

LAPORAN TUGAS BESAR I

IF2211 STRATEGI ALGORITMA



Disusun oleh:

Kelompok Sanss Kuy

Raden Francisco Trianto Bratadiningrat	13522091
--	----------

Fabian Radenta Bangun	13522105
-----------------------	----------

Hafizh Hananta Akbari	13522132
-----------------------	----------

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2024

DAFTAR ISI

DAFTAR ISI.....	2
DAFTAR GAMBAR.....	3
DAFTAR TABEL.....	4
BAB I DESKRIPSI TUGAS.....	5
1.1 Deskripsi Permainan.....	5
1.2 Cara Kerja Permainan.....	6
BAB II LANDASAN TEORI.....	7
2.1 Algoritma Greedy.....	7
2.2 Elemen-Elemen Algoritma Greedy.....	7
2.3 Game Engine Diamonds.....	8
2.4 Cara Kerja Program.....	8
BAB III APLIKASI STRATEGI GREEDY.....	12
3.1 Mapping Persoalan Diamonds.....	12
3.2 Eksplorasi Alternatif Solusi.....	13
3.3 Faktor Pengaruh Efektivitas Secara Umum.....	18
3.4 Strategi Greedy yang Dipilih.....	19
BAB IV IMPLEMENTASI DAN PENGUJIAN.....	21
4.1. Struktur Data Program.....	21
4.2 Implementasi dalam Pseudocode.....	24
4.3 Analisis dan Pengujian.....	27
BAB V KESIMPULAN DAN SARAN.....	36
5.1 Kesimpulan.....	36
5.2 Saran.....	36
5.3 Refleksi.....	36
LAMPIRAN.....	37
DAFTAR PUSTAKA.....	38

DAFTAR GAMBAR

Gambar 4.3.1.1 Bot Sebelum Mengambil diamond

Gambar 4.3.1.2 Bot Setelah Mengambil diamond

Gambar 4.3.2.1 Bot Sebelum Inventory Penuh/Hampir Penuh

Gambar 4.3.2.2 Inventory Penuh/Hampir Penuh

Gambar 4.3.2.3 Gerak Bot Ketika Inventory Penuh/Hampir Penuh

Gambar 4.3.3.1 Bot Sebelum Melakukan Tackle

Gambar 4.3.3.2 Bot Setelah Melakukan Tackle

Gambar 4.3.3.3 Inventory Sebelum Tackle

Gambar 4.3.3.4 Inventory Setelah Tackle

Gambar 4.3.4.1 Bot Sebelum Menggunakan Teleporter

Gambar 4.3.4.2 Bot Setelah Menggunakan Teleporter

Gambar 4.3.5.1 Bot Sebelum Deteksi Waktu Akan Habis

Gambar 4.3.5.2 Waktu Sebelum Deteksi Waktu Akan Habis

Gambar 4.3.5.3 Bot Setelah Deteksi Waktu Akan Habis

Gambar 4.3.5.4 Bot Sesaat Waktu Akan Habis

DAFTAR TABEL

Tabel 3.1.1 Mapping Persoalan Diamonds

BAB I

DESKRIPSI TUGAS

1.1 Deskripsi Permainan

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.

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

Komponen-komponen dari permainan Diamonds antara lain:

1. Diamond
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. 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.

1.2 Cara Kerja Permainan

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. 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 Algoritma Greedy

Algoritma greedy adalah algoritma yang memecahkan suatu persoalan dengan mengambil pilihan yang terbaik untuk setiap langkahnya. Algoritma greedy berfokus terhadap pengambilan pilihan optimum global di setiap langkahnya tanpa memperhatikan konsekuensi atau pengaruh dari pilihan tersebut untuk kedepannya. Dengan memilih pilihan yang terbaik (optimum lokal) di setiap langkah, algoritma greedy diharapkan mampu untuk mencapai optimum global. Contohnya, dalam masalah knapsack. Algoritma greedy akan memilih barang dengan rasio nilai banding berat yang paling tinggi dan memasukkannya ke dalam tas hingga tas penuh.

Algoritma greedy memiliki beberapa kelebihan. Pertama adalah kecepatan dari algoritma greedy yang cenderung cepat karena algoritma ini hanya meninjau pilihan-pilihan secara lokal tanpa melihat langkah-langkah berikutnya. Kemudian, algoritma greedy mampu mencapai optimal global di beberapa kasus dimana pencarian optimum lokal untuk setiap langkahnya mampu menghasilkan optimum global. Lalu, algoritma greedy cenderung mudah dipahami dan digunakan terutama dalam kasus yang sederhana.

Selain kelebihan, algoritma greedy juga memiliki beberapa kekurangan. Pertama adalah ketidakpastian solusi yang didapat. Dalam berbagai kasus berbeda, algoritma greedy tidak mampu menghasilkan solusi optimum global karena pencarian optimum lokal di setiap langkah algoritma tersebut mampu mempengaruhi hasil yang didapat sehingga tidak dapat mencapai optimum global. Lalu, algoritma greedy tidak mampu menyelesaikan persoalan-persoalan tertentu.

2.2 Elemen-Elemen Algoritma Greedy

Algoritma greedy terdiri dari beberapa komponen, yaitu:

1. Himpunan kandidat (C)
Himpunan yang berisi kandidat yang akan dipilih pada setiap langkah.
Misal: simpul/sisi di dalam graf, job, task, koin, benda, karakter, dsb.
2. Himpunan solusi (S)
Himpunan yang berisi kandidat yang sudah dipilih.
3. Fungsi solusi
Fungsi yang menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi.
4. Fungsi seleksi (selection function)
Fungsi yang memilih kandidat berdasarkan strategi greedy tertentu.
Strategi greedy ini bersifat heuristik.
5. Fungsi kelayakan (feasible)
Fungsi yang memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak).

6. Fungsi objektif:

Fungsi ini menunjukkan sifat memaksimumkan atau meminimumkan.

2.3 Game Engine Diamonds

Game engine merupakan perangkat lunak yang berperan sebagai fondasi esensial dalam pengembangan video game. *Game engine* menyediakan berbagai komponen penting untuk menjalankan game, seperti *rendering*, *physics*, *audio*, dan *scripting*. Untuk menjalankan suatu permainan maka diperlukan *game engine* yang berjalan dan mengatur alur permainan.

Untuk permainan Diamonds diperlukan *game engine*. *Game engine* dapat ditemukan pada tautan <https://github.com/haziqam/tubes1-IF2211-game-engine/releases/tag/v1.1.0>. Pada tautan tersebut, sudah diberikan panduan yang berisi: komponen-komponen yang diperlukan, cara melakukan instalasi, dan cara menjalankan *game engine*.

2.4 Cara Kerja Program

Game engine adalah dasar dari permainan Diamonds. Ketika *Game engine* menyala, maka akan dibuatnya sebuah papan permainan Diamonds. *Game engine* yang aktif akan membuat *game object* (diamond, teleporter, dsb) yang dihasilkan secara acak baik posisi maupun jumlahnya. *Game engine* yang aktif juga dapat menerima bot yang dijalankan melalui Python yang kemudian dapat berkompetisi dalam papan permainan.

Setiap bot diberikan waktu hidup yang sama, untuk mendapatkan poin sebanyak mungkin. Semua bot yang diterima oleh game engine akan diletakkan secara acak pada papan permainan. Pada setiap detik Game engine akan meminta gerakan dari setiap bot melalui fungsi *next_move*. Gerakan dari semua bot akan dilakukan secara bersamaan ketika satu detik tersebut berlalu. Jika suatu bot memerlukan waktu lebih dari satu detik tersebut, maka gerakan yang dilakukan oleh bot tersebut akan dijalankan dengan konsekuensi total gerakan lebih sedikit dibanding dengan bot yang memproses gerakannya dengan cepat. Sehingga algoritma yang dimiliki oleh suatu bot harus dibuat secara optimal agar waktu pemrosesan gerakan tidak lama, namun dapat berkompetisi dengan bot-bot lain dalam mendapatkan poin sebanyak mungkin.

2.4.1 Menjalankan Bot

Untuk menjalankan Bot, terlebih dahulu diperlukannya Python. Python sendiri dapat diunduh dari tautan <https://www.python.org/>. Pada Terminal, pastikan dulu, lokasi berada pada direktori “src”. Terdapat dua cara menjalankan bot, menjalankan satu bot dan menjalankan beberapa bot secara bersamaan.

2.4.2.1 Menjalankan Satu Bot

Untuk menjalankan satu bot, maka dapat menggunakan perintah berikut:


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

2.4.2.1 Menjalankan Beberapa Bot Secara Bersamaan

Untuk menjalankan beberapa bot secara bersamaan, maka dapat menggunakan file yang menjalankan beberapa perintah sekaligus. Untuk Windows kita menggunakan file berekstensi *.bat* dan untuk Linux digunakan file berekstensi *.sh*.

Berikut isi dari file *run-bots.bat* untuk digunakan pada Windows:

```
@echo off  
start cmd /c "python main.py --logic MyBot --email=example1@email.com  
--name=bot1 --password=123456 --team etimo"  
start cmd /c "python main.py --logic MyBot --email=example2@email.com  
--name=bot2 --password=123456 --team etimo"  
start cmd /c "python main.py --logic MyBot --email=example3@email.com  
--name=bot3 --password=123456 --team etimo"  
start cmd /c "python main.py --logic MyBot --email=example4@email.com  
--name=bot4 --password=123456 --team etimo"
```

Berikut isi dari file *run-bots.sh* untuk digunakan pada Linux:

```
python main.py --logic MyBot --email=example1@email.com --name=bot1  
--password=123456 --team etimo etimo &  
python main.py --logic MyBot --email=example2@email.com --name=bot2  
--password=123456 --team etimo etimo &  
python main.py --logic MyBot --email=example3@email.com --name=bot3  
--password=123456 --team etimo etimo &  
python main.py --logic MyBot --email=example4@email.com --name=bot4  
--password=123456 --team etimo etimo &
```

Untuk menjalankan file tersebut dapat menggunakan perintah sebagai berikut:

```
# Untuk Windows  
./run-bots.bat  
  
# Untuk Linux  
./run-bots.sh
```

2.4.2 Hubungan Antara *Game Engine* dengan Bot

Game engine dapat menggunakan suatu bot dari file Python dengan cara memanggil sebuah fungsi milik suatu object bot. *Game engine* akan melakukan inisialisasi class milik bot

masing-masing. Hasil inisialisasi adalah objek bot, yang kemudian dapat digunakan oleh *game engine*. Dengan menggunakan panggilan kepada fungsi *next_move* dari objek bot tersebut, maka suatu bot dapat bergerak sesuai dengan algoritma yang dimiliki oleh sebuah bot.

2.4.2 Mengembangkan Bot

Pengembangan bot dimulai dengan *bot-starter-pack* v1.0.1 yang dapat ditemukan pada tautan <https://github.com/haziqam/tubes1-IF2211-bot-starter-pack/releases/tag/v1.0.1>. Struktur data starter pack tersebut adalah sebagai berikut:

```
bot-starter-pack/  
├── game/  
│   ├── logic/  
│   │   ├── otherBots/  
│   │   ├── random.py  
│   │   ├── base.py  
│   │   └── random.py  
│   ├── __init__.py  
│   ├── api.py  
│   ├── board_handler.py  
│   ├── bot_handler.py  
│   ├── models.py  
│   └── util.py  
├── .gitignore  
├── decode.py  
├── main.py  
├── README.md  
├── requirement.txt  
├── run-bots.bat  
└── run-bots.sh
```

Pengembangan bot dilakukan pada folder *bot-starter-pack/game/logic/*. Pada folder tersebut dapat dibuat file berekstensi *.py*. Dalam file tersebut, diperlukan dibuatnya sebuah *Class* yang nanti akan dipanggil oleh *main.py* untuk menjalankan suatu bot. Contoh bot yang telah diberikan pada *bot starter pack* sebagai berikut (*random.py*) :

```
import random  
from typing import Optional  
  
from game.logic.base import BaseLogic  
from game.models import GameObject, Board, Position  
from game.util import get_direction  
  
class Random(BaseLogic):  
    def __init__(self):  
        directions = [(1, 0), (0, 1), (-1, 0), (0, -1)]  
        goal_position: Optional[Position] = None
```

```

current_direction = 0

def next_move(self, board_bot: GameObject, board: Board):
    props = board_bot.properties
    # Analyze new state
    if props.diamonds == 5:
        # Move to base
        base = board_bot.properties.base
        goal_position = base
    else:
        # Just roam around
        goal_position = None

    current_position = board_bot.position
    if goal_position:
        # We are aiming for a specific position, calculate delta
        delta_x, delta_y = get_direction(
            current_position.x,
            current_position.y,
            goal_position.x,
            goal_position.y,
        )
    else:
        # Roam around
        delta = directions[current_direction]
        delta_x = delta[0]
        delta_y = delta[1]
        if random.random() > 0.6:
            current_direction = (current_direction + 1) % len(
                directions
            )
    return delta_x, delta_y

```

Dapat terlihat bahwa diperlukannya sebuah class yang memiliki fungsi *next_move*. Fungsi tersebut akan dipanggil oleh *game engine* sebagai penghubung algoritma yang dimiliki oleh bot dan gerakan pada *board*. Bot dapat mengakses variabel-variabel *game object* dari *game engine*. Dengan menggunakan informasi yang diberikan, maka dapat dibuatnya algoritma untuk memenangkan permainan Diamonds.

BAB III

APLIKASI STRATEGI GREEDY

3.1 Mapping Persoalan Diamonds

Algoritma greedy memiliki beberapa elemen atau istilah yang sering menyertainya, diantaranya:

1. Himpunan kandidat (C): berisi kandidat yang akan dipilih pada setiap langkah.
2. Himpunan solusi (S): berisi kandidat yang sudah dipilih.
3. Fungsi 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 (layak atau tidak).
6. Fungsi objektif: memaksimumkan atau meminimumkan

Persoalan di dalam Diamonds dibagi menjadi *game objects* dan *property* dari *game objects* yang ada, sehingga mapping yang dihasilkan dari persoalan tersebut dapat dirangkum menjadi:

Tabel 3.1.1 Mapping Persoalan Diamonds

No	Persoalan	Kegunaan	Fungsi Proses
1	Diamond	<ul style="list-style-type: none"> - Sumber utama poin - Menambahkan poin bot, jika dibawa kembali ke base - Diproses dengan strategi <i>greedy by path</i> 	<i>moveToObjective</i>
2	Red Button/Diamond Button	<ul style="list-style-type: none"> - Pembuatan kembali diamond secara acak di papan permainan 	<i>moveToDiamondButton</i>
3	Teleporters (Portal)	<ul style="list-style-type: none"> - Terdapat 2 teleporter - Bot dapat memasuki suatu teleporter dan mengalami perpindahan posisi menuju lokasi teleporter yang lainnya - Penggunaan Teleporter diproses dengan strategi <i>greedy by path</i> 	<i>moveToObjective</i>
4	Bot	<ul style="list-style-type: none"> - Representasi pemain dalam permainan Diamonds - Bot mampu mengambil diamond serta menyimpannya 	-

		di base	
5	Base	- Tempat penyimpanan diamond secara permanen	<i>moveToBase</i>
6	Inventory	- Tempat penyimpanan diamond sementara di dalam bot	-
7	Tackle	- Sebuah bot dapat melakukan <i>tackle</i> kepada musuh untuk mengambil diamond yang dimiliki musuh - Bot yang terkena tackle akan kembali ke base	<i>moveToClosestEnemy</i>
8	Aksi (Action)	- Suatu rangkaian gerakan untuk melakukan suatu tujuan - Contoh: gerak menuju diamond, gerak menuju teleporter, tackle, kembali ke base, dsb.	-

3.2 Eksplorasi Alternatif Solusi

Menurut Kelompok Kami, terdapat beberapa alternatif solusi baik menggunakan algoritma berbasis *greedy* maupun *non-greedy*. Terdapat beberapa alternatif algoritma yang kami eksplorasi adalah sebagai berikut:

3.2.1 Greedy by points

Greedy by points adalah pendekatan algoritma greedy dengan memprioritaskan mendapatkan poin dalam suatu *game state*. Bot akan mencari sumber poin terbesar dan mencoba mendapatkannya. Pada Diamonds terdapat 2 macam diamond, merah yang bernilai 2 poin dan biru yang bernilai 1 poin. Bot ini akan selalu menuju diamond merah jika masih terdapat diamond merah pada *board*. Bot juga mencari musuh yang memiliki diamond yang banyak dan jarak yang dapat dicapai untuk melakukan *tackle*.

a. Mapping Elemen Greedy

- Himpunan kandidat : Semua Aksi yang tersedia.
- Himpunan solusi : Aksi yang terpilih.
- Fungsi solusi : Memeriksa apakah aksi menghasilkan poin terbesar.
- Fungsi seleksi : Pilih aksi yang paling menghasilkan poin.

- v. Fungsi kelayakan : Memeriksa apakah aksi menghasilkan poin.
- vi. Fungsi objektif : Mencari poin terbesar yang mungkin.

b. Analisis Efisiensi Solusi

Aksi utama yang dilakukan oleh bot adalah aksi menuju diamond merah, aksi melakukan *tackle*, dan kembali ke base. Bot akan memprioritaskan gerakan mendapatkan diamond merah. Bot pertama mencari diamond terdekat dengan melakukan traversal terhadap semua diamond dan mencari diamond dengan jarak terdekat. Proses tersebut relatif cepat, namun pemilihan aksi masih kurang efisien. Hal tersebut dikarenakan bobot poin yang dihasilkan aksi tidak sebanding dengan jumlah gerakan yang diperlukan untuk melakukan aksi tersebut. Dalam hal ini jumlah gerakan untuk mendapatkan diamond merah tanpa melihat jaraknya tidak sebanding dengan jumlah gerakan yang diperlukan untuk mendapatkan diamond biru namun dengan jarak yang sering kali lebih sedikit.

c. Analisis Efektivitas Solusi

Greedy by points sangat bergantung pada ada tidaknya diamond merah dan posisinya dari base milik bot. Jarak dari musuh juga mempengaruhi bisa tidaknya melakukan *tackle* terhadap bot musuh yang memiliki diamond.

Strategi ini efektif jika:

- Jumlah diamond merah lebih banyak dibandingkan dengan jumlah diamond biru pada jarak yang relatif sama.
- Terdapat banyak diamond merah di sekitar base.

Strategi ini tidak efektif jika:

- Lokasi diamond merah yang sangat jauh.
- Semua base musuh jauh dari base milik bot.

3.2.2 Greedy by enemy

Greedy by enemy adalah pendekatan algoritma greedy dengan yang berbasis musuh. Bot ini memiliki sumber utama poin, berasal dari musuh. Dasar dari bot ini adalah menyerang musuh dan melakukan *tackle* untuk mendapatkan diamond yang dimiliki oleh musuh. Ketika tidak ada musuh yang memiliki diamond atau tidak dapat dicapai sebelum musuh kembali ke rumah, maka bot akan bergerak menuju diamond terdekat.

a. Mapping Elemen Greedy

- i. Himpunan kandidat : Semua Aksi *tackle* terhadap semua bot musuh
- ii. Himpunan solusi : Aksi *tackler* yang terpilih.
- iii. Fungsi solusi : Memeriksa apakah aksi *tackle* menghasilkan poin.
- iv. Fungsi seleksi : Pilih aksi *tackle* yang dapat dilakukan dengan gerakan paling sedikit dan menghasilkan poin

- v. Fungsi kelayakan : Memeriksa apakah aksi *tackle* dapat dilakukan terhadap suatu bot dan dapat menghasilkan poin.
- vi. Fungsi objektif : Mencari aksi *tackle* yang paling efisien..

b. Analisis Efisiensi Solusi

Ketika bot menggunakan pendekatan *greedy by enemy*, bot akan melakukan aksi berdasarkan bot musuh. Aksi yang dilakukan bot adalah mencari musuh, melakukan *tackle* terhadap musuh sebelum kembali ke base, serta mencari diamond ketika musuh tidak memiliki diamond atau musuh tidak dapat dicapai sebelum kembali ke base musuh. Proses yang dilakukan bot untuk menentukan aksi yang dilakukan tidak rumit karena hanya menyangkut diamond yang dimiliki musuh. Bot hanya melihat jumlah diamond yang dimiliki musuh serta jarak dari musuh ke base-nya sehingga proses bisa berlangsung dengan cepat. Meskipun memiliki proses yang cepat, proses tersebut kurang efisien karena langkah yang diambil untuk berusaha melakukan *tackle* terhadap musuh memiliki kemungkinan untuk tidak sebaik mengambil diamond yang lebih mudah didapat di papan permainan.

c. Analisis Efektivitas Solusi

Solusi yang menggunakan pendekatan *greedy by enemy* memiliki ketergantungan terhadap diamond yang dimiliki musuh serta jarak dari musuh dengan base-nya.

Solusi ini dinilai efektif apabila:

- Musuh berada di jarak yang dekat dengan bot serta memiliki diamond dengan jumlah banyak
- Diamond di papan permainan sudah habis dan musuh memiliki diamond

Solusi ini dinilai kurang efektif apabila:

- Musuh memiliki strategi untuk menghindari *tackle* dari bot
- Terdapat banyak diamond yang berada di dekat bot ketika musuh sudah memiliki diamond

3.2.3 Greedy by safe

Greedy by safe adalah pendekatan algoritma *greedy* dengan yang berbasis strategi yang mementingkan diamond yang sedang dimiliki. Bot ini akan sering kembali ke base dan memiliki algoritma yang menjauhi bot-bot musuh. Bot akan mencari diamond yang dapat diambil secara aman tanpa resiko *tackle* oleh musuh. Pada kondisi inventory yang kosong bot ini mencari diamond terdekat dari basenya.

a. Mapping Elemen Greedy

- i. Himpunan kandidat : Semua aksi yang bisa dipilih
- ii. Himpunan solusi : Aksi yang lebih aman untuk diambil

- iii. Fungsi solusi : Memeriksa apakah aksi mampu menjamin keamanan bot
- iv. Fungsi seleksi : Memilih aksi yang mampu mendapatkan poin di dekat base
- v. Fungsi kelayakan : Menentukan apakah aksi mampu membuka resiko bot terkena *tackle* dari musuh
- vi. Fungsi objektif : Mencari poin dengan aman

b. Analisis Efisiensi Solusi

Bot yang melakukan pendekatan *greedy by safe* akan mementingkan keamanan bot. Untuk itu, aksi yang dilakukan bot adalah mencari letak musuh, mencari diamond yang tidak dekat dengan musuh, serta menghindari musuh ketika musuh berada di jarak yang dekat dengan bot. Proses yang dilakukan bot untuk menentukan aksi yang diambil menyangkut peninjauan letak musuh serta letak diamond yang berada di papan permainan sehingga membuat proses yang dilakukan memakan waktu yang sedikit lebih lama. Aksi yang diambil bisa cukup efisien hanya apabila terdapat banyak diamond yang berada di base. Selain itu, pengambilan aksi yang terlalu aman menghentikan bot dari melakukan *tackle* terhadap musuh serta mengambil diamond tambahan sehingga bisa kurang efisien mengenal jumlah langkah yang diambil dan poin yang bisa didapat.

c. Analisis Efektivitas Solusi

Solusi yang menggunakan pendekatan *greedy by safe* memiliki ketergantungan letak musuh di papan permainan serta letak diamond terhadap musuh.

Solusi ini dinilai efektif apabila:

- Letak musuh jauh dari area yang memiliki diamond dalam jumlah banyak
- Diamond yang berada di dekat base memiliki jumlah yang banyak

Solusi ini dinilai kurang efektif apabila:

- Musuh berada di area dengan banyak diamond

3.2.4 Prediction

Prediction (prediksi) adalah pendekatan algoritma yang menggabungkan teknik *backtracking* dan *greedy*. Bot akan menggunakan *game state* saat ini untuk melakukan prediksi rangkaian aksi yang terbaik untuk mendapatkan poin yang terbanyak. Prediksi ini kemudian menggunakan *backtracking* ketika gerakan terjadi dan *game state* berubah. Bot juga dapat menyimpan data gerakan setiap bot untuk menentukan pola gerakan bot lain. Dengan mengetahui pola gerakan bot lain maka bot dapat menambahkan aksi *tackle* ke dalam rangkaian aksi yang dieksekusikan.

a. Mapping Elemen Greedy

- i. Himpunan kandidat : Semua aksi yang bisa dipilih

- ii. Himpunan solusi : Aksi yang paling optimal berdasarkan pola dari aksi-aksi sebelumnya untuk bot musuh serta perubahan game state
- iii. Fungsi solusi : Memeriksa apakah aksi mampu menghasilkan poin
- iv. Fungsi seleksi : Memeriksa apakah aksi sesuai dengan pola yang ditangkap
- v. Fungsi kelayakan : Memeriksa apakah aksi yang akan dilakukan mampu menghasilkan poin serta optimal berdasarkan pola dari game state sebelumnya
- vi. Fungsi objektif : Mencari aksi optimal sesuai dengan pola musuh serta game state

b. Analisis Efisiensi Solusi

Bot yang menggunakan pendekatan prediction akan meninjau perubahan game state seperti letak diamond, semua bot, dan objek lainnya pada permainan. Bot juga meninjau pola gerakan dari musuh. Lalu, bot mengambil aksi yang optimal sesuai dengan hal yang telah ditinjau. Karena banyaknya hal yang menjadi faktor serta kalkulasi langkah yang optimal berdasarkan faktor-faktor tersebut, proses dalam mengambil aksi yang akan dilakukan cukup lama. Meskipun cukup lama, aksi yang dilakukan bisa cukup efisien karena bisa menyesuaikan diri dengan game state serta pola pergerakan musuh. Hal tersebut memungkinkan bot untuk mengambil langkah yang optimal untuk mengambil poin yang banyak dengan lebih konsisten meskipun berbagai perubahan pada game state juga mampu membuat langkah yang diambil kurang efisien.

c. Analisis Efektivitas Solusi

Solusi yang menggunakan pendekatan *greedy by safe* memiliki ketergantungan terhadap letak musuh di papan permainan serta letak diamond terhadap musuh.

Solusi ini dinilai efektif apabila:

- Informasi yang didapat bot cukup untuk membuat prediksi yang baik
- Tidak adanya perubahan yang besar pada game state sehingga aksi yang diambil bot cukup relevan di masa depan

Solusi ini dinilai kurang efektif apabila:

- Terdapat kurangnya informasi sehingga bot tidak mampu membuat prediksi yang optimal
- Terjadi perubahan-perubahan tertentu pada game state yang tidak memungkinkan bot untuk mengambil langkah terbaik di masa depan

3.2.5 Machine Learning

Machine Learning (ML) adalah pendekatan bot yang menggunakan data-data dari pertandingan yang digunakan untuk melatih *Machine Learning*. ML yang telah dilatih dapat menghasilkan gerakan terbaik di semua *game state* yang mungkin. Batasannya adalah jumlah data pertandingan yang dibutuhkan sangat banyak. Optimasi dan ketepatan bot juga bergantung pada tipe ML yang digunakan. ML akan dilatih dengan tujuan mendapatkan poin sebanyak mungkin dan secara konsisten.

a. Mapping Elemen Greedy

- i. Himpunan kandidat : Semua aksi yang bisa dipilih
- ii. Himpunan solusi : Aksi yang paling optimal berdasarkan data-data sebelumnya
- iii. Fungsi solusi : Memeriksa aksi yang sesuai dengan data yang digunakan
- iv. Fungsi seleksi : Memilih aksi yang optimal yang sesuai dengan data yang digunakan
- v. Fungsi kelayakan : Menentukan apakah aksi yang akan diambil sesuai dengan data yang digunakan
- vi. Fungsi objektif : Mencari aksi yang optimal yang sesuai dengan data

b. Analisis Efisiensi Solusi

Bot yang menggunakan pendekatan *Machine Learning* menganalisis data-data yang digunakan untuk melatih bot serta melakukan aksi yang optimal berdasarkan data-data tersebut. Pada proses awal dimana bot mengalami pelatihan serta pemrosesan data, proses yang dilakukan bisa sangat lama tergantung model *Machine Learning* yang digunakan, penyiapan data yang digunakan dalam proses pelatihan, serta banyaknya data yang diambil untuk melatih bot tersebut. Proses bot dalam mengambil aksi juga bisa cepat atau lama tergantung model *Machine Learning* yang digunakan serta jumlah data yang digunakan. Meskipun waktu yang digunakan bisa cukup lama, aksi yang diambil cenderung efisien karena bot sudah dilatih dengan data dengan jumlah banyak sehingga bot mampu mengambil keputusan yang baik di berbagai situasi dengan efisien.

c. Analisis Efektivitas Solusi

Solusi yang menggunakan pendekatan *Machine Learning* memiliki ketergantungan terhadap data-data yang digunakan untuk melatih bot tersebut.

Solusi ini dinilai efektif apabila:

- Data yang digunakan untuk melatih bot sesuai untuk game state yang sedang berjalan

Solusi ini dinilai kurang efektif apabila:

- Data yang digunakan untuk melatih bot tidak sesuai untuk game state yang sedang berjalan

3.3 Faktor Pengaruh Efektivitas Secara Umum

Seperti yang terlihat pada bagian eksplorasi alternatif solusi, semua strategi memiliki kelebihan dan kekurangannya masing-masing. Namun, kami juga menemukan faktor-faktor yang mempengaruhi semua bot tanpa melihat algoritma masing-masing. Berikut adalah faktor-faktor yang mempengaruhi efektivitas bot secara besar:

- ➔ Lokasi Base yang berada pada ujung-ujung dari papan permainan

Suatu gerakan keluar dari base, memerlukan jumlah gerakan yang sama untuk kembali ke base. Sumber poin utama adalah diamond yang dibawa kembali ke base. Oleh karena itu, posisi Base yang berada pada ujung papan permainan, sangat mempengaruhi jarak menuju *game object* termasuk diamond.

→ Lokasi Objective yang jauh

Mirip dengan poin sebelumnya, suatu gerakan keluar dari base, memerlukan jumlah gerakan yang sama untuk kembali ke base. Sumber poin utama adalah diamond yang dibawa kembali ke base. Semua *game object* dibuat secara acak pada papan permainan. Hal tersebut menyebabkan kemungkinan bahwa semua diamond dibuat pada lokasi yang jauh dari base pemain.

→ Diamond yang terkumpul pada satu tempat saja

Semua diamond dibuat secara acak oleh sistem. Hal ini dapat menyebabkan terbentuknya kumpulan-kumpulan diamond pada area yang kecil. Kumpulan ini menyebabkan ketika suatu bot mencapai salah satu diamond pada kumpulan tersebut, maka dia dapat mengambil semua diamond yang berada pada kumpulan tersebut dengan cepat. Sehingga lokasi terbentuknya kumpulan diamond sangat mempengaruhi jumlah diamond yang dapat diambil dari satu kali aksi keluar dari base.

→ teleporter yang menghalangi base

teleporter dibuat secara acak oleh sistem. Suatu teleporter dapat muncul di depan base (1 langkah dari base) suatu bot. Hal ini dapat mempengaruhi jarak gerakan yang diperlukan untuk keluar dan kembali ke base. Tentu teleporter yang dihasilkan dapat juga menjadi bantuan besar dalam aksi mengambil diamond, namun hal tersebut bergantung pada keberuntungan suatu bot.

Dari faktor-faktor tersebut dapat terlihat bahwa keberuntungan merupakan faktor yang mempengaruhi semua bot tanpa melihat algoritma yang digunakan. Sehingga dapat dikatakan bahwa keberuntungan merupakan faktor besar dalam memenangkan permainan Diamonds.

3.4 Strategi Greedy yang Dipilih

Strategi yang pada akhirnya digunakan adalah strategi *greedy by path*. Strategi *greedy by path* adalah strategi greedy yang melihat jarak antara bot dengan objek terdekat dengan bot. Strategi ini dinilai paling baik untuk digunakan karena strategi ini mampu pengambilan keputusan bot yang lebih fleksibel dan aman. Pengimplementasian strategi *greedy by path* memungkinkan bot untuk menjadi lebih fleksibel pada pengambilan keputusannya karena bot mengambil keputusan dengan melihat objek-objek yang terdekat dari bot. Hal tersebut memungkinkan bot untuk mengambil keputusan yang lebih optimal di berbagai situasi berbeda, terutama dengan cara kerja permainan Diamonds dimana semua posisi objek pada permainan seperti bot dan base diacak di papan permainan.

Contoh pengaplikasian strategi *greedy by path* pada bot ini dapat dilihat dengan pengambilan keputusan bot untuk mengambil diamond yang berada di posisi yang terdekat. Selain itu,

pengimplementasiannya juga dapat dilihat dengan keputusan bot untuk kembali ke base apabila inventory sudah setengah terisi untuk menyimpan diamond ketika bot berada di posisi yang dekat dengan base.

Rincian algoritma kami dari prioritas paling tinggi adalah sebagai berikut:

1. Jika waktu tersisa (dalam detik) dikurang dua, lebih kecil dari jumlah langkah atau jarak untuk kembali ke base, maka bot akan kembali ke base.
2. Jika terdapat musuh dalam jarak 1 langkah dari bot dan bot tersebut tidak berada pada basenya, maka bot akan mencoba melakukan tackle dengan maksimal coba tackle adalah 3 kali.
3. Jika inventory bot sudah penuh, maka bot akan kembali ke base
4. Jika inventory bot setengah penuh dan base berada dalam 2 langkah, maka bot akan kembali ke base
5. Jika ruang dalam inventory lebih besar dari 2, maka bot akan mengambil diamond terdekat
6. Jika inventory bot hampir penuh (terisi 4 dari maksimum 5) namun terdapat diamond biru dalam jarak 2 langkah, maka bot akan mengambil diamond tersebut
7. Jika tidak ada diamond yang ditemukan dan inventory bot tidak kosong, maka bot kembali ke base
8. Bot akan menuju diamond button.

Algoritma tersebut bukan murni *greedy by path*, namun merupakan modifikasi dari *greedy by path*. Hal ini disebabkan jika menggunakan algoritma murni *greedy by path*, maka banyak kejadian yang tidak teratasi. Namun mayoritas dari algoritma kami berdasar pada *greedy by path*.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1. Struktur Data Program

Secara keseluruhan, struktur folder untuk bot Diamonds adalah sebagai berikut:

```
src/  
├─ game/  
│   ├─ logic/  
│   │   ├─ otherBots/  
│   │   │   ├─ chase.py  
│   │   │   ├─ random.py  
│   │   │   ├─ stay.py  
│   │   │   └─ points.py  
│   │   └─ __init__.py  
│   │   └─ base.py  
│   │   └─ myBot.py  
│   └─ __init__.py  
│   └─ api.py  
│   └─ board_handler.py  
│   └─ bot_handler.py  
│   └─ models.py  
│   └─ util.py  
├─ .gitignore  
├─ decode.py  
├─ main.py  
├─ README.md  
├─ requirement.txt  
├─ run-bots.bat  
└─ run-bots.sh
```

Seperti yang terlihat pada struktur tersebut, terdapat `myBot.py` dan terdapat juga bot-bot lainnya pada folder `otherBots`. Bot `myBot` adalah bot utama yang berisi implementasi algoritma greedy yang kami pilih. Bot `chase` adalah bot yang algoritmanya adalah mengejar pemain lain, khususnya yang memiliki inventory yang paling banyak. Bot `random` adalah bot yang algoritmanya berbasis gerakan acak. Bot `stay` adalah bot yang tidak memiliki algoritma dan tidak bergerak. Bot `points` adalah bot yang algoritmanya berbasis greedy terhadap poin.

Bot `MyBot` yang merupakan hasil implementasi dari algoritma kami, terdiri dari fungsi, prosedur dan variabel sebagai berikut:

4.1.1 Variabel

Berikut adalah variabel yang terdapat pada *myBot.py*:

- **step_variation** (boolean) → Untuk gerakan zig-zag pada bot
- **step_ignore_portal** (integer) → Mengatasi gerakan setelah menggunakan teleporter
- **step_chase_enemy** (integer) → Membatasi jumlah gerakan mengejar musuh
- **current_distance_to_base** (integer) → Menyimpan jarak bot menuju base
- **current_inventory_space** (integer) → Menyimpan ruang inventory milik bot
- **distance_self_to_closest_portal** (integer) → Menyimpan jarak bot menuju portal terdekat
- **closest_portal_pair** (tuple of GameObject) → Menyimpan pasangan portal dengan salah satu portal adalah portal terdekat diantara semua portal
- **base** (GameObject) → Menyimpan *game object* base milik bot
- **objects_base** (array of GameObject) → Menyimpan kumpulan *game object* base
- **objects_portal** (array of GameObject) → Menyimpan kumpulan *game object* portal
- **object_button** (array of GameObject) → Menyimpan kumpulan *game object* button
- **red_diamonds** (array of GameObject) → Menyimpan kumpulan *game object* diamond merah
- **blue_diamonds** (array of GameObject) → Menyimpan kumpulan *game object* diamond biru
- **closest_enemy** (GameObject) → Menyimpan musuh terdekat
- **closest_diamond** (GameObject) → Menyimpan diamond terdekat

4.1.2 Prosedur

Berikut adalah prosedur yang terdapat pada *myBot.py*:

- **getGameObjects(board : Board)**
Mendapatkan semua *game object* dan menyimpan pada masing-masing koleksi mulai dari base, portal dan red button.
- **getDiamondsObject(board : Board)**
Mendapatkan semua *game object diamond* dan menyimpannya berdasarkan tipe diamond merah atau diamond biru.

4.1.3 Fungsi

Fungsi dikategorikan menjadi getter, check, movement, distance, dan fungsi utama.

4.1.3.1 Getter

- **def getBots(board: Board) → array of GameObject**
- **def getEnemyBots(this_bot: GameObject, board: Board) → array of GameObject**
- **def getBases() → array of GameObject**
- **def getHomeBaseObject(this_bot: GameObject) → GameObject**
- **def getDiamonds(board: Board) → array of GameObject**

- `def getRedDiamonds() → array of GameObject`
- `def getBlueDiamonds() → array of GameObject`
- `def getPortals() → array of GameObject: return objects_portal`
- `def getClosestPortalPair(this_bot: GameObject) → array of GameObject`
- `def getDiamondButton() → GameObject`
- `def getClosestRedDiamond(this_bot: GameObject, board: Board) → GameObject`
- `def getClosestBlueDiamond(this_bot: GameObject, board: Board) → GameObject`
- `def getClosestDiamond(this_bot: GameObject, board: Board) → GameObject`
- `def getClosestEnemy(this_bot: GameObject, board: Board) → GameObject`
- `def getUsedInventorySpace(this_bot: GameObject) → int`
- `def getEmptyInventorySpace(this_bot: GameObject) → int`
- `def getTimeRemaining(this_bot: GameObject) → int`

4.1.3.2 Checks

- `def isInventoryFull() → bool`
- `def isInventoryEmpty(this_bot: GameObject) → bool`
- `def isBotHome(bot: GameObject) → bool`

4.1.3.3 Distance

- `def distance(objectFrom: GameObject, objectTo: GameObject, board: Board) → int`
- `def distanceWithoutPortal(objectFrom: GameObject, objectTo: GameObject) → int`
- `def distanceSelfUsingPortal(objectTo: GameObject) → int`
- `def distanceUsingPortal(objectFrom: GameObject, objectTo: GameObject) → int`
- `def distanceToClosestEnemy(this_bot: GameObject, board: Board) → int`
- `def distanceToBase(this_bot: GameObject, board: Board) → int`
- `def distanceToClosestPortal(this_bot: GameObject)`

4.1.3.4 Movement

- `def moveRight() → tuple[int, int]`
- `def moveLeft() → tuple[int, int]`
- `def moveUp() → tuple[int, int]`
- `def moveDown() → tuple[int, int]`
- `def moveToObjective(this_bot: GameObject, objective: GameObject, board: Board) → tuple[int, int]`
- `def moveToBase(this_bot: GameObject, board: Board) → tuple[int, int]`

- `def moveToClosestEnemy(this_bot: GameObject, board: Board) → tuple[int, int]`
- `def moveToDiamondButton(this_bot: GameObject, board: Board) → tuple[int, int]`
- `def next_move(this_bot: GameObject, board: Board) → tuple[int, int]`

4.2 Implementasi dalam Pseudocode

Implementasi dari algoritma kami adalah sebagai berikut:

```
{MOVEMENT FUNCTION}
function moveRight() -> tuple[int, int] :
    → (1,0)
function moveLeft() -> tuple[int, int] :
    → (-1,0)
function moveUp() -> tuple[int, int] :
    → (0,1)
function moveDown() -> tuple[int, int] :
    → (0,-1)
function moveToObjective(this_bot, objective, board) -> tuple[int, int] :
    x_diff:int ← this_bot.position.x - objective.position.x
    y_diff:int ← this_bot.position.y - objective.position.y
    {After Teleport}

    if (objective.type="TeleportGameObject" and distance(this_bot, base, board)=0) then
        temp:int ← randint(1,4)
        if (temp = 1) then
            → moveUp(this_bot, board)
        elif (temp = 2) then
            → moveDown(this_bot, board)
        elif (temp = 3) then
            → moveLeft(this_bot, board)
        else then
            → moveRight(this_bot, board)

    {Go To Portal if faster}
    if (objective.type≠"TeleportGameObject" and distanceWithoutPortal(this_bot, base) >
        distanceSelfUsingPortal(base) then
        → moveToObjective(this_bot, closest_portal_pair[0], board)

    if (objective.type≠"TeleportGameObject") then
        if distance_self_to_closest_portal = 1) then
            x_portal_diff:int ← this_bot.position.x -
                closest_portal_pair[0].position.x
            y_portal_diff:int ← this_bot.position.y -
                closest_portal_pair[0].position.y

            {because portal is near make sure to avoid it}
            if (x_diff < 0 and x_portal_diff = -1) then
                step_ignore_portal ← 1
            if (y_diff < 0) then
```



```

        → moveUp()
    else then
        → moveDown()

    elif (x_diff > 0 and x_portal_diff = 1) then
        step_ignore_portal ← 2
        if (y_diff < 0) then
            → moveUp()
        else then
            → moveDown()

    elif (y_diff < 0 and y_portal_diff = -1) then
        step_ignore_portal ← 3
        if (x_diff < 0) then
            → moveRight()
        else then
            → moveLeft()

    elif (y_diff > 0 and y_portal_diff = 1) then
        step_ignore_portal ← 4
        if (x_diff < 0) then
            → moveRight()
        else then
            → moveLeft()

if (step_ignore_portal = 1) then
    step_ignore_portal ← 0
    output("IGNORE PORTAL Follow up")
    → moveRight()
elif (step_ignore_portal = 2) then
    step_ignore_portal ← 0
    output("IGNORE PORTAL Follow up")
    → moveLeft()
elif (step_ignore_portal = 3) then
    step_ignore_portal ← 0
    output("IGNORE PORTAL Follow up")
    → moveUp()
elif (step_ignore_portal = 4) then
    step_ignore_portal ← 0
    output("IGNORE PORTAL Follow up")
    → moveDown()

{Move directly toward objective}
if step_variation then
    step_variation ← False
    if (x_diff < 0) then
        → moveRight()
    elif (x_diff > 0) then
        → moveLeft()
    elif (y_diff < 0) then
        → moveUp()

```

```

        elif (y_diff > 0) then
            → moveDown()
    else then
        step_variation ← True
        if (y_diff < 0) then
            → moveUp()
        elif (y_diff > 0) then
            → moveDown()
        elif (x_diff < 0) then
            → moveRight()
        elif (x_diff > 0) then
            → moveLeft()
    return None

{Move to Object}
function moveToBase(this_bot, board) -> tuple[int, int]:
    if (distanceWithoutPortal(this_bot, base) > distanceSelfUsingPortal(base)) then
        → moveToObjective(this_bot, closest_portal_pair[0], board)
    → moveToObjective(this_bot, base, board)

function moveToClosestEnemy(this_bot: GameObject, board: Board) -> tuple[int, int]:
    → moveToObjective(this_bot, getClosestEnemy(this_bot, board), board)

function moveToDiamondButton(this_bot: GameObject, board: Board) -> tuple[int, int]
    → moveToObjective(this_bot, getDiamondButton(), board)

{MAIN FUNCTION}
function next_move (this_bot, board) :
    {bot function call by engine}
    getGameObjects(board)
    getDiamondsObject(board)

    if (not base) then
        base ← getHomeBaseObject(this_bot)

    closest_portal_pair ← getClosestPortalPair(this_bot)
    distance_self_to_closest_portal ← distanceToClosestPortal(this_bot)
    current_distance_to_base ← distanceToBase(this_bot, board)
    current_inventory_space ← getEmptyInventorySpace(this_bot)

    {If the time left is less than the distance to the base, then the bot will go to the
    base}

    if (not isInventoryEmpty(this_bot) and getTimeRemaining(this_bot) - 2 <=
    current_distance_to_base) then
        → moveToBase(this_bot, board)

    {If the Enemy is within one move, then the bot will attack it, unless the enemy is in
    it's base}
    closest_enemy ← getClosestEnemy(this_bot, board)
    if (distanceToClosestEnemy(this_bot, board)=1 and step_chase_enemy<3 and not

```

```

isBotHome(closest_enemy)) then
step_chase_enemy ← step_chase_enemy + 1
    → moveToClosestEnemy(this_bot, board)
{logic to stop chasing attacking enemy after 2 tries}
elif (step_chase_enemy = 5) then
    step_chase_enemy ← 0
elif (step_chase_enemy < 5) then
    step_chase_enemy ← step_chase_enemy + 1

{If the bot's inventory is full, it will go to the base to deposit the diamonds}
if (isInventoryFull()) then
    → moveToBase(this_bot, board)

{If the bot's inventory is half full but the base is near, deposit the diamonds}
if (current_distance_to_base <= 2 and current_inventory_space <=
(this_bot.properties.inventory_size // 2)) then
    → moveToBase(this_bot, board)

{Go to the nearest diamond if inventory space is more than equal to 2 and distance to
diamond button}
closest_diamond ← getClosestDiamond(this_bot, board)
if (closest_diamond and current_inventory_space >= 2) then
    → moveToObjective(this_bot, closest_diamond, board)

{Go to the nearest blue diamond that is within 2 moves, if the bot's inventory is not
full}
if (closest_diamond.properties.points = 2) then
    closest_diamond ← getClosestBlueDiamond(this_bot, board)
if (closest_diamond and not isInventoryFull() and closest_diamond.properties.points ==
1 and distance(this_bot, closest_diamond, board)) <= 2 then
    → moveToObjective(this_bot, closest_diamond, board)

{If no diamonds are found, and the bot's inventory is not empty, then it will go to the
base}
if (not isInventoryEmpty(this_bot)) then
    → moveToBase(this_bot, board)

{If no diamonds are found and the bot's inventory is empty, then it will go to the
diamond button}
    → moveToDiamondButton(this_bot, board)

```

4.3 Analisis dan Pengujian

Kami melakukan analisis dan pengujian dengan cara menjalankan permainan dengan memasukkan bot kami dengan bot milik tim lain yang tidak diketahui algoritmanya sebagai. Setelah kami jalankan permainannya, kami melakukan pengamatan terhadap kasus-kasus dan reaksi dari bot kami terhadap kasus tersebut.

Berikut adalah analisis dari kasus-kasus yang kami amati (bot bernama mybot1):

4.3.1 Kasus Diamond Terdekat



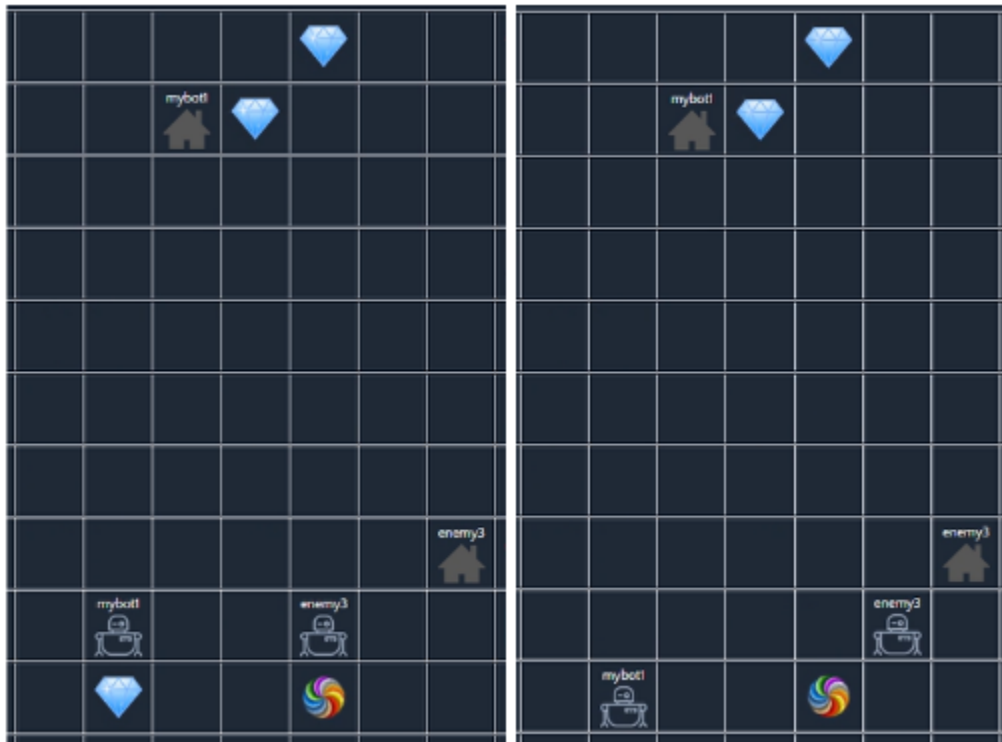
Gambar 4.3.1.1 Bot Sebelum Mengambil diamond



Gambar 4.3.1.2 Bot Setelah Mengambil diamond

Seperti yang terlihat pada gambar 4.3.1.1 dan 4.3.1.2, bot kami dengan baik melakukan aksi mengambil diamond terdekat. Dalam kasus ini terdapat 2 diamond yang dekat dengan bot. Bot melakukan rangkaian aksi berupa: Pertama, mengambil diamond yang berada di kanannya. Kedua kemudian mengambil diamond yang ada di atas kanan. Aksi tersebut sesuai dengan solusi Greedy by Path yang memprioritaskan jarak dibandingkan poin.

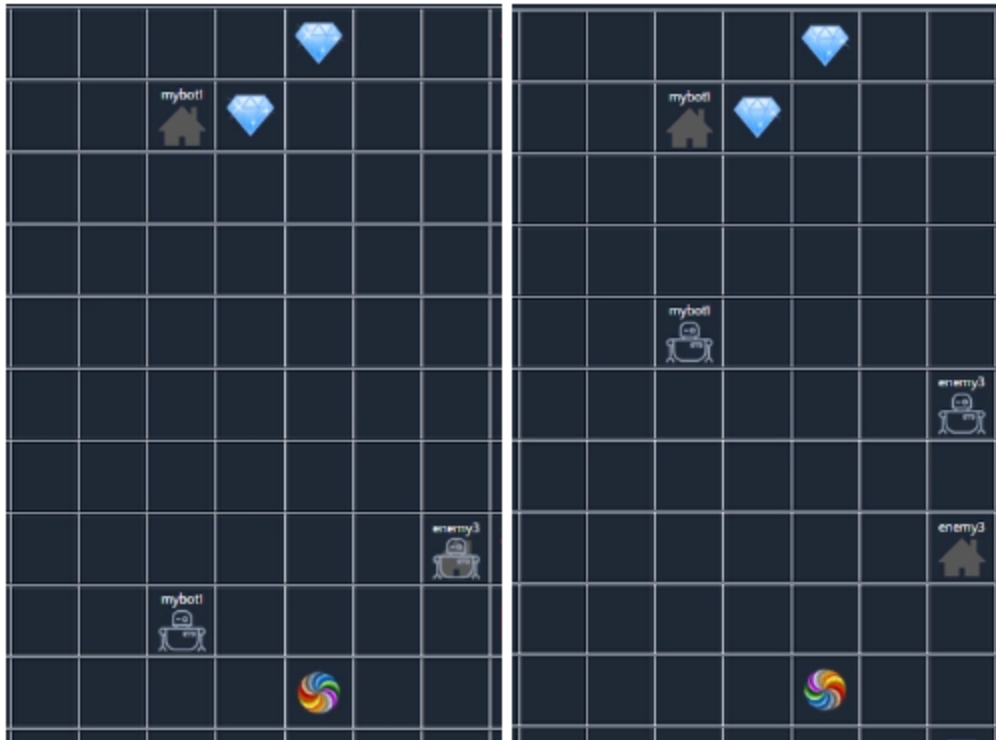
4.3.2 Kasus Inventory Penuh atau Hampir Penuh



Gambar 4.3.2.1 Bot Sebelum Inventory Penuh/Hampir Penuh

Name	Diamonds	Score	Time
enemy1	💎	3	26s
mybot1	💎💎💎💎	1	26s
enemy3	💎💎💎	4	26s
enemy2		5	26s

Gambar 4.3.2.2 Inventory Penuh/Hampir Penuh



Gambar 4.3.2.3 Gerak Bot Ketika Inventory Penuh/Hampir Penuh

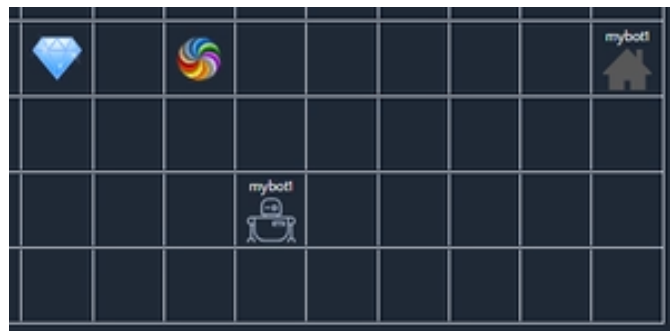
Inventory yang penuh saat kondisi normal yaitu maximum inventory adalah 5. Seperti yang terlihat pada gambar 4.3.2.1 Inventory Bot masih berisi 3 diamond dari maximum 5. Setelah mengambil diamond biru, terlihat pada gambar 4.3.2.2 bahwa inventory bot menjadi 4 dari 5, yang artinya inventori hampir penuh. Hal tersebut menyebabkan bot untuk kembali ke base, seperti algoritma yang kami buat. Terlihat pada gambar 4.3.2.3 bot telah mendeteksi inventory yang hampir penuh sehingga kembali ke base.

Pada Algoritma kami jika dalam perjalanan terdapat diamond dalam jarak 2 langkah dan inventory masih cukup untuk menyimpan, maka bot kami akan mengambil diamond tersebut. Ketika Inventori bot penuh maka, bot tidak akan mencoba mengambil diamond yang lewat, melainkan melakukan aksi kembali ke base.

4.3.3 Kasus Tackle Musuh






Gambar 4.3.3.1 Bot Sebelum Melakukan Tackle



Gambar 4.3.3.2 Bot Setelah Melakukan Tackle

Name	Diamonds	Score	Time
enemy3		1	54s
enemy2		0	54s
mybot1		0	54s
enemy1		0	54s

Gambar 4.3.3.3 Inventory Sebelum Tackle

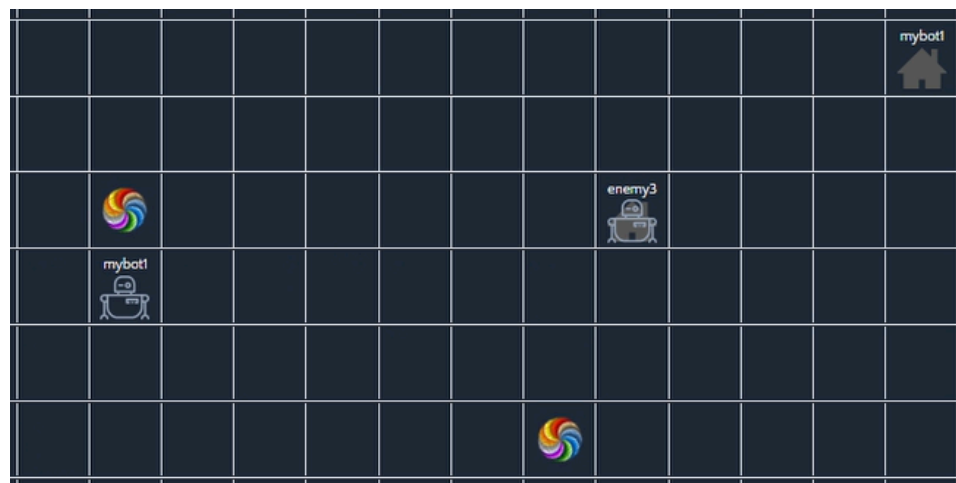
Name	Diamonds	Score	Time
enemy3		1	53s
enemy2		0	53s
mybot1		0	53s
enemy1		0	53s

Gambar 4.3.3.4 Inventory Setelah Tackle

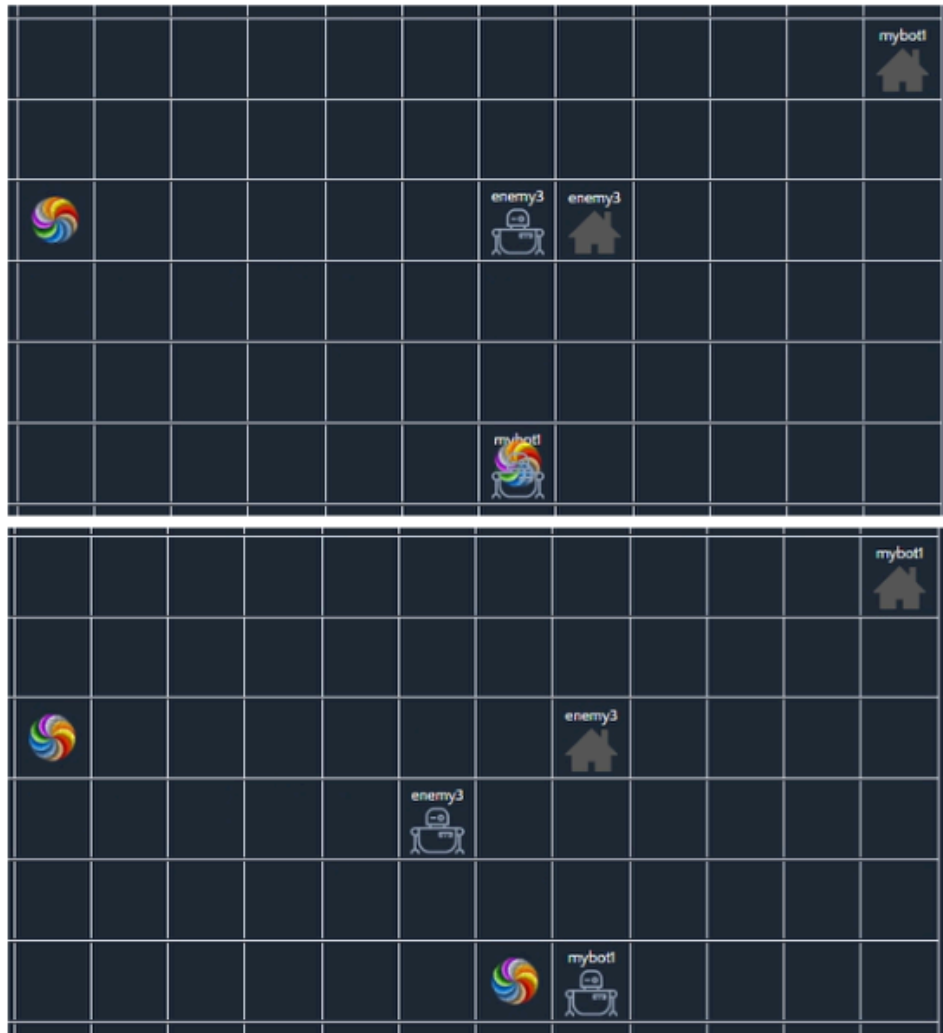
Pada Gambar 4.3.3.1, terlihat kedua bot sedang mengincar diamond. Ketika bot enemy1 berhasil mendapatkan diamond, posisi bot kami dengan musuh adalah 1 langkah yang memungkinkan untuk dilakukannya tackle. Seperti yang terlihat pada gambar 4.3.3.2, mybot1 berhasil melakukan tackle terhadap musuh dan mengambil diamond miliknya.

Pada Gambar 4.3.3.3, perhatikan bahwa inventory enemy1 berisi 2 diamond dan inventory mybot1 berisi 1 diamond. Setelah tackle dapat dilihat pada gambar 4.3.3.4 bahwa mybot1 mengambil semua diamond milik enemy1 dan enemy1 kembali hidup di basenya. Aksi ini sesuai dengan algoritma kami. Tujuan dari aksi ini adalah sebagai cara melindungi diri dari tackle musuh. Kami membuat algoritma jika jarak dari bot adalah 1 langkah maka kita akan melakukan tackle, namun jika tidak maka kita tidak akan pernah mencoba mengincar bot musuh.

4.3.4 Kasus Gunakan Teleporter



Gambar 4.3.4.1 Bot Sebelum Menggunakan Teleporter



Gambar 4.3.4.2 Bot Setelah Menggunakan Teleporter

Seperti yang terlihat pada gambar 4.3.4.1 dan 4.3.4.2, Bot dapat menggunakan teleporter. Dalam contoh tersebut dapat dilihat bot sedang menuju base, namun dengan menggunakan teleporter maka bot akan tiba lebih cepat.

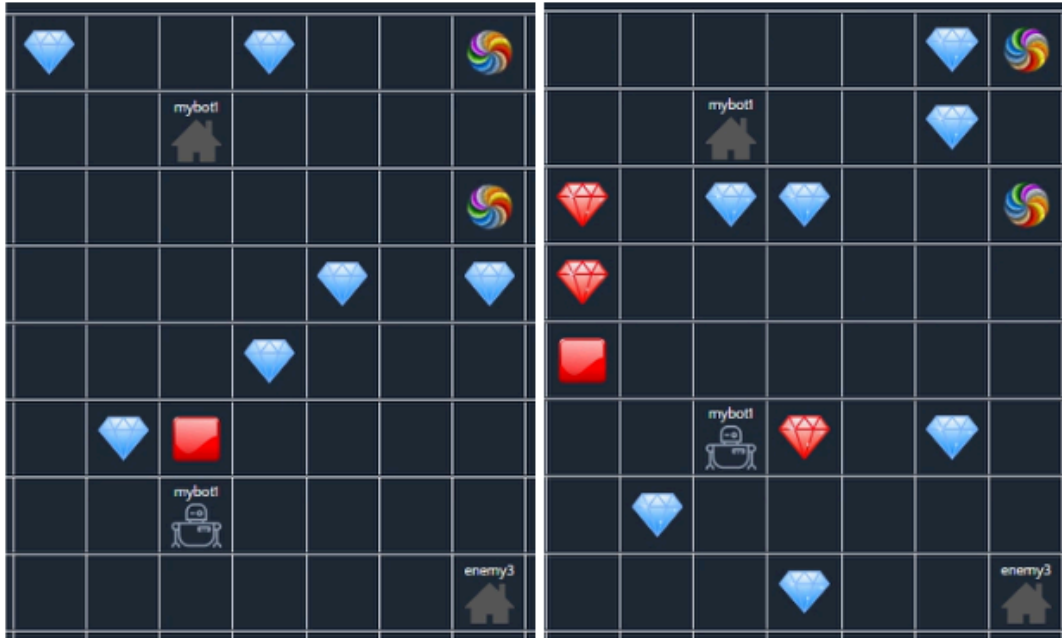
4.3.5 Kasus Waktu Akan Habis



Gambar 4.3.5.1 Bot Sebelum Deteksi Waktu Akan Habis

Name	Diamonds	Score	Time
enemy1		8	8s
mybot1	💎💎	5	8s
enemy3		9	8s
enemy2	💎💎	9	8s

Gambar 4.3.5.2 Waktu Sebelum Deteksi Waktu Akan Habis



Gambar 4.3.5.3 Bot Setelah Deteksi Waktu Akan Habis

Name	Diamonds	Score	Time
enemy1		8	1s
mybot1		8	1s
enemy3	💎💎	9	1s
enemy2	💎💎💎	9	1s

Gambar 4.3.5.4 Bot Sesaat Waktu Akan Habis

Seperti yang terlihat pada gambar 4.3.5.1 dan 4.3.5.2, Bot belum mendeteksi bahwa waktu akan habis, namun Inventory bot masih berisi. Pada algoritma kami, jika terjadi kasus seperti ini, maka bot akan kembali ke base. Hal ini dikarenakan jika melakukan aksi lain, maka bot tidak memiliki waktu yang cukup untuk kembali ke base dan menyimpan diamond yang sudah diambilnya.

Ketika waktu menunjukkan tujuh detik dan jarak gerakan bot menuju rumah adalah 5 langkah, bot mendeteksi bahwa waktu akan habis sehingga dia melakukan aksi untuk kembali ke base. Dengan syarat jumlah detik - 2 lebih kecil sama dengan jarak ke base, maka bot akan selalu kembali ke base jika waktu akan habis. Hasilnya dapat dilihat pada gambar 4.3.5.3 yang menunjukkan aksi kembali ke base oleh bot. Pada gambar 4.3.5.4 terlihat bahwa mybot1 berhasil menyimpan diamond yang dibawanya ke base dengan sisa waktu 1 detik.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dengan ini, kami berhasil membuat strategi bot untuk permainan Diamonds dengan pendekatan algoritma greedy. Dari tugas ini dapat ditarik kesimpulan bahwa algoritma greedy dapat menjadi alternatif strategi pemecahan masalah yang tepat dalam menentukan solusi optimum dari kondisi sekarang (maksimum lokal), walaupun solusi yang berupa maksimum lokal belum tentu merupakan solusi yang paling tepat (maksimum global). Namun, solusi yang dihasilkan algoritma greedy memiliki optimasi yang cukup baik dalam *decision making* seperti pada kasus bot di permainan Diamonds, sebab dalam kebanyakan kasus, solusi yang dihasilkan oleh algoritma *greedy* cukup mendekati solusi maksimum global.

5.2 Saran

Saran untuk kelompok ini diantaranya :

1. Membuat *timeline* pengerjaan tugas yang lebih teratur
2. Memulai pengerjaan tugas lebih awal
3. Memperbanyak diskusi selama pengerjaan tugas
4. Mendalami pengetahuan tentang algoritma greedy dan *game engine*
5. Memperbanyak eksplorasi mengenai ilmu yang dibutuhkan dalam mengerjakan tugas

5.3 Refleksi

Refleksi yang kami dapatkan dari tugas ini antara lain adalah untuk meningkatkan kinerja pengerjaan tugas besar dengan cara pembagian tugas yang lebih cepat dan lebih baik sehingga pembagian waktu kerja masing-masing anggota lebih merata. Kami juga merasa kurangnya komunikasi antar anggota, mengganggu kelancaran dari pengerjaan tugas ini. Hal yang kami dapatkan dari tugas ini adalah pengalaman penting dalam pengembangan bot dan perancangan algoritma yang paling optimal.

LAMPIRAN

Link Repository: https://github.com/NoHaitch/Tubes1_SanssKuy

Link Video: <https://www.youtube.com/watch?v=dpHv0tuuF3M>

Link Game Engine: <https://github.com/haziqam/tubes1-IF2211-game-engine>

Link Bot Starter Pack: <https://github.com/haziqam/tubes1-IF2211-bot-starter-pack>

DAFTAR PUSTAKA

Rinaldi Munir. Algoritma Greedy (Bagian 1). Diakses pada 8 Maret 2024 dari
[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)

Rinaldi Munir. Algoritma Greedy (Bagian 2). Diakses pada 9 Maret 2024 dari
[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)

Rinaldi Munir. Algoritma Greedy (Bagian 3). Diakses pada 9 Maret 2024 dari
[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)

Geeksforgeeks. Greedy Algorithms. Diakses pada 8 Maret 2024 dari
<https://www.geeksforgeeks.org/greedy-algorithms/>