

# **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 12(TURTLE)**

<b>Aprililianti</b>	<b>123140041</b>
<b>Annisa Salsabila</b>	<b>123140070</b>
<b>Yeremia Situmorang</b>	<b>123140048</b>



Dosen Pengampu: Winda Yulita, M.Cs.

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

## DAFTAR ISI

<b>BAB I.....</b>	<b>3</b>
<b>DESKRIPSI TUGAS.....</b>	<b>3</b>
<b>BAB II.....</b>	<b>4</b>
<b>LANDASAN TEORI.....</b>	<b>6</b>
2.1 Dasar Teori.....	7
2.2 Cara Implementasi Program.....	8
2.3 Menjalankan Bot Program.....	
<b>BAB III.....</b>	<b>10</b>
<b>APLIKASI STRATEGI GREEDY.....</b>	
3.1 <b>Proses Mapping.....</b>	<b>12</b>
3.2 Eksplorasi Alternatif Solusi Greedy.....	14
3.3 Strategi Greedy yang Dipilih.....	14
<b>BAB IV.....</b>	<b>15</b>
<b>IMPLEMENTASI DAN PENGUJIAN.....</b>	
4.1 Implementasi Algoritma Greedy.....	15
1. Pseudocode.....	15
2. Penjelasan Alur Program.....	18
4.2 Struktur Data yang Digunakan.....	19
4.3 Pengujian Program.....	20
1. Skenario Pengujian.....	21
2. Hasil Pengujian dan Analisis.....	21
<b>BAB V.....</b>	<b>22</b>
<b>KESIMPULAN DAN SARAN.....</b>	<b>22</b>
5.1 Kesimpulan.....	22
5.2 Saran.....	22
<b>LAMPIRAN.....</b>	<b>23</b>
<b>DAFTAR PUSTAKA.....</b>	<b>24</b>

# BAB I

## DESKRIPSI TUGAS

### 1.1 Abstraksi

Diamonds adalah sebuah tantangan pemrograman kompetitif di mana setiap peserta akan membangun dan mengendalikan sebuah bot. Tujuan utama dari bot ini adalah mengumpulkan diamond sebanyak mungkin di dalam arena permainan. Namun, mengumpulkan diamond tidaklah semudah yang dibayangkan. Arena permainan dipenuhi dengan berbagai rintangan dan elemen pengganggu yang dirancang untuk membuat strategi menjadi lebih menantang, seru, dan kompleks. Untuk meraih kemenangan, setiap pemain harus mengembangkan dan menerapkan strategi yang efektif pada bot mereka, termasuk kemampuan navigasi, pengambilan keputusan, dan adaptasi terhadap situasi yang berubah-ubah.



**Gambar 1.** Permainan Diamonds

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
- c. 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

Terdapat pula komponen-komponen dari permainan Diamonds antara lain, yakni sebagai berikut.

### 1) Diamonds



**Gambar 2.** Diamond Biru dan Merah

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



**Gambar 3.** Red/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



**Gambar 4.** 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



**Gambar 5.** Bots and Bases

Pada game ini kita akan menggerakkan bot untuk mendapatkan diamond sebanyak banyaknya. Semua bot memiliki sebuah Base dimana Base ini akan digunakan untuk menyimpan diamond yang sedang dibawa. Apabila diamond disimpan ke base, score bot akan bertambah senilai diamond yang dibawa dan inventory (akan dijelaskan di bawah) bot menjadi kosong. Posisi Base tidak akan berubah sampai akhir game.

### 5) Inventory

Name	Diamonds	Score	Time
stima		0	43s
stima2		0	43s
stima1		0	44s
stima3		0	44s

**Gambar 6.** 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.

## **BAB II**

### **LANDASAN TEORI**

#### **2.1 Dasar Teori**

Algoritma Greedy merupakan metode yang paling populer dan sederhana untuk memecahkan persoalan optimasi, serta ada dua macam persoalan optimasi, yakni maksimasi dan minimasi. Algoritma greedy adalah algoritma yang memecahkan persoalan secara langkah per langkah (step by step) sedemikian sehingga pada setiap langkah itu mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan dan “berharap” bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global. Berikut elemen-elemen yang terdapat dalam Algoritma Greedy yang harus ditentukan dan dipertimbangkan.

- 1) Himpunan kandidat,  $C$  : berisi kandidat yang akan dipilih pada setiap Langkah (misal: simpul/sisi di dalam graf, job, task, koin, benda, karakter, dsb)
- 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 obyektif : memaksimumkan atau meminimumkan

Namun, optimum global belum tentu merupakan solusi optimum (terbaik), bisa jadi merupakan solusi sub-optimum atau pseudo-optimum. Hal ini dikarenakan Algoritma greedy tidak beroperasi secara menyeluruh terhadap semua kemungkinan solusi yang ada dan terdapat beberapa fungsi seleksi yang berbeda sehingga harus memilih fungsi yang tepat jika ingin algoritma menghasilkan solusi optimal.

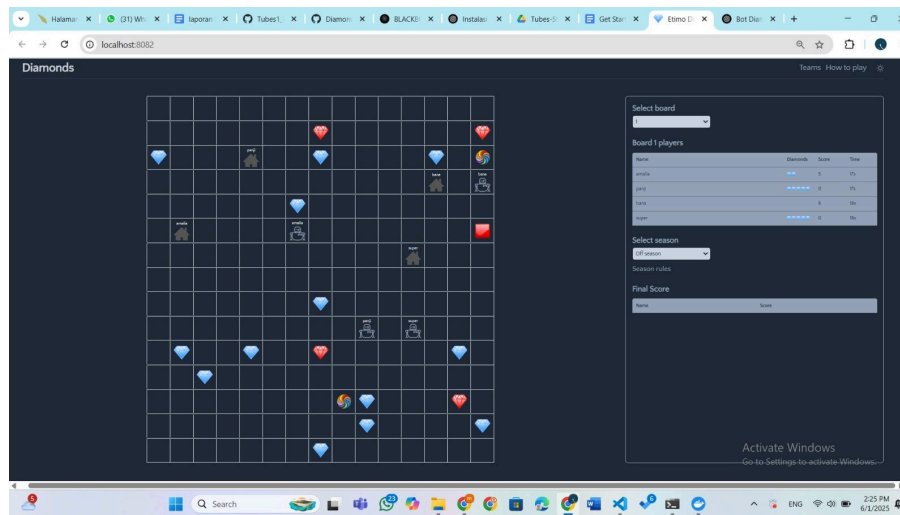
## 2.2 Cara Kerja Program

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 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).
- 6) Selain itu, terdapat beberapa fitur tambahan seperti teleporter dan red button yang dapat digunakan apabila bot menuju posisi objek tersebut.
- 7) Apabila waktu seluruh bot telah berakhir, maka permainan berakhir. Score masing-masing pemain akan ditampilkan pada tabel Final Score di sisi kanan layar.

## 2.3 Implementasi Algoritma Greedy

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



**Gambar 7.1** Implementasi Algoritma Greedy Bot-Diamond

untuk mengimplementasikan konsep Algoritma Greedy *by density* untuk mendekati kasus optimal. Selain itu, ini detail dari keenam elemen Algoritma Greedy sebagai bahan pertimbangan langkah yang dilakukan oleh bot jika mendapati ketiga komponen tersebut.

- 1) Himpunan kandidat,  $C$  : Semua langkah yang mungkin dilakukan oleh bot untuk mendekati dan mengambil diamond. Ini termasuk pergerakan ke berbagai lokasi di peta di mana diamond tersedia.
- 2) Himpunan solusi,  $S$  : Langkah dari Himpunan Kandidat yang menghasilkan pengambilan jumlah diamond terbanyak, termasuk mempertimbangkan



penggunaan Red Button untuk memaksimalkan jumlah diamond yang dapat diambil.

- 3) Fungsi solusi: Fungsi yang mengevaluasi apakah kondisi akhir telah tercapai, bisa jadi berupa bot telah mengumpulkan jumlah diamond tertentu atau waktu permainan telah habis. Tujuan akhir adalah untuk memaksimalkan jumlah diamond yang dikumpulkan sebelum kembali ke base.
- 4) Fungsi seleksi (selection function): Memilih langkah selanjutnya berdasarkan kriteria jumlah diamond terbanyak yang bisa diambil dalam satu langkah, memperhitungkan posisi diamond, Red Button, dan Teleporters.
- 5) Fungsi kelayakan (feasible): Semua langkah dianggap layak selama memungkinkan bot untuk bergerak ke lokasi diamond atau Red Button tanpa atau kehilangan diamond karena inventory penuh.
- 6) Fungsi obyektif : Memaksimalkan jumlah diamond yang dikumpulkan oleh bot. Strategi ini memprioritaskan pengambilan diamond merah dan menggunakan Red Button secara strategis.

## **BAB III**

### **APLIKASI STRATEGI GREEDY**

#### **3.1 Proses Mapping Persoalan Diamonds ke Elemen-Elemen Algoritma Greedy**

Algoritma greedy memecahkan masalah optimasi dengan membuat pilihan lokal terbaik pada setiap langkah, dengan harapan menghasilkan solusi global yang optimal. Pendekatan greedy dalam persoalan Diamonds harus dilakukan pemetaan elemen elemen masalah yaitu:

##### **3.1.1 Eksplorasi Alternatif Solusi Greedy**

Himpunan kandidat merupakan semua kemungkinan langkah atau tujuan yang dapat dipilih oleh bot di setiap iterasi. Dalam persoalan game diamonds, himpunan kandidatnya adalah:

1. Posisi berlian yang dapat diambil
2. Posisi teleport
3. Posisi base atau tempat asal
4. Posisi bot musuh

Masing-masing kandidat memiliki bentuk  $(x,y)$  yang merupakan koordinat mereka di bidang kartesian

##### **3.1.2 Himpunan Solusi**

Himpunan solusi adalah urutan langkah-langkah yang diambil oleh bot untuk mencapai tujuan, yaitu mendapatkan skor sebanyak mungkin dan kembali ke base, sehingga dalam konteks game diamonds himpunan solusi merupakan daftar posisi yang dikunjungi oleh bot dalam bentuk koordinat. Solusi akhir dari himpunan solusi merupakan jalur lengkap atau daftar koordinat yang dilalui bot untuk memaksimalkan skor. Dalam hal ini daftar himpunan solusinya berupa:

1. Posisi diamond yang terdekat dan di saat cukup untuk dimasukkan ke inventory
2. Posisi musuh (jika terlalu dekat)
3. Posisi teleport yang mempercepat atau jarak terdekat menuju base atau diamond
4. Posisi base jika inventory penuh

Himpunan solusi dihasilkan dengan mempertimbangkan kombinasi

1. Jarak ke Object
2. Kapasitas inventory (masih mencukupi atau sudah penuh)
3. Jarak ke musuh

##### **3.1.3 Fungsi Solusi**

Fungsi solusi merupakan fungsi yang menentukan kapan suatu masalah dapat dianggap selesai, sehingga dalam permainan diamonds kita memiliki fungsi solusi sebagai berikut:

5. Inventory bot sudah penuh dan bot sudah kembali ke base
6. Berlian yang tersedia di papan sudah habis dan bot sudah kembali ke base

7. Waktu permainan habis.

#### **3.1.4 Fungsi Seleksi**

Fungsi seleksi menjelaskan bagaimana algoritma memilih kandidat terbaik pada setiap langkah. Dalam bot yang kami buat fungsi seleksinya adalah sebagai berikut:

1. Bot mencari berlian terdekat berdasarkan jarak euclidean yang poinnya tidak melebihi kapasitas inventory.
2. Bot mencari teleportasi terdekat berdasarkan jarak euclidean dengan asumsi bahwa hal tersebut akan mempersingkat jarak terhadap diamond
3. Jika inventory penuh bot memilih
  - Langsung menuju base, atau
  - Menggunakan kombinasi teleportasi yaitu sebagai pintu masuk dan teleport terjauh sebagai pintu keluar menuju base, lalu membandingkan total jarak dengan rute langsung . Bot memilih mana jalur yang paling pendek/singkat
4. Jika ada musuh berada di jarak yang ditentukan, maka bot mengutamakan bergerak menuju posisi musuh sebagai bentuk prioritas untuk menghindari atau menghadapi musuh
5. Pemilihan jalur ke berlian atau teleportasi di prioritaskan melalui
  - Jarak ke berlian melalui teleport lebih pendek daripada jarak langsung
  - Poin berlian masih muat di inventory, maka bot akan mengakses berlian melalui jalur teleportasi

#### **3.1.5 Fungsi Kelayakan**

Fungsi kelayakan menentukan apakah suatu kandidat yang dipilih dapat ditambahkan ke himpunan solusi parsial yang sedang dibangun. Fungsi ini memeriksa apakah sebuah langkah atau pilihan memenuhi syarat-syarat tertentu pada kondisi saat itu. Suatu langkah dikatakan layak jika memenuhi beberapa syarat berikut:

1. Teleportasi yang dipilih memiliki koordinat tujuan yang valid dan bukan terblokir atau sudah ditempati.
2. Tidak berisiko terlalu dekat dengan musuh (misalnya dalam radius bahaya tertentu).
3. Langkah tersebut tidak mengakibatkan bot terjebak atau jauh dari base ketika waktu hampir habis.
4. Posisi tujuan masih ada di papan permainan dan belum diambil/diklaim oleh bot lain.

### 3.1.6 Fungsi Objektif

Fungsi objektif merupakan tujuan akhir dari algoritma yang ingin dicapai melalui serangkaian langkah-langkah yang diambil. Fungsi objektif dari algoritma greedy yang digunakan oleh bot adalah:

5. Meminimalkan jarak tempuh ke diamond, teleport, atau base, sehingga waktu tidak terbuang dan bot bisa melakukan lebih banyak aksi dalam satu pertandingan.
6. Menggunakan teleportasi secara strategis bila memungkinkan mempercepat rute ke diamond atau ke base ketika inventory penuh.
7. Mengosongkan inventory di base secara berkala agar tetap bisa mengangkut diamond baru tanpa membuang langkah sia-sia.
8. Kembali langsung ke base bila jarak ke teleportasi lebih jauh dibandingkan ke base dengan tepat waktu jika inventory penuh

### 3.2 Eksplorasi alternatif solusi greedy yang mungkin dipilih dalam persoalan Diamonds

Dalam pengembangan strategi greedy untuk bot pengumpul diamond, terdapat beberapa pendekatan yang dapat dipertimbangkan untuk meningkatkan performa bot dalam pengumpulan diamonds seperti:

1. Greedy by Diamond Value Density
  - Memprioritaskan diamond dengan nilai tertinggi per unit jarak
  - Lebih efisien dalam meningkatkan skor per langkah
2. Greedy with Time Constraint
  - Memastikan bot punya cukup waktu untuk kembali ke base
  - Mencegah bot terjebak terlalu jauh dari base
3. Greedy with Aggressive Collection
  - Mencari cluster diamond terdekat
  - Memaksimalkan pengumpulan per trip
4. Hybrid Greedy Approach
  - Menyeimbangkan berbagai faktor penting
  - Dapat disesuaikan bobot parameternya
5. Greedy with Risk Avoidance
  - Meminimalisir konflik dengan bot lain
  - Lebih konservatif dalam menghindari risiko

Dalam menganalisis berbagai pendekatan greedy untuk bot pengumpul diamond, kita dapat melihat trade-off yang menarik antara efisiensi komputasi dan efektivitas strategi seperti:

1. Greedy by Diamond Value

Density Efisiensi:

- Kompleksitas tetap  $O(n)$  untuk mengevaluasi semua diamond
- Perhitungan value/distance sederhana namun berdampak besar
- Optimal untuk papan dengan distribusi nilai

diamond bervariasi Efektivitas:

- Maksimisasi poin per unit waktu
- Cocok untuk early game saat inventory masih kosong.

2. Greedy with Time

Constraint Efisiensi:

- Memerlukan perhitungan jarak ganda (ke diamond + ke base)
- Kompleksitas  $O(2n) \approx$  masih linear
- Overhead minimal untuk

pengecekan waktu Efektivitas:

- Mencegah game over karena waktu habis
- Sangat efektif di late game.

3. Cluster-based Aggressive

Collection Efisiensi:

- Kompleksitas  $O(n^2)$  untuk deteksi cluster
- Resource-intensive untuk papan besar
- Cocok untuk papan dengan diamond terkonsentrasi Efektivitas:
- Optimalisasi movement dengan multi-collection
- Efek domino positif pada skor akhir

4. Hybrid Greedy

Approach

Efisiensi:

- Perhitungan multifaktor ( $0.4 \cdot \text{dist} + 0.4 \cdot \text{value} + 0.2 \cdot \text{time}$ )
- Kompleksitas tetap  $O(n)$
- Fleksibel melalui penyesuaian bobot Efektivitas:
- Balance antara berbagai strategi
- Adaptif terhadap fase permainan
- Membutuhkan tuning bobot yang presisi

## 5. Risk-Averse

Greedy

Efisiensi:

- $O(n*m)$  untuk  $n$  diamond dan  $m$  bot
- Mahal secara komputasi
- Cocok untuk multiplayer competitive Efektivitas:
- Minimasi konflik dengan bot lain
- Konsistensi pengumpulan
- Trade-off: mungkin melewatkan oppor

### 3.3 Strategi greedy yang dipilih

Strategi utama yang diterapkan dalam bot ini mengadopsi pendekatan greedy berbasis jarak terdekat dengan pertimbangan keamanan dari bot lawan dan teleportasi. Bot secara konsisten memilih diamond terdekat yang sesuai dengan kapasitas inventory saat ini, dengan mekanisme fallback ke penghindaran musuh ketika terdeteksi bahaya. Implementasi ini menggabungkan tiga lapisan strategi:

1. Prioritas keamanan melalui pengecekan musuh dalam radius 2 unit
2. Optimasi rute menggunakan portal teleport ketika menguntungkan
3. Default ke algoritma greedy konvensional pencarian

diamond terdekat. Dari segi efektivitas, strategi ini mengutamakan dalam beberapa aspek seperti:

1. Manajemen risiko dengan melakukan penghindaran musuh yang dimulai ketika inventory penuh
2. Optimalisasi gerakan melalui analisis biaya-manfaat penggunaan teleport yang dihitung berdasarkan jarak antara Bot dengan teleportasi dan jarak teleportasi dengan GameObject lain seperti Diamond maupun Base
3. Keputusan yang berbeda saat inventory kosong dan saat penuh

## **BAB IV**

### **IMPLEMENTASI DAN PENGUJIAN**

#### **4.1 Implementasi Algoritma Greedy**

##### **1. Pseudocode**

```
from typing import Optional, List

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

class April(BaseLogic):
    def __init__(self):
        self.directions = [(1, 0), (0, 1), (-1, 0), (0, -1)]
        self.goal_position: Optional[Position] = None
        self.current_direction = 0
        self.diamonds = []

    def getDiamond_inRange(self, board: Board, pos1: Position, pos2: Position)
    -> List[GameObject]:
        x1 = pos1.x
        x2 = pos2.x
        y1 = pos1.y
        y2 = pos2.y

        dirX, dirY = clamp(x2 - x1, -1, 1), clamp(y2 - y1, -1, 1)
        if (dirY == 0):
            dirY += 1
        if (dirX == 0):
            dirX += 1
        print(dirX, dirY)
        diamonds = [d for d in board.game_objects if ((d.type ==
"DiamondGameObject") and (d.position.x in range(x1, x2+dirX, dirX)) and
(d.position.y in range(y1, y2+dirY, dirY)))]
        print(len(diamonds))
        diamondBaru = [d.position for d in diamonds]
```

```

        for i in range(len(self.diamonds)):
            print(i,'X ', self.diamonds[i].x)
            print(i,'Y ', self.diamonds[i].y)
        print("Get Range")

        return diamondBaru

    def getDistance(self, pos1: Position, pos2: Position, byX: bool = True, byY:
bool = True) -> int:
        jarak = 0
        if(byX):
            jarak += abs(pos1.x - pos2.x)
        if(byY):
            jarak += abs(pos1.y - pos2.y)
        return jarak

    def next_move(self, board_bot: GameObject, board: Board):
        props = board_bot.properties
        current_position = board_bot.position

        # Get base, red button, and diamonds in between
        base = board_bot.properties.base
        redButton = [d.position for d in board.game_objects if d.type ==
"DiamondButtonGameObject"][0]

        if(len(self.diamonds) == 0):
            self.diamonds.append(base)

        isRedButton = position_equals(current_position, self.diamonds[-1])
        isBase = position_equals(current_position, base)
        if (isBase or isRedButton):
            if (not(isRedButton and isBase)):
                self.diamonds = self.getDiamond_inRange(board, base,
self.diamonds[-1])
            else:
                self.diamonds = self.getDiamond_inRange(board, base, redButton)

        if (self.getDistance(base, redButton, byY=False) >
self.getDistance(base, redButton, byX=False)):
            self.diamonds = sorted(self.diamonds, key=lambda diamond:
diamond.y)[0:4]
        else:
            self.diamonds = sorted(self.diamonds, key=lambda diamond:

```



```

diamond.x)[0:4]

        self.diamonds = sorted(self.diamonds, key=lambda diamond:
self.getDistance(diamond, current_position))
        if (not(isRedButton and isBase)):
            self.diamonds.append(base)
        else:
            self.diamonds.append(redButton)

    if (position_equals(current_position, self.diamonds[0])):
        self.diamonds.pop(0)

    # Analyze new state
    if props.diamonds == 5:
        # Move to base
        self.goal_position = base
    else:
        # Just roam around
        self.goal_position = self.diamonds[0]

    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,
        )
    return delta_x, delta_y

```

## Penjelasan Alur Program

Program bot pada permainan ini dibangun dengan memanfaatkan dua algoritma utama, yaitu Breadth-First Search (BFS) untuk pencarian jalur terpendek dan Greedy Algorithm yang menggunakan rasio poin terhadap jarak sebagai kriteria pemilihan diamond. Berikut penjelasan rinci dari alur program:

### 1. Inisialisasi Bot dan Struktur Data

- Bot didefinisikan dalam class yang diturunkan dari BaseLogic.
- Dua struktur data utama yang digunakan adalah:
  - Self.diamonds: menyimpan semua objek diamond yang ditemukan di peta.
  - self.obstacles: menyimpan posisi semua objek penghalang (seperti dinding dan bot lain).

### 2. Fungsi get\_objects

- Fungsi ini memindai seluruh objek pada peta.
- Jika objek bertipe Diamond, maka posisinya ditambahkan ke dalam daftar self.diamonds.
- Jika objek bertipe Bot atau Wall, maka posisinya dimasukkan ke self.obstacles.
- Tujuannya adalah memisahkan objek yang dapat diambil dan yang harus dihindari.

### 3. Algoritma Breadth-First Search (BFS)

- Digunakan dalam fungsi bfs() untuk menghitung jarak terpendek dari posisi bot ke target diamond.
- BFS menjelajahi peta secara menyeluruh menggunakan struktur antrian deque.
- Setiap simpul yang dikunjungi akan ditandai agar tidak dikunjungi kembali.
- Fungsi ini menghindari posisi yang terdapat dalam self.obstacles.
- Jika diamond dapat dicapai, maka dikembalikan jaraknya. Jika tidak, maka hasilnya infinity.

### 4. Pengambilan Keputusan Arah Gerak

- Jika tidak ada target yang valid, bot akan kembali ke base.
- Jika ada target terbaik, bot akan bergerak ke arah diamond tersebut.
- Pergerakan hanya dilakukan satu langkah per giliran, dengan arah (kanan, kiri, atas, bawah) tergantung posisi relatif bot terhadap target.
- Arah gerak dikembalikan dalam bentuk dx dan dy.

## 5. Strategi Greedy dan Rasio Poin/Jarak

Dalam fungsi `next_move()`, bot akan:

- Mengambil informasi kondisi tas: posisi, jumlah diamond, dan kapasitas.
- Menghitung sisa ruang tas untuk menentukan apakah diamond dapat diambil.

Untuk setiap diamond:

- Dicek apakah diamond memiliki poin dan cukup ruang di tas.
- Hitung jarak ke diamond menggunakan BFS.
- Hitung rasio:  $\text{poin} / \text{jarak}$ .
- Jika rasio lebih tinggi dari sebelumnya, simpan sebagai target terbaik.

## 4.2 Struktur Data yang Digunakan

Program merupakan logika pengendalian bot otomatis dalam permainan pengumpulan diamond. Untuk mencapai tujuan tersebut, program ini menggunakan beberapa struktur data penting untuk menyimpan dan mengelola informasi posisi, objek permainan, serta strategi bot. Berikut penjelasan masing-masing struktur data yang digunakan:

### 1. Position

- Merupakan struktur data yang digunakan untuk merepresentasikan posisi di dalam papan permainan (grid).
- Memiliki dua atribut utama: **x** dan **y**, masing-masing menyatakan koordinat horizontal dan vertikal.
- Digunakan dalam:
  - Posisi bot saat ini
  - Posisi diamond
  - Posisi base
  - Posisi tombol merah (red button)
  - Posisi tujuan bot

### 2. GameObject

- Merupakan objek di dalam permainan yang memiliki jenis (type) dan posisi (position).
- Dapat berupa:
  - Bot itu sendiri
  - Diamond ("DiamondGameObject")
  - Tombol merah ("DiamondButtonGameObject")
  - Base pemain
- Semua objek ini tersimpan di dalam `board.game_objects`, yang merupakan daftar semua objek di papan.

### 3. Board

- Mewakili seluruh papan permainan.
- Menyimpan semua objek permainan (dalam `game_objects`).
- Menjadi parameter yang diteruskan ke fungsi utama `next_move()`.

### 4. `self.diamonds: List[Position]`

- Merupakan daftar posisi diamond yang ditargetkan oleh bot.
- Diurutkan berdasarkan strategi tertentu:
  - Jarak terdekat dari posisi bot
  - Arah tertentu (horizontal/vertikal)
- Bot akan bergerak mengikuti urutan posisi dalam list ini.
- List ini bisa di-update setiap kali bot mencapai base atau tombol merah.

### 5. `self.goal_position: Optional[Position]`

- Menyimpan posisi tujuan utama bot saat ini.
- Bisa bernilai:
  - Posisi diamond berikutnya (jika belum penuh)
  - Posisi base (jika bot sudah membawa 5 diamond)
- Bertipe optional, artinya bisa kosong (`None`) jika belum ditentukan

### 6. `self.directions: List[Tuple[int, int]]`

- Daftar tuple yang menyatakan arah gerak:
  - $(1, 0) \rightarrow$  kanan
  - $(0, 1) \rightarrow$  atas
  - $(-1, 0) \rightarrow$  kiri
  - $(0, -1) \rightarrow$  bawah
- Walaupun didefinisikan, data ini belum digunakan secara aktif dalam logika program.

## 4.3 Pengujian Program

### 1. Skenario Pengujian

Pengujian dilakukan untuk mengevaluasi kinerja bot dalam berbagai kondisi peta permainan.

Tujuan pengujian adalah memastikan bahwa bot dapat:

1. Menghindari rintangan seperti tembok dan bot lain.
2. Memilih diamond yang paling efisien berdasarkan rasio nilai terhadap jarak.
3. Kembali ke base saat tas sudah penuh atau tidak ada diamond yang sesuai kapasitas

Skenario yang diuji antara lain:

Skenario	Deskripsi Skenario	Diharapkan
1	Diamond berada dekat dan tidak terhalang rintangan	Bot bergerak langsung ke diamond

2	Diamond bernilai besar tetapi jauh	Bot memilih diamond dengan rasio terbaik
3	Tidak ada diamond yang muat di tas	Bot kembali ke base
4	Rintangan menghalangi jalur langsung ke diamond	Bot menggunakan jalur alternatif (via BFS)
5	Diamond lebih dari satu, dengan berbagai nilai dan jarak	Bot memilih berdasarkan rasio terbaik (greedy)

## 2. hasil Pengujian dan Analisis

Berikut adalah hasil dari beberapa pengujian yang dilakukan:

Skenario	Hasil Bot	Status
1	Bot berhasil menuju diamond terdekat secara langsung	Sesuai
2	Bot lebih memilih diamond terdekat dengan rasio lebih baik	Sesuai
3	Bot langsung kembali ke base karena tidak ada diamond sesuai	Sesuai
4	Bot menghindari rintangan dan tetap mencapai diamond	Sesuai
5	Bot menghitung rasio semua diamond dan memilih yang terbaik	Sesuai

## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Berdasarkan hasil implementasi dan pengujian bot menggunakan strategi algoritma Greedy dalam permainan *Diamonds*, dapat disimpulkan bahwa pendekatan ini mampu memberikan solusi yang cukup efisien dalam konteks waktu terbatas dan kompleksitas peta permainan. Strategi greedy yang diterapkan berfokus pada pemilihan diamond terdekat berdasarkan jarak dan nilai, serta mempertimbangkan kondisi inventory, keberadaan musuh, dan pemanfaatan objek khusus seperti red button dan teleporters.

Penggunaan pendekatan *Greedy by Density* (nilai per jarak) memberikan peningkatan skor yang signifikan dibandingkan greedy konvensional karena mampu memaksimalkan skor per langkah. Selain itu, integrasi strategi tambahan seperti pengecekan musuh dalam radius bahaya serta pemanfaatan teleportasi memberikan keunggulan kompetitif dalam menghindari risiko kehilangan diamond akibat *tackle*. Bot juga mampu melakukan adaptasi terhadap kondisi dinamis peta dengan menyusun ulang target diamond berdasarkan prioritas dan situasi saat itu.

Secara keseluruhan, strategi greedy yang diterapkan telah memenuhi elemen-elemen penting dari algoritma greedy klasik, mulai dari himpunan kandidat, fungsi seleksi, kelayakan, hingga fungsi objektif, dan mampu menyelesaikan permasalahan pengumpulan diamond dengan cara yang cukup efektif.

#### **5.2 Saran**

1. Penggunaan algoritma greedy saat ini hanya mempertimbangkan jarak terdekat dan jumlah poin diamond. Bot dapat dikembangkan lebih lanjut dengan mengimplementasikan *Hybrid Greedy Algorithm* atau algoritma lain seperti A\*, Dijkstra, atau algoritma reinforcement learning untuk meningkatkan pengambilan keputusan, terutama dalam kondisi kompleks atau saat berhadapan dengan musuh.
2. Red Button dan Teleporter dapat digunakan lebih strategis, bukan hanya sekadar akses cepat. Bot bisa belajar kapan waktu terbaik untuk menekan Red Button (misalnya saat diamond tersisa sedikit), atau memanfaatkan teleportasi sebagai bagian dari rute optimal ke base maupun ke diamond bernilai tinggi.
3. Dalam strategi hybrid greedy, bobot terhadap nilai diamond, jarak, waktu, dan risiko dapat disesuaikan selama permainan berlangsung. Mengembangkan mekanisme adaptif terhadap fase permainan (awal, tengah, akhir) akan membuat bot lebih fleksibel dan efektif.

## LAMPIRAN

A. Repository Github ([link](#))

B. Video Penjelasan (link GDrive)

<https://drive.google.com/file/d/1-9jvMGjSlh1wL4kszlK7OeDMacopi/view?usp=sharing>

## DAFTAR PUSTAKA

- [1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA: The MIT Press, 2009.
- [2] S. Dasgupta, C. Papadimitriou, and U. Vazirani, *Algorithms*. New York: McGraw-Hill, 2008.
- [3] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.
- [4] R. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. London: Pearson, 2021.
- [5] "Greedy Algorithm - GeeksforGeeks," [Online]. Available: <https://www.geeksforgeeks.org/greedy-algorithms/>. [Accessed: 1-Jun-2025].
- [6] "Game AI Pro: Collected Wisdom of Game AI Professionals," S. Rabin, Ed. Boca Raton, FL: CRC Press, 2013.
- [7] Diamonds Game – Github Repository. [Online]. Available: [repository link yang akan kalian isi di lampiran]