

PEMANFAATAN ALGORITMA GREEDY DALAM PEMBUATAN BOT PERMAINAN DIAMONDS

**Strategi Algoritma
Semester Genap 2024/2025
Kelas IF2211**



Oleh:

- | | |
|--|------------------|
| 1. KHAIRUL RIJAL SYAUQI | 123140143 |
| 2. FALIH FAIQ FADHLURRAHMAN | 123140129 |
| 3. MUHAMMAD DAFFANSYAH DESUANDI | 123140127 |

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

Daftar Isi

Daftar Isi	2
BAB I	
Deskripsi Tugas	4
1.1 Spesifikasi Tugas.....	4
BAB II	
Landasan Teori	6
2.1 Dasar Teori Algoritma Greedy	6
2.2 Cara Kerja Program.....	6
BAB III	
Aplikasi Strategi Greedy	8
3.1 Proses Mapping Persoalan Diamonds ke Algoritma Greedy	8
Himpunan Kandidat (Candidate Set)	8
Himpunan Solusi (Solution Set)	8
Fungsi Solusi (Solution Function)	8
Fungsi Seleksi (Selection Function)	8
Fungsi Kelayakan (Feasibility Function).....	8
Fungsi Objektif (Objective Function).....	9
3.2 Eksplorasi Alternatif Solusi Greedy	9
Alternatif 1: Greedy by Distance	9
Alternatif 2: Greedy by Value.....	9
Alternatif 3: Greedy by Efficiency (Value per Distance)	10
Alternatif 4: Greedy by Opportunity.....	10
Alternatif 5: Aggressive Greedy	10
3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy	11
Analisis Alternatif 1: Greedy by Distance	11
Analisis Alternatif 2: Greedy by Value.....	11
Analisis Alternatif 3: Greedy by Efficiency	11
Analisis Alternatif 4: Greedy by Opportunity.....	12
Analisis Alternatif 5: Aggressive Greedy	12
3.4 Strategi Greedy yang digunakan	12
BAB IV	
Implementasi dan Pengujian.....	13
4.2 Struktur Data dalam Algoritma Greedy.....	15
4.2.1 Game.....	15
4.2.2 Logic.....	18
4.3 Pengujian Program	19
4.3.1 Skenario Pengujian	19
4.3.2 Hasil Pengujian dan Analisis.....	22
BAB V	
Kesimpulan dan Saran	24

5.1 Kesimpulan	24
5.2 Saran	24
LAMPIRAN.....	25
DAFTAR PUSTAKA.....	26

BAB I

Deskripsi Tugas

1.1 Spesifikasi Tugas

- Buatlah program sederhana dalam bahasa Python yang mengimplementasikan algoritma Greedy pada bot permainan Diamonds dengan tujuan memenangkan permainan.
- Tugas dikerjakan berkelompok dengan anggota minimal 2 orang dan maksimal 3 orang, boleh lintas kelas dan lintas kampus.
- Strategi greedy yang diimplementasikan setiap kelompok harus dikaitkan dengan fungsi objektif dari permainan ini, yaitu memenangkan permainan dengan memperoleh diamond sebanyak banyak nya dan jangan sampai diamond tersebut diambil oleh bot lain. Buatlah strategi greedy terbaik, karena setiap bot dari masing-masing kelompok akan diadu dalam kompetisi Tubes 1.
- Strategi greedy yang kelompok anda buat harus dijelaskan dan ditulis secara eksplisit pada laporan, karena akan diperiksa saat demo apakah strategi yang dituliskan sesuai dengan yang diimplementasikan. Tiap kelompok dapat menggunakan kreativitas yang bermacam macam dalam menyusun strategi greedy untuk memenangkan permainan. Implementasi pemain harus dapat dijalankan pada game engine yang telah disebutkan diatas serta dapat dikompetisikan dengan bot dari kelompok lain.
- Program harus mengandung komentar yang jelas, dan untuk setiap strategi greedy yang disebutkan, harus dilengkapi dengan kode sumber yang dibuat.
- Mahasiswa dilarang menggunakan kode program yang diunduh dari Internet. Mahasiswa harus membuat program sendiri, diperbolehkan untuk belajar dari program yang sudah ada.
- Mahasiswa dianggap sudah melihat dokumentasi dari game engine, sehingga tidak terjadi kesalahpahaman spesifikasi antara mahasiswa dan asisten.
- BONUS (maks 10): Membuat video tentang aplikasi greedy pada bot serta simulasinya pada game kemudian mengunggahnya di Youtube. Video dibuat harus memiliki audio dan menampilkan wajah dari setiap anggota kelompok. Untuk contoh video tubes stima tahun-tahun sebelumnya dapat dilihat di Youtube dengan kata kunci “Tubes Stima”, “strategi algoritma”, “Tugas besar stima”, dll.

- Jika terdapat kesulitan selama mengerjakan tugas besar sehingga memerlukan bimbingan, maka dapat melakukan asistensi tugas besar kepada asisten (opsional). Dengan catatan asistensi hanya bersifat membimbing, bukan memberikan “jawaban”.
- Terdapat juga demo dari program yang telah dibuat. Pengumuman tentang demo menunggu pemberitahuan lebih lanjut dari asisten.
- Bot yang telah dibuat akan dikompertisikan dengan kelompok lain dan disaksikan oleh seluruh peserta kuliah. Terdapat hadiah menarik bagi kelompok yang memenangkan kompetisi.
- Setiap kelompok harap melaporkan nama kelompok dan anggotanya dan diserahkan kepada asisten untuk didata.
- Kelompok yang terindikasi melakukan kecurangan akan diberikan nilai 0 pada tugas besar.
- Program disimpan dalam repository yang bernama Tubes1_NamaKelompok dengan nama kelompok. Berikut merupakan struktur dari isi repository tersebut:
 - a. Folder src berisi source code.
 - b. Folder doc berisi laporan tugas besar dengan format NamaKelompok.pdf
 - c. README untuk tata cara penggunaan yang minimal berisi:
 - i. Penjelasan singkat algoritma greedy yang diimplementasikan
 - ii. Requirement program dan instalasi tertentu bila ada
 - iii. Command atau langkah-langkah dalam meng-compile atau build program
 - iv. Author (identitas pembuat)

BAB II

Landasan Teori

2.1 Dasar Teori Algoritma Greedy

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

Program permainan Diamonds yang dibahas pada laporan ini dijalankan dengan menggunakan 2 program, yaitu program permainan yang menjalankan papan permainan dan program bot yang akan memberi perintah untuk gerakan bot yang dimainkan di dalam Diamonds.

Program yang menjalankan permainan akan menginisialisasi papan permainan di server. Selama permainan berlangsung, program ini akan terus memperbarui dan menyediakan data permainan terkini yang dapat diakses oleh program bot untuk menjalankan fungsinya.

Setelah program permainan dijalankan, program bot dapat dieksekusi. Bot ini akan mengambil data dari papan permainan, memproses informasi tersebut, lalu memberikan masukan berupa arah gerak bot di papan permainan. Proses ini berlangsung secara berulang selama permainan berjalan. Di sinilah strategi greedy diterapkan. Bot akan menganalisis data dari papan untuk menentukan arah geraknya pada giliran berikutnya. Contoh dari penerapan strategi greedy adalah bot yang memilih untuk segera mengambil diamond terdekat dan membawanya langsung ke base. Alternatif lainnya, bot dapat memilih untuk mengumpulkan sebanyak mungkin diamond terlebih dahulu sebelum kembali ke base.

Berbagai pendekatan inilah yang menjadi sarana implementasi algoritma greedy. Setiap bot dirancang untuk menjalankan metode tertentu yang menggunakan prinsip greedy, dengan tujuan memperoleh jumlah diamond sebanyak mungkin dalam waktu 60 detik.

BAB III

Aplikasi Strategi Greedy

3.1 Proses Mapping Persoalan Diamonds ke Algoritma Greedy

Persoalan permainan Diamonds dipetakan ke dalam elemen-elemen algoritma greedy untuk memungkinkan implementasi strategi pengambilan keputusan lokal yang optimal. Berikut adalah pemetaan yang dilakukan:

Himpunan Kandidat (Candidate Set)

Himpunan kandidat terdiri dari semua pilihan aksi yang dapat dilakukan bot pada setiap langkah permainan, yaitu empat arah gerakan (NORTH, SOUTH, EAST, WEST) untuk mencapai berbagai target seperti diamond biru (1 poin), diamond merah (2 poin), red button, teleporter, atau kembali ke base.

Himpunan Solusi (Solution Set)

Himpunan solusi merupakan urutan keputusan dan gerakan yang telah dibuat bot sepanjang permainan. Setiap langkah yang dipilih akan membentuk path pengumpulan diamond yang mengarah pada pencapaian score maksimal.

Fungsi Solusi (Solution Function)

Fungsi ini menentukan apakah strategi bot telah lengkap dan valid, yaitu ketika bot berhasil mencapai target yang ditentukan tanpa melanggar aturan permainan atau ketika waktu permainan telah habis.

Fungsi Seleksi (Selection Function)

Fungsi seleksi menentukan pilihan terbaik dari kandidat yang tersedia berdasarkan kriteria tertentu, seperti prioritas nilai diamond (diamond merah lebih diprioritaskan), jarak terdekat menggunakan Manhattan distance, atau rasio nilai diamond terhadap jarak yang diperlukan.

Fungsi Kelayakan (Feasibility Function)

Fungsi kelayakan memastikan setiap pilihan tidak melanggar batasan permainan, termasuk tidak keluar dari boundaries board, tidak melebihi kapasitas inventory maksimum, memiliki waktu yang cukup untuk kembali ke base, dan menghindari posisi yang berpotensi tackle dengan bot lawan.

Fungsi Objektif (Objective Function)

Fungsi objektif bertujuan memaksimalkan total score yang diperoleh dari pengumpulan diamond sambil meminimalkan risiko kehilangan diamond akibat tackle dan mengoptimalkan efisiensi waktu pergerakan bot.

3.2 Eksplorasi Alternatif Solusi Greedy

Dalam mengembangkan strategi bot untuk permainan Diamonds, kami mengeksplorasi lima alternatif solusi greedy yang berbeda. Setiap alternatif memiliki pendekatan yang unik dalam menentukan pilihan optimal pada setiap langkah:

Alternatif 1: Greedy by Distance

Deskripsi: Strategi yang selalu memilih diamond terdekat dari posisi bot saat ini.

Cara Kerja:

1. Pada setiap turn, hitung jarak Manhattan dari posisi bot ke semua diamond yang tersedia
2. Pilih diamond dengan jarak terpendek sebagai target
3. Bergerak menuju target menggunakan jalur langsung (prioritas horizontal kemudian vertikal)
4. Ulangi proses sampai inventory penuh atau tidak ada diamond tersisa
5. Kembali ke base untuk menyimpan diamond

Alternatif 2: Greedy by Value

Deskripsi: Strategi yang memprioritaskan diamond berdasarkan nilai poin yang diberikan.

Cara Kerja:

1. Urutkan semua diamond berdasarkan nilai (diamond merah prioritas tertinggi)
2. Dari diamond dengan nilai sama, pilih yang memiliki jarak terdekat
3. Selalu kejar diamond merah terlebih dahulu, baru diamond biru
4. Abaikan pertimbangan jarak jika masih ada diamond merah tersedia

Alternatif 3: Greedy by Efficiency (Value per Distance)

Deskripsi: Strategi yang memilih diamond berdasarkan rasio nilai terhadap jarak tempuh.

Cara Kerja:

1. Untuk setiap diamond, hitung efisiensi = $\text{nilai_diamond} / \text{jarak_manhattan}$
2. Pilih diamond dengan nilai efisiensi tertinggi
3. Pertimbangkan baik nilai diamond maupun jarak tempuh secara seimbang
4. Dinamis menyesuaikan pilihan berdasarkan posisi bot saat ini

Alternatif 4: Greedy by Opportunity

Deskripsi: Strategi yang mempertimbangkan kompetisi dengan musuh dalam mengambil diamond.

Cara Kerja:

1. Identifikasi posisi semua musuh di peta
2. Hitung kemungkinan setiap diamond akan diambil musuh berdasarkan jarak
3. Prioritaskan diamond yang kemungkinan besar akan diambil musuh jika tidak segera diambil
4. Kombinasikan dengan faktor jarak untuk menentukan pilihan final
5. "Potong" jalur musuh menuju diamond berharga

Alternatif 5: Aggressive Greedy

Deskripsi: Strategi yang fokus menyerang musuh untuk mengambil diamond mereka melalui mekanisme tackle.

Cara Kerja:

1. Monitor inventory semua musuh di peta
2. Identifikasi musuh dengan diamond terbanyak atau bernilai tinggi
3. Jika ada musuh yang menguntungkan untuk diserang, kejar dan tackle
4. Pertimbangkan risiko vs reward: hanya serang jika inventory musuh > threshold tertentu
5. Fallback ke strategi pengumpulan normal jika tidak ada target musuh

3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy

Analisis Alternatif 1: Greedy by Distance

Efisiensi:

- Kompleksitas waktu: $O(n)$ dimana n adalah jumlah diamond
- Kompleksitas ruang: $O(1)$ untuk penyimpanan target
- Sangat efisien untuk implementasi real-time

Efektivitas:

- Baik untuk peta dengan distribusi diamond merata
- Kurang optimal jika banyak diamond merah berjauhan
- Konsisten dalam performa, jarang stuck

Analisis Alternatif 2: Greedy by Value

Efisiensi:

- Kompleksitas waktu: $O(n \log n)$ karena perlu sorting berdasarkan nilai
- Memerlukan tracking lebih kompleks

Efektivitas:

- Lebih optimal dalam hal poin total
- Risiko terjebak mengejar diamond merah yang jauh
- Bisa kalah dalam kecepatan pengumpulan

Analisis Alternatif 3: Greedy by Efficiency

Efisiensi:

- Kompleksitas waktu: $O(n)$ dengan perhitungan rasio
- Balanced antara waktu dan ruang

Efektivitas:

- Optimal secara teoritis untuk memaksimalkan poin per waktu
- Kompleks implementasi untuk real-time decision
- Bagus untuk endgame strategy

Analisis Alternatif 4: Greedy by Opportunity

Efisiensi:

- Kompleksitas waktu: $O(n \times m)$ dimana m adalah jumlah musuh
- Memerlukan tracking posisi dan pergerakan musuh

Efektivitas:

- Sangat baik dalam kompetisi melawan bot lain
- Memerlukan prediksi pergerakan musuh yang akurat
- Risiko kalah dalam kecepatan jika prediksi salah

Analisis Alternatif 5: Aggressive Greedy

Efisiensi:

- Kompleksitas waktu: $O(m)$ untuk tracking musuh
- Sederhana dalam implementasi

Efektivitas:

- Sangat menguntungkan jika berhasil tackle musuh
- Berisiko tinggi jika gagal tackle
- Memerlukan timing yang tepat

3.4 Strategi Greedy yang digunakan

Berdasarkan hasil evaluasi terhadap berbagai strategi greedy pada bagian sebelumnya, mulai dari *Greedy by Distance*, *Greedy by Value*, *Greedy by Efficiency (Value per Distance)*, *Greedy by Opportunity*, hingga *Aggressive Greedy* kelompok kami memilih untuk menggunakan ***Greedy by Distance*** sebagai pendekatan utama dalam pengembangan bot. Keputusan ini diambil dengan mempertimbangkan durasi pertandingan Diamonds yang hanya berlangsung selama 60 detik, di mana diasumsikan bot dapat melakukan satu gerakan setiap detik. Dalam rentang waktu yang sangat terbatas tersebut, tidak cukup hanya mengejar diamond terdekat atau diamond bernilai tinggi saja, melainkan dibutuhkan keseimbangan antara jarak tempuh dan poin yang diperoleh. Strategi Greedy Berdasarkan Rasio Nilai-Jarak memungkinkan bot untuk memprioritaskan diamond yang memberikan “nilai per satuan jarak” tertinggi yakni diamond dengan rasio terbesar—sehingga setiap detik gerakan bot dioptimalkan untuk mendapatkan total poin maksimum dalam waktu singkat.

BAB IV

Implementasi dan Pengujian

4.1 Implementasi Algoritma Greedy dalam Pseudocode

Implementasi pada program naruto.py

```
prosedur next_move (board_bot: GameObject, board: Board) ;

props := board_bot.properties ;
board_game := board ;
diamonds := board.diamonds ;
bots := board.bots ;
teleporter := [d for d in board.game_objects if d.type = "TeleportGameObject"] ;
tombolMerah := [d for d in board.game_objects if d.type = "DiamondButtonGameObject"] ;
musuh := [d for d in bots if d.id != board_bot.id] ;
diamondMusuh := [d.properties.diamonds for d in musuh] ;

{ Reset static ketika di base }
jika board_bot.position = props.base maka
    mulai
        static_goals := [] ;
        static_goal_teleport := null ;
        static_temp_goals := null ;
        static_direct_to_base_via_teleporter := false ;
    akhir ;

{ Update static goals }
jika static_goal_teleport != null dan board_bot.position = find_other_teleport(static_goal_teleport)
maka
    mulai
        static_goals.hapus(static_goal_teleport.position) ;
        static_goal_teleport := null ;
    akhir ;

jika static_goal_teleport = null dan board_bot.position di static_goals maka
    static_goals.hapus(board_bot.position) ;

{ Reset temp goal jika tercapai }
jika board_bot.position = static_temp_goals maka
    static_temp_goals := null ;

{ Atur strategi utama }
```

```

jika props.diamonds = 5 atau (props.milliseconds_left < 5000 dan props.diamonds > 1) maka
    mulai
        goal_position := find_best_way_to_base() ;
        jika tidak static_direct_to_base_via_teleporter maka
            mulai
                static_goals := [] ;
                static_goal_teleport := null ;
            akhir ;
        akhir
    lain
        mulai
            jika panjang(static_goals) = 0 maka
                find_nearest_diamond() ;
                goal_position := static_goals[0] ;
            akhir ;

{ Strategi tambahan }
jika calculate_near_base() dan props.diamonds > 2 maka
    mulai
        goal_position := find_best_way_to_base() ;
        jika tidak static_direct_to_base_via_teleporter maka
            mulai
                static_goals := [] ;
                static_goal_teleport := null ;
            akhir ;
    akhir ;

jika static_temp_goals != null maka
    goal_position := static_temp_goals ;

{ Strategi serang musuh }
posisiSekarang := board_bot.position ;
jika musuh dan props.diamonds < 3 maka
    mulai
        urutkan musuh berdasarkan jarak dari posisiSekarang ;
        jarakGoal := |goal_position.x - posisiSekarang.x| + |goal_position.y - posisiSekarang.y| ;
        jarakMusuh := |musuh[0].position.x - posisiSekarang.x| + |musuh[0].position.y - posisiSekarang.y|
    ;
        jika musuh[0].properties.diamonds > 2 dan jarakMusuh < jarakGoal maka
            goal_position := musuh[0].position ;
        akhir ;

{ Hitung arah gerakan }

```

```

jika goal_position != null maka
  mulai
    jika static_temp_goals = null maka
      obstacle_on_path('teleporter', posisiSekarang.x, posisiSekarang.y, goal_position.x,
goal_position.y) ;

    jika props.diamonds = 4 maka
      obstacle_on_path('redDiamond', posisiSekarang.x, posisiSekarang.y, goal_position.x,
goal_position.y) ;

    delta_x, delta_y := get_direction(posisiSekarang.x, posisiSekarang.y, goal_position

```

4.2 Struktur Data dalam Algoritma Greedy

Struktur data pada permainan Diamonds ini terdiri dari kelas-kelas yang terbagi dalam beberapa file. Terdapat dua folder yang digunakan dalam pengembangan bot yaitu, game dan logic.

4.2.1 Game

Folder game berisi pendefinisian model-model objek, posisi, peta, serta aksi yang valid dalam permainan. Berikut file yang ada dalam folder ini.

A. api.py

api.py	
Atribut	<ul style="list-style-type: none"> • String url
Method	<ul style="list-style-type: none"> • def _get_url(self, endpoint: str) -> str • def _req(self, endpoint: str, method: str, body: dict) -> Response • def bots_get(self, bot_token: str) -> Optional[Bot] • def bots_register(self, name: str, email: str, password: str, team: str) -> Optional[Bot] • def boards_list(self) -> Optional[List[Board]] • def bots_join(self, bot_token: str, board_id: int) -> bool • def boards_get(self, board_id: str) -> Optional[Board] • def bots_move(self, bot_token: str, direction: str) -> Optional[Board] • def bots_recover(self, email: str, password: str) -> Optional[str] • def _return_response_and_status(self, response: Response) -> Tuple[Union[dict, List], int]

B. board_handler.py

board_handler.py	
Atribut	<ul style="list-style-type: none"> • Api api
Method	<ul style="list-style-type: none"> • def list_boards(self) -> List[Board] • def get_board(self, board_id: int) -> Board

C. bot_handler.py

bot_handler.py	
Atribut	<ul style="list-style-type: none"> • Api api
Method	<ul style="list-style-type: none"> • def _get_direction(dx: int, dy: int) • def get_my_info(self, token: str) -> Bot • def join(self, token: str, board_id: int) -> bool • def move(self, token: str, board_id: int, dx: int, dy: int) -> Optional[Board] • def register(self, name: str, email: str, password: str, team: str) -> Optional[Bot] • def recover(self, email: str, password: str) -> Optional[str]

D. models.py

Kelas Bot	
Atribut	<ul style="list-style-type: none"> • String name ○ • String email ○ • String id
Method	-
Kelas Position	
Atribut	<ul style="list-style-type: none"> • Int y • Int x

Method	-
Kelas Properties	
Atribut	<ul style="list-style-type: none"> ● Optional[int] points ○ Optional[int] pair_id ○ ● Optional[int] diamonds ○ ● Optional[int] score ○ ● Optional[int] name ○ ● Optional[int] inventory_size ○ ● Optional[int] can_tackle ○ ● Optional[int] milliseconds_left ○ ● Optional[int] time_joined ○ ● Optional[int] base
Method	-
Kelas Game Object	
Atribut	<ul style="list-style-type: none"> ● Position position ○ ● String type ○ ● Optional[Properties] properties
Method	-
Kelas Config	
Atribut	<ul style="list-style-type: none"> ● Optional[float] generation_ratio ○ ● Optional[float] min_ratio_for_generation ○ ● Optional[float] red_ratio ○ ● Optional[int] seconds ○ ● Optional[int] pairs ○ ● Optional[int] inventory_size ○ ● Optional[bool] can_tackle
Method	-
Kelas Feature	

Atribut	<ul style="list-style-type: none"> ● String name ○ ● Optional[Config] config
Method	-
Kelas Board	
Atribut	<ul style="list-style-type: none"> ● Int id ○ Int width ○ ● Int height ○ ● List[Feature] features ○ ● Int minimum_delay_between_moves ○ ● Optional[List[GameObject]] game_objects
Method	<ul style="list-style-type: none"> ● def bots(self) -> List[GameObject] ○ ● def diamonds(self) -> List[GameObject] ○ ● def get_bot(self, bot: Bot) -> Optional[GameObject] ○ ● def is_valid_move(self, current_position: Position, delta_x: int, delta_y: int) -> bool

E. util.py

util.py	
Atribut	-
Method	<ul style="list-style-type: none"> ● def clamp(n, smallest, largest) ● def get_direction(current_x, current_y, dest_x, dest_y) 36 Lap ● def position_equals(a: Position, b: Position)

4.2.2 Logic

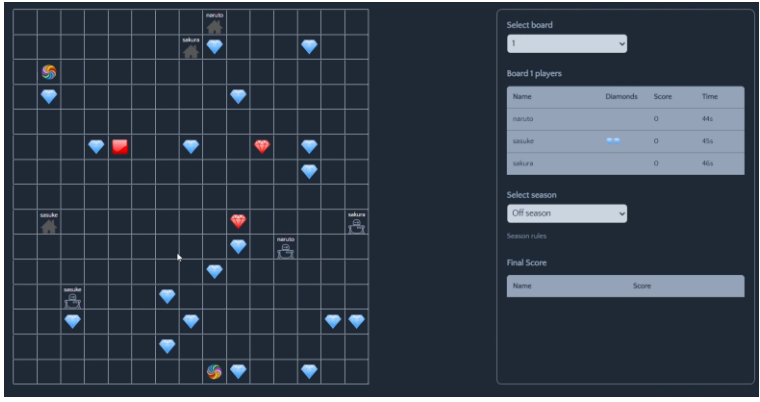
Naruto.py	
Atribut	<ul style="list-style-type: none"> ● List[int] direction ○ ● Optional[Position] goal_position ○ ● Int current_direction ○ ● Int distance

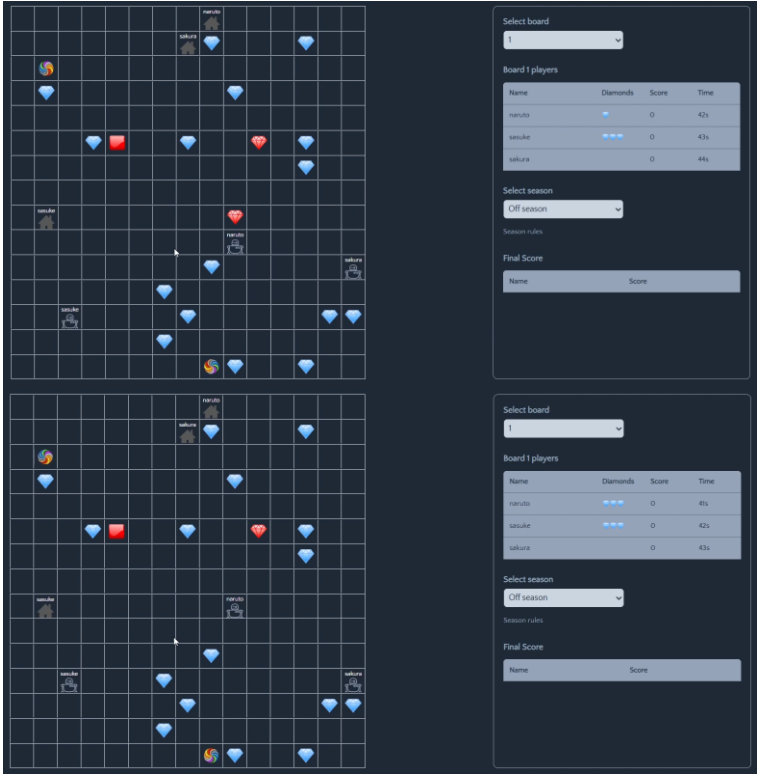
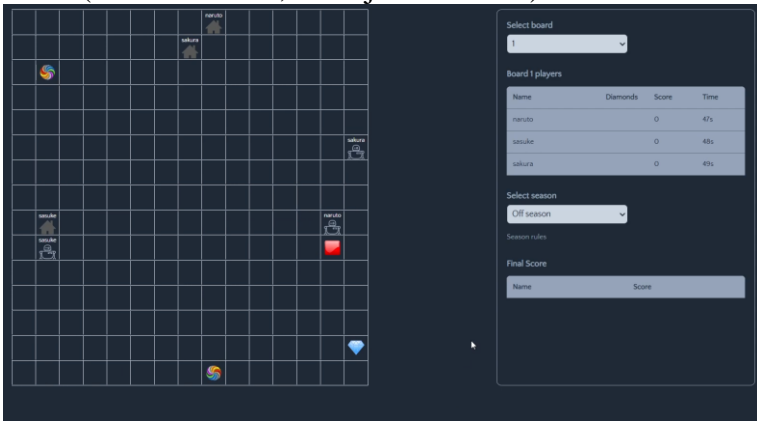
Method	<ul style="list-style-type: none"> • def next_move(self, board_bot: GameObject, board: Board) • def find_best_way_to_base(self) • def calculate_near_base(self) • def find_base_distance_teleporter(self) • def find_nearest_diamond(self) • def find_nearest_red_button(self) • def find_nearest_teleport(self) • def find_other_teleport(self, teleport: GameObject) • def find_nearest_diamond_teleport(self) • def find_nearest_diamond_direct(self) • def obstacle_on_path(self, type, current_x, current_y, dest_x, dest_y)
--------	--

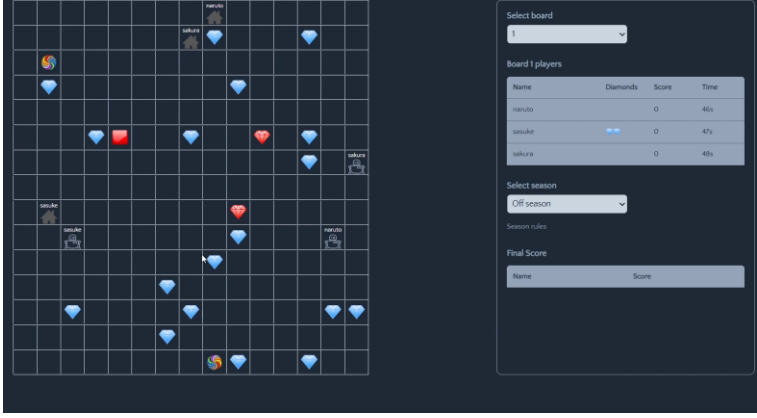

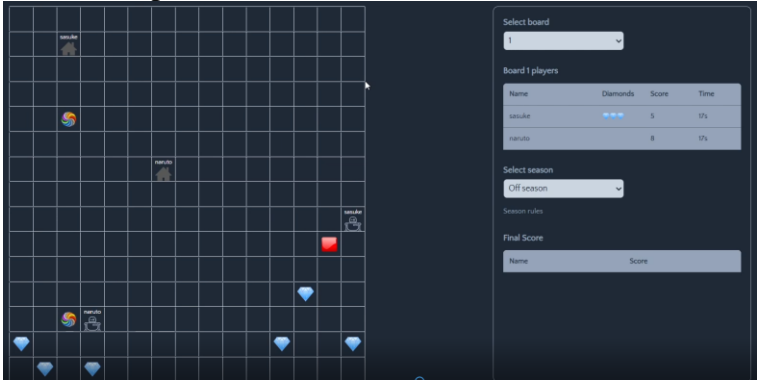
4.3 Pengujian Program

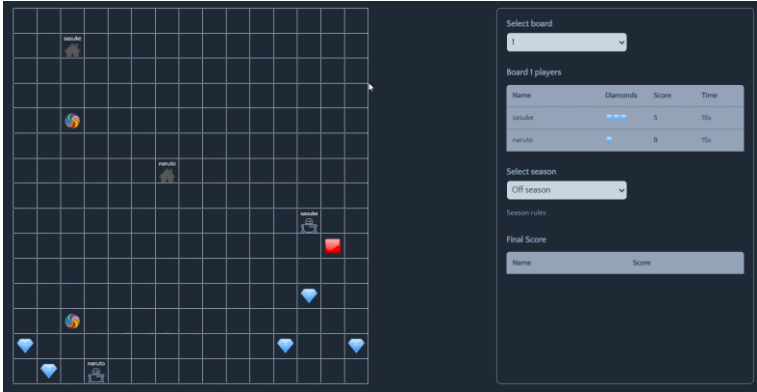
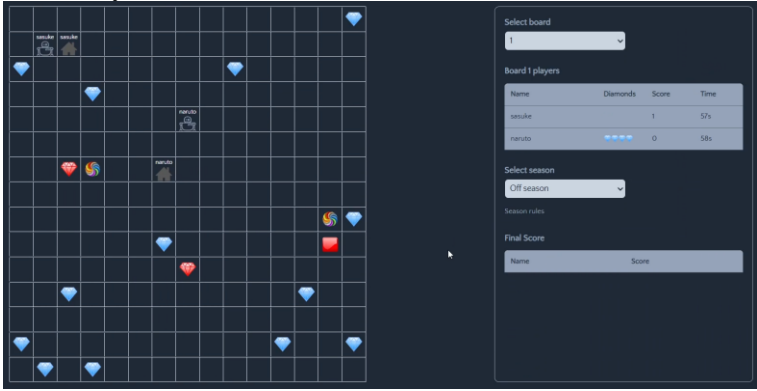
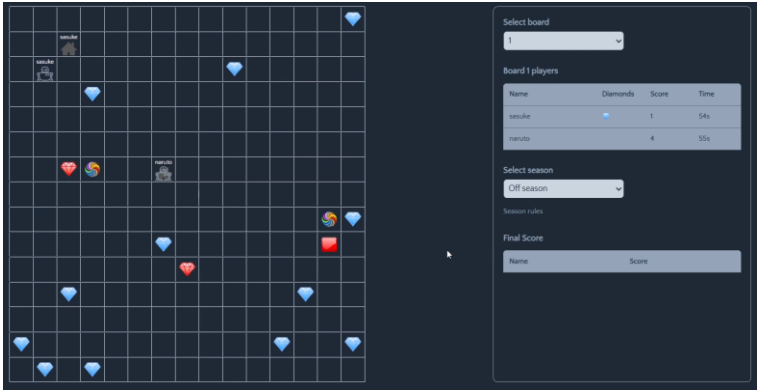
Pada bagian ini akan dilakukan pengujian terhadap beberapa fitur dari bot naruto yang dipilih, yakni ***Greedy by Distance***

4.3.1 Skenario Pengujian

No	Kondisi	Keterangan
1.	<p>Awal</p>  <p>Setelah mengambil diamond</p>	<p>Memperlihatkan bahwa bot mengambil diamond biru yang terdekat dari posisi awal bot, setelah itu mencari lagi diamond yang terdekat dari posisi bot. Ini karena bot menggunakan algoritma <i>Greedy by Distance</i>, Strategi yang selalu memilih diamond terdekat dari posisi bot saat ini.</p>

		
2.	<p>Awal (diamond habis, menuju red button)</p>  <p>Setelahnya (red button dilewati, diamond di-generate)</p>	<p>Memperlihatkan dari kedua gambar, bot segera menuju ke red button dikarenakan di board hanya tersisa satu diamond dan lebih jauh daripada red button. Dipastikan bahwa implementasi untuk red button sukses.</p>

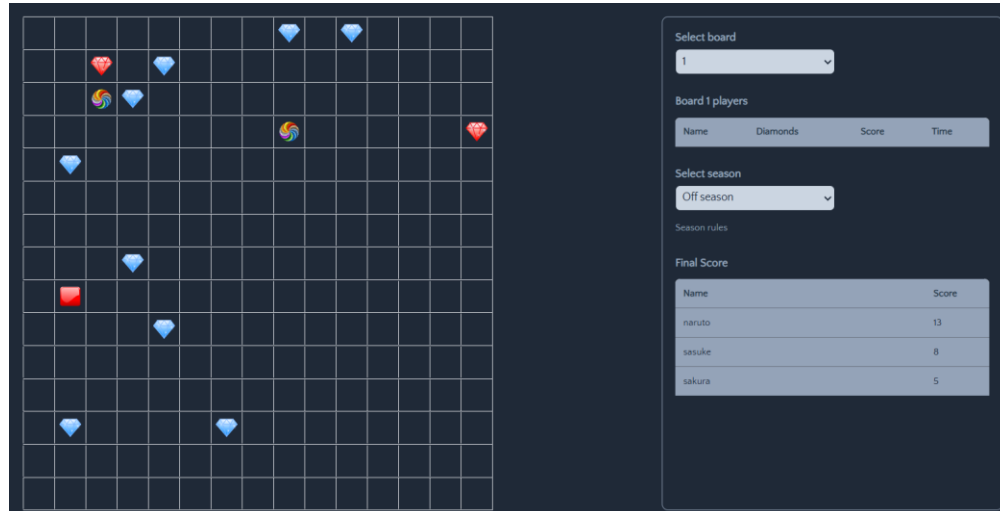
		
<p>3.</p>	<p>Sebelum teleport</p>  <p>Setelah teleport</p> 	<p>Memperlihatkan bot sebelum dan setelah teleportasi, bot melakukan teleportasi dikarenakan jarak antara bot dan diamond lebih dekat melewati teleport daripada berjalan secara langsung menuju diamond. Dipastikan bahwa implementasi untuk teleport berjalan dengan baik.</p>

		
4.	<p>Inventory - 1</p>  <p>Kembali ke base</p> 	<p>Memperlihatkan bahwa bot akan kembali ke base saat jumlah diamond yang dibawanya mencapai 4. Dalam perjalanan kembali, bot akan memeriksa apakah ada diamond di sekitarnya. Jika ditemukan, bot akan mengambil diamond tersebut terlebih dahulu, lalu melanjutkan ke base. Jika tidak ada diamond di sekitar, bot akan langsung kembali ke base.</p>

4.3.2 Hasil Pengujian dan Analisis

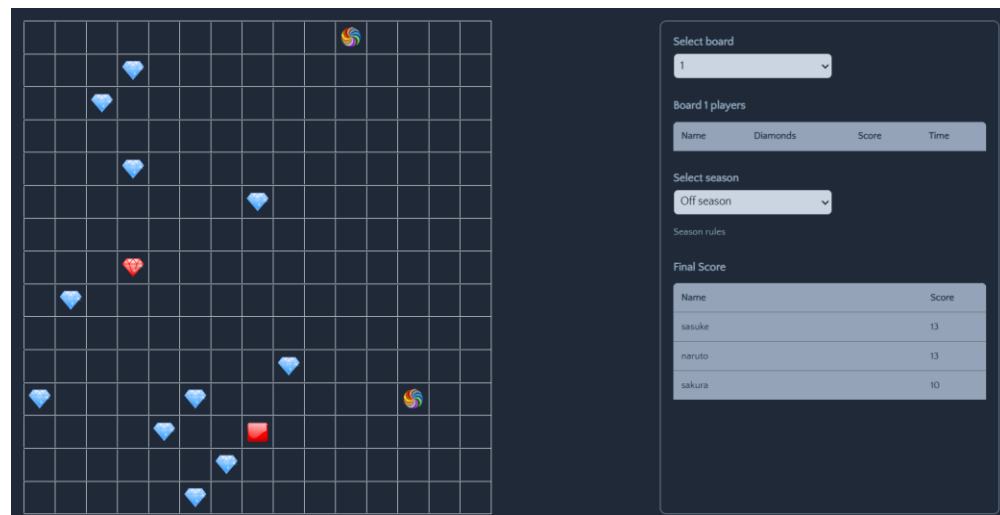
Pengujian bot naruto versus bot sasuke dan sakura.

Pengujian 1:



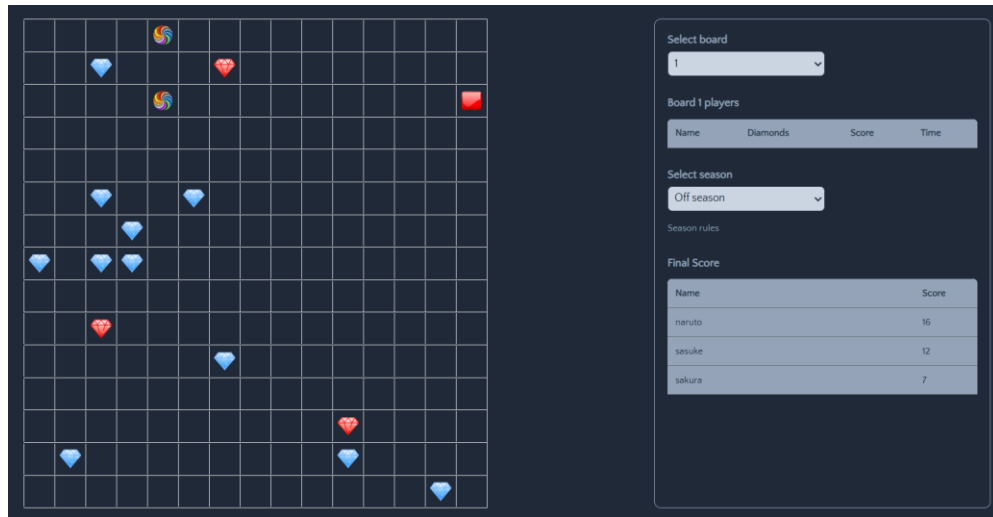
Pengujian pertama dimenangkan oleh bot naruto diikuti dengan sasuke lalu sakura.

Pengujian 2:



Pengujian kedua terjadi seri antara bot naruto dan sasuke, disini sasuke men-tackle naruto sehingga diamondnya terambil.

Pengujian 3:



Pengujian ketiga dimenangkan kembali oleh bot naruto.

Hasil analisis kami terhadap pengujian yang telah dilakukan menunjukkan bahwa bot Naruto, yang menggunakan strategi ***Greedy by Distance***, cenderung lebih cepat dalam merespons diamond terdekat dan memiliki performa yang baik dalam pathfinding. Namun, pendekatan greedy ini memiliki kelemahan dalam beberapa kasus, seperti ketika diamond yang dituju diambil terlebih dahulu oleh bot lawan atau ketika terjadi tackle di jalan dengan bot lain. Hal ini menyebabkan bot belum mampu memaksimalkan diamond yang dikumpulkan secara optimal.

BAB V

Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian yang telah dilakukan, dapat disimpulkan bahwa algoritma Greedy dapat diterapkan secara efektif dalam pembuatan bot permainan *Diamonds*. Strategi utama yang digunakan adalah **Greedy by Distance**, yaitu strategi yang memprioritaskan pengambilan diamond terdekat pada setiap langkah permainan. Strategi ini terbukti efisien dalam pengambilan keputusan real-time dan menghasilkan performa yang cukup kompetitif dalam beberapa skenario pengujian.

Meskipun strategi ini memiliki kelebihan dalam hal kecepatan respon dan kesederhanaan implementasi, terdapat juga beberapa kelemahan, seperti kerentanan terhadap gangguan dari bot lawan (misalnya tackle) dan tidak mempertimbangkan nilai diamond atau peluang kompetisi. Oleh karena itu, strategi greedy ini bersifat *lokal optimum* dan belum tentu menghasilkan solusi *global optimum*.

Secara keseluruhan, implementasi yang dilakukan sudah memenuhi spesifikasi tugas besar dan berhasil menunjukkan aplikasi nyata dari algoritma Greedy dalam konteks permainan strategi berbasis waktu.

5.2 Saran

Ke depan, strategi bot bisa ditingkatkan dengan menyesuaikan cara bermain berdasarkan situasi di permainan, seperti berpindah strategi saat menghadapi lawan atau saat diamond sulit dijangkau. Selain itu, bot juga sebaiknya mampu memperkirakan gerakan lawan agar tidak mudah ditackle atau kalah cepat. Menambahkan perhitungan jalur yang lebih cerdas serta melakukan lebih banyak pengujian di berbagai kondisi peta juga bisa membantu meningkatkan performa bot secara keseluruhan.

LAMPIRAN

A. Repository Github

<https://github.com/DaffaDesuandi2/Team7>

B. Video Penjelasan (link GDrive)

https://drive.google.com/file/d/1qi_oIknViVYnQKMSl0WI59jgnIUBJQ1T/view?usp=sharing

DAFTAR PUSTAKA

- [1] Institut Teknologi Sumatera, *Algoritma Greedy - Bagian 1*, Mata Kuliah Strategi Algoritma (IF2211), 2021. [Online]. Available: <https://kuliah.itera.ac.id/mod/resource/view.php?id=32758>
- [2] Institut Teknologi Sumatera, *Algoritma Greedy - Bagian 2*, Mata Kuliah Strategi Algoritma (IF2211), 2021. [Online]. Available: <https://kuliah.itera.ac.id/mod/resource/view.php?id=33150>
- [4] R. Munir, *Algoritma Greedy - Bagian 3*, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 2022. [Online]. Available: [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)
- [5] Haziqam, *Tubes1-IF2211 Game Engine v1.1.0*, GitHub repository, 2024. [Online]. Available: <https://github.com/haziqam/tubes1-IF2211-game-engine/releases/tag/v1.1.0>
- [6] Haziqam, *Tubes1-IF2211 Bot Starter Pack, Version 1.1.0*, GitHub repository, 2024. [Online]. Available: <https://github.com/haziqam/tubes1-IF2211-bot-starter-pack/releases/tag/v1.0.1>