IMPLEMENTASI ALGORITMA GREEDY DALAM PEMECAHAN BOT PERMAINAN DIAMOND

Tugas Besar

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

Oleh: Pilar 23

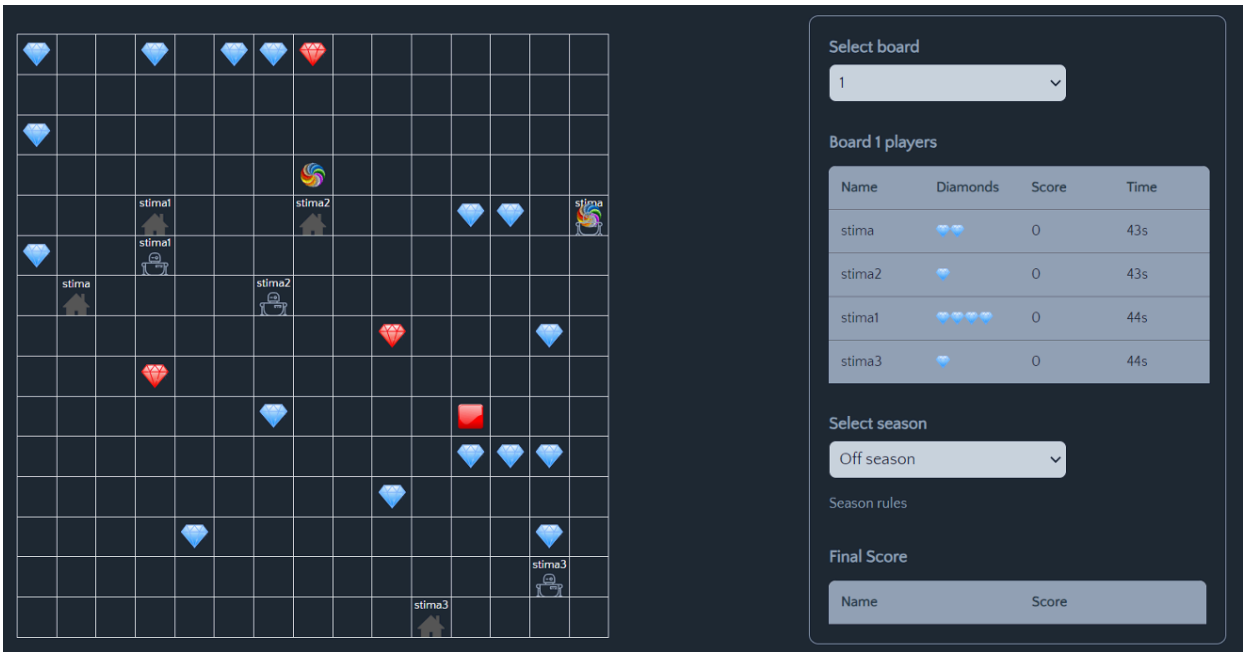
Muhammad Nufal Fikri akmal	123140132
Ausyaf Naufal Dinata	123140201
Aji Nur Astondinata	123140189

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

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SUMATERA
2025

DAFTAR ISI

BAB I
DESKRIPSI TUGAS



Diamonds merupakan tantangan pemrograman di mana setiap pemain membuat bot yang akan bersaing dengan bot milik pemain lainnya. Setiap pemain memiliki satu bot dengan tujuan utama mengumpulkan diamond sebanyak mungkin. Tentunya, proses mengumpulkan diamond ini tidak semudah kelihatannya berbagai rintangan akan membuat permainan terasa lebih seru dan menantang. Untuk bisa menang, pemain perlu menerapkan strategi yang tepat pada bot mereka. Penjelasan lebih lanjut mengenai aturan permainan dapat ditemukan pada bagian berikutnya.

Pada tugas pertama mata kuliah Strategi Algoritma ini, mahasiswa diminta untuk membuat sebuah bot yang nantinya akan diadu dengan bot dari kelompok lain. Bot tersebut harus dirancang menggunakan strategi greedy sebagai dasar pengambilan keputusannya.

Tugas ini terdiri dari dua bagian utama:

1. Game Engine, yang secara umum mencakup:
    - Kode backend permainan, yang berisi logic permainan secara keseluruhan serta API yang disediakan untuk berkomunikasi dengan frontend dan program bot
    - Kode frontend permainan, yang berfungsi untuk memvisualisasikan permainan.
2. Bot Starter Pack, yang secara umum mencakup:
    - Program untuk mengakses API yang tersedia dari backend.
    - Program logika bot, yaitu bagian yang akan kalian kembangkan menggunakan algoritma *greedy*.
    - Program utama (main) serta utilitas tambahan lainnya yang mendukung jalannya bot.

Adapun 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. 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.

5. 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.

Untuk mengetahui alur dari game ini, berikut ini adalah cara kerja permainan Diamonds.

1. Inisialisasi Pertandingan: Ketika game engine dijalankan, pertandingan akan dimulai dengan meng-host permainan pada hostname yang telah ditentukan.

2. Koneksi antara Engine dan Bot: Setelah engine berhasil dijalankan, engine akan menunggu koneksi dari bot-bot yang dijalankan oleh para pemain. Engine akan melakukan proses logging dan menyiapkan semua informasi yang diperlukan untuk pertandingan.

3. Jumlah bot yang akan berpartisipasi dalam satu pertandingan diatur oleh atribut BotCount yang ada dalam file "appsettings.json" di dalam folder "engine-publish".

Pengaturan ini memungkinkan penyesuaian jumlah pemain dalam pertandingan sesuai dengan keinginan.

4. Setiap bot juga memiliki sebuah inventory, dimana inventory berfungsi sebagai tempat penyimpanan sementara diamond yang telah diambil. Inventory ini sewaktu-waktu bisa penuh, maka dari itu bot harus segera kembali ke home base.

5. Apabila bot menuju ke posisi home base, score bot akan bertambah senilai diamond yang tersimpan pada inventory dan inventory bot akan menjadi kosong kembali.

6. Usahakan agar bot anda tidak bertemu dengan bot lawan. Jika bot A menimpa posisi bot B, bot B akan dikirim ke home base dan semua diamond pada inventory bot B akan hilang, diambil masuk ke inventory bot A (istilahnya tackle).

7. Selain itu, terdapat beberapa fitur tambahan seperti teleporter dan red button yang dapat digunakan apabila anda menuju posisi objek tersebut.

8. Apabila waktu seluruh bot telah berakhir, maka permainan berakhir. Score masing-masing pemain akan ditampilkan pada tabel Final Score di sisi kanan layar.

BAB II
LANDASAN TEORI

2.1 Dasar Teori Algoritma Greedy

Algoritma Greedy adalah salah satu pendekatan dalam pemrograman yang digunakan untuk menyelesaikan masalah optimasi dengan cara memilih solusi yang terlihat paling menguntungkan pada setiap langkahnya. Pendekatan ini menekankan pada pengambilan keputusan secara langsung, dengan harapan bahwa setiap pilihan lokal terbaik akan mengarah pada solusi akhir yang optimal.

Dalam strategi greedy, keputusan diambil berdasarkan keuntungan terbesar saat ini, tanpa mempertimbangkan dampak atau konsekuensinya di kemudian hari. Pendekatannya dapat dianalogikan seperti mengambil uang tunai sebanyak mungkin dari

2.2 Cara Kerja Program

Dalam game ini, program dibagi menjadi dua komponen utama: **Game Engine** dan **Bot**. Kedua komponen ini berinteraksi melalui API yang sudah disediakan dalam sistem. Di sisi Bot, program akan menggunakan API tersebut dengan cara mengirimkan permintaan **POST** untuk

mendaftarkan diri menggunakan email dan password. Setelah proses pendaftaran berhasil, Bot bisa **bergabung ke dalam permainan** dengan melakukan POST ke endpoint khusus untuk join, lalu mendapatkan data kondisi board.

Dengan data tersebut, Bot melakukan perhitungan untuk menentukan langkah berdasarkan strategi atau logika yang dimilikinya. Bot lalu mengirimkan langkah tersebut ke backend melalui endpoint **move**. Sebagai balasannya, backend akan mengirimkan kondisi board terbaru setelah langkah dijalankan. Proses ini berulang Bot menganalisis kondisi terkini,

menghitung langkah terbaik, lalu mengirimkannya kembali hingga permainan dinyatakan selesai.

1. **Cara Implementasi Program**

   Berikut ini adalah langkah-langkah untuk menjalankan program:

   1. Pastikan Node.js, Docker, dan Yarn telah terpasang. Pemasangan dapat dilakukan dengan command dibawah.

      ```
      npm install --global yarn
      ```

   2. Jalankan game engine dengan mengunduh folder game engine pada tautan ini https://github.com/haziqam/tubes1-IF2211-game-engine/releases/tag/v1.1.0

      a. Setelah berhasil mengunduh, lakukan ekstraksu file.zip dan masuk ke root dari folder hasil ekstraksi tadi, lalu buka terminal

      b. Jalankan command berikut ini di terminal untuk masuk ke root directory dari game engine

      ```
      cd tubes1-IF2110-game-engine-1.1.0
      ```

      c. Install dependencies dengan menggunakan yarn

      ```
      yarn
      ```

d. Setup default environment variable dengan menjalankan script berikut Untuk Windows

```
./scripts/copy-env.bat
```

e. Setup local database dengan membuka aplikasi docker desktop, lalu masukkan command berikut di terminal

```
docker compose up -d database
```

Kemudian jalankan script berikut untuk windows

```
./scripts/setup-db-prisma.bat
```

f. Masukkan comand berikut untuk melakukan proses build pada game engine

```
npm run build
```

g. Masukkan command berikut untuk memulai game engine

```
npm run start
```

h. Jika berhasil, maka tampilan terminal akan seperti ini

3. Jalankan bot starter pack dengan mengunduh folder bot pada tautan ini

https://github.com/haziqam/tubes1-IF2211-bot-starter-pack/releases/tag/v1.0.1

1. Setelah berhasil diunduh, lakukan ekstraksi file.zip tersebut dan masuk ke root dari folder hasil ekstraksi tadi, lalu buka terminal
2. Jalankan command berikut pada terminal untuk masuk ke root directory dari bot starter pack

```
cd tubes1-IF2211-bot-starter-pack-1.0.1
```

3. Install dependencies dengan pip

```
pip install -r requirements.txt
```

4. Gunakan command berikut untuk menjalankan satu bot

```
python main.py --logic MyBot --email=your_email@example.com --name=your_name
```

```
--password=your_password --team etimo
```

User juga dapat menjalankan lebih dari satu bot dengan menjalankan script berikut Untuk Windows

```
./run-bots.bat
```

Untuk Linux/macOS

```
./run-bots.sh
```

User dapat mengatur script yang ada di run-bots.bat atau run-bots.s dari **logic, email, nama, dan password yang digunakan**.

5. Untuk merancang bot, buatlah file baru pada direktori/game/logic (misalnya MyBot.py)
6. Buatlah sebuah kelas yang meng-inherit kelas BaseLogic, lalu implementasikan constructor beserta method next_move pada kelas tersebut

7. Import kelas yang telah dibuat tadi dalam *main.py* dan daftarkan pada dictionary *CONTROLLERS*
8. Jalankan program seperti pada step d pada bagian 3 dengan menyesuaikan argumen logic pada argumen logic pada command/script dengan nama yang terdaftar pada *CONTROLLERS.* User juga masih bisa menjalankan satu bot saja atau beberapa bot dengan run-bots.bat atau run-bots.sh.

BAB III
APLIKASI STRATEGI *GREEDY*

3.1 Proses *Mapping*

Dalam algoritma greedy, proses mapping memiliki peran krusial karena berfungsi untuk menyederhanakan data agar lebih mudah diproses oleh program. Mapping memungkinkan transformasi informasi yang rumit seperti nama objek atau koordinat menjadi bentuk yang lebih terstruktur, seperti indeks, bobot, atau nilai numerik lainnya. Tujuan utamanya adalah untuk mempermudah proses pengambilan keputusan lokal terbaik pada setiap langkah algoritma. Karena pendekatan greedy mengandalkan pemilihan solusi optimal secara bertahap, mapping menjadi sangat penting agar setiap alternatif dapat dievaluasi dan dibandingkan dengan cepat serta akurat. Selain itu, mapping juga berguna dalam melacak elemen-elemen yang sudah dipilih, sehingga dapat mencegah duplikasi atau kesalahan. Representasi data yang efisien melalui mapping turut berkontribusi terhadap peningkatan performa algoritma, baik dari segi waktu pemrosesan maupun pemakaian memori. Dengan demikian, mapping menjadi salah satu komponen penting yang mendukung keberhasilan algoritma greedy dalam menyelesaikan berbagai masalah optimasi.

3.1.1 Himpunan Kandidat

Dalam permasalahan ini, terdapat sejumlah elemen yang dapat dimasukkan ke dalam himpunan kandidat, di antaranya adalah:

Nama objek	Penjelasan
Base	Objek ini berperan sebagai markas utama bagi bot. Saat bot kembali ke markas sambil membawa sejumlah N diamond, maka bot akan mendapatkan N poin sesuai dengan jumlah diamond yang dibawa. Selain itu, markas juga berfungsi sebagai titik respawn bagi bot apabila bot tersebut terkena tackle oleh bot lawan.
Red button	Saat bot melewati objek ini, seluruh diamond di papan permainan akan diacak ulang dan ditempatkan kembali secara acak. Selain itu, posisi tombol merah (red button) juga akan berubah secara acak setelah objek ini dilewati.
Diamond	Ketika bot melewati objek ini, ia akan menambah N diamond ke dalam inventarisnya. Jumlah N bergantung pada jenis diamond yang diambil red diamond memberikan 2 diamond, sementara blue diamond hanya menambahkan 1 diamond.
Teleporter	Dalam setiap permainan, hanya dua teleporter yang berfungsi secara bersamaan. Saat bot menyentuh salah satu dari teleporter tersebut (teleporter A), bot akan langsung berpindah dan muncul kembali di lokasi teleporter lainnya (teleporter B).
Inventory	Inventory adalah tempat penyimpanan sementara bagi diamond yang dikumpulkan bot selama permainan

	berlangsung. Bot hanya bisa membawa maksimal 5 diamond dalam inventory-nya. Jika bot mencoba mengambil diamond saat slot sudah penuh, maka diamond tersebut tidak akan bisa diambil. Untuk bisa mengambil diamond lagi, bot harus kembali ke base terlebih dahulu untuk mengosongkan inventory-nya.
Bot Musuh	Di setiap permainan, terdapat bot musuh yang ikut berpartisipasi dan memiliki kemampuan yang setara dengan bot kita. Mereka dapat menggunakan base, memanfaatkan teleporter, mengumpulkan diamond, serta memiliki sistem inventory yang serupa. Bot musuh bisa melakukan tackle dengan menyentuh posisi bot kita. Jika hal ini terjadi, semua diamond dalam inventory kita akan berpindah ke bot musuh. Sebaliknya, kita juga dapat men-tackle bot musuh untuk merebut seluruh diamond yang mereka bawa.

3.1.2 Himpunan Solusi

Dengan pemilihan objek-objek tersebut, dapat disimpulkan bahwa himpunan solusi dari algoritma ini merupakan jalur atau lintasan yang dilalui oleh bot.

3.1.3 Fungsi Solusi

Algoritma ini mengevaluasi jarak terdekat menuju Base, RedButton, dan Diamond, lalu menentukan pilihan berdasarkan kriteria yang ditetapkan dalam fungsi seleksi.

3.1.4 Fungsi Seleksi

Pemilihan tujuan dalam permainan dapat dirangkum sebagai berikut:

Nama objek	Syarat
Base	Bot akan mengutamakan kembali ke base jika memenuhi salah satu kondisi berikut: 1. Inventory sudah penuh, yaitu ketika bot membawa 5 diamond. 2. Bot membawa lebih dari 2 diamond dan base berada dalam jarak kurang dari 2 satuan.

	3. Bot sedang membawa diamond dan waktu yang tersisa kurang dari 10 detik.
Red button	Bot akan memilih red button sebagai target apabila jumlah diamond yang tersisa di papan kurang dari atau sama dengan 6, dan jarak menuju red button lebih dekat dibandingkan dengan jarak ke diamond terdekat.
Diamond	Meski masih ada beberapa diamond di papan, bot akan memprioritaskan diamond yang paling dekat sebagai tujuan.
Teleporter	Bot akan menentukan teleporter sebagai tujuan apabila: 1. Jumlah jarak dari bot ke teleporter A ditambah jarak dari teleporter B ke diamond lebih kecil dibandingkan jarak langsung menuju diamond. 2. Jumlah jarak dari bot ke teleporter A ditambah jarak dari teleporter B ke base lebih singkat dibandingkan jarak langsung menuju base.
Bot musuh	Bot akan membidik bot musuh sebagai target jika: 1. Bot musuh membawa lebih dari 2 diamond dan jaraknya kurang dari 3 satuan dari bot kita. 2. Bot musuh berada tepat 1 satuan dari posisi bot kita.

3.1.5 Fungsi Kelayakan

Setiap kali akan bergerak, bot akan memeriksa apakah jumlah diamond yang dibawa sudah mencapai 4 poin. Jika sudah, bot akan segera bergerak kembali ke base.

3.1.6 Fungsi Objektif

Jumlah diamond yang didapat maksimum

3.2 Eksplorasi Alternatif Solusi Greedy

Beberapa Alternatif Solusi Greedy yang kami eksplorasi untuk persoalan ini adalah sebagai berikut:

3.2.1 Greedy By Highest Value

Greedy by Highest Value adalah algoritma Greedy yang menitikberatkan pada pengambilan diamond dengan nilai poin tertinggi di papan permainan. Karena hanya terdapat dua jenis diamond, yaitu Blue Diamond bernilai 1 poin dan Red Diamond bernilai 2 poin, bot akan mengutamakan pengambilan Red Diamond terdekat selama masih ada di papan. Jika semua Red Diamond sudah habis diambil, bot akan beralih untuk mengincar Blue Diamond terdekat, dan kembali mengejar Red Diamond begitu diamond jenis ini muncul kembali.

3.2.2 Greedy By Highest Density

Greedy by Highest Density adalah strategi greedy yang memprioritaskan diamond dengan rasio nilai poin terhadap jarak terbesar. Bot akan terlebih dahulu menghitung nilai poin dibagi jarak untuk setiap diamond di papan, kemudian memilih diamond yang memiliki rasio poin per jarak tertinggi.

3.2.3 Greedy By Tackling

Greedy by Point per Tackling merupakan algoritma Greedy yang menargetkan bot musuh terdekat yang membawa diamond. Bot akan bergerak ke arah bot musuh tersebut dengan tujuan melakukan tackling untuk merebut diamond yang dibawanya.

3.2.4 Greedy By Closet Base

Greedy by Closest Base adalah algoritma Greedy yang mengutamakan diamond dengan lokasi terdekat ke base. Strategi ini dirancang agar bot tidak terlalu jauh dari base, sehingga dapat segera kembali untuk menyimpan diamond sebagai poin tanpa menghabiskan waktu secara sia-sia.

3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy

Alternatif Solusi (Greedy by)	Kelebihan	Kekurangan
Highest Density	1. Memprioritaskan rasio poin terhadap biaya pergerakan 2. Efektif digunakan pada awal permainan 3. Sesuai untuk area dengan jumlah diamond yang banyak	1. Tidak mempertimbangkan nilai diamond 2. Jika diamond langka, bot bisa menjauh dari base tanpa mendapatkan hasil yang optimal 3. Tidak mampu menentukan waktu yang tepat untuk kembali ke base
Highest Value	1. Selalu memilih diamond dengan poin tertinggi 2. Tepat digunakan di area yang memiliki sedikit diamond namun bernilai tinggi	1. Sering kembali ke base dengan jarak tempuh yang terlalu jauh 2. Kurang adaptif, karena jika diamond bernilai tinggi diambil oleh musuh, dapat menyebabkan kebingungan
Closest Base	1. Tepat untuk peta yang memiliki banyak teleporter atau rintangan sehingga waktu kembali ke base menjadi lama 2. Selalu mengutamakan jarak terdekat ke base	1. Skor per perjalanan yang rendah membuatnya sering kalah poin dibanding strategi lain 2. Sering kembali ke base tanpa memperoleh hasil yang optimal
Tackling	1. Bersifat adaptif dengan mengambil diamond yang ada di	1. Tidak selalu berhasil mendapatkan diamond bernilai

	sepanjang jalur pulang 2. Sangat cocok digunakan pada peta dengan bentuk linear	tinggi 2. Rentan terhalang oleh musuh

3.4 Strategi Greedy yang Dipilih

Setelah menilai berbagai strategi greedy yang telah dibahas pada bagian 3.1, kelompok kami memutuskan untuk mengadopsi strategi Greedy by Highest Density sebagai metode utama dalam pengembangan bot. Keputusan ini didasarkan pada durasi pertandingan Diamonds yang hanya berlangsung selama 60 detik, dengan asumsi satu aksi dilakukan setiap detik. Dalam kondisi tersebut, kecepatan dan ketepatan pengumpulan diamond menjadi sangat krusial. Strategi Greedy by Highest Density memungkinkan bot untuk memprioritaskan diamond dengan rasio nilai terhadap jarak tertinggi, sehingga pengumpulan poin menjadi lebih optimal. Pendekatan ini juga unggul dalam memaksimalkan total poin yang dapat dikumpulkan oleh bot dengan cara memberikan prioritas pada diamond yang menawarkan rasio poin per jarak terbaik. Keunggulan ini memberikan bot kelebihan kompetitif selama pertandingan, karena bot dapat meraih skor tertinggi dalam waktu yang terbatas. Berdasarkan pertimbangan tersebut, strategi Greedy by Highest Density terbukti sebagai solusi yang tepat dan efisien untuk bot kami dalam game Diamonds berdurasi 60 detik. Strategi ini meningkatkan peluang bot meraih kemenangan melalui pengumpulan poin yang strategis dan terencana.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma Greedy

1. Pseudocode

```
# Tiki Taka Toe (122140623)
# Nas Lalalala (122140832)
# Kelompok: Anak Pak Eko
#
# DISCLAIMER:
# Ini adalah contoh dummy sederhana untuk algoritma Greedy Huffman
Coding.
# BUKAN strategi untuk pembuatan bot Diamonds.
# Digunakan hanya untuk contoh penulisan laporan

import heapq  # Mengimpor pustaka heapq untuk membuat priority
queue

# Membuat class Node untuk pohon Huffman
class Node:
    def __init__(self, char, freq):
        self.char = char  # Karakter yang disimpan di node
        self.freq = freq  # Frekuensi kemunculan karakter
        self.left = None  # Anak kiri
        self.right = None  # Anak kanan

    # Menentukan urutan node berdasarkan frekuensi (untuk heapq)
    def __lt__(self, other):
        return self.freq < other.freq

# Fungsi untuk membuat pohon Huffman
def build_huffman_tree(char_freq):
    # Membuat priority queue (min heap)
    heap = []
    for char, freq in char_freq.items():
```

```
        heapq.heappush(heap, Node(char, freq))  # Masukkan node ke
heap berdasarkan frekuensinya

    # Membuat pohon Huffman
    while len(heap) > 1:
        # Ambil dua node dengan frekuensi terkecil
        left = heapq.heappop(heap)
        right = heapq.heappop(heap)

        # Gabungkan menjadi node baru
        merged = Node(None, left.freq + right.freq)
        merged.left = left
        merged.right = right

        # Masukkan node gabungan kembali ke heap
        heapq.heappush(heap, merged)

    # Kembalikan root pohon
    return heap[0]

# Fungsi untuk membuat kode Huffman
def generate_codes(node, current_code="", codes={}):
    if node is None:
        return

    # Jika node adalah daun (leaf)
    if node.char is not None:
        codes[node.char] = current_code  # Simpan kode untuk
karakter tersebut
        return

    # Rekursi ke anak kiri (tambah '0') dan anak kanan (tambah '1')
    generate_codes(node.left, current_code + "0", codes)
    generate_codes(node.right, current_code + "1", codes)

    return codes
```

```
# Contoh frekuensi karakter
char_freq = {
    'A': 5,
    'B': 9,
    'C': 12,
    'D': 13,
    'E': 16,
    'F': 45
}

# Bangun pohon Huffman
root = build_huffman_tree(char_freq)

# Generate kode Huffman
huffman_codes = generate_codes(root)

# Cetak hasil
print("Kode Huffman untuk masing-masing karakter:")
for char, code in huffman_codes.items():
    print(f"Karakter: {char} | Kode: {code}")
```

2. **Penjelasan Alur Program**

4.2 Struktur Data yang Digunakan

Bahasa pemrograman yang digunakan dalam implementasi bot ini adalah Python. Struktur data pada program didefinisikan dalam bentuk class yang terdapat pada file *models.py*.

Seluruh class yang digunakan dalam proses pengembangan dan implementasi bot ditempatkan di dalam folder *game*.

Class yang ada di dalam *models.py* adalah sebagai berikut:

○ Points yang merupakan properti milik diamond yang berisi informasi tentang jumlah poin dari setiap jenis diamond, diamond biru memiliki poin 1 dan diamond merah memiliki poin 2,

○ Pair_id merupakan properti milik teleporter yang isi nya id untuk teleporter agar dapat dilakukan pairing,

○ Diamonds merupakan informasi dari jumlah diamond yang dibawa oleh bot,
○ Score adalah informasi dari jumlah poin dari diamond yang berhasil dibawa ke bot ke base,

○ Name adalah informasi mengenai nama dari objek tersebut,

○ Inventory_size merupakan properti milik bot yang berisi jumlah poin maksimal yang dapat dibawa oleh bot,

○ Can_tackle berisikan informasi apakah sebuah objek dapat ditabrak atau tidak,

○ Milisecond_left berisikan informasi mengenai sisa waktu yang dimiliki oleh bot dalam permainan,
○ Time_joined berisikan informasi mengenai waktu masuk dari objek ke dalam permainan,

○ Base berisi informasi mengenai koordinat dari base sebuah bot.

● Class GameObject merupakan kelas yang menyimpan informasi mengenai objek dalam permainan, seperti id, position, type, dan properties. Atribut id bertipe integer, position merupakan instance dari class Position, type berupa string, dan properties merupakan instance dari class Properties.

● Class Board berfungsi untuk menyimpan data yang berkaitan dengan papan permainan, termasuk id, lebar (width), tinggi (height), daftar fitur (features), minimum_delay_between_moves, serta game_objects. Atribut minimum_delay_between_moves menentukan waktu jeda minimum antar pergerakan bot, sementara game_objects berisi informasi mengenai seluruh objek yang ada di dalam papan. Kelas ini juga menyediakan fungsi untuk memeriksa apakah suatu pergerakan bot dapat dianggap valid.

Program juga memerlukan file *util.py* agar dapat berjalan. Fungsi-fungsi yang ada di dalam file ini adalah sebagai berikut:

- Fungsi *clamp* menerima tiga parameter: *n*, *smallest*, dan *largest*, lalu mengembalikan nilai yang dibatasi di antara *smallest* dan *largest*. Nilai yang dikembalikan adalah maksimum dari *smallest* dan minimum dari *n* dan *largest*.

- Fungsi *get_direction* memerlukan koordinat *x* dan *y* milik bot serta koordinat *x* dan *y* dari objek lain di papan. Fungsi ini mengembalikan *delta_x* dan *delta_y* yang menunjukkan arah pergerakan bot relatif terhadap posisi objek tersebut.

- Fungsi *position_equals* menerima dua parameter, yaitu *position a* dan *position b*, untuk memeriksa apakah kedua posisi tersebut memiliki nilai yang sama atau tidak.

Program ini juga memanfaatkan beberapa file logic yang berfungsi sebagai versi alternatif dari bot dengan berbagai strategi greedy. Masing-masing file tersebut berisi fungsi *next_move* yang menentukan langkah bot berdasarkan strategi greedy yang diterapkan. File-file logic ini juga mengimpor fungsi *get_direction* dari *util.py* untuk membantu dalam menentukan arah pergerakan bot. Selain itu, terdapat file *main.py* yang berperan sebagai penghubung utama antar seluruh komponen dalam bot starter pack.

4.3 Pengujian Program

Pada bagian ini akan dilakukan pengujian terhadap beberapa fitur utama dari bot yang dipilih, yaitu bot dengan strategi greedy by highest density

4.3.1 Skenario Pengujian

Bagian ini berfokus pada pengujian perilaku bot utama yang menerapkan strategi *greedy by highest density*, dengan tujuan memastikan bahwa seluruh komponen logika berfungsi sesuai harapan. Pengujian dilakukan dalam berbagai skenario permainan untuk mengevaluasi ketepatan keputusan yang diambil bot dalam situasi yang beragam. Berikut adalah rincian dari skenario-skenario pengujian yang telah dilakukan.

No	Kondisi	Deskripsi	Input (Singkat)	Perilaku Diharapkan (Output)
1	Inventory penuh	Bot sudah membawa diamond maksimal	diamonds = 5/5	Menuju base
2	Waktu hampir habis	Waktu < jarak ke base + 1	time_left = 3, dist_to_base = 4	Menuju base
3	Di base, inventory penuh	Bot sudah sampai di base & penuh	pos = base, diamonds = max	Stay (0, 0)
4	Ada diamond dengan rasio value/distance tertinggi	Beberapa diamond, pilih terbaik	3 diamond, value:distance = 3:1, 5:3, 2:2	Bergerak ke diamond dengan rasio terbaik
5	Semua diamond melebihi kapasitas	Setiap diamond akan melebihi inventory	diamonds = 4/5, all new diamonds worth 2	Abaikan, cari tombol merah atau balik
6	Tidak ada diamond, ada red button	Tidak ada diamond tersedia	diamonds = [], redButton exists	Gerak ke tombol merah
7	Tidak ada diamond dan red button	Kosong semua	No diamond, no redButton	Pulang ke base
8	Sudah di base & tidak ada opsi lain	Di base, inventory kosong, tidak ada target	pos = base, no diamond/redButton	Stay (0, 0)

9	Berdiri di posisi diamond	pos = diamond position	Jarak dist = 0 → atur dist = 1	Tetap memilih target atau pickup
10	Bot di tengah game normal	Waktu cukup, ada diamond, belum penuh	time = 50s, inventory = 2/5	Pilih diamond terbaik berdasarkan value/distance

4.3.2 Hasil Pengujian dan Analisis

Fase	Penjelasan
Kondisi Awal Kondisi Akhir	Ilustrasi di samping memperlihatkan posisi bot pada saat sebelum dan sesudah berpindah menggunakan teleporter menuju area diamond. Teleportasi dipilih karena jalur menuju lokasi diamond lebih efisien ditempuh melalui teleporter dibandingkan dengan berjalan langsung. Berdasarkan ilustrasi tersebut, dapat disimpulkan bahwa fitur teleportasi telah berjalan dengan baik dan sesuai fungsinya.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dalam tugas besar Strategi Algoritma ini, kami telah melakukan sejumlah percobaan dengan berbagai jenis bot pada permainan Diamonds yang menerapkan strategi greedy. Setelah melakukan serangkaian uji coba dan perbandingan antar strategi, kami akhirnya menetapkan bot

dengan pendekatan strategi kepadatan tertinggi (*highest density*) sebagai pilihan akhir. Keputusan ini didasarkan pada hasil pertandingan antar bot yang menggunakan berbagai pendekatan.

Dari hasil pertandingan, terlihat bahwa bot dengan strategi kepadatan tertinggi mampu mengumpulkan diamond dengan rata-rata jumlah paling tinggi dibandingkan bot lain yang menerapkan strategi *Closest Base*. Berdasarkan pengamatan selama pertandingan, bot ini menunjukkan kinerja yang cukup efektif karena mempertimbangkan area dengan konsentrasi diamond tertinggi, memanfaatkan teleporter untuk mempercepat akses ke diamond, serta menjaga efisiensi dalam perjalanan kembali ke base.

Pengujian dilakukan pada papan permainan berukuran 15x15 dengan durasi permainan selama satu menit. Berdasarkan hasil dan observasi, dapat disimpulkan bahwa bot dengan strategi kepadatan tertinggi menunjukkan performa yang cukup optimal, baik dari segi fitur yang digunakan, rata-rata skor yang dicapai, maupun konfigurasi papan permainan yang diterapkan.

5.2 Saran

Beberapa saran untuk pengembangan bot dalam tugas besar ini antara lain:

1. Penulisan kode sebaiknya dilakukan secara lebih terorganisir agar proses debugging menjadi lebih mudah dan efisien.

2. Sebaiknya hindari menyelesaikan tugas besar secara tergesa-gesa menjelang deadline, agar pengembangan bot dapat dilakukan secara lebih optimal dan menghasilkan solusi yang lebih matang.

LAMPIRAN

A. Repository Github (link)

Repository Gith ub

DAFTAR PUSTAKA

[1] Diamond Game Engine. Diakses 30 Mei 2025
https://github.com/haziqam/tubes1-IF2211-game-engine/releases/tag/v1.1.0
[2] Bot Starter Pack. Diakses 30 Mei 2025
https://github.com/haziqam/tubes1-IF2211-game-engine/releases/tag/v1.1.0
[3] R. Munir, “Algoritma Greedy (Bagian 1),” *informatika.stei.itb.ac.id*, [Online]. Tersedia pada:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag

1. pdf. Diakses 31 Mei 2025

[4] R. Munir, “Algoritma Greedy (Bagian 2),” *informatika.stei.itb.ac.id*, [Online]. Tersedia pada:.
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag

2. pdf. Diakses 01-Juni-2025

[5 R. Munir, “Algoritma Greedy (Bagian 3),” *informatika.stei.itb.ac.id*, [Online]. Tersedia pada:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag

3. pdf. Diakses: 01 Juni 2025.