IMPLEMENTASI ALGORITMA GREEDY DALAM PEMECAHAN BOT PERMAINAN DIAMOND

Tugas Besar

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

Oleh: Kelompok BOTDFI

Danar Prayogo	**123140015**
Ilyas Ramadhan	**123140016**
Fanisa Aulia Safitri	**123140121**

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

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

2025

DAFTAR ISI

BAB I
DESKRIPSI TUGAS

1.1 Latar Belakang

Dalam tugas besar mata kuliah Strategi Algoritma memberikan kesempatan kepada mahasiswa untuk menerapkan pemahaman mereka dalam bentuk permaian kompetitif berjudul Diamonds. Permainan ini merupakan sebuah tantangan pemrograman dimana setiap peserta diminta untuk mengembangkan suatu bot yang otomatis dan dapat bersaing dengan bot peserta lawan. Tujuan dari permainan ini adalah mengumpulkan diamonds sebanyak-banyaknya. Namun permainan tersebut tidaklah mudah, karena permainan ini dilengkapi dengan berbagai rintangan seperti inventory yang terbatas, serta interaksi antar bot yang saling menjatuhkan.

Agar permainan menjadi unggul, setiap peserta harus merancang strategi yang efisien dan adaptif, terutama dengan menerapkan pendekatan algoritma Greedy. Algoritma ini dipilih karena mampu mengambil keputusan yang terbaik secara cepat, dan algoritma ini sangat sesuai dalam permainan dengan waktu terbatas serta kondisi lingkungan yang dapat berubah-ubah. Dengan strategi yang tepat, bot dapat secara efektif mengejar diamond, menghindari bahaya dan menyimpan hasil diamond ke dalam base.

1.2 Tujuan Tugas Besar

Tujuan dari tugas besar permainan Diamonds adalah sebagai berikut:

1. Menerapkan algoritma Greedy dalam konteks yang nyata khususnya dalam perancangan bot otomatis.
2. Dapat mengembangkan kemampuan berpikir analisis dan strategis.
3. Dapat meningkatkan keterampilan pemrograman serta penguasaan struktur data
4. Mengevaluasi efektivitas strategi algortima yang telah dirancang

1.3 Ruang Lingkup Tugas

Ruang Lingkup tugas besar permainan diamonds adalah sebagai berikut:

1. Peranvangan bot

   Bot dirancang untuk dapat bergerak secara otomatis berdasarkan strategi algoritma yang diteraokan, bot harus mampu mengambil keputusan untuk mengambil diamond, kembali ke base atau menghindari serangan dari musuh.

2. Penerapan Algritma Greedy

   Strategi utama yang diterapkan adalah Algoritma Greedy, dengan logika pemilihan langkah terbaik berdasarkan kondisi lokal saat itu.

3. Lingkungan Permainan

   Bot dioperasikan dalam papan dua dimensi yang dinamis, dan terdapat berbagai elemen seperti diamond yang memiliki warna merah dan biru, red button, base, teleport serta interaksi antar bot.

4. Batasan Teknis

   Bot harus dikembangkan dalam bahasa pemrograman Python dengan memanfaatkan game engine dan bot stater pack. Bot harus dapat dijalankan dan siap untuk berkompetisi dengan bot peserta lain.

5. Evaluasi dan Dokumentasi

   Evaluasi dilakukan dengan melakukan pengujian strategi bot dalam simulasi ataupun dalam kometisi, seluruh proses bot harus didokumentasikan dalam bentuk laporan.

1.4 Spesifikasi Tugas

Spesifikasi Tugas ini adalah

1. Tujuan Bot

   Bot harus dapat mengumpulkan diamond sebanyak mungkin, dengan keterangan diamond merah berisikan 2 point dan diamond biru berisikan 1 point, ketika inventory bot sudah penuh bot harus membawa diamond ke dalam base untuk menambahkan skor, karena diamond yang berhasil dibawa ke base akan dihitung sebagai point.

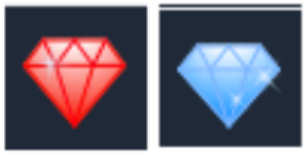
2. Ketentuan Bot

   Bot memiliki inventory yang terbatas, jika inventory penuh bot harus kembali ke base untuk menyimpan diamond, bot yang ditabrak oleh bot lawan akan

kehilangan seluruh diamond yang ada di inventorynya dan bot akan kembali lagi ke dalam base.

3. Elemen Permainan

   (a) Diamond



Gambar 1.1: Diamonds

Terdapat 2 jenis diamond yaitu red diamond dan blue diamond dengan point yang bereda. Red diamond memiliki 2 point dan blue diamond hanya memiliki 1 point

   (b) Red Buttom

Ketika Bot Peserta menginjak Red Button, posisi diamond akan diubah secara acak

   (c) Teleporter

Terdapat dua teleportasi yang terhubung satu sama lain dan dapat diguanakan untuk berpindah tempat.

   (d) Bots dan Base

Bot akan menampung beberapa diamond diinventorynya, jika inventory bot sudah penuh maka bot akan kembali ke base untuk menyimpan diamond. Apabila diamond disimpan ke dalam base maka score bot akan bertambah,

Gambar 1.2: Red Button

Gambar 1.3: Teleporter

Gambar 1.4: Bots dan Base

dengan posisi base tidak akan pernah berubah sampai akhir permainan.

BAB II
LANDASAN TEORI

2.1 Dasar Teori

Algoritma Greedy adalah metode yang paling populer untuk memecahkan persoalan optimasi, dalam bahasa inggris greedy diartikan sebagai tamak atau rakus. Algoritma Greedy adalah algoritma yang memecahkan masalah langkah per langkah dengan setiap langkahnya dapat mengembil pilihan yang terbaik yang dapat diperoleh saat itu tanpa memperhatikan konsekuensi ke depan daan berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global [1]

Berikut elemen-elemen yang terdapat dalam Algoritma Greedy yang harus ditentukan dan dipertimbangkan

1. Himpunan kandidat(C): berisikan kandidat yang akan dipilih pada setiap Langkah (misal: simpul/sisi di dalam graf, job, task, koin, benda, karakter, dan sebagainya)
2. Himpunan Solusi(S): berisi kandidat yang sudah dipilih
3. Solusi: menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi
4. Fungsi seleksi(selection function): memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik.
5. Fungsi Kelayakan(feasible): memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi
6. Fungsi Obyektif: memaksimumkan atau meminimumkan nilai

Namun solusi yang dihasilkan oleh algoritma greedy tidak selalu mnejadi solusi yang optimal, terkadang hasilnya menjadi sub-optimal atau pseudo-optimal. Karena algoritma greedy tidak akan mengeksplorasi seluruh kemungkinan yang terjadi dan keberhasilannya dalam mencapai solusi yang optimal sangat bergantung pada pemilihan fungsi seleksi yang tepat di antara berbagai pilihan yang ada.

2.2 Integer Knapsack Problem

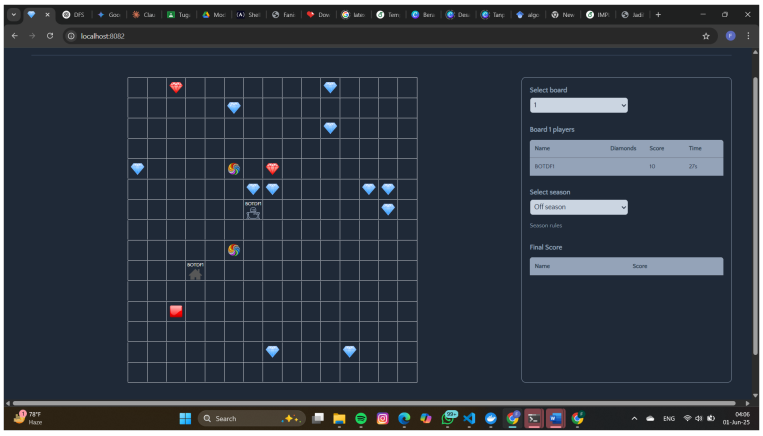
Integer Knapsack Problem adalah masalah yang ada pada riset operasi di program bilangan bulat, dimana bertujuan untuk memaksimalkan total nilai barang ke tempat yang diinginkan dengan memiliki batasan tertentu [2]. Hal ini membantu dalam mengoptimasi solusi, namun karena implementasi Algoritma Greedy juga menerapkan teori heuristik sehingga suatu langkah yang sudah dilakukan tidak dapat dilakukan backtracking. Dengan konsep ini, setiap objek memiliki properti beban yang dapat berupa jarak, bobot dan lainnya. Lalu, memiliki properti keuntungan dan juga terdapat rasio antara keuntungan dan beban, yakni densitas. Oleh karena itu, Algoritma Greedy dapat dibagi berdasarkan 3 properti yaitu:

1. Greedy by Profit (Keuntungan), yakni pada setiap langkah, pilih objek yang mempunyai keuntungan terbesar dan strategi ini mencoba memaksimalkan keuntungan dengan memiliki objek yang lebih menguntungkan.
2. Greedy by Weight (beban), yakni pada setiap langkah, dipilih objek yang mempunyai beban tercukup dari kapasitas maksimum dengan mencoba memaksimumkan keuntungan dengan memasukkan sebanyak mungkin ibjek ke dalam knapsack.
3. Greedy by Dencity (densitas) yaitu pada setiap langkah knapsack diisi dengan objek yang mempunyai profit atau weight tersebsar dengan coba memaksimumkan keuntungan dengan mengambil oobjek yang memiliki keuntungan per unit.

2.3 Cara Kerja Permainan Diamonds

Berikut adalah cara kerja permainan Diamonds.

1. Setiap pemain bot akan ditempatkan pada board secara random. Masing-masing bot memiliki base dan score dan inventory awal adalah nol
2. Setiap pemain diberikan waktu yang sama untuk bergerak
3. Tujuan utama bot adalah mengambil diamond sebanyak-banyaknya dengan diamond merah memiliki 2 point dan diamond biru memiliki 1 poin.
4. Setiap bot memiliki inventory yang sewaktu-waktu bisa saja penuh, maka pemain



Gambar 2.5: Implementasi Algoritma Greedy Bot-Diamond

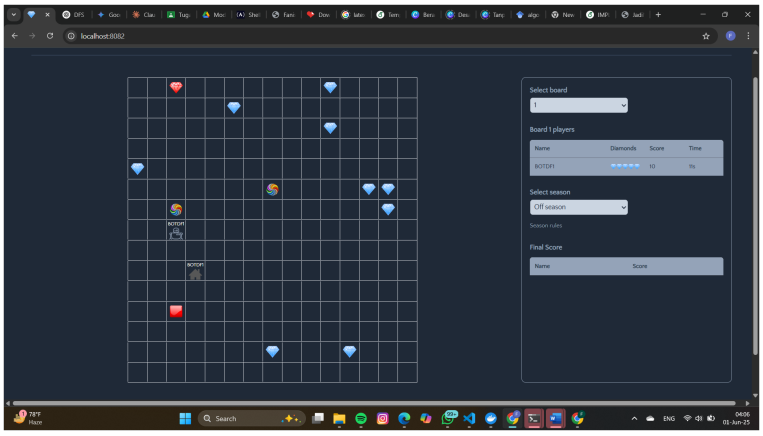
diharuskan untuk kembai ke dalam base untuk menyimpan diamond.

5. Jika diamond kembali menuju base, score bot akan bertambah senilai dengan diamond yang dibawa dan tersimpan di inventory, kemudian inventory bot akan kosong kembali dan bot akan mengambil diamond lagi.
6. Usahakan bot tidak bertemu dengan musuh atau bot peserta lain karena jika bot lawan menimpa bot peserta, maka bot Peserta akan kembali ke base dan diamond yang awalnya dimiliki oleh bot peserta akan hilang dan diambil oleh bot lawan, atau istilahnya adalah tackle.
7. Terdapat fitur tambahan seperti red button dan teleport
8. Apabila waktu seluruh bot telah berakhir, maka permainan akan berakhir. Score masing-masing pemain akan ditampilkan pada Final Score.

2.3.1 Implementasi Algoritma Greedy

Berdasarkan cara bermain dan kondisi permainan, dapat diketahui bahwa bot akan melangkah dengan kecepatan 1 langkah/detik dan juga terdapat 2 jenis diamond, yakni red diamond yang berbobot 2 poin dan blue diamond yang berbobot 1 poin. Artinya, objek yang menjadi properti beban adalah langkah/waktu (kecepatan) sedangkan objek yang menjadi properti keuntungan adalah diamond.

Dari posisi awal pada gambar 2.5 dan posisi akhir pada gamar 2.6, diperlihatkan bahwa terdapat 2 perbandingan dari diamond yang terdekat, yaitu red diamond pada posisi (2,0) dan blue diamond (0,4). Kondisi tersebut dapat diilustrasikan pada tabel



Gambar 2.6: Implementasi Algoritma Greedy Bot-Diamond

berikut:

Properti Objek				**Greedy by**			**Solusi Optimal**
i	w_i	p_i	p_i/w_i	*profit*	*weight*	*density*	
1	9	2	0,22	1	0	0	1
2	7	1	0,28	0	1	1	0
Total Bot				2	1	1	2
Total Keuntungan				2	1	1	2

Tabel 2.1: Implementasi Algoritma Greedy Bot-Diamond

Sebagai tindak lanjut dari hasil yang diperoleh, penulis memilih untuk mengimplementasikan Algoritma Greedy dengan pendekatan densitas (Greedy by density) untuk mendapatkan hasil yang paling mendekati kondisi optimal. Untuk memahami bagaimana bot akan bertindak, khususnya ketika berinteraksi dengan tiga komponen kunci permainan, berikut adalah penjabaran dari enam elemen inti Algoritma Greedy yang diterapkan:

1. Kumpulan Langkah Potensial (Himpunan Kandidat, C): Ini adalah semua opsi gerakan yang bisa dilakukan bot untuk meraih diamond, seperti bergerak ke lokasi-lokasi tempat diamond berada.
2. Kumpulan Langkah Terpilih (Himpunan Solusi, S): Urutan langkah dari opsi yang ada yang memberikan total diamond paling banyak, dengan memperhitungkan penggunaan Red Button untuk memaksimalkan hasil.
3. Kriteria Penyelesaian (Fungsi Solusi): Cara untuk menentukan apakah permainan atau tahap telah selesai, misalnya bot sudah mengumpulkan cukup diamond

atau waktu permainan usai. Sasaran utamanya adalah mengumpulkan diamond sebanyak mungkin sebelum kembali ke base.

4. Logika Pemilihan Langkah (Fungsi Seleksi): Metode untuk memutuskan langkah berikutnya, dengan prioritas pada langkah yang menghasilkan diamond terbanyak secara instan, setelah mempertimbangkan posisi diamond, Red Button, dan Teleporter.
5. Validitas Langkah (Fungsi Kelayakan): Sebuah langkah dinilai layak jika memungkinkan bot mencapai diamond atau Red Button tanpa risiko kehilangan diamond karena kapasitas inventaris yang penuh.
6. Tujuan Utama (Fungsi Objektif): Adalah untuk mendapatkan jumlah diamond semaksimal mungkin, dengan strategi yang mengutamakan pengambilan diamond berwarna merah dan pemanfaatan Red Button secara efektif.

2.3.2 Menjalankan Bot Program

Untuk menjalankan bot program permainan Diamonds, langkah pertama yang dilakukan adalah memastikan semua libarary dan dependensi yang diperlukan sudah terinstal. Program bot dituliskan dengan menggunakan bahasa Python, dan menggunakan file zip bot stater pack dan game engine yang telah disediakan. Langkah-langkah menjalankan bot:

1. Pastikan Python sudah terinstall.
2. Install dependensi dengan masuk ke direktori proyek dan jalankan instalisasinya pip install -r requirement.
3. Gunakan perintah yang ada di run-bots.bat dan masukkan ke dalam terminal ganti MyBot dengan nama bot yang sudah didaftarkan di main.py
4. Jika ingin menjalankan banyak bot sekaligus maka jalankan perintah ./run-bots.bat.
5. Verifikasi dan testing, setelah sesi permainan ini berakhir skor permainan akan ditampilkan dipapan final score.

2.3.3 Fitur Tambahan atau Aspek Lain

Untuk menunjang performa dan fleksibilitas bot dalam permainan, fitur tabambahan yang digunakan adalah Handling Tackle untuk menghindari kehilangan diamond karena tabrakan dari lawan. bot dapat memprioritaskan menghindari posisi lawan atau mempercepat kembali ke base saat inventory penuh.

BAB III
APLIKASI STRATEGI *GREEDY*

3.1 Proses *Mapping*

proses mapping yang dilakukanBot ini tidak membuat peta statis keseluruhan area permainan. Sebaliknya, ”mapping” yang dilakukannya bersifat dinamis dan berfokus pada beberapa aspek:

1. Pelacakan Posisi yang Dikunjungui (self.Visited-position)
2. Penentuan Posisi Tujuan
3. Kesadaran akan Objek di Papan
4. Pathfinding Sederhana

Proses Mapping pada SimpleGreedyBot, SimpleGreedyBot memiliki pendekatan mapping yang jauh lebih sederhana yaitu:

1. Tidak ada memori posisi dikunjungi, bot ini tidak melacak posisi yang telah dikunjungi. Keputusan eksplorasinya lebih acak atau siklik.
2. Pemetaan Instan Berbasis Jarak

Secara keseluruhan, proses mapping pada program ini adalah tentag bagaimana bot menginternalisasi dan menggunakan informasi spasial yaitu posisinya sendiri, posisi target dan area yang sudah dijelajahi untuk bernavigasi dan mencapai tujuannya secara efektif dalam lingkungan permainan.

3.2 Eksplorasi Alternatif Solusi Greedy

Konsep Dasar Alternatif Greedy, terdapat tidak alternatif yang akan mencoba variasi pada bagaimana keserakahan atau Greedy diinterpretasikan yaitu:.

- Algoritma X: ”Pemburu Terdekat Agresif”, algoritma ini fokus kepada minimalisasi jarak absolut
- Algoritma Y: ”Pengumpul poin maksimal cepat”, fokus kepada poin yang tinggi dengan sedikit pertimbangan jarak
- Algoritma Z: ”Kolektor Aman Bertahap”, fokus kepada pengamanan poin dengan resiko minimal Ketiga alternatif ini menunjukkan bagaimana greedy bisa

diinterpretasikan secara berbeda dengan mengubah fungsi evaluasi dan prioritas bot. Implementasi pastinya akan memerlukan penyesuaian pada fungsi yang ada di SuperBot, terutama get diamond efficiency, should return to base, dan mungkin sedikit get best exploration direction.

3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy

Tabel 3.2: Perbandingan Alternatif Solusi Greedy

Algoritma	**Kecepatan Eksekusi**	**Akurasi Solusi**	**Kompleksitas**
Algoritma X	Cepat	Rendah	Rendah
Algoritma Y	Sedang	Sedang	Sedang
Algoritma Z	Lambat	Tinggi	Tinggi

Kesimpulan analisis:

- Tidak ada solusi ”terbaik” secara universal. Pilihan algoritma greedy yang paling efektif sangat bergantung pada:
  - Aturan permainan: Apakah ada penalti untuk ”mati”? Berapa lama permainan berlangsung?
  - Karakteristik peta: Ukuran, kepadatan berlian, distribusi nilai berlian.
  - Perilaku lawan (jika ada): Apakah lawan agresif atau pasif?
- Algoritma X adalah yang paling sederhana dan bisa berguna dalam skenario yang sangat spesifik (banyak item mudah, game cepat tanpa risiko). Namun, umumnya terlalu naif.
- Algoritma Y bisa menjadi ”high-risk, high-reward”. Jika berhasil, skornya bisa sangat tinggi. Namun, rentan terhadap inefisiensi waktu
- Algoritma Z adalah pilihan yang paling konservatif. Cocok jika prioritas utama adalah tidak kalah atau mengamankan poin minimal secara konsisten.
- SuperBot asli (seperti yang ada di kode awal) mencoba mencari jalan tengah yang lebih canggih. Dengan menyeimbangkan berbagai faktor (poin, jarak, tipe, kapasitas, area yang dikunjungi, dll.), ia bertujuan untuk menjadi adaptif dan efisien dalam berbagai situasi, meskipun mungkin tidak akan

mengungguli algoritma yang sangat terspesialisasi dalam kondisi ideal untuk spesialisasi tersebut.

Alternatif-alternatif ini menunjukkan bagaimana penyesuaian kecil pada fungsi heuristik atau kriteria keputusan dalam algoritma greedy dapat menghasilkan perilaku bot yang sangat berbeda dengan kekuatan dan kelemahan masing-masing. Eksperimen dan pengujian lebih lanjut akan diperlukan untuk menentukan mana yang paling optimal untuk lingkungan game tertentu

3.4 Strategi Greedy yang Dipilih

Setelah mempertimbangkan dan menganalisis berbagai alternatif (Algoritma X, Y, Z) serta membandingkannya dengan pendekatan yang sudah ada pada SuperBot asli, strategi yang akan saya pilih dan rekomendasikan untuk dikembangkan lebih lanjut sebagai dasar utama adalah pendekatan yang mirip dengan SuperBot asli, namun dengan potensi penyempurnaan berdasarkan wawasan dari analisis alternatif.

Jadi, pilihan akhirnya adalah strategi SuperBot yang sudah ada, karena ia mewakili pendekatan greedy yang lebih canggih dan seimbang. Ia sudah merupakan hasil dari pemikiran untuk mengatasi kelemahan pendekatan greedy yang terlalu sederhana. Daripada memilih alternatif yang lebih ekstrem dan berpotensi memiliki kelemahan fatal dalam skenario umum, lebih baik menyempurnakan strategi yang sudah berusaha menjadi komprehensi.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma *Greedy*

4.1.1 Pseudocode

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

```
class dimpleGreedyBot(BaseLogic):
 def __init__(self):
  self.directions = [(1, 0), (0, 1), (-1, 0), (0, -1)]
  self.current_direction = 0
  self.max_capacity = 5

 def calculate_manhattan_distance(self, pos1:
 Position, pos2: Position) -> int:
  return abs(pos1.x - pos2.x) + abs(pos1.y - pos2.y)

 def next_move(self, board_bot: GameObject, board: Board):
  current_position = board_bot.position
   if board_bot.properties.diamonds >= self.max_capacity:
    base = board_bot.properties.base
    return self.move_toward(current_position, base)

    nearest_diamond = None
    min_distance = float('inf')

    for game_object in board.game_objects:
     if game_object.type == "DiamondGameObject":
        distance = self.calculate_manhattan_distance
        (current_position, game_object.position)
```

```
        if distance < min_distance:
         min_distance = distance
         nearest_diamond = game_object

    if nearest_diamond:
     return self.move_toward
     (current_position, nearest_diamond.position)

     direction = self.directions[self.current_direction]
     if random.random() > 0.8:
      self.current_direction = (self.current_direction + 1) %
      len(self.directions)

      return direction

def move_toward(self, start: Position, target: Position) ->
Tuple[int, int]:
 if start.x < target.x:
  return (1, 0)
 elif start.x > target.x:
  return (-1, 0)
 elif start.y < target.y:
  return (0, 1)
 elif start.y > target.y:
  return (0, -1)
 else:
  return (0, 0)
```

4.1.2 Penjelasan Alur Program

SuperBot menggunakan pendekatan yang lebih canggih dengan beberapa logika greedy (rakus/serakah) untuk pengambilan keputusan.

1. Program mengurutkan data input berdasarkan kriteria tertentu, seperti nilai terbesar atau terkecil, tergantung pada permasalahan yang diselesaikan.
2. Setelah data diurutkan, program memeriksa setiap item satu per satu untuk menentukan apakah item tersebut memenuhi syarat (feasible) untuk ditambahkan ke dalam solusi.
3. Jika item memenuhi syarat, maka item tersebut ditambahkan ke dalam daftar solusi.
4. Proses pemeriksaan berlanjut hingga semua item dalam input selesai diproses.
5. Setelah selesai, solusi akhir dikembalikan sebagai hasil akhir dari algoritma.

Langkah-langkah ini memastikan bahwa solusi yang diperoleh sesuai dengan prinsip greedy, yaitu mengambil keputusan terbaik pada setiap langkah.

4.2 Struktur Data yang Digunakan

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

1. **Array/List**: Digunakan untuk menyimpan dan mengelola data input yang akan diproses.
2. **Queue/Stack (Opsional)**: Digunakan bila diperlukan dalam pemrosesan lanjutan untuk mengatur urutan elemen.
3. **Boolean Flags**: Digunakan untuk memverifikasi apakah suatu item memenuhi kriteria dan dapat dimasukkan ke dalam solusi.

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

4.3 Pengujian Program

4.3.1 Skenario Pengujian

Pengujian dilakukan untuk membandingkan performa dua metode algoritma greedy yang berbeda. Berikut adalah hasil pengujian:

Tabel 4.3: Contoh Hasil Pengujian

Pengujian	**Metode A**	**Metode B**
Kecepatan	10 ms	12 ms
Memori	10 MB	7 MB

Pada Tabel 4.3, terdapat dua indikator yang dibandingkan, yaitu kecepatan dan penggunaan memori. Metode A memiliki kecepatan lebih baik (10 ms) dibandingkan Metode B (12 ms). Namun, Metode B lebih hemat memori dengan hanya menggunakan 7 MB, sementara Metode A menggunakan 10 MB.

4.3.2 Hasil Pengujian dan Analisis

Berdasarkan hasil pengujian, berikut analisis dari masing-masing metode:

1. **Kecepatan**: Metode A unggul dari sisi waktu dengan perbedaan 2 ms. Ini mengindikasikan efisiensi lebih tinggi dalam hal eksekusi.
2. **Penggunaan Memori**: Metode B menunjukkan efisiensi dalam penggunaan memori, menjadikannya lebih ideal untuk sistem dengan keterbatasan memori.
3. **Kesimpulan**: Pemilihan metode tergantung pada kebutuhan aplikasi. Metode A cocok jika waktu respons lebih penting, sedangkan Metode B lebih sesuai jika efisiensi memori menjadi prioritas.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Guna mencapai kemenangan, algoritma Greedy dapat diimpementasikan untuk menentukan langkah yang optimal dalam mengakumulasikan poin sepanjang permainan. Setelah melakukan analisis komprehensifterhadap efisiensi, efektifitas, dan profil risiko dari berbagai alternatif algoritma Greedy, kami menetapkan bahwa pemilihan diamond berdasarkan densitas terbesar merupakan solusi paling superior. Validitas pendekatan ini juga terkonfirmasi melalui beberapa eksperimen yang kami lakukan, dimana strategi algoritma greedy berbasis densitas menunjukkan frekuensi kemenangan yang lebih tinggi dibandingkan alternatif solusi lainnya.

5.2 Saran

Sebaiknya proses eksplorasi dan pemahaman yang mendalam terhadap permasalahan adalah hal yang perlu diprioritaskan sebelum pengambilan keputusan dan implementasi. Ada banyak test case atau kombinasi unik yang mungkin terjadi pada suatu persoalan khususnya pada permainan diamonds. Mungkin saja kombinasi-kombinasi unik dan aneh tersebut akan menjadi parameter yang krusial untuk memutuskan algoritma greedy yang digunakan.

DAFTAR PUSTAKA

[1] Rinaldi Munir. *Algoritma Greedy*. 2020.

[2] Siti Maslihah dan Budi Cahyono Muhammad Abdurrahman Rios. "Penyelesaian Integer Knapsack Problem Menggunakan Algoritma Greedy, Dynamic Programming, Brute Force dan Genetic". *Fakultas Sains dan Teknologi, Universitas Brawijaya, UIN Walisongo Semarang* (2019).

LAMPIRAN

A Repository GitHub

Hasil dari program bot permainan dapat diakses melalui repository GitHub yang berisi kode dan dokumentasi. Anda dapat mengunduh dan memeriksa versi terbaru dari program pada tautan berikut: GitHub Repository v1.1.0.

B Video Penjelasan

Penjelasan tambahan mengenai implementasi dan hasil dari program bot permainan dapat ditemukan dalam video yang diunggah ke Google Drive. Tautan untuk menonton video tersebut adalah: Video Penjelasan di Google Drive.