LAPORAN TUGAS BESAR
STRATEGI ALGORITMA



Disusun Oleh:

Faiz Akbar Al Kalabadzi (123140091)

Ivan Nandira Mangunang (123140094)

Ahmad Mulia Saputra (123140168)

Dosen Pengampuh:

Winda Yulita, M.Cs.

TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI SUMATERA
2025

BAB I
Deskripsi Tugas

Tugas besar ini merupakan bagian dari mata kuliah IF2211 Strategi Algoritma yang bertujuan untuk menguji kemampuan mahasiswa dalam mengimplementasikan algoritma greedy dalam bentuk aplikasi nyata. Pada tugas ini, mahasiswa diminta untuk membuat sebuah bot permainan yang akan bertanding dalam mengumpulkan diamond sebanyak-banyaknya di sebuah lingkungan permainan yang telah ditentukan. Permainan tersebut bernama Diamonds, yang merupakan sebuah programming challenge di mana setiap pemain akan memiliki satu buah bot.

Bot tersebut harus dapat bergerak di dalam papan permainan dan mengambil keputusan secara otomatis untuk mengumpulkan diamond, menyimpannya ke base, serta menghindari interaksi negatif dengan bot lawan. Permainan ini memiliki berbagai elemen seperti dua jenis diamond (biru dan merah), teleporter, red button, base, dan inventory dengan kapasitas terbatas. Seluruh elemen tersebut menjadi tantangan tambahan dalam pengambilan keputusan bot selama permainan berlangsung.

Algoritma yang wajib digunakan untuk mengatur perilaku bot adalah algoritma greedy, di mana strategi pengambilan keputusan harus berdasarkan pemilihan aksi terbaik secara lokal pada setiap langkah. Mahasiswa diminta untuk tidak hanya mengimplementasikan bot tersebut, tetapi juga menjelaskan strategi greedy yang digunakan dalam laporan ini secara rinci, mulai dari proses pemetaan persoalan ke elemen-elemen greedy hingga analisis efektivitas strategi tersebut.

Tujuan akhir dari tugas besar ini adalah untuk menghasilkan bot dengan strategi greedy terbaik yang mampu bersaing dengan bot dari kelompok lain dalam kompetisi. Selain implementasi teknis, laporan ini juga mencakup penjelasan teoritis, eksplorasi strategi alternatif, analisis efektivitas, serta pengujian terhadap solusi yang dibuat. Oleh karena itu, pemahaman mendalam tentang algoritma greedy dan kemampuan mengadaptasinya ke dalam konteks permainan menjadi aspek utama yang diuji dalam tugas besar ini.

BAB II
Landasan Teori

2.1 Algoritma Greedy

Algoritma greedy merupakan salah satu pendekatan dalam pemecahan masalah optimasi yang bekerja dengan cara mengambil keputusan terbaik pada setiap langkah berdasarkan informasi lokal yang tersedia saat itu. Pendekatan ini bersifat myopic, yaitu hanya mempertimbangkan kondisi saat ini tanpa memperhitungkan konsekuensi jangka panjang dari keputusan tersebut.

Dua karakteristik utama dari algoritma greedy adalah: (1) Greedy-choice property, yaitu pilihan terbaik secara lokal akan mengarah pada solusi terbaik secara global; dan (2) Optimal substructure, yaitu solusi optimal dari suatu masalah dapat dibentuk dari solusi optimal dari sub-masalahnya. Algoritma ini banyak digunakan karena kesederhanaannya serta efisiensi waktu komputasinya. Namun, tidak semua masalah cocok diselesaikan dengan pendekatan greedy, terutama jika keputusan lokal tidak menjamin tercapainya solusi global yang optimal.

2.2 Cara Kerja Program Secara Umum

Program yang dikembangkan dalam proyek ini bertujuan untuk menyelesaikan persoalan pengumpulan diamond pada permainan menggunakan pendekatan algoritma greedy. Program ini diimplementasikan dalam bentuk sebuah bot yang mampu melakukan aksi seperti bergerak, melakukan pemindaian lingkungan (scanning), dan mengambil keputusan secara otomatis berdasarkan strategi greedy.

2.3 Proses Scanning oleh Bot

Bot secara berkala melakukan proses scanning atau pemindaian untuk memetakan kondisi lingkungan sekitarnya. Pemindaian ini dilakukan dalam radius tertentu untuk mendeteksi posisi diamond, rintangan, musuh (jika ada), dan objek-objek penting lainnya. Hasil scanning biasanya dikembalikan dalam bentuk grid atau daftar objek dengan posisi relatif terhadap bot. Informasi yang diperoleh dari hasil scanning ini kemudian menjadi dasar dalam proses pengambilan keputusan berikutnya. Bot mengevaluasi semua objek yang terdeteksi untuk menentukan target mana yang akan dikejar terlebih dahulu.

2.4 Evaluasi dan Pemilihan Aksi

Setelah melakukan scanning, bot akan mengevaluasi semua kandidat diamond berdasarkan beberapa kriteria, antara lain: jarak terdekat (misalnya menggunakan Manhattan Distance), nilai atau prioritas diamond (jika ada bobot), dan kemudahan akses (apakah jalurnya terhalang atau terbuka). Dari hasil evaluasi tersebut, bot kemudian memilih aksi terbaik saat itu juga, yaitu aksi dengan nilai tertinggi menurut kriteria greedy. Pemilihan ini tidak mempertimbangkan kondisi masa depan atau perencanaan jangka panjang.

2.5 Eksekusi dan Perulangan Proses

Setelah menentukan langkah, bot akan menjalankan aksi tersebut, misalnya dengan bergerak ke arah diamond yang dipilih. Proses ini kemudian diulang: bot melakukan scanning kembali, mengevaluasi kondisi terbaru, dan memilih aksi berikutnya. Siklus ini terus berjalan hingga semua diamond berhasil dikumpulkan atau tidak ada lagi aksi yang dapat dilakukan.

2.6 Implementasi Strategi Greedy

Strategi greedy dalam bot ini difokuskan pada kecepatan pengambilan keputusan dan efisiensi pelaksanaan. Bot tidak menyimpan riwayat langkah atau membuat rencana jangka panjang. Keputusan selalu diambil berdasarkan kondisi lingkungan saat ini yang diperoleh dari hasil scanning. Pendekatan ini cocok digunakan dalam lingkungan permainan yang dinamis. Namun, perlu diakui bahwa dalam beberapa kasus, strategi greedy dapat mengarah pada solusi yang tidak optimal. Misalnya, bot bisa saja terjebak atau terlalu fokus pada satu area tertentu sehingga melewatkan diamond yang lebih bernilai di tempat lain.

BAB III
Aplikasi Strategi Greedy

3.1 Proses Mapping Persoalan Diamonds menjadi Elemen-elemen Algoritma Greedy

Dalam penyelesaian persoalan pengambilan berlian (diamonds) pada bot, strategi greedy diaplikasikan dengan memetakan masalah ke dalam elemen-elemen algoritma greedy berikut:

- Himpunan kandidat: Semua berlian yang tersedia di papan permainan (board.diamonds)
- Himpunan solusi: Urutan pengambilan berlian oleh bot hingga mencapai kapasitas maksimum (5 diamonds)
- Fungsi solusi: Posisi target berlian berikutnya yang akan diambil oleh bot
- Fungsi seleksi: Pemilihan berlian terdekat dari posisi bot saat ini
- Fungsi kelayakan: Bot hanya mengambil berlian jika kapasitas inventaris belum penuh (belum mencapai 5 diamond)
- Fungsi objektif: Meminimalkan jarak total tempuh bot dalam mengambil berlian dan kembali ke base

3.2 Eksplorasi Alternatif Solusi Greedy yang Mungkin Dipilih dalam Persoalan Diamonds

Strategi greedy yang diterapkan dalam program adalah memilih berlian terdekat dari posisi bot saat ini sebagai langkah berikutnya, dengan batas kapasitas inventaris sebanyak 5 berlian. Alternatif solusi greedy lain yang bisa dieksplorasi antara lain:

- Memprioritaskan berlian dengan warna atau nilai tertentu terlebih dahulu (misalnya diamond merah).
- Memilih berlian berdasarkan kombinasi jarak dan nilai berlian.
- Menggunakan strategi pengambilan berlian dengan mempertimbangkan rute kembali ke base untuk efisiensi waktu.

Namun, dalam kode yang diberikan, fokus adalah pada pengambilan berlian terdekat dengan cara menghitung jarak Manhattan sebagai metrik seleksi.

3.3 Analisis Efisiensi dan Efektivitas dari Kumpulan Alternatif Solusi Greedy yang Dirumuskan

Strategi greedy pemilihan berlian terdekat ini memiliki keunggulan dari sisi efisiensi komputasi, karena hanya perlu mencari berlian dengan jarak minimum di setiap langkah, tanpa perlu menghitung rute-rute kompleks yang memakan waktu.

Efektivitas strategi ini dalam mendapatkan solusi optimal bergantung pada distribusi berlian di papan permainan. Jika berlian tersebar merata, strategi ini cenderung menghasilkan jarak tempuh yang pendek. Namun, ada kemungkinan solusi yang dihasilkan bukan solusi global optimal karena tidak mempertimbangkan seluruh rute.

3.4 Strategi Greedy yang Dipilih dan Pertimbangan Pemilihannya

Strategi greedy yang dipilih adalah:

- Bot akan mengambil berlian yang terdekat dari posisi saat ini selama kapasitas inventaris belum penuh.
- Jika kapasitas inventaris sudah penuh (5 berlian), bot akan kembali ke base untuk menyimpan berlian tersebut.

Pertimbangan pemilihan strategi ini adalah:

- Kesederhanaan implementasi yang sesuai dengan kebutuhan bot.
- Kecepatan eksekusi yang penting dalam game dengan waktu nyata.
- Kecukupan hasil yang memadai untuk memperoleh nilai optimal atau mendekati optimal dalam konteks pengambilan berlian secara efisien.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi

Kode ini menggunakan strategi *greedy* berbasis jarak terdekat (*greedy on nearest distance*). Artinya, bot selalu mengambil keputusan lokal terbaik berdasarkan posisi diamond yang paling dekat, tanpa mempertimbangkan nilai diamond atau efek jangka panjang

4.1.1 Inventori penuh

Jika inventori sudah penuh, Bot langsung menetapkan base sebagai tujuan, tanpa mempertimbangkan lokasi diamond lain. Ini adalah bentuk greedy karena bot menganggap tidak ada tindakan yang lebih menguntungkan daripada langsung pulang saat inventori penuh.

PSEUDOCODE Strategi Pulang ke base:

```
FUNCTION next_move_if_inventory_full(bot):
    props ← bot.properties
    current_position ← bot.position
    base_position ← props.base

    // Hitung arah gerakan menuju base
    delta_x ← SIGN(base_position.x - current_position.x)
    delta_y ← SIGN(base_position.y - current_position.y)

    RETURN (delta_x, delta_y)  // Gerak ke base
END FUNCTION
```

4.1.2 Inventori belum penuh

Jika inventori belum penuh, Bot akan mencari satu diamond dengan jarak terpendek dari posisinya sekarang. Perhitungan jarak menggunakan Manhattan Distance: $Jarak = |x1 - x2| + |y1 - y2|$. Tidak membedakan apakah diamond tersebut bernilai 1 poin (biru) atau 2 poin (merah) — semuanya diperlakukan sama.
PSEUDOCODE Strategi Cari dan ambil diamond terdekat:

```
FUNCTION next_move_if_inventory_not_full(bot, board):
    current_position ← bot.position
    nearest_diamond ← NULL
    min_distance ← ∞

    // Iterasi semua diamond yang tersedia
    FOR each diamond IN board.diamonds DO
        diamond_position ← diamond.position

        // Hitung jarak Manhattan dari posisi bot ke diamond
        distance ← ABS(diamond_position.x - current_position.x) +
                   ABS(diamond_position.y - current_position.y)

        // Simpan diamond dengan jarak paling kecil sejauh ini
        IF distance < min_distance THEN
            min_distance ← distance
            nearest_diamond ← diamond
        END IF
    END FOR

    // Jika ditemukan diamond terdekat
    IF nearest_diamond ≠ NULL THEN
        target_position ← nearest_diamond.position
    ELSE
        // Jika tidak ada diamond, fallback ke base
        target_position ← bot.properties.base
    END IF

    // Hitung arah gerakan ke diamond atau base
    delta_x ← SIGN(target_position.x - current_position.x)
    delta_y ← SIGN(target_position.y - current_position.y)

    RETURN (delta_x, delta_y)  // Gerak ke target
END FUNCTION
```

4.2 Struktur data

4.2.1 Struktur data yang digunakan

4.2.1.1. Kelas GameObject

GameObject adalah representasi dari semua entitas dalam permainan, seperti:

- Bot (Pemain)
- Diamond (Biru dan Merah)
- Base (Markas untuk menyimpan diamond)

Atribut:

- position : Posisi objek pada papan, bertipe Position
- properties : Menyimpan atribut tambahan seperti:
    - diamonds : Jumlah diamond yang dimiliki bot
    - base : Posisi base bot
    - name : Tipe diamond (contoh: "RED" atau "BLUE")

4.2.1.2. Kelas Position

Position menyimpan koordinat x dan y suatu objek di papan permainan.
Tipe ini digunakan untuk merepresentasikan:

- Posisi bot saat ini
- Lokasi diamond
- Lokasi base
- Tujuan gerak (goal)

4.2.1.3. Kelas Board

Board adalah representasi dari keadaan papan permainan saat ini.
Atribut:

- diamonds : daftar semua diamond yang tersedia di papan
  → Tipe: list[GameObject]

4.2.1.4. Properti goal_position

Properti ini dimiliki oleh kelas GreedyLogic.
Digunakan untuk menyimpan tujuan bot saat ini, yaitu:

- Posisi diamond terdekat, jika inventori belum penuh
- Posisi base, jika inventori sudah penuh

Nilainya dapat berupa None jika tidak ada tujuan yang ditentukan.

4.2.1.5. Fungsi Bantu get_direction

Merupakan fungsi utilitas yang mengembalikan arah gerakan berdasarkan dua koordinat:

get_direction(x1, y1, x2, y2) → (delta_x, delta_y)

Hasilnya berupa tuple gerakan (arah):

- (1, 0) = ke kanan
- (-1, 0) = ke kiri
- (0, 1) = ke bawah
- (0, -1) = ke atas
- (0, 0) = diam

Digunakan untuk mengarahkan bot dari posisinya saat ini menuju goal_position

4.2.1 Tipe Data Python yang Terkait

Tipe data	Keterangan
class / object	Digunakan untuk struktur kompleks: GameObject, Position, Board
list	Menyimpan banyak diamond dalam papan (board.diamonds)
Optional	Untuk nilai yang bisa kosong seperti goal_position
tuple	Menyatakan arah gerakan dalam (delta_x, delta_y)
lambda	Digunakan dalam pencarian diamond terdekat dengan fungsi min()

4.3. Analisis

4.3.1 Pengujian dan Analisis Performa Strategi Greedy

4.3.1.1 Pengujian 1 Diamond Tersebar Merata

Kondisi: Diamond tersebar secara merata, tidak ada hambatan atau zona bahaya.

Hasil: Bot dapat dengan mudah menemukan dan mengambil diamond secara efisien serta kembali ke base tanpa kendala berarti.

Analisis: Dalam situasi ini, strategi greedy bekerja secara optimal karena keputusan lokal (mengambil diamond terdekat) berkontribusi langsung terhadap tujuan global (mengumpulkan poin dan menyetor ke base).

4.3.1.2 Pengujian 2 Diamond Merah Lebih Menguntungkan Tetapi Sedikit Lebih Jauh

Kondisi: Terdapat diamond merah (bernilai 2 poin) sedikit lebih jauh dibandingkan diamond biru (1 poin) yang lebih dekat.

Hasil: Bot selalu memilih diamond biru karena jaraknya lebih dekat, meskipun nilai poin total yang diperoleh menjadi lebih rendah.

Analisis: Strategi greedy tidak optimal dalam kasus ini karena hanya mempertimbangkan jarak, bukan nilai diamond. Bot mengabaikan potensi perolehan poin lebih tinggi dari diamond merah.

4.3.1.3 Pengujian 3 Posisi Diamond Mengarah ke Area Terpencil

Kondisi: Diamond terdekat berada di ujung papan dan menjauh dari base.

Hasil: Setelah mengambil diamond tersebut, bot memerlukan waktu lebih lama untuk kembali ke base.

Analisis: Strategi greedy gagal mempertimbangkan efisiensi perjalanan kembali, sehingga menyebabkan peningkatan waktu tempuh yang seharusnya bisa dihindari.

4.3.1.4 Pengujian 4 Tidak Ada Diamond yang Dapat Diambil

Kondisi: Bot sudah memiliki 4 poin, dan hanya tersedia diamond merah (2 poin) yang akan melebihi kapasitas.

Hasil: Bot tetap memilih mendekati diamond terdekat (merah), lalu tidak bisa mengambilnya, akhirnya kembali ke base dengan langkah tambahan yang sia-sia.

Analisis: Strategi greedy tidak memiliki logika pengecualian untuk diamond yang tidak dapat diambil, sehingga menyebabkan tindakan yang tidak efisien.

4.3.2 Ringkasan Hasil Pengujian

Kondisi Pengujian	Hasil Strategi Greedy	Optimal
Diamond merata, tanpa hambatan	Gerakan efisien, cepat ke base	Ya
Diamond merah lebih menguntungkan tetapi lebih jauh	Pilih diamond biru terdekat, poin lebih kecil	Tidak
Diamond terdekat menjauhkan dari base	Waktu pulang lama, langkah boros	Tidak
Tidak ada diamond valid karena kapasitas penuh sebagian	Gerak mendekati diamond sia-sia	Tidak

BAB V
KESIMPULAN DAN SARAN

Strategi greedy yang diterapkan dalam program ini efisien dan optimal dalam kondisi permainan yang sederhana dan terbuka, tanpa pertimbangan nilai variatif atau perencanaan jangka panjang. Namun, dalam kondisi permainan yang lebih kompleks, strategi ini memiliki keterbatasan signifikan, terutama dalam hal:

- Memprioritaskan jarak tanpa memperhitungkan nilai diamond.
- Tidak mempertimbangkan efisiensi rute balik ke base.
- Tidak mengantisipasi kehilangan peluang akibat keputusan lokal yang keliru.

Untuk meningkatkan performa bot dalam permainan dengan kompleksitas lebih tinggi, strategi greedy ini dapat dikombinasikan dengan pendekatan lain, seperti:

- Greedy + Value-based planning: mempertimbangkan rasio nilai per langkah.
- Greedy + Zone prioritization: memprioritaskan area sekitar base.
- Greedy + Lookahead simulation: mengevaluasi beberapa langkah ke depan sebelum bergerak.

LAMPIRAN

Link Github: https://github.com/V4nzz/Tubes_Stigma_Pie-Sigma

DAFTAR PUSTAKA

Modul Algoritma Greedy Bagian 1 oleh Rinaldi Munir

Modul Algoritma Greedy Bagian 2 & 3 dan Algoritma Devide & Conqure Bagian 1

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.