebih dahulu elemen- elemen yang ada di dalam game Diamonds menjadi elemen- elemen algoritma greedy, yaitu sebagai berikut:

1. Himpunan kandidat (C)

Himpunan kandidat dalam permasalahan ini akan berisi koordinat- koordinat objek yang ada di dalam game. Objek- objek tersebut adalah blue diamond, red diamond, red button, teleporter, base bot, bot musuh, dan base musuh. Selain itu, ada juga elemen penting yang perlu diperhatikan dalam game, yaitu jarak bot ke koordinat tujuan.

2. Himpunan solusi (S)

Himpunan solusi berisi koordinat yang akan memberikan keuntungan paling banyak ketika didatangi oleh bot. Berdasarkan pendekatan greedy yang dilakukan, himpunan solusi bisa hanya berisi satu koordinat atau beberapa koordinat yang harus dilalui secara berurutan.

3. Fungsi solusi

Fungsi solusi berguna untuk menentukan apakah kandidat yang dipilih sudah memberikan solusi. Secara umum, fungsi ini hanya akan digunakan ketika permainan akan berakhir. Fungsi ini akan melakukan perhitungan apakah waktu yang tersisa cukup untuk menempuh suatu koordinat dalam map dan untuk kembali ke base. Alasan kenapa fungsi ini hanya dilakukan di akhir permainan adalah karena fungsi-fungsi untuk 13 menentukan kandidat solusi pada fungsi seleksi sudah menghilangkan kandidat yang tidak bisa menjadi solusi.

4. Fungsi Seleksi (Selection)

Fungsi seleksi berguna untuk memilih kandidat berdasarkan strategi greedy tertentu. Fungsi seleksi akan dijelaskan lebih lanjut pada bagian alternatif solusi karena pendekatan greedy yang berbeda akan memiliki fungsi seleksi yang berbeda.

5. Fungsi Kelayakan (Feasible)

Fungsi kelayakan berguna untuk memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi. Dalam game Diamonds, fungsi kelayakan akan digunakan untuk menentukan apakah objek dalam game bisa menjadi tujuan berikutnya atau tidak. Contoh yang paling sederhana adalah ketika red diamond menjadi tujuan selanjutnya, namun bot sudah memiliki 4 diamonds dalam inventory sehingga red diamond tidak bisa diambil. Fungsi kelayakan akan mencegah bot untuk menuju koordinat tersebut, karena apabila bot tetap menuju koordinat tersebut ada kemungkinan terjadi bug yaitu bot akan berputar- putar di sekitar red diamond atau bahkan memberikan invalid move.

6. Fungsi objektif

Fungsi objektif digunakan untuk memaksimalkan atau meminimumkan dari himpunan kandidat menjadi solusi. Dalam permasalahan ini himpunan kandidat akan melalui fungsi objektif untuk ditentukan mana kandidat yang memiliki keuntungan terbesar. Jadi, fungsi objektif akan diterapkan untuk memaksimalkan keuntungan yang didapat dari himpunan kandidat

1. **Alternatif Solusi**
2. **Algoritma Greedy by Time-Weighted**

Algoritma greedy yang digunakan dalam proyek ini merupakan versi modifikasi dari greedy klasik, yang dinamakan Time-Weighted Greedy Algorithm. Strategi ini tidak hanya memperhitungkan nilai diamond terhadap jarak tempuh (value per distance), tetapi juga mempertimbangkan sisa waktu permainan (time-weight). Tujuannya adalah agar bot mampu mengambil keputusan yang cepat namun tetap adaptif terhadap kondisi permainan yang dinamis, seperti tekanan waktu dan jumlah diamond yang sudah dikumpulkan.

1. Himpunan Kandidat (Candidate Set)

Pada setiap langkah permainan, bot harus menentukan ke mana ia akan bergerak selanjutnya. Opsi-opsi yang bisa dipilih meliputi semua diamond yang tersebar di papan permainan, termasuk juga tombol-tombol khusus yang bisa memberi keuntungan besar serta jalur portal yang dapat mempercepat perpindahan. Semua objek tersebut merupakan kandidat yang dapat dipertimbangkan oleh algoritma. Pemindaian terhadap kandidat dilakukan terus-menerus selama permainan berjalan, disesuaikan dengan posisi bot, kondisi papan, dan sisa waktu yang tersedia.

1. Himpunan Solusi (Solution Set)

Solusi dalam konteks ini adalah rangkaian langkah atau lintasan yang dilalui bot dari posisi awal, ke target yang dituju, hingga kembali ke base. Solusi ini dibentuk secara bertahap—langkah demi langkah—berdasarkan hasil pemilihan target terbaik di setiap siklus. Meskipun tidak disimpan sebagai daftar, solusi dibangun dalam bentuk arah gerakan yang ditentukan dari posisi bot menuju target yang telah dipilih. Bot akan terus bergerak menuju target tersebut sampai ia mencapai tujuan atau terjadi evaluasi ulang terhadap target baru.

1. Fungsi Seleksi (Selection Function)

Setiap kali dihadapkan pada berbagai kandidat, algoritma akan melakukan pemilihan berdasarkan perbandingan antara nilai suatu diamond dengan jarak tempuh yang diperlukan untuk mencapainya. Namun, karena permainan ini dibatasi oleh waktu, maka algoritma juga memasukkan pertimbangan waktu: semakin sedikit sisa waktu, semakin diprioritaskan target yang lebih dekat, meskipun nilainya lebih kecil. Skor dari masing-masing kandidat dihitung dengan menyesuaikan antara nilai, jarak, dan urgensi waktu tersebut. Kandidat dengan skor tertinggi akan ditetapkan sebagai target selanjutnya.

1. Fungsi Kelayakan (Feasibility Function)

Tidak semua kandidat bisa langsung dijadikan target. Beberapa kondisi tertentu membuat kandidat menjadi tidak layak untuk dipilih. Sebagai contoh, jika bot sudah membawa sejumlah diamond tertentu, maka mengambil diamond tambahan justru berisiko karena bisa membuat waktu kembali ke base menjadi terlalu singkat. Selain itu, jika di jalur menuju target terdapat hambatan seperti portal atau diamond merah yang bisa berbahaya, maka algoritma akan menghindari jalur tersebut dan memilih alternatif jalur sementara. Dengan demikian, hanya kandidat yang memenuhi syarat kelayakan secara waktu, jarak, dan keamanan jalur yang akan diproses lebih lanjut.

1. Fungsi Solusi (Solution Function)

Algoritma tidak hanya fokus pada tujuan jangka pendek. Pada titik-titik tertentu, ia akan mengevaluasi apakah melanjutkan pencarian diamond merupakan pilihan terbaik, atau justru saatnya kembali ke base. Evaluasi ini mempertimbangkan sisa waktu dan jumlah diamond yang sudah dikumpulkan. Jika waktu dirasa tidak cukup untuk mengambil diamond tambahan, atau posisi bot terlalu jauh dari base, maka algoritma akan mengubah target menjadi base agar diamond yang sudah dikumpulkan tidak hilang. Evaluasi ini bersifat dinamis dan akan terus diperbarui setiap kali kondisi permainan berubah.

1. Fungsi Objektif (Objective Function)

Tujuan utama dari algoritma ini adalah memaksimalkan jumlah diamond yang berhasil dikumpulkan dan disimpan dengan aman di base sebelum waktu habis. Untuk itu, setiap keputusan yang diambil harus mengarah pada peningkatan skor akhir, baik melalui pemilihan diamond bernilai tinggi maupun dengan memastikan semua diamond yang telah dikumpulkan tidak hilang karena kehabisan waktu. Oleh karena itu, logika algoritma selalu mempertimbangkan keseimbangan antara eksplorasi (mencari diamond) dan eksploitasi (segera pulang ke base).

1. **Algoritma Greedy Value-to-Cost Ratio**

Algoritma greedy pada file ini menggunakan pendekatan rasio nilai terhadap biaya (value-to-cost ratio) dalam proses pemilihan diamond. Artinya, setiap diamond tidak hanya dinilai berdasarkan jarak, tetapi dibagi dengan poin (nilai) yang dimilikinya, sehingga diamond yang bernilai tinggi cenderung lebih diprioritaskan walaupun mungkin sedikit lebih jauh. Dalam implementasinya, hal ini menciptakan sebuah pendekatan greedy yang lebih ekonomis—mengusahakan hasil terbaik dengan biaya gerak serendah mungkin

1. Himpunan Kandidat (Candidate Set)

Setiap kali bot harus bergerak, ia melakukan identifikasi terhadap semua objek yang mungkin dituju. Objek-objek ini mencakup diamond biasa, diamond merah (bernilai 2), tombol khusus yang memberikan banyak diamond sekaligus, serta rute melalui portal yang bisa mempercepat perjalanan. Seluruh objek tersebut menjadi elemen dalam himpunan kandidat yang akan dipertimbangkan oleh bot saat memilih target berikutnya.

1. Himpunan Solusi (Solution Set)

Solusi dibangun dari serangkaian langkah atau perpindahan posisi bot menuju target yang terpilih, dan dilanjutkan hingga bot kembali ke base. Rangkaian langkah ini tidak ditampung dalam satu daftar eksplisit, melainkan terbentuk secara bertahap seiring bot bergerak dari satu posisi ke posisi berikutnya, berdasarkan target aktif yang terus diperbarui. Hasil akhirnya berupa lintasan penuh yang dilalui bot dari awal permainan hingga permainan berakhir.

1. Fungsi Seleksi (Selection Function)

Algoritma ini akan memilih kandidat terbaik dari semua opsi yang tersedia berdasarkan perbandingan antara jarak tempuh menuju diamond dan nilai dari diamond itu sendiri. Nilai ini dihitung sebagai rasio jarak dibagi poin. Semakin kecil nilai rasio tersebut, semakin besar kemungkinan kandidat tersebut dipilih. Strategi ini memungkinkan bot untuk lebih memilih dia mond yang nilainya besar meskipun sedikit lebih jauh, daripada diamond bernilai kecil yang lebih dekat. Selain itu, jika jalur melalui portal menghasilkan rasio yang lebih efisien, maka rute portal juga bisa dipilih sebagai prioritas.

1. Fungsi Kelayakan (Feasibility Function)

Tidak semua kandidat dapat langsung dijadikan target. Misalnya, ketika bot telah memiliki 4 diamond, ia tidak akan memilih diamond merah yang bernilai 2, karena pengambilan diamond kelima akan memicu keharusan segera kembali ke base. Logika ini juga memperhatikan hambatan-hambatan di jalur seperti portal dan diamond merah yang dapat mengganggu perjalanan. Dalam kasus tersebut, bot akan menghitung rute alternatif atau menggunakan target sementara untuk menghindari hambatan.

1. Fungsi Solusi (Solution Function)

Di sepanjang permainan, bot secara berkala mengevaluasi apakah ia masih aman untuk terus mengumpulkan diamond atau harus kembali ke base. Jika waktu hampir habis dan jumlah diamond yang dibawa cukup banyak (misalnya lebih dari 1), maka target akan segera diganti ke arah base. Evaluasi ini juga dilakukan jika jarak antara bot dan base sudah cukup dekat dan dianggap layak untuk kembali. Dengan begitu, solusi yang sedang dibangun bisa langsung dihentikan dan diarahkan untuk disimpan ke base demi menghindari kehilangan skor.

1. Fungsi Objektif (Objective Function)

Tujuan dari algoritma ini adalah untuk memperoleh skor sebanyak mungkin dalam waktu permainan yang terbatas. Skor diperoleh dari jumlah diamond yang berhasil dibawa pulang ke base. Maka dari itu, semua keputusan dalam permainan diarahkan untuk mengumpulkan diamond dengan nilai terbaik dalam jumlah gerak terkecil, sambil memastikan bahwa bot mampu kembali ke base tepat waktu. Dengan pendekatan perbandingan antara jarak dan nilai diamond, algoritma ini bertujuan untuk mencapai efisiensi maksimum secara lokal di setiap langkah pengambilan keputusan.

# **BAB IV**

# **IMPLEMENTASI DAN PENGUJIAN**

# **BAB V**

# **KESIMPULAN DAN SARAN**

# LAMPIRAN

# DAFTAR PUSTAKA