

IMPLEMENTASI ALGORITMA X DALAM PEMECAHAN BOT PERMAINAN DIAMOND

Tugas Besar

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



Oleh: Kelompok 7 (Mangu)

Afifa Aulia	123140073
Bening Apni Prameswari	123140089
Raisya Syifa Saleh	123140169

Dosen Pengampu: Winda Yulita, M.Cs.

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SUMATERA
2025**

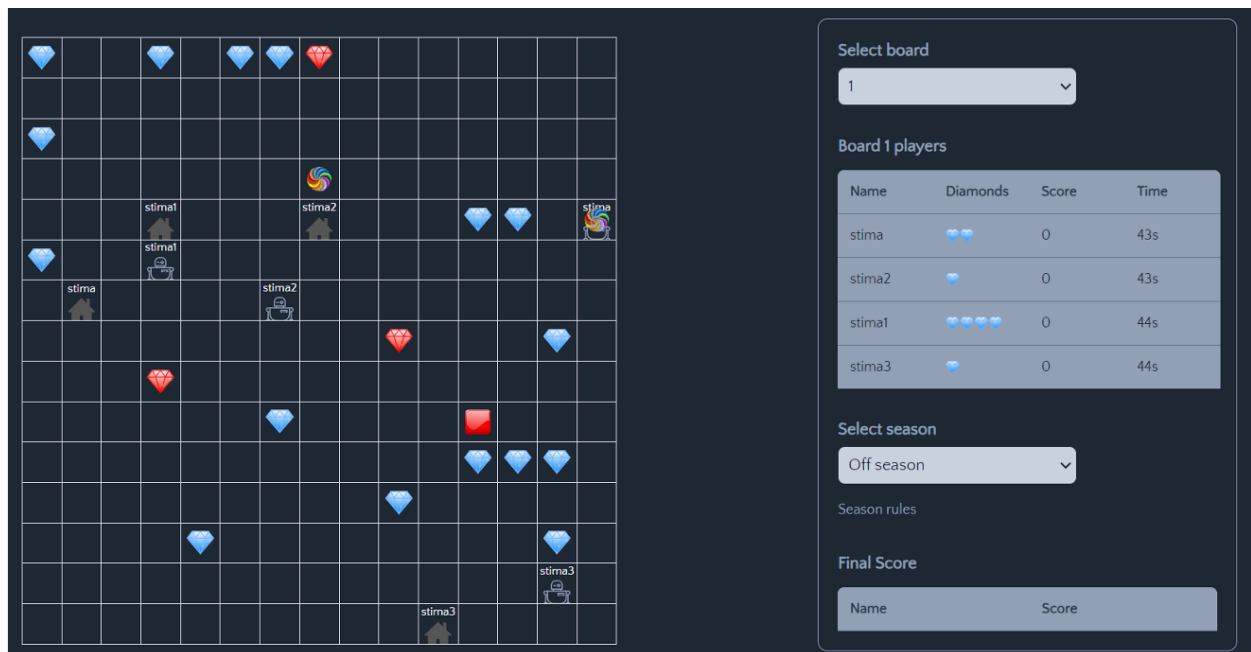
DAFTAR ISI

BAB I DESKRIPSI TUGAS.....	2
BAB II LANDASAN TEORI.....	5
2.1 Dasar Teori Algoritma Greedy.....	5
2.2 Cara Kerja Program.....	5
BAB III APLIKASI STRATEGI GREEDY.....	7
3.1 Proses Mapping Persoalan Menjadi Elemen Algoritma Greedy.....	7
3.1.1 Himpunan Kandidat.....	7
3.1.2 Himpunan Solusi.....	8
3.1.3 Fungsi Solusi.....	8
3.1.4 Fungsi Seleksi.....	8
3.1.5 Fungsi Kelayakan.....	9
3.1.6 Fungsi Objektif.....	9
3.2 Eksplorasi Alternatif Solusi Greedy.....	9
3.2.2 Greedy by Point.....	9
3.2.3 Greedy by Point per Distance.....	10
3.2.4 Greedy by Tackling.....	10
3.2.5 Greedy by Point per Area.....	10
3.2.6 Greedy by Base Distance.....	10
3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy.....	10
3.4 Strategi Greedy yang Dipilih.....	12
BAB IV IMPLEMENTASI DAN PENGUJIAN.....	13
4.1 Implementasi Algoritma Greedy.....	13
4.1.1 Pseudocode.....	13
4.1.2 Struktur Project.....	16
4.1.3 Penjelasan Fungsi-Fungsi.....	17
4.2 Struktur Data yang Digunakan.....	17
4.3 Pengujian Program.....	18
4.3.1 Skenario Pengujian.....	18
4.3.2 Hasil Pengujian dan Analisis.....	19
BAB V KESIMPULAN DAN SARAN.....	21
5.1 Kesimpulan.....	21
LAMPIRAN.....	22
DAFTAR PUSTAKA.....	23

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 sederhana 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.

1. Game engine, yang secara umum berisi:
 - 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 berisi:

- Program untuk memanggil API yang tersedia pada backend.
- Program bot logic (bagian ini yang akan kalian implementasikan dengan algoritma greedy untuk bot kelompok kalian).
- Program utama (main) dan utilitas lainnya.

Adapun komponen-komponen dari permainan Diamonds antara lain:

1. Diamonds

Untuk memenangkan pertandingan, kita harus mengumpulkan diamond ini sebanyak-banyaknya dengan melewati/melangkahnya. 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. 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.

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.

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 menempa 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. 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 merupakan salah satu metode yang paling populer dan sederhana untuk memecahkan masalah optimasi. Pengertian dari Algoritma Greedy sendiri adalah pendekatan dalam pemrograman untuk memecahkan persoalan optimasi dengan cara yang tampaknya rakus (Greedy). Pendekatan dari Algoritma Greedy memfokuskan algoritma untuk mengambil keputusan optimal pada tiap langkahnya dengan harapan bahwa setiap langkah akan membawa kita lebih dekat ke solusi akhir yang optimal.

Algoritma Greedy akan mengambil solusi optimal pada setiap langkahnya tanpa memepertimbangkan konsekuensi di masa depan. Oleh sebab itu, Algoritma Greedy tidak selalu menghasilkan solusi optimal pada beberapa persoalan seperti pada Coin Exchange Problem. Tapi, pada persoalan-persoalan tertentu seperti persoalan Minimum Spanning Tree, Algoritma Greedy akan selalu menghasilkan solusi optimal dengan pendekatan Algoritma Kruskal atau Prim. Dalam konteks permainan Diamonds, algoritma greedy digunakan untuk memilih langkah terbaik yang menghasilkan poin terbanyak dalam waktu tercepat berdasarkan kondisi board saat ini.

2.2 Cara Kerja Program

Pada game ini, program dibagi menjadi 2 bagian besar yaitu bagian Game Engine dan Bot. Kedua bagian program ini dapat berkomunikasi menggunakan API yang telah tersedia pada program. Pada bagian Bot, program akan memanggil API yang tersedia pada backend. Bot melakukan POST request terhadap endpoint API untuk mendaftarkan dirinya berdasarkan email dan password. Setelah terdaftar, bot dapat bergabung/join ke dalam permainan dengan melakukan POST terhadap endpoint join dan mendapatkan informasi dari board. Berdasarkan informasi tersebut, Bot dapat melakukan kalkulasi move berdasarkan logic masing-masing bot dan mengirimkannya melalui POST request ke backend melalui endpoint move. Backend akan memberikan response dengan body berisi kondisi board setelah move tersebut, dan bot kembali mengkalkulasi move selanjutnya dan diulang sampai permainan selesai. Proses ini berlangsung secara iteratif hingga waktu permainan habis, dan skor akhir bot ditentukan berdasarkan jumlah

diamond yang berhasil dikumpulkan dan disimpan ke base. Untuk menjalankan bot tersebut, kita menjalankan main menggunakan :

Untuk menjalankan satu bot

```
python main.py --logic Random --email=your_email@example.com --name=your_name  
--password=your_password --team etimo
```

Untuk menjalankan beberapa bot (windows)

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

BAB III

APLIKASI STRATEGI *GREEDY*

3.1 Proses *Mapping* Persoalan Menjadi Elemen Algoritma Greedy

3.1.1 Himpunan Kandidat

Pada permasalahan ini terdapat beberapa objek yang dapat dimasukkan ke dalam himpunan kandidat yaitu:

Nama Objek	Penjelasan
Base	Objek ini merupakan markas dari bot dimana jika bot kembali ke base membawa N diamonds, maka bot ini akan mendapatkan N points sesuai dengan banyaknya diamonds. Kemudian base juga merupakan tempat muncul kembalinya bot jika botter-tackle oleh bot lain.
Red Button	Jika objek ini dilewati oleh bot maka semua diamond yang terletak di board akan generate ulang dan diacak posisinya, posisi dari red button pun akan diacak setelah dilewati.
Diamond	Jika objek ini dilewati oleh bot maka bot akan menambahkan diamond sebanyak N kedalam inventory-nya. Jumlah N tergantung dengan tipe diamond di mana jika bot mengambil red diamond maka N adalah dua dan jika bot mengambil blue diamond maka N adalah satu.
Inventory	Inventory merupakan tempat bot menyimpan diamonds yang terambil oleh bot selama berjalannya game. Inventory memiliki batasan slot 5 diamonds dimana jika kita mengambil diamond saat slot sudah 5 maka diamond tidak akan terambil dan bot pun harus kembali ke base untuk mengosongkan inventory.
Bot Musuh	Pada setiap game terdapat bot musuh yang ikut bermain. Bot musuh memiliki sifat yang sama dengan bot kita dalam hal base,

	teleporter, dan diamonds, dan inventory. Mereka men-tackle kita dengan cara melewati bot kita. Jika kita ter-tackle maka semua diamond pada inventory kita akan terambil oleh bot yang menyerang kita. Sebaliknya juga kita dapat men-tackle bot musuh dan mendapatkan semua diamond yang terdapat pada inventory mereka.
--	---

3.1.2 Himpunan Solusi

Dengan pemilihan objek-objek tersebut maka dapat dikatakan bahwa himpunan solusi dari algoritma ini adalah path yang diambil oleh bot.

3.1.3 Fungsi Solusi

Algoritma ini memeriksa jarak terdekat untuk ke Base, RedButton, dan Diamond dan memilih berdasarkan syarat pada fungsi seleksi.

3.1.4 Fungsi Seleksi

Seleksi tujuan dalam permainan dapat disimpulkan sebagai berikut :

Nama Objek	Syarat
Base	Bot akan menjadikan base sebagai tujuan jika: <ol style="list-style-type: none"> 1. Inventory penuh, bot membawa 5 diamond. 2. Membawa lebih dari 2 diamond dan jarak base kurang dari 2. 3. Membawa diamond dan batas waktu kurang dari 10 detik.
Red Button	Bot akan menjadikan red button sebagai tujuan jika jumlah diamond yang tersisa pada board kurang dari sama dengan 6 dan jarak dari red button kurang dari jarak diamond terdekat.
Diamond	Terdapat beberapa diamond pada board namun diamond yang dijadikan tujuan oleh

	bot adalah diamond terdekat.
Bot Musuh	<p>Bot musuh akan dijadikan tujuan oleh bot jika:</p> <ol style="list-style-type: none"> 1. Bot musuh membawa lebih dari 2 diamond dan jarak dengan bot musuh kurang dari 3 2. Bot musuh terletak tepat 1 satuan dari bot kita

3.1.5 Fungsi Kelayakan

Algoritma ini akan memeriksa apakah inventory sudah penuh dan juga apakah waktu permainan kurang dari sama dengan 10 detik.

3.1.6 Fungsi Objektif

Algoritma ini memaksimumkan jumlah poin yang didapatkan oleh bot.

3.2 Eksplorasi Alternatif Solusi Greedy

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

3.2.1 Greedy by Distance

Greedy by Distance adalah Algoritma Greedy dengan mencari Diamond mana yang memiliki jarak paling dekat terhadap bot. Jarak yang digunakan pada persoalan ini adalah *Manhattan Distance*.

3.2.2 Greedy by Point

Greedy by Point adalah Algoritma Greedy dengan mencari Diamond mana yang memiliki poin paling besar pada board. Karena dalam permainan ini hanya terdapat 2 kemungkinan nilai point Diamond, yaitu 1 point (Blue Diamond) dan 2 point (Red Diamond), maka bot

akan menuju Red Diamond dengan jarak terdekat dengan bot selama masih ada Red Diamond pada board. Setelah Red Diamonds habis, baru bot menuju ke Blue Diamond terdekat sampai muncul kembali Red Diamond pada Board.

3.2.3 Greedy by Point per Distance

Greedy by Point per Distance adalah Algoritma Greedy dengan mencari Diamond mana yang memiliki nilai Point/Distance paling besar terhadap bot. Bot akan menuju ke Diamond dengan nilai Point/Distance yang paling besar.

3.2.4 Greedy by Tackling

Greedy by Point per Tackling adalah Algoritma Greedy dengan mencari Bot lawan yang memiliki Diamond dan berjarak terdekat dengan Bot. Bot akan menuju ke posisi bot lawan dan mencoba meng-*tackle* bot lawan tersebut.

3.2.5 Greedy by Point per Area

Greedy by Point per Area adalah Algoritma Greedy dengan mencari Area dengan total point pada Area tersebut yang paling besar. Bot akan menghitung jumlah point pada area yang telah dibagi-bagi pada board, kemudian menuju area dengan point total paling besar.

3.2.6 Greedy by Base Distance

Greedy by Base Distance adalah Algoritma Greedy dengan mencari Diamond dengan jarak yang dekat dengan base. Tujuan dari pendekatan Algoritma Greedy ini adalah untuk mencegah bot bergerak terlalu jauh dari base sehingga tidak memerlukan waktu lama untuk kembali ke base dan menyimpan diamond menjadi point.

3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy

No	Alternatif Solusi (Greedy by)	Kelebihan	Kekurangan
1	Distance	1. Semakin dekat diamond, semakin banyak diamond yang dapat diperoleh dalam	1. Bot mengambil diamond tanpa mempertimbangkan nilainya.

		<p>satu permainan.</p> <ol style="list-style-type: none"> 2. Bot bergerak semakin jauh dari base jika diamond sudah sedikit. 	<ol style="list-style-type: none"> 2. Bot melewatkan diamond merah (2 poin) padahal dengan jarak yang sangat kecil.
2	<i>Point</i>	<ol style="list-style-type: none"> 1. Bot akan memprioritaskan untuk mengambil diamond merah terlebih dahulu. 2. Memaksimalkan jumlah poin yang dikumpulkan dalam satu pengisian inventory. 	<ol style="list-style-type: none"> 1. Bot memprioritaskan pengambilan diamond merah walaupun dengan jarak sangat jauh. 2. Bot tidak memperhitungkan posisi dan jarak dengan diamond sehingga terjadi ketidakseimbangan antara waktu yang ditempuh dengan jumlah poin yang didapatkan.
3	<i>Point per Distance</i>	<ol style="list-style-type: none"> 1. Bot memilih waktu dan gerakan yang paling efisien untuk mendapatkan diamond. 2. Bot menyeimbangkan antara poin diamond yang akan diambil dengan jarak yang ditempuh, contohnya bot akan mengambil diamond merah dahulu dibanding diamond biru dengan jarak yang sama. 3. Bot akan menghindari gerakan perjalanan yang tidak efektif. 	<ol style="list-style-type: none"> 1. Bot mengabaikan jarak yang ditempuh hanya untuk mengambil diamond merah. 2. Bot akan menghabiskan banyak waktu untuk bergerak mengambil diamond merah antara satu sama lain dengan jarak yang berjauhan. 3. Bot tidak mempertimbangkan waktu permainan dan perhitungan antara rasio jarak dengan poin diamond bisa salah.
4	<i>Tackling</i>	<ol style="list-style-type: none"> 1. Bot mendapatkan banyak poin dalam waktu yang singkat tanpa mengambil diamond sendiri. 2. Bot lawan akan kehilangan poin yang mana membuat kemungkinan menang lebih besar. 3. Bot akan menghambat progress bot lawan dan ini membuat waktu yang digunakan efektif. 	<ol style="list-style-type: none"> 1. Bot memiliki risiko untuk gagal melakukan tackle bot lawan ataupun bot lawan dapat melakukan tackle juga terhadap bot kami. 2. Bot memiliki kemungkinan banyak menghabiskan waktu untuk saling mentackle satu sama lain dengan bot lawan, apalagi ketika bot lawan sampai terlebih dahulu ke base. 3. Bot akan memiliki risiko untuk berjarak jauh dengan

			base sehingga ketika waktu habis bot kesulitan untuk dapat mencapai kembali ke base.
5	<i>Point per Area</i>	<ol style="list-style-type: none"> 1. Bot melakukan pergerakan untuk pengambilan diamond pada area yang memiliki lebih banyak penyebaran diamond sehingga jarak antar diamond lebih kecil. 2. Bot akan secara otomatis memilih kelompok diamond yang memberikan nilai maksimal per langkah, yang sangat berguna ketika terjadi regenerasi diamond, karena bot langsung dapat mengevaluasi ulang semua target dan bergerak ke kumpulan diamond yang paling efisien. 	<ol style="list-style-type: none"> 1. Bot akan menghabiskan banyak waktu ketika area yang ingin dicapai terletak sangat jauh dari base. 2. Bot tidak responsif terhadap diamond-diamond yang memiliki jarak terdekat. 3. Bot perlu melakukan banyak perhitungan area dalam waktu singkat, dan jika terlalu kompleks atau lambat, bisa menyebabkan keterlambatan saat menentukan arah gerak di setiap langkahnya.
6	<i>Base Distance</i>	<ol style="list-style-type: none"> 1. Bot berada pada posisi yang terletak tidak jauh dari base sehingga tidak memakan waktu yang lama untuk kembali ke base. 	<ol style="list-style-type: none"> 1. Bot kehilangan kesempatan untuk mengambil diamond terdekat karena berada pada jarak yang jauh dengan base.

3.4 Strategi Greedy yang Dipilih

Berdasarkan hasil analisis dan pengujian, strategi Greedy yang dipilih kami merupakan gabungan dari mekanisme permainan dan kondisi yang ada sesuai dengan Greedy Bot, yaitu sebagai berikut:

1. Greedy by Point per Distance

Greedy by Point per Distance kami pilih karena strategi ini berpacu pada pemilihan target diamond, bot menghitung rasio antara point diamond dengan jarak Manhattan dari posisi bot saat ini ke arah diamond. Bot akan mengurutkan diamond berdasarkan jarak atau point sehingga diamond dengan jarak terkecil akan diprioritaskan. Strategi ini mengoptimalkan efisiensi pergerakan bot dalam mengumpulkan diamond untuk dapat

mengumpulkan diamond sebanyak banyaknya dengan langkah seminimal mungkin, bot akan fokus pada target yang benar benar menguntungkan.

2. Greedy by Base Distance

Greedy by Base Distance dipilih kami karena strategi ini memastikan bot menjaga jarak ke base dengan baik, terutama saat inventory diamond sudah penuh. Jika bot sudah membawa 3 diamond, bot akan langsung berusaha kembali ke base untuk menyimpan terlebih dahulu diamond yang sudah didapat agar bot tidak mengalami kehabisan ruang untuk mengumpulkan diamond yang lain. Strategi ini juga dilakukan untuk menghindari tackle musuh saat membawa diamond banyak dan berisiko kehilangan diamond karena dicuri. Strategi ini sangatlah penting, mengingat waktu yang terbatas dan risiko kehilangan point sehingga bot akan mengutamakan keamanan dan efisiensi dengan memprioritaskan perjalanan kembali ke base ketika kapasitas sudah maksimal.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma Greedy

4.1.1 Pseudocode

```
from game.logic.base import BaseLogic
from game.models import GameObject, Board, Position
from game.util import get_direction
import random

class GreedyBot(BaseLogic):
    def __init__(self):
        super().__init__()
```

```

        self.dikunjungi = set()
        self.target_gagal = {}
        self.jumlah_melangkah = 0

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

    def dekat_Musuh(self, posisi: Position, papan: Board, id_bot: str) ->
bool:
        posisi_Musuh = [
            objek.position for objek in papan.game_objects
            if objek.type == "bot" and objek.id != id_bot
        ]
        return any(self.jarak_Manhattan(posisi, musuh) <= 1 for musuh in
posisi_Musuh)

    def cari_arah_ke(self, posisi_awal: Position, posisi_tujuan: Position,
papan: Board):
        dx, dy = get_direction(posisi_awal.x, posisi_awal.y,
posisi_tujuan.x, posisi_tujuan.y)
        opsi = [(dx, dy), (dx, 0), (0, dy)]
        for gerak in opsi:
            if papan.is_valid_move(posisi_awal, gerak[0], gerak[1]):
                return gerak

        cadangan_geraknya = [(1, 0), (0, 1), (-1, 0), (0, -1)]
        random.shuffle(cadangan_geraknya)
        for gerak in cadangan_geraknya:
            if papan.is_valid_move(posisi_awal, gerak[0], gerak[1]):
                return gerak
        return 0, 0

    def langkah_berikutnya(self, bot: GameObject, papan: Board):
        self.jumlah_melangkah += 1
        properti = bot.properties
        posisi_sekarang = bot.position

        # Mereset kunjungan setiap 30 langkah
        if self.jumlah_melangkah % 30 == 0:
            self.dikunjungi.clear()

        # Kembali ke base jika diamond penuh
        if properti.diamonds >= 3:

```

```

        if posisi_sekarang == properti.base:
            return 0, 0
        return self.cari_arah_ke(posisi_sekarang, properti.base, papan)

    # Mengumpulkan diamond
    semua_diamond = [objek for objek in papan.game_objects if
objek.type == "diamond"]
    target_valid = []
    for diamond in semua_diamond:
        posisi_diamond = diamond.position
        kunci = (posisi_diamond.x, posisi_diamond.y)
        jarak = self.jarak_Manhattan(posisi_sekarang, posisi_diamond)
        jarak_gagal = self.target_gagal.get(kunci)
        if jarak_gagal is None or jarak < jarak_gagal:
            if not self.dekat_Musuh(posisi_diamond, papan, bot.id):
                poin = diamond.properties.get("points", 1)
                target_valid.append((posisi_diamond, poin, jarak))

    if target_valid:
        target_valid.sort(key=lambda item: (item[2] / item[1])) #
jarak / poin
        for posisi_target, poin, _ in target_valid:
            gerak = self.cari_arah_ke(posisi_sekarang, posisi_target,
papan)

            if papan.is_valid_move(posisi_sekarang, *gerak):
                self.dikunjungi.add((posisi_target.x, posisi_target.y))
                return gerak
            else:
                self.target_gagal[(posisi_target.x, posisi_target.y)] =
self.jarak_Manhattan(posisi_sekarang, posisi_target)

    # Eksplorasi
    arah_eksplorasi = [(1, 0), (-1, 0), (0, 1), (0, -1)]
    random.shuffle(arah_eksplorasi)
    for gerak in arah_eksplorasi:
        posisi_baru = Position(posisi_sekarang.x + gerak[0],
posisi_sekarang.y + gerak[1])
        if papan.is_valid_move(posisi_sekarang, gerak[0], gerak[1]) and
(posisi_baru.x, posisi_baru.y) not in self.dikunjungi:
            self.dikunjungi.add((posisi_baru.x, posisi_baru.y))
            return gerak

    # Melakukan gerakan fallback jika semua sudah dikunjungi

```



```

        for gerak in arah_eksplorasi:
            if papan.is_valid_move(posisi_sekarang, gerak[0], gerak[1]):
                return gerak

    # Hanya Diam
    return 0, 0

def next_move(self, bot: GameObject, board: Board):
    return self.langkah_berikutnya(bot, board)

```

4.1.2 Struktur Project

tubes1-IF2211-bot-starter-pack-1.0.1/

```

├── __pycache__/
│   └── decode.cpython-313.pyc
├── game/
│   ├── __pycache__/
│   ├── logic/
│   │   ├── __pycache__/
│   │   ├── __init__.py
│   │   ├── base.py
│   │   ├── greedybot.py
│   │   ├── mybot.py
│   │   └── random.py
│   ├── __init__.py
│   ├── api.py
│   ├── board_handler.py
│   ├── bot_handler.py
│   ├── models.py
│   └── util.py
├── .gitignore
├── decode.py
├── main.py
├── README.md
├── requirements.txt
├── run-bots.bat
└── run-bots.sh

```

4.1.3 Penjelasan Fungsi-Fungsi

Berikut adalah penjelasan singkat dari fungsi yang dibuat:

Fungsi	Penjelasan
Langkah_berikutnya(self, bot: GameObject, papan: Board)	Untuk menentukan langkah gerak selanjutnya bot.
Cari_arah_ke(self, posisi_awal: Position, posisi_tujuan: Position, papan: Board)	Untuk mencari arah langkah satu langkah dari posisi saat ini menuju posisi tujuan.
Jarak_Manhattan(self, pos1: Position, pos2: Position) -> int	Menghitung jarak Manhattan antara dua titik di papan. Digunakan untuk memperkirakan seberapa jauh bot ke diamond atau base. Penting untuk menentukan target diamond yang paling efisien
Dekat_Musuh(self, posisi: Position, papan: Board, id_bot: str) -> bool	Mengecek apakah posisi yang dituju dekat dengan bot musuh. Membantu bot memilih diamond yang lebih aman untuk diambil.

4.2 Struktur Data yang Digunakan

Dalam melakukan pengembangan bot pada permainan diamonds, berbagai struktur data yang berasal dari starter pack bot dan data yang diimplementasikan digunakan dengan baik. Struktur data berperan penting dalam pelaksanaan strategi greedy yang digunakan bot.

Dalam implementasi bot bernama Greedy Bot, digunakan beberapa struktur data untuk mendukung strategi Greedy yang digunakan. Terdapat struktur list, yang digunakan untuk menyimpan diamond dan diurutkan berdasarkan rasio antara jarak dan nilai poin, agar bot memilih target yang paling menguntungkan. Struktur set (dikunjungi) juga berfungsi menyimpan posisi yang telah tereksplorasi oleh bot agar bot menghindari pergerakan berulang. Struktur dict (target_gagal) membantu bot agar tidak mencoba target yang sama secara terus menerus.

Variabel (jumlah_melangkah) int juga digunakan untuk mencatat jumlah langkah yang dilakukan untuk melakukan reset terhadap memori secara berkala agar eksplorasi berjalan efisien. Seluruh logika bot diimplementasikan dalam bentuk sebuah kelas Python bernama Greedy.Bot yang merupakan turunan dari BaseLogic yang disediakan di starter pack. Implementasi next-move() akan dijalankan pada setiap giliran untuk menentukan langkah bot

selanjutnya berdasarkan strategi greedy yang telah dipersiapkan. Dengan penggunaan struktur data ini, bot dapat berjalan secara cepat, efisien terhadap kondisi permainan.

Struktur Data	Tipe	Digunakan untuk
dikunjungi	set	Menyimpan posisi yang sudah dilewati.
target_gagal	dict	Mencatat diamond yang gagal dijangkau.
target_valid	list	Menyimpan kandidat diamond yang layak diambil
arah_eksplorasi	list	Menentukan arah eksplorasi acak
jumlah_melangkah	int	Menghitung jumlah langkah untuk reset
GameObject, Board, Position	class	Objek-objek utama dalam permainan

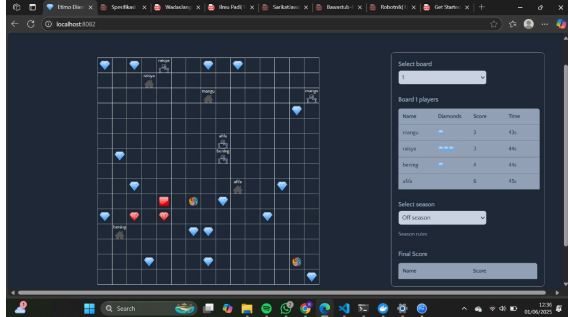
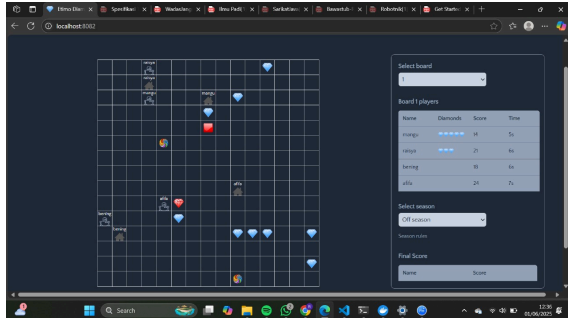
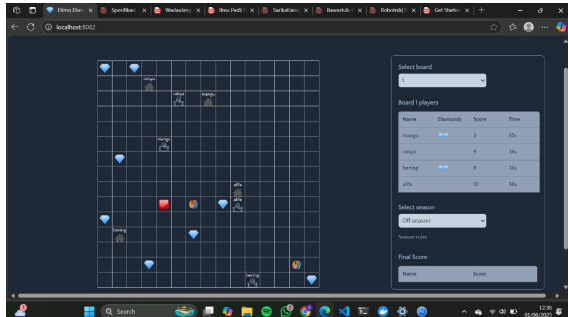
4.3 Pengujian Program

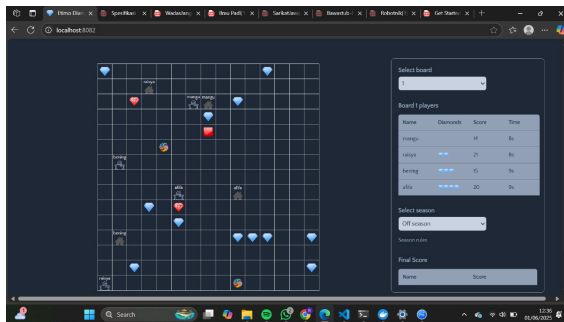
4.3.1 Skenario Pengujian

No.	Skenario Pengujian	Tujuan
1.	Pemilihan diamond dengan risiko terbaik (point per distance)	Memastikan strategi greedy berdasarkan rasio point per jarak diterapkan dengan benar, bukan sekadar memilih yang paling dekat atau paling tinggi point.
2.	Keputusan pulang ke base saat inventory penuh	Memverifikasi bahwa bot langsung kembali ke base saat inventory penuh, mengabaikan diamond di sekitar, sesuai strategi greedy by base distance.
3.	Menghindari diamond yang dekat musuh	Menguji apakah fungsi dekat_Musuh() bekerja dengan benar dalam mencegah bot mendekati diamond berisiko tinggi karena berada di sekitar musuh.

4.	Eksplorasi saat tidak ada diamond yang aman	Memastikan bot masuk ke mode eksplorasi dengan menjelajah ke lokasi lain saat tidak ada target yang aman atau layak dikejar.
----	---	--

4.3.2 Hasil Pengujian dan Analisis

No.	Gambar	Penjelasan
1.		Dalam gambar ini terlihat bahwa bot berada di antara beberapa diamond dengan nilai dan jarak yang bervariasi. Meskipun terdapat diamond yang lebih dekat, bot memilih bergerak menuju diamond yang memiliki rasio point/jarak tertinggi, bukan sekadar yang paling dekat atau paling tinggi poinnya. Hal ini menunjukkan bahwa strategi greedy yang diterapkan telah mempertimbangkan efisiensi nilai per langkah, bukan sekadar keuntungan langsung, dan itu sesuai dengan skenario pengujian pertama.
2.		Dalam gambar ini, bot terlihat telah membawa 3 diamond (penuh) dan meskipun masih ada diamond lain yang menguntungkan di sekitar, bot langsung bergerak ke arah base tanpa mencoba mengambil diamond tambahan. Ini menunjukkan bahwa fungsi pemeriksaan kapasitas berjalan baik dan strategi greedy memprioritaskan menyimpan diamond ke base saat inventory penuh, bukan tergoda mengambil lebih banyak.
3.		Gambar ini memperlihatkan bahwa ada beberapa diamond yang lokasinya berdekatan dengan bot musuh (jarak Manhattan 1 atau 2). Meskipun secara nilai dan jarak diamond tersebut menarik, bot tidak mendekatinya dan memilih arah lain, bahkan cenderung menjauh. Ini membuktikan bahwa fungsi pertimbangan risiko terhadap musuh (<code>dekat_Musuh()</code>)

		telah dijalankan, sesuai dengan skenario pengujian ketiga.																				
4.	 <p>The screenshot shows a game interface with a grid map on the left and a sidebar menu on the right. The map contains various icons, including blue diamonds, red hearts, and yellow stars. The sidebar menu has a 'Select board' dropdown, a 'Board Layouts' table, a 'Select weapon' dropdown, and a 'Final Score' section.</p> <table> <tr> <th>Board</th> <th>Dimensions</th> <th>Score</th> <th>Time</th> </tr> <tr> <td>Board 1</td> <td>10 x 10</td> <td>10</td> <td>10</td> </tr> <tr> <td>Board 2</td> <td>10 x 10</td> <td>10</td> <td>10</td> </tr> <tr> <td>Board 3</td> <td>10 x 10</td> <td>10</td> <td>10</td> </tr> <tr> <td>Board 4</td> <td>10 x 10</td> <td>10</td> <td>10</td> </tr> </table> <p>The 'Select weapon' dropdown is set to 'Off location'. The 'Final Score' section shows 'Name' and 'Score'.</p>	Board	Dimensions	Score	Time	Board 1	10 x 10	10	10	Board 2	10 x 10	10	10	Board 3	10 x 10	10	10	Board 4	10 x 10	10	10	<p>Gambar ini menunjukkan kondisi di mana tidak ada diamond yang aman atau layak untuk dikejar tidak peduli karena terlalu dekat dengan musuh atau posisi yang sulit dijangkau. Bot dalam gambar terlihat tidak diam di tempat, tapi bergerak menjelajahi peta ke arah yang sebelumnya belum dikunjungi. Gerakan acak namun sah ini menunjukkan bahwa bot telah memasuki mode eksplorasi, mencari kemungkinan lokasi diamond baru seperti yang ditargetkan dalam skenario pengujian keempat.</p>
Board	Dimensions	Score	Time																			
Board 1	10 x 10	10	10																			
Board 2	10 x 10	10	10																			
Board 3	10 x 10	10	10																			
Board 4	10 x 10	10	10																			

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan dari tugas besar ini adalah kami dapat mengaplikasikan strategi greedy dalam permainan diamonds ini. Algoritma greedy mampu memberikan solusi yang cukup efektif dalam mengumpulkan poin secara cepat dan efisien, terutama saat kondisi permainan stabil dan target diamond mudah dijangkau. Kombinasi pemilihan diamond berdasarkan rasio poin dan jarak serta pengelolaan inventory dengan kembalinya ke base mampu membuat bot memaksimalkan skor, ini juga membantu meminimalisir risiko kehilangan diamond akibat tackle bot lawan.

5.2 Saran

Adapun beberapa saran pengembangan untuk tugas besar ini adalah:

1. Menghindari penyelesaian tugas besar secara tergesa-gesa mendekati deadline agar bot yang dihasilkan dapat dikembangkan dengan lebih baik dan sempurna.
2. Melakukan uji coba antar tim secara rutin selama proses pengembangan akan sangat berguna untuk mengidentifikasi kelemahan strategi dan memperkuat daya saing bot dalam perlombaaan nanti.
3. Saat ini, bot hanya menghindari diamond yang berada dekat dengan musuh. Strategi dapat ditingkatkan dengan kemampuan membaca pola pergerakan musuh dan memperkirakan kemungkinan arah serangan mereka. Perbanyak variasi strategi greedy untuk meningkatkan adaptasi
4. Melakukan evaluasi secara rutin selama proses pengembangan untuk mendeteksi bug sehingga dapat dilakukan perbaikan.

LAMPIRAN

A. Repository Github

<https://github.com/beningapniprameswari/Tubes1-Mangu>

B. Video Penjelasan

https://drive.google.com/drive/folders/1bIJ8ItuNbxhd8ZDbYdMvGrEJq8lECwO0?usp=drive_link

DAFTAR PUSTAKA

- [1] GeeksforGeeks, “Greedy Algorithm.” [Online]. Tersedia pada: <https://www.geeksforgeeks.org/greedy-algorithms/>. [Diakses: 01-Juni-2025].
- [2] 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-\(2021\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf) [Diakses: 01-Juni-2025].
- [3] 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-\(2021\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf) [Diakses: 01-Juni-2025].
- [4] 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-\(2022\)-Bag3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf) [Diakses: 01-Juni-2025].