IF2211-53 STRATEGI ALGORITMA

PEMANFAATAN ALGORITMA GREEDY DALAM PENGEMBANGAN BOT PERMAINAN “DIAMONDS”

Laporan Tugas Besar 1

Kelompok: Reela Indo Indo Indo

Oleh :
Jimly Nur Arif (13522123)
Yosef Rafael Joshua (13522133)
Rayhan Ridhar Rahman (13522160)

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024

Daftar Isi

BAB I PENDAHULUAN

Diamonds merupakan suatu *programming challenge* yang mempertandingkan bot yang anda buat dengan bot dari para pemain lainnya. Setiap pemain akan memiliki sebuah bot dimana tujuan dari bot ini adalah mengumpulkan *diamond* sebanyak-banyaknya. Cara mengumpulkan *diamond* tersebut tidak akan sesederhana itu, tentunya akan terdapat berbagai rintangan yang akan membuat permainan ini menjadi lebih seru dan kompleks. Untuk memenangkan pertandingan, setiap pemain harus mengimplementasikan strategi tertentu pada masing-masing bot-nya. Penjelasan lebih lanjut mengenai aturan permainan akan dijelaskan di bawah.

Pada tugas besar pertama Strategi Algoritma ini, kami diminta untuk membuat sebuah bot yang nantinya akan dipertandingkan satu sama lain. Tentunya dalam pengembangan bot, kami harus mengimplementasikan strategi greedy.

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. 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.

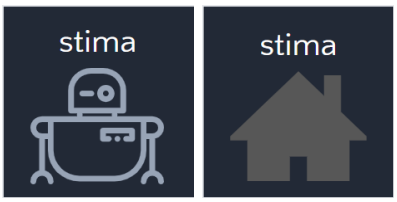
2. Red Button/Diamond Button

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

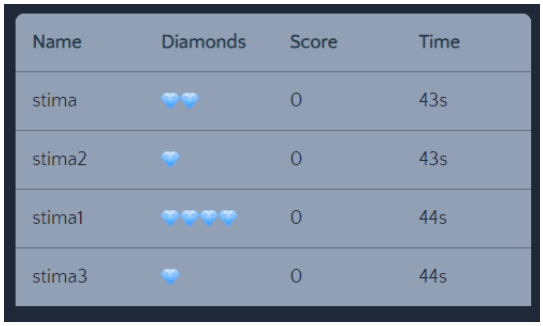
Terdapat 2 teleporter yang saling terhubung satu sama lain. Jika bot melewati sebuah teleporter maka bot akan berpindah menuju posisi teleporter yang lain.

4. 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.

Berikut adalah alur permainan Diamonds secara umum.

1. Pertama, setiap pemain (bot) akan ditempatkan pada board secara random. Masing-masing bot akan mempunyai home base, serta memiliki score dan inventory awal bernilai nol.
2. Setiap bot diberikan waktu untuk bergerak, waktu yang diberikan semua sama untuk setiap pemain.
3. Objektif utama bot adalah mengambil diamond-diamond yang ada di peta sebanyak-banyaknya. Seperti yang sudah disebutkan di atas, diamond yang berwarna merah memiliki 2 poin dan diamond yang berwarna biru memiliki 1 poin.
4. Setiap bot juga memiliki sebuah inventory, dimana inventory berfungsi sebagai tempat penyimpanan sementara diamond yang telah diambil. Inventory ini sewaktu-waktu bisa penuh, maka dari itu bot harus segera kembali ke home base.
5. Apabila bot menuju ke posisi home base, score bot akan bertambah senilai diamond yang tersimpan pada inventory dan inventory bot akan menjadi kosong kembali.
6. Usahakan agar bot anda tidak bertemu dengan bot lawan. Jika bot A menimpa posisi bot B, bot B akan dikirim ke home base dan semua diamond pada inventory bot B akan hilang, diambil masuk ke inventory bot A (istilahnya tackle).
7. Selain itu, terdapat beberapa fitur tambahan seperti teleporter dan red button yang dapat digunakan apabila anda menuju posisi objek tersebut.
8. Apabila waktu seluruh bot telah berakhir, maka permainan berakhir. Score masing-masing pemain akan ditampilkan pada tabel Final Score di sisi kanan layar.

BAB II LANDASAN TEORI

2.1. Algoritma Greedy

Algoritma Greedy adalah algoritma langkah per langkah yang memilih pilihan terbaik saat ini dengan harapan akan menghasilkan solusi optimal untuk permasalahan yang sedang diselesaikan. Di setiap langkah kita akan memilih keputusan yang paling baik saat ini berdasarkan informasi yang kita miliki, dan berharap keputusan terbaik yang dibuat di setiap langkah akan memberikan solusi optimal. Algoritma Greedy sangat populer dan sering digunakan di berbagai masalah seperti knapsack problem, minimum spanning tree, dan shortest path.

Algoritma Greedy tidak melakukan backtracking. Ini berarti, berbeda dengan algoritma Brute Force dan Exhaustive Search, hasil yang diperoleh dari algoritma Greedy bisa saja bukan merupakan solusi terbaik. Hal ini karena algoritma Greedy tidak memeriksa setiap kemungkinan yang bisa terjadi seperti yang dilakukan pada algoritma Brute Force. Greedy hanya akan memilih keputusan terbaik di setiap langkah. Sehingga bisa saja keputusan terbaik di suatu langkah (local) adalah keputusan yang kurang baik secara keseluruhan (global).Namun hasil dari algoritma Greedy merupakan hampiran (approximation) solusi terbaik, sehingga dapat dipakai jika solusi terbaik tidak terlalu penting dalam masalah yang sedang diselesaikan.

Keuntungan yang diberikan algoritma Greedy adalah waktu yang lebih cepat dalam memberikan solusi dibandingkan dengan algoritma lain seperti Brute Force. Hal ini karena Greedy tidak memeriksa semua solusi, tetapi hanya memilih keputusan terbaik di setiap langkah. Dengan begitu Greedy dapat memberikan waktu yang relatif lebih cepat. Keuntungan lainnya yang diberikan algoritma Greedy adalah hampiran (approximation) solusi terbaik. Meskipun hanya berupa hampiran, tapi sering kali hampiran tersebut sudah layak dipakai sebagai solusi dari masalah yang sedang diselesaikan. Keuntungan berikutnya dari algoritma Greedy adalah langkah-langkahnya yang mudah dipahami dan intuitif.

Secara keseluruhan, algoritma Greedy adalah algoritma yang relatif cepat serta memberikan solusi berupa hampiran dari solusi terbaik untuk suatu masalah. Oleh karena itu algoritma Greedy cocok dipakai ketika kita menginginkan solusi yang cepat walaupun bukan solusi terbaik. Dalam tugas kali ini kami akan menerapkan algoritma Greedy dalam mengembangkan bot untuk permainan Diamonds. Hasil yang kami harapkan adalah bot yang dapat melakukan semua aksinya dengan efektif dan efisien.

2.2. Elemen Algoritma Greedy

Algoritma Greedy memiliki beberapa elemen umum dalam penerapannya. Elemen-elemen tersebut adalah sebagai berikut.

1. Himpunan kandidat (C)
   Berisi kandidat yang akan dipilih pada setiap langkah
2. Himpunan solusi
   Berisi kandidat yang sudah dipilih
3. Fungsi solusi:
   Sebuah fungsi yang menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi
4. Fungsi seleksi:
   Fungsi yang memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy dapat bersifat heuristik.
5. Fungsi kelayakan:
   Fungsi yang memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak sebagai solusi)
6. Fungsi obyektif:
7. Fungsi yang memaksimumkan atau meminimumkan

2.3. Cara Kerja Program

Permainan “Diamonds” memiliki beberapa prasyarat agar bisa dijalankan dengan baik. Berikut prasyarat yang harus diinstal sebelum menjalankan permainan “Diamonds” di komputer.

1. Node.js
2. Docker desktop
3. Yarn
4. Game engine
5. Bot starter pack
6. Python
7. IDE (seperti visual studio code)

Pastikan semua prasyarat sudah diinstal sebelum menjalankan program

2.3.1. Menjalankan Program

Sebelum menjalankan program dapat dilakukan konfigurasi berapa jumlah bot yang ingin dipasang dan informasi mengenai bot tersebut.

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

```
@echo off
start cmd /c "python main.py --logic Logic1 --email=test@email.com
--name=stima --password=123456 --team etimo"
start cmd /c "python main.py --logic Random --email=test1@email.com
--name=stima1 --password=123456 --team etimo"
start cmd /c "python main.py --logic Random --email=test2@email.com
--name=stima2 --password=123456 --team etimo"
start cmd /c "python main.py --logic Random --email=test3@email.com
--name=stima3 --password=123456 --team etimo"
```

Untuk menambahkan jumlah bot yang akan dimainkan maka file run-bots.bat dapat diubah dengan menambahkan bot beserta dengan informasi logic, email, nama, password, dan juga tim bot tersebut.

```
// file run-bots.sh

#!/bin/bash

python main.py --logic Random --email=test@email.com --name=stima
--password=123456 --team etimo &
python main.py --logic Random --email=test1@email.com --name=stima1
--password=123456 --team etimo &
python main.py --logic Random --email=test2@email.com --name=stima2
--password=123456 --team etimo &
python main.py --logic Random --email=test3@email.com --name=stima3
--password=123456 --team etimo &
```

File run-bots.sh juga dapat dikonfigurasi dengan menambahkan bot dan informasi mengenai bot tersebut seperti pada file run-bots.bat

Setelah konfigurasi selesai, program dapat dijalankan dengan menjalankan perintah “./run-bots.bat”

2.3.2. Mengembangkan Bot

Pengembangan bot baru dapat dilakukan di folder /game/logic. Di direktori tersebut sudah ada file bernama “random.py” yang berisi logic template untuk sebuah bot. Untuk mengembangkan bot, yang harus dilakukan adalah

menambah logika pada file tersebut atau bisa juga dengan membuat file baru di direktori yang sama, menyalin dari “random.py” dan menambahkan logic untuk bot. Tersedia file “random.py” yang berasal dari kit bot starter-pack v1.0.1 yang dapat digunakan sebagai dasar logika yang nanti dikembangkan.

```python
# file random.py
from random import random
from typing import Optional

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


class RandomLogic(BaseLogic):
    def __init__(self):
        self.directions = [(1, 0), (0, 1), (-1, 0), (0, -1)]
        self.goal_position: Optional[Position] = None
        self.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
            self.goal_position = base
        else:
            # Just roam around
            self.goal_position = None

        current_position = board_bot.position
        if self.goal_position:
            # We are aiming for a specific position, calculate delta
            delta_x, delta_y = get_direction(
                current_position.x,
                current_position.y,
                self.goal_position.x,
```

```
                self.goal_position.y,
            )
        else:
            # Roam around
            delta = self.directions[self.current_direction]
            delta_x = delta[0]
            delta_y = delta[1]
            if random.random() > 0.6:
                self.current_direction = (self.current_direction + 1) %
len(
                    self.directions
                )
        return delta_x, delta_y
```

Pada file tersebut sudah ada beberapa fungsi yang sudah disediakan dari awalnya, yaitu fungsi **_init_** dan fungsi **next_move**. Untuk mengembangkan bot, yang dapat dilakukan adalah mengubah fungsi **next_move** dan menambah fungsi-fungsi lain di file tersebut.

BAB III APLIKASI STRATEGI “GREEDY”

3.1. Alternatif Solusi

Dalam mengembangkan bot untuk permainan “Diamonds” kami menemukan beberapa alternatif solusi Greedy yang menarik dan layak diimplementasikan. Solusi-solusi alternatif yang kami temukan sebagai berikut.

3.1.1. Greedy by Distance

Greedy by Distance adalah penerapan algoritma Greedy yang lebih memprioritaskan pengambilan diamond terdekat terlebih dahulu. Jadi pada dasarnya bot akan mengevaluasi semua diamond yang ada dan mencari yang terdekat. Setelah ditemukan diamond terdekat maka bot akan pergi ke lokasi tempat diamond itu berada dan mengambilnya.

a. Mapping Elemen Algoritma Greedy
   - Himpunan Kandidat: Semua diamonds yang tersedia di board
   - Himpunan Solusi: Diamond yang terdekat
   - Fungsi Solusi: Memeriksa apakah diamond yang dipilih memiliki jarak di dalam radius base
   - Fungsi Seleksi: Memilih diamond dengan jarak terdekat dari base
   - Fungsi Kelayakan: Memeriksa apakah bot dapat melakukan langkah-langkah untuk mendapatkan diamond
   - Fungsi Obyektif: Mencari jarak terdekat ke sasaran (diamond) dari titik asal (base)

b. Analisis Efisiensi

   Strategi ini memprioritaskan pengambilan diamond terdekat dari base. Untuk menemukan diamond terdekat maka dilakukan linear search terhadap koordinat masing-masing diamond untuk menemukan diamond terdekat. Kompleksitas dari proses ini adalah

$$T(n) = O(n)$$ **dengan n adalah banyaknya elemen**

c. Analisis Efektivitas

   Strategi ini kami buat karena tujuan utama dari permainan “Diamonds” adalah mengumpulkan diamond sebanyak-banyaknya. Oleh karena itu salah satu cara adalah dengan mengumpulkan diamond yang terdekat dengan base sehingga mudah untuk diamankan di base.

   Efektif dalam situasi:
   - Banyak diamond yang jaraknya dekat dengan base
   - Tidak banyak bot lain di sekitar base

Tidak efektif dalam situasi:

- Diamond spawn berjauhan dari base
- Bot lain memprioritaskan aksi tabrak dan dekat dengan base

3.1.2. Greedy by Weight

Greedy by Weight adalah penerapan algoritma Greedy yang lebih memprioritaskan pengambilan diamond dengan nilai tertinggi. Di permainan ini terdapat diamond merah dan diamond biru. Dengan pendekatan ini maka bot akan memprioritaskan mengambil diamond merah sebanyak-banyaknya.

a. Mapping Elemen Algoritma Greedy
   - Himpunan Kandidat: Semua diamond merah yang tersedia di board
   - Himpunan Solusi: Diamond merah yang terpilih
   - Fungsi Solusi: Memeriksa apakah diamond tersebut lebih dekat
   - Fungsi Seleksi: Memilih diamond merah dengan jarak terdekat dengan posisi saat ini (base atau bot).
   - Fungsi Kelayakan: Memeriksa apakah bot dapat melakukan langkah-langkah untuk mengambil diamond merah (inventory)
   - Fungsi Objektif: Mencari jarak dan langkah terdekat yang harus diambil bot untuk mengambil diamond merah

b. Analisis Efisiensi

   Strategi ini memprioritaskan pengambilan diamond dengan nilai tertinggi (diamond merah) dan terdekat. Untuk menemukan koordinat diamond merah, digunakan linear search. Kompleksitas proses ini adalah:

$$T(n) = O(n)_{\text{dengan n adalah banyaknya elemen}}$$

c. Analisis Efektivitas

   Strategi ini kami buat karena tujuan utama permainan “Diamonds” adalah mengumpulkan poin sebanyak-banyaknya. Salah satu caranya adalah mengambil diamond yang paling besar nilainya (diamond merah).

   Efektif dalam situasi:

   - Lebih banyak diamond merah yang spawn
   - Tidak banyak bot lain di sekitar base

   Tidak efektif dalam situasi:

   - Bot lain memprioritaskan aksi tabrak
   - Ada sekumpulan diamond biru yang sedikit lebih jauh dari diamond merah

3.1.3. Greedy by Tackle

Greedy by Tackle adalah penerapan algoritma Greedy yang lebih memprioritaskan mendapatkan poin dengan memanfaatkan aksi tabrak (tackle). Dengan pendekatan ini bot akan fokus untuk tabrak bot lain daripada mengambil diamond.

a. Mapping Elemen Algoritma Greedy
   - Himpunan Kandidat: Semua bot selain bot diri sendiri
   - Himpunan Solusi: Bot yang terpilih
   - Fungsi Solusi: Memeriksa apakah bot lain dapat dikenakan aksi tabrak oleh bot kita
   - Fungsi Seleksi: Memilih bot lain dengan inventory paling besar (membawa diamond paling banyak)
   - Fungsi Kelayakan: Memeriksa apakah bot dapat melakukan aksi tabrak
   - Fungsi Objektif: Mencari langkah yang dapat memberikan diamond terbanyak

b. Analisis Efisiensi

   Strategi ini memprioritaskan pengambilan diamond dengan nilai tertinggi (diamond merah) dan terdekat. Untuk menemukan koordinat diamond merah, digunakan linear search. Kompleksitas proses ini adalah:

$$T(n) = O(n)$$ **dengan n adalah banyaknya elemen**

c. Analisis Efektivitas

   Strategi ini kami buat karena cara lain untuk mendapatkan diamond adalah dengan “mencuri” dari bot lain. Hal ini dicapai dengan aksi tabrak yang dapat dilakukan oleh bot. Ketika bot kita menabrak bot lain maka semua diamond yang sedang dibawa bot lain akan ditransfer ke inventory bot kita secara otomatis.

   Efektif dalam situasi:
   - Banyak bot lain di sekitar bot kita
   - Bot kita sedang tidak memiliki diamond dalam inventory

   Tidak efektif dalam situasi:
   - Ada bot lain di dekat kita yang juga menerapkan aksi tabrak
   - Semua bot menerapkan aksi tabrak

3.2. Strategi Greedy yang Terpilih

Berdasarkan pertimbangan kami memilih untuk menggunakan kombinasi dari strategi Greedy by Distance, Greedy by Weight, dan Greedy by Tackle. Setiap strategi memiliki keunggulan dan kelemahannya masing-masing. Namun karena banyak sekali strategi yang dapat dipakai oleh lawan, maka kami memutuskan untuk menggunakan kombinasi dari tiga strategi agar bot kami lebih fleksibel dalam menghadapi berbagai situasi saat permainan berjalan.

Pada bot yang kami buat, kami memutuskan untuk membuat logika bot dengan prioritas tertinggi hingga terendah sebagai berikut.

1. Mengambil diamond (Greedy by Distance)
2. Mengambil diamond merah jika harus memilih antara diamond biru dan diamond merah (Greedy by Weight)
3. Melakukan tackle jika bot musuh berpapasan dengan bot kita (Greedy by Tackle)

Prioritas tertinggi yang akan dipertimbangkan bot kami adalah mengambil diamond yang paling dekat dengan base. Hal ini kami lakukan untuk memaksimalkan perolehan diamond kami. Dengan adanya tombol reset object-object dalam board akan sering berganti. Oleh karena itu kami memaksimalkan jumlah diamond yang kami dapat dengan mengambil diamond terdekat sebanyak-banyaknya kemudian kembali ke base.

Prioritas kedua dari logika bot kami adalah mengambil diamond merah jika memungkinkan. Untuk memaksimalkan poin, maka jika ada dua diamond dengan jarak yang sama terhadap base dan salah satunya adalah diamond merah bot akan mengambil diamond yang merah. Dengan begitu inventory akan penuh lebih cepat dan bot dapat pulang lebih sering.

Prioritas terakhir adalah untuk melakukan tackle bot lain jika memungkinkan. Bot kami akan melakukan tackle kepada bot lain hanya jika bot lain berjarak satu langkah dari posisi bot kami (berpapasan). Kami melakukan ini untuk mencapai dua hal sekaligus. Yang pertama adalah melindungi diri sendiri karena jika ada bot lawan yang berpapasan maka ada kemungkinan bot lawan tersebut akan melakukan aksi tabrak kepada bot kami. Yang kedua adalah untuk “mencuri” diamond lawan. Jika bot lawan tidak sengaja berpapasan dengan bot kami, maka bot kami akan melakukan tackle dan mencuri diamond lawan jika inventory tidak penuh.

BAB IV IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi dalam Pseudocode

GreedyTomerry init

```
class TomerryLogic(BaseLogic)
    __init__: {melakukan inisialisasi objek bot Greedy Tomerry}
        self.goal_position: Position <- NULL {Posisi yang dituju}
        self.goal: GameObject <- NULL {Objek yang dituju}
        self.portals: List of Position <- [] {List kedua portal}
        self.wrath: bool <- true {bool untuk melakukan cooldown tackle, supaya
tidak terus mengejar jika gagal}
```

next_move function

```
function next_move(board_bot: GameObject, board: Board) -> integer, integer:
    props <- board_bot.properties
    current_position <- board_bot.position
    base <- board_bot.properties.base

    {Fungsi atau prosedur yang ada dalam kelas diberi self.}
    self.update_portals(board, current_position)

    if props.diamonds = 5 or (board_bot.properties.milliseconds_left <
(board.minimum_delay_between_moves + 500) * self.distance(current_position,
base) and  props.diamonds > 0)then
        self.goal_position <- base
    else if not self.goal in board.game_objects or props.diamonds = 4 or
(self.goal_position.x = current_position.x and self.goal_position.y =
current_position.y) or current_position in self.portals then
        if len(self.objects_near_coords(base, board, "DiamondGameObject")) < 1
then
            self.goal <- self.square(board, current_position)
            self.goal_position <- self.goal.position
        else
            self.goal <- self.nearest_diamond(board, current_position, 5 -
props.diamonds)
            self.goal_position <- self.goal.position
        endif
        if not (base.x = current_position.x and base.y = current_position.y)
and props.diamonds > 0 then
            self.triangle(board_bot)
    endif
```

```
    if self.warp_train(current_position)then
        delta_x, delta_y <- get_bot_direction(
            current_position.x,
            current_position.y,
            self.portals[0].x,
            self.portals[0].y,
        )
    else
        delta_x, delta_y <- self.lets_play(current_position, board)
    endif
    -> delta_x, delta_y
```

update_portals procedure

```
procedure update_portals(input board: Board, input current_position:
Position) -> None:
        self.portals <- [p.position for p in board.game_objects if p.type =
"TeleportGameObject"] {mendapatkan portal dari list game object milik board}
        if self.distance(current_position, self.portals[0]) >
self.distance(current_position, self.portals[1])then
        {Memastikan portal dengan indeks paling awal yang terdekat dengan
lokasi bot}
            self.portals[0], self.portals[1] <- self.portals[1],
self.portals[0]
        endif
```

lets_play function

```
function lets_play(current_position: Position, board: Board) -> integer,
integer:
{menentukan arah gerak bot selanjutnya}
    adjacent <- [obj for obj in board.game_objects if
self.distance(current_position, obj.position) = 1 and not
(obj.type="DiamondGameObject" or obj.type="DiamondButtonGameObject" or
obj.type="BaseGameObject")] {mendapat objek dengan tipe sesua}

    for i in adjacent do
        if i.properties.can_tackle and self.wrath then {Jika terdapat bot
lain, coba untuk tackle}
            self.wrath <- false
            -> get_bot_direction(
                current_position.x,
                current_position.y,
                i.position.x,
```

```
                i.position.y,
            )
        endif
    endfor
    self.wrath <- true
    -> get_bot_direction(
        current_position.x,
        current_position.y,
        self.goal_position.x,
        self.goal_position.y,
    )
```

Distance-related functions

```
function warp_train(current_position: Position) -> bool:
{Mendapatkan boolean untuk menggunakan portal atau tidak}
    -> self.distance(current_position, self.goal_position) >
self.distance(current_position, self.portals[0]) +
self.distance(self.goal_position, self.portals[1]) and not current_position in
self.portals

function objects_near_coords(coords: Position, board: Board, object_type:
str) -> List of GameObject:
{Mengembalikan list berisi objek dengan tipe tertentu dari posisi tertentu}
    -> [d for d in board.game_objects if d.type = object_type and
absolute(d.position.x - coords.x) + absolute(d.position.y - coords.y) < 7]

function distance(object: Position, target: Position) -> integer:
{Mengembalikan jarak antara dua posisi (baru dibuat setelah fungsi sebelum)}
    -> absolute(target.x - object.x) + absolute(target.y - object.y)

function nearest_diamond(self, board: Board, current_position: Position,
point_cap: integer) -> GameObject:
{Mengembalikan objek diamond terdekat}
    distance <- 10000
    temp <- None
    tpoints <- 0

    for d in board.diamonds do
        if d.properties.points <= point_cap then
            compared_distance <- min(self.distance(d.position,
current_position), self.distance(self.portals[0], current_position) +
self.distance(self.portals[1], d.position))

            if compared_distance < distance or (compared_distance <= distance
and d.properties.points > tpoints) do
```

```
                distance <- compared_distance
                tpoints <- d.properties.points
                temp <- d
            endif
        endif
    -> temp

procedure triangle(input board_bot: GameObject) -> None:
{Menetapkan jarak yang mungkin ditempuh atau ke base dahulu}
    current_position <- board_bot.position
    base <- board_bot.properties.base

    min_distance <- min(self.distance(self.goal_position, current_position),
self.distance(self.portals[0], current_position) +
self.distance(self.portals[1], self.goal_position))
    if min_distance * 2 > self.distance(self.goal_position, base) +
self.distance(base, current_position) then
        self.goal <- base
        self.goal_position <- base
    endif

function square(board: Board, current_position: Position) ->
Optional[GameObject]:
    for p in board.game_objects do
        if p.type = "DiamondButtonGameObject" then
            button <- p
            break
        endif
    endfor

    for d in board.diamonds do
        if in_between(d.position, button.position, current_position):
            -> d
        endif
    endfor
    -> button
```

Out of class functions

```
function in_between(checked: Position, onep: Position, twop: Position) ->
boolean:
    -> checked.x >= min(onep.x, twop.x) and checked.x <= max(onep.x, twop.x)
and checked.y >= min(onep.y, twop.y) and checked.y <= max(onep.y, twop.y)

function get_bot_direction(current_x: int, current_y: int, dest_x: int,
dest_y: int) -> List[int]:
```

```
{Modifikasi get_direction dari file util.py}
delta_x <- dest_x - current_x
delta_y <- dest_y - current_y
if abs(delta_x) > abs(delta_y)then
    delta_x <- max(-1, min(delta_x, 1))
    delta_y <- 0
else
    delta_y <- max(-1, min(delta_y, 1))
    delta_x <- 0
endif
-> (delta_x, delta_y)
```

4.2. Struktur Data yang Digunakan

Struktur data proyek ini adalah board. Board merupakan array yang terdiri dari sekumpulan data GameObject bertipe seperti berikut:
GameObject(id=254, position=Position(y=6, x=7), type='tipenya', properties=Properties(points=2, pair_id=None, diamonds=None, score=None, name=None, inventory_size=None, can_tackle=None, milliseconds_left=None, time_joined=None, base=None))

Nilai dari masing masing atribut bergantung pada type. Type misalnya TeleportGameObject, 'DiamondButtonGameObject', 'DiamondGameObject', 'BotGameObject', dan 'BaseGameObject'. Template program sudah menyediakan fungsi untuk mengakses tipe tipe tersebut melalui properti pada kelas board.

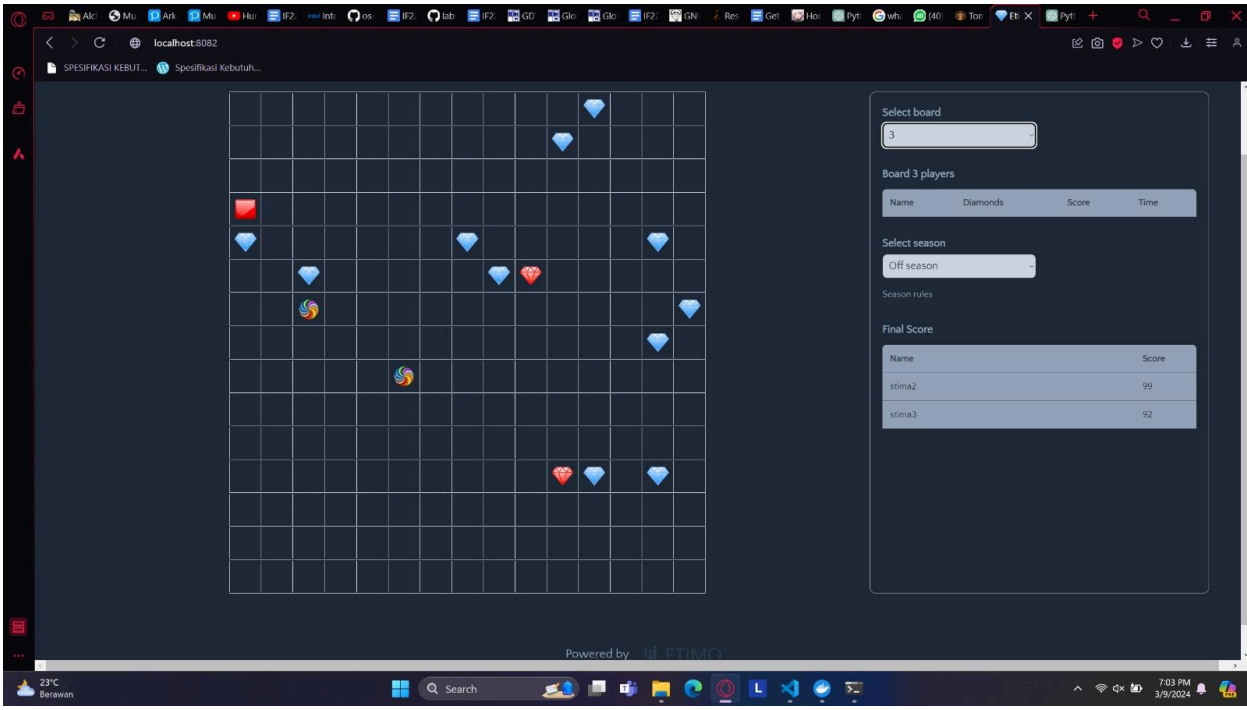
4.3. Analisis Pengujian

4.3.1. Pengujian

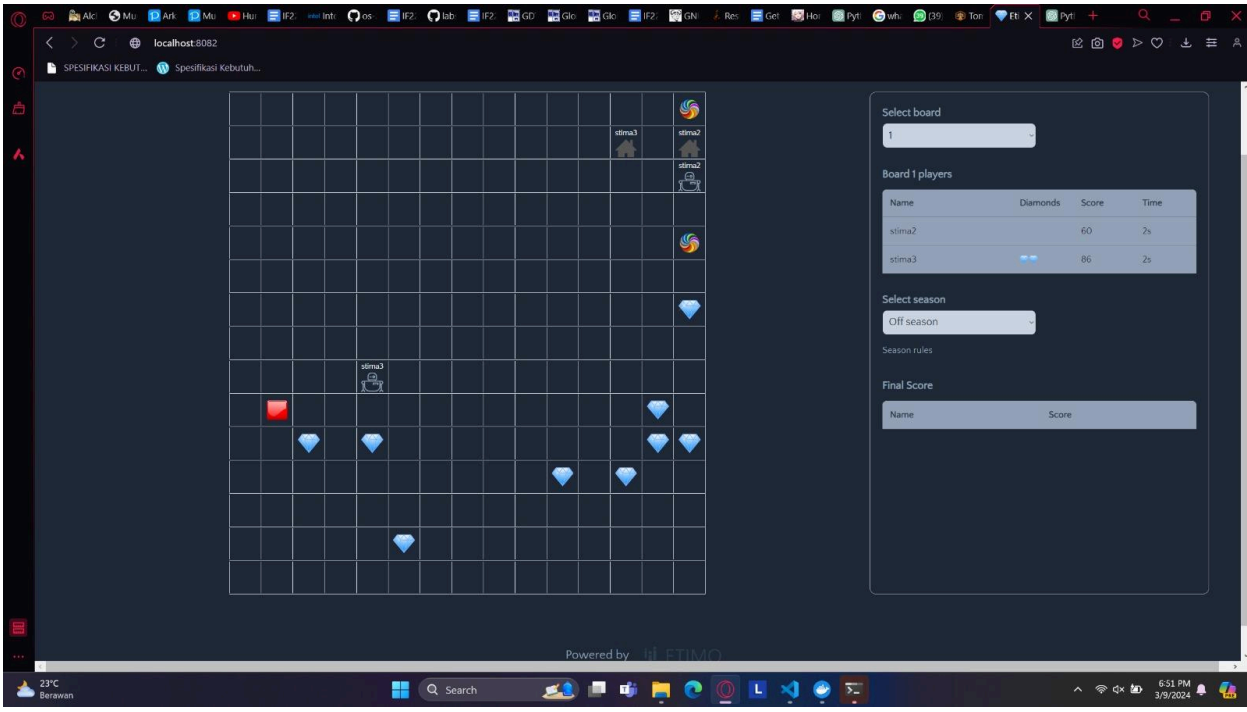
Algoritma greedy terkadang tidak mengembalikan hasil yang optimal. Misalnya ada satu diamond yang dekat yakni di sisi kiri, namun diamond lainnya berada di sisi kanan, ini mengakibatkan inefisiensi langkah. Kelemahan algoritma greedy terasa minim bila sleep nya kecil. Namun akan sangat berpengaruh ketika nilai sleep besar. Secara overall algoritma greedy cukup efisien. Karena kami juga menanggulangi inefisiensi algoritma greedy dengan hanya memilih diamond dalam radius tertentu agar jangan sampai malah terlalu jauh dari base.

4.3.2. Screenshoot

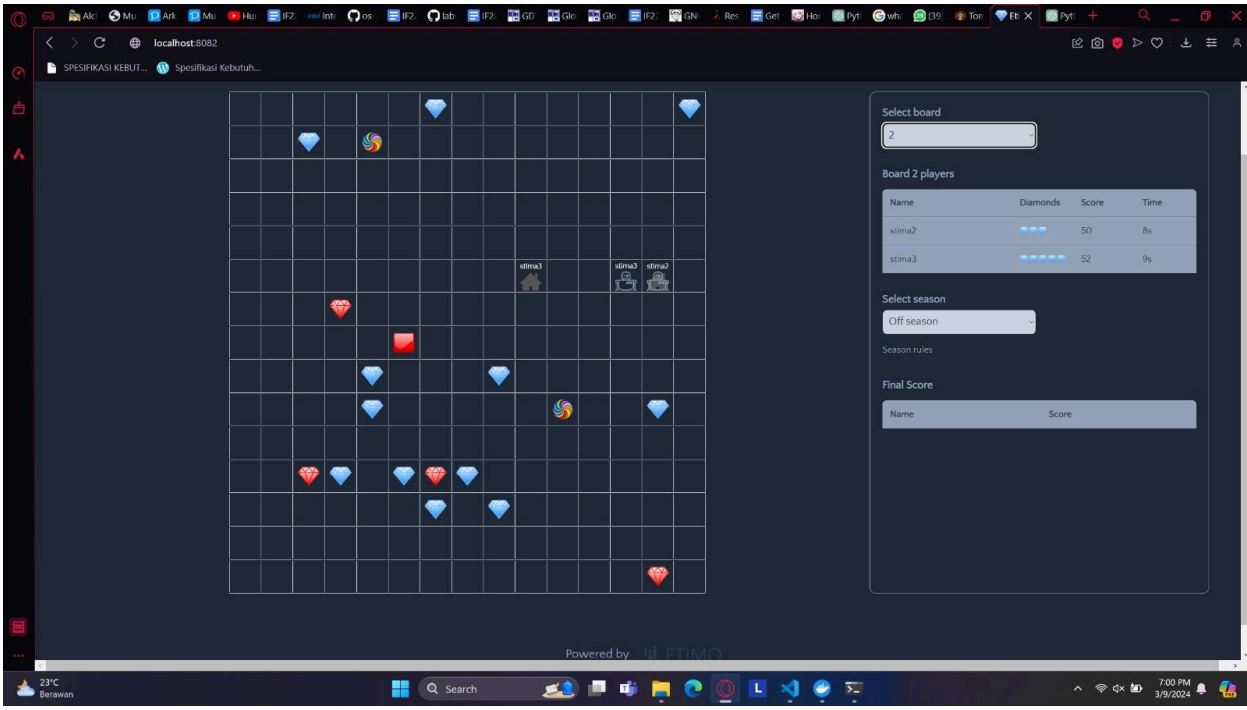
4.3.2.1. Picture 1



4.3.2.2. Picture 2



4.3.2.3. Picture 3


localhost:8082
SPESIFIKASI KEBUT...
Spesifikasi Kebutuh...
stima3
stima3
stima2
Select board
2
Board 2 players
Name
Diamonds
Score
Time
stima2
50
8s
stima3
52
9s
Select season
Off season
Season rules
Final Score
Name
Score
Powered by
23°C
Berawan
Search
7:00 PM
3/9/2024

BAB V PENUTUP

5.1. Kesimpulan

Kami menyimpulkan bahwa dengan ini, kami telah berhasil mengembangkan bot untuk permainan “Diamonds” dengan memanfaatkan algoritma Greedy. Program yang kami buat adalah gabungan dari beberapa aksi Greedy yang dapat dilakukan bot dalam permainan. Kami telah melakukan proses pengujian dan menemukan bahwa program bot kami telah memiliki strategi Greedy yang baik dan layak dipakai.

5.2. Saran

Saran untuk pengembang bot lainnya:

- Lakukanlah pengujian yang banyak, karena ada banyak sekali aksi yang dapat dilakukan oleh bot, sehingga kemungkinan perilaku bot lawan juga sangat banyak.

Saran untuk Tugas Besar selanjutnya:

- Pada awal ada sedikit kebingungan karena spesifikasi tugas besar tidak diberikan lewat email. Alangkah baiknya jika di tugas besar selanjutnya spesifikasi langsung diberikan lewat email di tanggal rilis tugas besar.

LAMPIRAN

***Repository* Github:**

github.com/jimlynurarif/Tubes1_ReelaIndoIndoIndo

Drive berisi video:

https://drive.google.com/drive/folders/1KB6E2H90Ucluv7X8dk4a2xwkun08vusN?usp=drive_link

DAFTAR PUSTAKA

Rinaldi, M. (n.d.). Algoritma Greedy (2021) Bag1. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

Rinaldi, M. (n.d.). Algoritma Greedy (2021) Bag2. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf

Rinaldi, M. (n.d.). Algoritma Greedy (2021) Bag3. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf

https://www.geeksforgeeks.org/greedy-algorithms/

https://www.freecodecamp.org/news/when-to-use-greedy-algorithms/