PEMANFAATAN ALGORITMA DALAM PEMBUATAN 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 5 (Malink Handal)

Muhammad Farisi Suyitno 123140152

Ali Akbar 123140162

Bima Aryaseta 123140177

Dosen Pengampu: Winda Yulita, M.Cs.

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SUMATERA
2025

DAFTAR ISI

BAB I

DESKRIPSI TUGAS

Tugas Besar Stima 2025 ini berfokus pada pembuatan bot untuk permainan "Diamonds" dengan mengimplementasikan algoritma Greedy. Tujuan utama bot adalah mengumpulkan diamond sebanyak mungkin untuk memenangkan pertandingan.

Komponen Utama Permainan:

1. Diamonds: Terdapat dua jenis, yaitu diamond biru (1 poin) dan diamond merah (2 poin). Diamond akan di-regenerate secara berkala dengan rasio yang dapat berubah.
2. Red Button/Diamond Button: Jika dilewati, semua diamond di papan akan di-generate ulang secara acak, dan posisi tombol ini juga akan berubah secara acak.
3. Teleporters: Dua teleporter saling terhubung; melewati satu akan memindahkan bot ke teleporter lainnya.
4. Bots and Bases: Setiap bot memiliki base untuk menyimpan diamond yang dibawa. Menyimpan diamond ke base akan menambah skor bot dan mengosongkan inventory bot.
5. Inventory: Tempat penyimpanan sementara diamond yang diambil bot, memiliki kapasitas maksimum. Untuk mengosongkannya, bot harus kembali ke base.

Cara Kerja Permainan:

1. Bot pemain ditempatkan secara acak di papan dengan home base, skor awal nol, dan inventory kosong.
2. Setiap bot memiliki waktu yang sama untuk bergerak.
3. Tujuan utama adalah mengumpulkan diamond (merah 2 poin, biru 1 poin).
4. Inventory yang penuh mengharuskan bot kembali ke base untuk menyimpan diamond dan mengosongkan inventory.
5. Jika bot A menimpa posisi bot B (tackle), bot B dikirim ke home base, dan semua diamond di inventory bot B hilang dan diambil oleh bot A.
6. Fitur tambahan seperti teleporter dan red button dapat digunakan dengan melewati posisinya.
7. Permainan berakhir jika waktu seluruh bot habis, dan skor akhir ditampilkan.

Spesifikasi Tugas:

1. Buat program bot dalam bahasa Python menggunakan algoritma Greedy.
2. Tugas dikerjakan berkelompok (2-3 orang), boleh lintas kelas dan kampus.
3. Strategi greedy harus dirancang sekreatif mungkin karena akan ada kompetisi antar bot.
4. Asistensi tugas besar bersifat opsional.
5. Bot akan dikompetisikan, dan ada hadiah bagi pemenang.
6. Kecurangan akan mengakibatkan nilai nol.

7. Laporan dikumpulkan paling lambat hari Minggu, 1 Juni 2025, pukul 23.59 melalui Google Form yang disediakan.
8. Program harus dapat dijalankan di Windows dan Linux; jika tidak, tidak akan dinilai.

Program Permainan Diamonds Terdiri Atas:

1. Game engine: Berisi kode backend (logika permainan dan API) dan frontend (visualisasi).
2. Bot starter pack: Berisi program untuk memanggil API, program bot logic (yang akan diimplementasikan mahasiswa), dan program utama serta utilitas lainnya.

BAB II

LANDASAN TEORI

2.1 Dasar Teori

Algoritma greedy merupakan sebuah metode yang populer digunakan untuk memecahkan persoalan optimasi. Persoalan optimasi adalah jenis masalah yang bertujuan untuk menemukan solusi terbaik (optimum) dari semua kemungkinan solusi yang ada, yang umumnya dapat berupa persoalan maksimasi atau minimasi.

Inti dari pendekatan greedy terletak pada prinsip "ambil apa yang bisa kamu dapatkan sekarang!". Berdasarkan prinsip ini, algoritma greedy membangun solusi secara bertahap, langkah demi langkah. Pada setiap langkah yang diambil, algoritma akan membuat keputusan dengan memilih opsi yang tampak paling baik (optimum lokal) pada saat itu tanpa melakukan pertimbangan mendalam mengenai dampak pilihan tersebut terhadap langkah-langkah berikutnya. Secara lebih formal, algoritma greedy bekerja dengan cara memecahkan masalah secara sekuensial di mana pada setiap tahapnya:

1. Algoritma akan mengambil pilihan yang dianggap memberikan keuntungan terbaik pada saat itu, tanpa mempertimbangkan konsekuensi jangka panjang dari pilihan tersebut.
2. Algoritma beroperasi dengan harapan bahwa akumulasi pilihan-pilihan optimum lokal ini akan menghasilkan sebuah solusi optimum secara global.

Dalam implementasinya, algoritma greedy melibatkan beberapa elemen fundamental, yaitu:

1. Himpunan Kandidat (C): Merupakan kumpulan dari semua elemen atau item yang berpotensi menjadi bagian dari solusi akhir.
2. Himpunan Solusi (S): Sebuah himpunan yang dibangun secara bertahap, berisi kandidat-kandidat yang telah terpilih dan memenuhi kriteria untuk membentuk solusi.
3. Fungsi Seleksi (Selection Function): Bertugas untuk memilih kandidat berikutnya yang dianggap paling menjanjikan dari himpunan kandidat yang belum terpilih.
4. Fungsi Kelayakan (Feasible Function): Digunakan untuk memeriksa apakah suatu kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi saat ini tanpa melanggar batasan-batasan (kendala) dari persoalan yang ada.
5. Fungsi Objektif (Objective Function): Mendefinisikan nilai yang ingin dioptimalkan (dimaksimalkan atau diminimalkan) oleh solusi yang dihasilkan.

Skema umum dari algoritma greedy biasanya dimulai dengan himpunan solusi yang masih kosong. Kemudian, dalam sebuah iterasi, algoritma akan menggunakan fungsi seleksi untuk memilih satu kandidat dari himpunan kandidat. Kandidat yang terpilih ini kemudian akan diuji kelayakannya, jika layak, maka kandidat tersebut akan ditambahkan ke dalam himpunan solusi, dan kandidat tersebut akan dihapus dari himpunan kandidat. Proses ini akan terus

berulang hingga himpunan solusi dianggap telah memenuhi kriteria sebagai solusi lengkap, atau hingga tidak ada lagi kandidat yang tersisa dalam himpunan kandidat.

Namun solusi yang dihasilkan oleh algoritma greedy tidak selalu merupakan solusi optimum secara global, terkadang solusi tersebut bersifat sub-optimum atau pseudo-optimum. Hal ini terjadi karena algoritma greedy tidak melakukan eksplorasi secara menyeluruh terhadap semua alternatif solusi yang mungkin ada, berbeda dengan metode pencarian menyeluruh .

2.2 Cara Kerja Program

program berjalan di dalam map untuk mengumpulkan diamond sebanyak banyaknya lalu kembali ke rumah untuk menyimpan diamond yang sudah di dapat, bot diprogram menggunakan algoritma greedy, bot selalu jalan ke arah terdekat seperti diamond terdekat atau pemain terdekat dan ketika diamond bot sudah penuh maka bot akan kembali ke rumah untuk menyimpan diamond.

2.2.1 cara implementasi program

1. Bot akan selalu menghitung diamond yang dibawa jika diamond sama dengan lima maka bot akan kembali ke dalam rumah sendiri
2. Jika diamond yang dimiliki bot masih belum lima maka bot kami akan mencari diamond terdekat dengan urutan diamond merah sebagai prioritas utama
3. Bot kami juga akan menyerang pemain terdekat dengan jumlah diamond terbanyak yang dibawa pemain lain
4. Bot kami akan segera pulang kerumah jika waktu permainan sudah akan berakhir

2.2.2 Menjalankan Bot Program

Pertama-tama pastikan Node.js, Docker desktop, Python dan Yarn sudah terinstal di pc terlebih dahulu. Kemudian kita akan melakukan intsallasi dan konfigurasi awal dengan menginstall folder game engine yang ditentukan lalu mengekstraknya. Setelah di ekstrak, masuk ke root directorinya menggunakan perintah cd tubes1-IF2211-game-engine-1.1.0. Didalamnya kita akan melakukan setup environment variable dengan memasukkan perintah ./scripts/copy-env.bat untuk windows dan local database (pada bagian ini, masuk dulu ke aplikasi Docker desktop) menggunakan perintah docker compose up -d database lalu ./scripts/setup-db-prisma.bat untuk windows. Kemudian jalankan perintah npm run build dan npm run start secara berurutan. Jika berhasil, bisa langsung mengunjungi frontend localhost dengan menekan link berikut http://localhost:8082/..

Setelah langkah-langkah sebelumnya telah selesai, kita lanjutkan dengan menginstall folder bot-starter-pack dan mengekstraknya. Setelah diekstrak masuk ke root directorinya dan menjalankan perintah pip install -r requirements.txt untuk menginstall dependencies yang diperlukan. Kemudian buka file bot-starter-pack di vscode untuk membuat logic greedy yang digunakan nanti. setelah logic botnya telah dibuat kita akan beralih ke file main.py. Di main py panggil dulu di main.py import
from game.logic.malinkhandal import Greedybot

kemudian buat dictionary CONTROLLERS = {
“Greedybot” : Greedybot }
kemudian di run-bots.bat
python main.py --logic GreedyBot --email=malinkhandal@gmail.com
--name=Malink--password=Malink123 --team etimo
run program run-bots.bat

BAB III

APLIKASI STRATEGI *GREEDY*

3.1 Proses *Mapping*

Untuk mengimplementasikan algoritma greedy pada permainan Diamonds, kami memetakan persoalan ke dalam elemen-elemen dasar algoritma greedy sebagai berikut:

Himpunan Kandidat Meliputi semua aksi yang dapat dilakukan oleh bot serta semua objek atau kondisi yang menjadi target aksi, seperti diamond (merah dan biru), posisi bot lawan, posisi base, serta arah gerak yang tersedia.

Himpunan Solusi Adalah semua aksi yang akhirnya dipilih bot selama permainan untuk mengoptimalkan skor. Misalnya, mengejar musuh yang membawa banyak diamond, mengumpulkan diamond terdekat, atau pulang ke base dengan aman.

Fungsi Solusi Fungsi ini mengevaluasi apakah suatu aksi dapat mendekatkan bot pada tujuannya, yaitu mengumpulkan skor sebanyak mungkin.

Fungsi Seleksi Menentukan aksi mana yang paling menguntungkan di antara semua kandidat. Prioritas seleksi disusun sebagai berikut: (1) Mengejar musuh yang membawa banyak diamond, (2) Pulang ke base saat inventory penuh atau waktu hampir habis, (3) Mengambil diamond merah atau biru terdekat, (4) Menghindari musuh saat dalam kondisi berisiko.

Fungsi Kelayakan Memastikan bahwa aksi yang dipilih dapat dijalankan, seperti memastikan langkah valid dan tidak menuju ke musuh.

Fungsi Objektif Mengoptimalkan skor akhir yang disimpan di base, bukan hanya mengumpulkan diamond tapi juga mengamankannya di markas.

3.2 Eksplorasi Alternatif Solusi Greedy

solusi alternatif Greedy yaitu :
-Greedy berdasarkan diamond terdekat
-Greedy berdasarkan prioritas pulang jika diamond penuh
-Greedy berdasarkan mengejar pemainlain

3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy

Pertama mari kami jelaskan kenapa solusi greedy yang kami bahas sebelumnya tidak terlalu efektif berikut penjelasan setiap algoritma greedy yang tidak kami gunakan:
Greedy berdasarkan diamond terdekat

Strategi ini bagus jika permainan hanya memiliki satu pemain jika terdapat pemain lain di game ini maka strategi ini sangat tidak di anjurkan.

Greedy berdasarkan prioritas pulang jika diamond penuh

Strategi ini juga sangat tidak bagus karena kita tidak memikirkan bagaimana ada pemain lain yang mengejar kita saat kita pulang maka diamond akan mudah dirampas pemain lain.

Greedy berdasarkan mengejar pemain lain

Strategi ini kami sebelumnya pertimbangkan tetapi setelah kami fikirkan strategi ini tidak terlalu bagus dikarenakan jika kita selalu mengejar pemain lain maka lebih susah untuk mengumpulkan diamondnya waktu akan habis duluan sebelum merampas diamond dari pemain lain.

3.4 Strategi Greedy yang Dipilih

Strategi greedy yang kami pilih adalah Greedy Prediksi adaptif strategi ini memiliki sistem pemetaan jarak posisi untuk menentukan target diamond,pemain yang lebih dekat, prediksi arah musuh bot kami akan memprediksi pergerakan musuh, bot kami juga akan pulang jika diamond yang dipegang sudah penuh / maksimal ,bot kami juga dapat mempertimbangkan untuk mengambil diamond merah atau diamond biru.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma Greedy

1. Greedybot (malinkhandal)

```
from game.logic.base import BaseLogic
from game.models import Board, GameObject, Position
from typing import Optional
from game.util import get_direction
import random
import math
#Muhammad Farisi Suyitno 123140152
#ali Akbar 123140162
#Bima Aryaseta 123140177
#kelompok 5 Malink handal
class GreedyBot(BaseLogic):
    def __init__(self):
        # arah gerak (kanan, bawah, kiri, atas)
        self.directions = [(1, 0), (0, 1), (-1, 0), (0, -1)]
        self.goal_position: Optional[Position] = None  # tujuan sekarang
        self.current_direction = 0  # arah terakhir
        self.chased_timer = 0  # ngitung berapa lama dikejar musuh
        self.last_position: Optional[Position] = None  # posisi sebelumnya
        self.run = 0  # buat ngitung berapa kali kabur
        self.diamound_counter_cd = 0  # cooldown ambil diamond
        self.stuck_counter = 0  # kalau nggak gerak

    def next_move(self, board_bot: GameObject, board: Board):
        # ambil semua data dari game
        props = board_bot.properties
        current_position = board_bot.position
        all_bots = board.bots
        diamonds = board.diamonds

        # ngitung jumlah diamond di sekitar target
        def count_near_diamond_target(center: Position, radius: int = 3)
-> int:
            return sum(
```

```
                1 for d in diamonds
                if abs(d.position.x - center.x) + abs(d.position.y -
center.y) <= radius
            )

        # jarak pake akar kuadrat
        def distance(pos1: Position, pos2: Position) -> int:
            return math.hypot(pos1.x - pos2.x, pos1.y - pos2.y)

        # cari musuh yang deket banget
        def steal(radius: int = 2):
            return [
                bot for bot in all_bots
                if bot.id != board_bot.id and
                abs(bot.position.x - current_position.x) +
abs(bot.position.y - current_position.y) <= radius
            ]

        # update status dikejar
        nearby_bots = steal(radius=2)
        self.chased_timer = self.chased_timer + 1 if nearby_bots else 0
        if self.diamound_counter_cd > 0:
            self.diamound_counter_cd -= 1
        if self.last_position == current_position:
            self.stuck_counter += 1
        else:
            self.stuck_counter = 0
        self.last_position = current_position

        # kalau stuck 2 turn, ganti arah
        if self.stuck_counter >= 2:
            self.goal_position = None
            self.current_direction = (self.current_direction + 1) %
len(self.directions)
            delta_x, delta_y = self.directions[self.current_direction]
            if board.is_valid_move(current_position, delta_x, delta_y):
                return delta_x, delta_y
            else:
                return 0, 0
```

```
        # kalau udah dikejar 5 turn, langsung balik ke base
        if self.chased_timer >= 5:
            base = props.base
            self.goal_position = base
            delta_x, delta_y = get_direction(current_position.x,
current_position.y, base.x, base.y)
            if board.is_valid_move(current_position, delta_x, delta_y):
                return delta_x, delta_y
            else:
                return 0, 0

        # kalau udah megang 4+ diamond, pulang ke base
        if props.diamonds >= 4:
            base = props.base
            if nearby_bots and self.run <= 2:
                closest = min(nearby_bots, key=lambda b:
distance(current_position, b.position))
                dx = current_position.x - closest.position.x
                dy = current_position.y - closest.position.y
                dodge_x = 1 if dx >= 0 else -1
                dodge_y = 1 if dy >= 0 else -1
                self.goal_position = Position(
                    min(max(0, base.x + dodge_x), board.width - 1),
                    min(max(0, base.y + dodge_y), board.height - 1),
                )
                self.run += 1
            else:
                self.goal_position = base
                self.run += 1
                if self.run >= 5:
                    self.run = 0
            delta_x, delta_y = get_direction(current_position.x,
current_position.y, self.goal_position.x, self.goal_position.y)
            if board.is_valid_move(current_position, delta_x, delta_y):
                return delta_x, delta_y
            else:
                return 0, 0

        # kejar musuh yang bawa banyak diamond
        enemy_carry_diamond = [
```

```
            bot for bot in all_bots
            if bot.id != board_bot.id
            and bot.properties.diamonds >= 2
            and distance(current_position, bot.position) <= 4
        ]

        if enemy_carry_diamond:
            target = max(enemy_carry_diamond, key=lambda b:
b.properties.diamonds)
            if target.position != props.base:
                self.goal_position = target.position
                delta_x, delta_y = get_direction(current_position.x,
current_position.y, self.goal_position.x, self.goal_position.y)
                if board.is_valid_move(current_position, delta_x,
delta_y):
                    return delta_x, delta_y
                else:
                    return 0, 0

        # cari diamond terdekat yang rame
        if self.diamound_counter_cd == 0:
            candidate_diamonds = [d for d in diamonds if
distance(current_position, d.position) <= 5]
            best_diamond = max(
                candidate_diamonds,
                key=lambda d: (
                    count_near_diamond_target(d.position, radius=2),
                    -distance(current_position, d.position)
                ),
                default=None
            )
            if best_diamond:
                self.goal_position = best_diamond.position
            elif diamonds:
                nearest = min(diamonds, key=lambda d:
distance(current_position, d.position))
                self.goal_position = nearest.position
            else:
                self.goal_position = None
```

```
        # kalau ada diamond deket banget
        elif self.diamound_counter_cd > 0:
            close_diamonds = [d for d in diamonds if
distance(current_position, d.position) <= 2]
            if close_diamonds:
                nearest = min(close_diamonds, key=lambda d:
distance(current_position, d.position))
                self.goal_position = nearest.position

        # kalau ada tujuan jelas, kejar
        if self.goal_position and self.goal_position != current_position:
            delta_x, delta_y = get_direction(current_position.x,
current_position.y, self.goal_position.x, self.goal_position.y)
            if board.is_valid_move(current_position, delta_x, delta_y):
                if self.goal_position == current_position:
                    self.diamound_counter_cd = 4
                return delta_x, delta_y
            else:
                return 0, 0
        else:
            # roaming nyari arah yang bisa dilalui
            for _ in range(4):
                delta_x, delta_y = self.directions[self.current_direction]
                if board.is_valid_move(current_position, delta_x,
delta_y):
                    if random.random() > 0.6:
                        self.current_direction = (self.current_direction +
1) % len(self.directions)
                    return delta_x, delta_y
                else:
                    self.current_direction = (self.current_direction + 1)
% len(self.directions)

        # fallback terakhir kalau nggak bisa kemana-mana
        return 0, 0
```

Dapat dilihat Greedybot dibuat dengan strategi greedy prediktif, bot ini dibuat untuk mengejar dengan prioritas terdekat yang terbaik, seperti mengambil diamond merah terdekat, memotong jalur pemain, dan pulang ke base jika diamond sudah penuh

2. **Penjelasan Alur Program**

Implementasi logika greedy dilakukan dalam fungsi next_move() dengan alur sebagai berikut:

Inisialisasi:

directions: daftar arah gerak dasar.

current_direction: arah acak saat roaming.

Prediksi Musuh:

Jika musuh memiliki 5 diamond, prediksi mereka akan ke base.

Jika musuh memiliki 3–4 diamond, prediksi mereka ke diamond terdekat.

Jika posisi prediksi berjarak ≤ 6 blok, bot memotong jalur.

Logika Diamond:

Ambil diamond merah terdekat jika ada dan aman.

Jika tidak ada, ambil biru terdekat.

Kondisi Pulang:

Jika diamond = 4 atau waktu tinggal < 1500 ms, bot pulang ke base.

Menghindari Musuh:

Jika musuh berada di sekitar dan bot membawa banyak diamond, hindari posisi mereka.

Fallback Movement:

Jika tidak ada target valid, bot melakukan roaming ke arah yang bisa dilewati.

4.2 Struktur Data yang Digunakan

List arah (directions): Navigasi arah gerak dasar.

Set posisi musuh: Menghindari tabrakan.

Fungsi prediksi (predict_enemy_target): Prediksi gerak musuh berdasarkan jumlah diamond.

List diamond merah dan biru: Prioritaskan pengambilan diamond sesuai urgensi dan nilai.

Fungsi try_move(): Validasi gerak dan menghindari posisi musuh.

4.3 Pengujian Program

1. **Skenario Pengujian**
    a. Bot mampu mengejar musuh saat membawa diamond.
    b. Bot menghindar saat inventory penuh.
    c. Bot konsisten mengumpulkan diamond terdekat bernilai tinggi.
    d. Bot dapat pulang saat waktu habis atau inventory penuh.
2. **Hasil Pengujian dan Analisis**
    a. Bot akan stuck jika area terlalu padat dan banyak musuh.
    b. Jika prediksi musuh meleset, bot kehilangan kesempatan.
    c. Bot mampu beradaptasi di papan dengan diamond padat atau musuh agresif.
    d. Bot tidak AFK dan mampu keluar dari kondisi stuck secara efisien

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan dari kami untuk bot Greedy bahwa strategi greedy dengan manajemen kondisi darurat seperti pengerjaran musuh,stuck detection, dan cooldown terhadap pengambilan diamond dapat meningkatkan efisiensi bot dalam permainan, bot harus dirancang untuk tidak hanya mencari diamond tapi juga menghitung kapan harus kembali ke base.

5.2 Saran

saran dari kami untuk mempermudah penggunaan user interface dari program etimo diamond karena system ini sangat tidak mudah untuk digunakan seperti cara setup dan lainnnya, dan terkadang juga sangat susah untuk memahami kode kode awal yang diberikan dari startup kit yang diberikan.

LAMPIRAN

A. **Repository Github**
   https://github.com/IcniP/tubes1-IF2211-bot-starter-pack-1.0.1
B. **Video Penjelasan (link GDrive)**

https://drive.google.com/file/d/1VRYJBDy3pwhaL44DumEU5OZZ-6LM5SSp/view?usp=drive_link

DAFTAR PUSTAKA

R. Munir, "Algoritma Greedy," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Informatika, Sekolah Teknik Elektro dan Informatika ITB, 2020.

Institut Teknologi Sumatera. (2024). Modul Strategi Algoritma greedy bagian 1 IF2211 [PDF]. https://kuliah.itera.ac.id/mod/resource/view.php?id=32758

Institut Teknologi Sumatera. (2024). Modul Strategi Algoritma greedy bagian 2&3 IF2211 [PDF]. https://kuliah.itera.ac.id/mod/resource/view.php?id=33150

Haziq Amrullah. (2024). tubes1-IF2211-game-engine (v1.1.0). GitHub. https://github.com/haziqam/tubes1-IF2211-game-engine/releases/tag/v1.1.0

Haziq Amrullah. (2024). tubes1-IF2211-bot-starter-pack (v1.0.1). GitHub. https://github.com/haziqam/tubes1-IF2211-bot-starter-pack/releases/tag/v1.0.1

Institut Teknologi Sumatera. (2024). Implementasi Algoritma Greedy dalam Permainan Diamonds [PDF]. https://drive.google.com/file/d/17_d7sRWhr0TspjS0ZqIIQCnQnElPaeDR/view

IF2211 Strategi Algoritma – Institut Teknologi Bandung. (2024). Tugas Besar 1 dan 2: Permainan Diamonds. Panduan dan dokumentasi teknis. https://docs.google.com/document/d/1L92Axb89yIkom0b24D350Z1QAr8rujvHof7-kXRAp7c