

LAPORAN TUGAS BESAR STRATEGI ALGORITMA

Diajukan sebagai syarat menyelesaikan mata kuliah Strategi Algoritma (IF2211) Kelas RB
di Program Studi Teknik Informatika, Fakultas Teknologi Industri, Institut Teknologi Sumatera



Oleh: Kelompok 1(Pesemka)

| | |
|------------------------|-----------|
| Roberto Charlos Sagala | 123140113 |
| Brahmantio Abimayu | 123140122 |
| Zacky Ghazi Al miqdad | 123140130 |

Dosen Pengampu: Imam Ekowicaksono, S.Si., M.Si.

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

DAFTAR ISI

| | |
|---|-----------|
| BAB I DESKRIPSI TUGAS..... | 3 |
| BAB II LANDASAN TEORI..... | 6 |
| 2.1 Dasar Teori..... | 6 |
| 2.2 Cara Kerja Program..... | 7 |
| BAB III APLIKASI STRATEGI GREEDY..... | 8 |
| 3.1 Process Mapping..... | 8 |
| 3.1.1 Himpunan Kandidat..... | 8 |
| 3.1.2 Himpunan Solusi..... | 8 |
| 3.1.3 Fungsi Solusi..... | 8 |
| 3.1.4 Fungsi Seleksi..... | 8 |
| 3.1.5 Fungsi Kelayakan..... | 8 |
| 3.1.6 Fungsi Objektif..... | 9 |
| 3.2 Eksplorasi Alternatif Solusi Greedy..... | 9 |
| 3.2.1 Greedy by Value..... | 9 |
| 3.2.2 Greedy by Distance..... | 9 |
| 3.2.3 Greedy by Density..... | 9 |
| 3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy..... | 9 |
| 3.3.1 Greedy by Value..... | 9 |
| 3.3.2 Greedy by Distance..... | 10 |
| 3.3.3 Greedy by Density..... | 10 |
| 3.4 Strategi Greedy yang Dipilih..... | 10 |
| BAB IV IMPLEMENTASI DAN PENGUJIAN..... | 11 |
| 4.1 Implementasi Algoritma Greedy..... | 11 |
| 4.1.1 Pseudocode..... | 11 |
| 4.2 Struktur Data yang Digunakan..... | 13 |
| 4.3 Pengujian Program..... | 13 |
| 4.3.1 Skenario Pengujian..... | 13 |
| 4.3.2 Hasil Pengujian dan Analisis..... | 14 |
| BAB V KESIMPULAN DAN SARAN..... | 16 |
| 5.1 Kesimpulan..... | 16 |
| LAMPIRAN..... | 17 |
| DAFTAR PUSTAKA..... | 18 |

BAB I

DESKRIPSI TUGAS

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

Pada tugas 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
2. Bot starter pack, yang secara umum berisi:
 - a. Program untuk memanggil API yang tersedia pada backend
 - b. Program bot logic (bagian ini yang akan kalian implementasikan dengan algoritma greedy untuk bot kelompok kalian)
 - c. Program utama (main) dan utilitas lainnya
3. Komponen-komponen dari permainan Diamonds antara lain:
 - a. Diamonds



Untuk memenangkan pertandingan, kita harus mengumpulkan diamond ini

sebanyak-banyaknya dengan melewati/melangkahnya. 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.

b. Red Button/Diamond Button



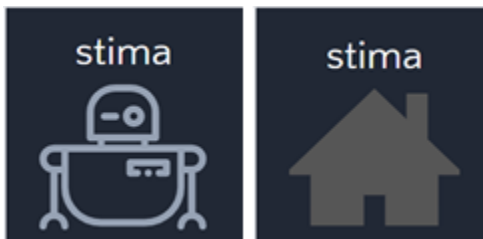
Ketika red button ini dilewati/dilangkahi, semua diamond (termasuk red diamond) akan di-generate kembali pada board dengan posisi acak. Posisi red button ini juga akan berubah secara acak jika red button ini dilangkahi.

c. Teleporters



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





d. 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.

e. Inventory

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

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* adalah metode populer dan sederhana untuk memecahkan persoalan optimasi, yaitu persoalan mencari solusi optimal, baik maksimasi maupun minimasi. Prinsip utama algoritma *greedy* adalah "*take what you can get now!*". Artinya, algoritma ini membentuk solusi langkah demi langkah, di mana pada setiap langkahnya, keputusan terbaik (optimum lokal) akan dibuat tanpa mempertimbangkan konsekuensi di masa depan, dengan harapan bahwa serangkaian optimum lokal ini akan mengarah pada solusi optimum global. Setelah keputusan dibuat, algoritma tidak dapat mundur lagi ke langkah sebelumnya.

Algoritma *greedy* bekerja dengan elemen-elemen seperti himpunan kandidat (C) yang berisi pilihan yang akan dievaluasi, himpunan solusi (S) untuk menyimpan kandidat yang sudah dipilih, fungsi solusi untuk menentukan apakah himpunan yang dipilih sudah menghasilkan solusi, fungsi seleksi untuk memilih kandidat berdasarkan strategi *greedy* tertentu, fungsi kelayakan untuk memeriksa apakah kandidat bisa dimasukkan ke dalam himpunan solusi, dan fungsi objektif yang ingin dimaksimalkan atau diminimalkan. Pada akhir setiap iterasi, solusi yang terbentuk adalah optimum lokal, dan pada akhir keseluruhan proses, diharapkan diperoleh optimum global.

Namun, algoritma *greedy* tidak selalu menjamin solusi optimal global. Hal ini karena algoritma *greedy* tidak beroperasi secara menyeluruh terhadap semua kemungkinan solusi seperti metode *exhaustive search*, dan pemilihan fungsi seleksi yang tepat sangat krusial untuk menghasilkan solusi optimal. Terkadang, algoritma *greedy* hanya menghasilkan solusi sub-optimal atau pseudo-optimal. Untuk masalah di mana solusi optimal mutlak tidak terlalu diperlukan, algoritma *greedy* dapat digunakan untuk menghasilkan solusi hampiran (aproksimasi) yang lebih cepat dibandingkan algoritma eksponensial. Jika algoritma *greedy* dapat menghasilkan solusi optimal, keoptimalannya harus dibuktikan secara matematis, yang seringkali merupakan tantangan. Akan tetapi, lebih mudah menunjukkan ketidakoptimalan algoritma *greedy* dengan menunjukkan *counterexample*. [1]

2.2 Cara Kerja Program

Cara kerja bot dimulai dengan pendaftaran pada board melalui HTTP request. Setelah terdaftar, bot akan secara berkala mengirimkan data arah geraknya ke backend game, juga melalui HTTP request. Untuk menentukan arah gerak ini, diperlukan sebuah algoritma. Meskipun template bot telah menyediakan algoritma Random, algoritma tersebut kurang optimal. Oleh karena itu, saya akan mengimplementasikan algoritma greedy untuk menentukan arah gerak bot yang lebih baik. Berikut adalah langkah-langkah untuk menjalankan bot dengan algoritma yang diimplementasikan sendiri:

1. Download / Clone source code game engine Etime Diamonds 2
2. Jalankan game dengan npm build dan npm start pada repository game
3. Download / Clone source code template bot
4. Buat file baru pada game/logic/
5. Implementasikan algoritma untuk menentukan arah gerak bot
6. Import class algoritma pada main file di main.py dan daftarkan pada CONTROLLERS
7. Jalankan bot dengan python main.py --logic {class yang diimport}
--email={email}
--name={nama untuk ditampilkan di board} --password={password} --team {team}
--board {nomor board} atau bisa menjalankan batch dengan menggunakan .sh atau .bat

BAB III

APLIKASI STRATEGI *GREEDY*

3.1 *Process Mapping*

3.1.1 Himpunan Kandidat

Himpunan kandidat merupakan semua kemungkinan langkah atau tujuan yang dapat dipilih oleh bot di setiap iterasi. Dalam kasus ini himpunan kandidatnya adalah semua objek yang ada di dalam board game.

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, dalam kasus ini himpunan solusi merupakan daftar posisi yang dikunjungi oleh bot dalam bentuk koordinat. Himpunan solusi dihasilkan dengan mempertimbangkan kombinasi posisi diamond, jarak diamond, posisi musuh. sehingga memungkinkan untuk mengambil keputusan yang paling menguntungkan.

3.1.3 Fungsi Solusi

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

1. Inventory bot sudah penuh sehingga harus kembali ke base untuk mengosongkan inventory
2. Berlian yang tersedia di papan sudah habis dan bot sudah kembali ke base
3. Waktu permainan habis

3.1.4 Fungsi Seleksi

Fungsi seleksi menjelaskan bagaimana algoritma memilih kandidat terbaik pada setiap langkah. Dalam kasus ini fungsi seleksinya adalah bot akan mengambil keputusan berdasarkan jarak dan nilai diamond yang tidak melebihi kapasitas inventory, jika inventory sudah penuh bot akan langsung memutuskan untuk kembali ke base terlebih dahulu.

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. Dalam kasus

ini bot kami akan mengecek apakah suatu objek masih layak untuk diambil atau tidak dengan cara memperhitungkan jarak dan nilai diamond serta jarak musuh.

3.1.6 Fungsi Objektif

Fungsi objektif merupakan tujuan akhir dari algoritma yang ingin dicapai melalui serangkaian langkah-langkah yang diambil. Dalam kasus kali ini fungsi objektifnya adalah mengambil diamond yang berada sedekat mungkin dengan bot berdasarkan kondisi inventory.

3.2 Eksplorasi Alternatif Solusi Greedy

3.2.1 Greedy by Value

Solusi Greedy by Value memprioritaskan diamond dengan nilai paling banyak terlebih dahulu tanpa memperhatikan jarak. Oleh karena itu, meskipun greedy by value efektif dalam memaksimalkan hasil di lingkup tempat yang kecil, ia belum tentu optimal secara keseluruhan jika aspek terdapat lain seperti waktu dan jarak.

3.2.2 Greedy by Distance

Solusi Greedy by Distance memprioritaskan diamond yang jaraknya lebih dekat dari posisi bot, dengan mengabaikan nilai diamond menjadikan kelemahan utama solusi ini karena bot bisa menghabiskan waktu untuk diamond terdekat padahal ada jalan lain dengan keuntungan lebih besar yang terlewatkan.

3.2.3 Greedy by Density

Solusi Greedy by Density melakukan perhitungan rasio antara nilai diamond dan jarak yang harus ditempuh bot untuk mencapainya, menggunakan rumus : $\text{nilai diamond} / \text{jarak}$, ini bertujuan untuk memaksimalkan perolehan nilai per unit jarak, sehingga secara efektif mengoptimalkan efisiensi waktu dan energi dengan menyeimbangkan antara berapa keuntungan sebuah diamond dan seberapa jauh jaraknya

3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy

3.3.1 Greedy by Value

Efisiensi : Berpotensi memiliki tingkat keefisienan yang rendah, terutama jika bot berada dalam kasus mengabaikan diamond yang lebih dekat walau memiliki nilai yang lebih kecil untuk mengambil diamond yang memiliki nilai lebih besar walaupun jaraknya lebih jauh.

Efektivitas : Berpotensi memiliki keefektivitasan yang tinggi, terutama jika terdapat diamond yang memiliki nilai besar berdekatan, tetapi rendah jika kasus dimana letak diamond berpencar

3.3.2 Greedy by Distance

Efisiensi : Berpotensi memiliki tingkat keefisienan yang tinggi, karena bot meminimalkan waktu untuk menempuh jarak untuk mengambil diamond yang ada karena memprioritaskan diamond yang lebih dekat

Efektivitas : Berpotensi memiliki keefektivitasan yang rendah, karena terdapat kasus dimana bot terus menerus mengambil diamond terdekat yang nilainya lebih kecil walau terdapat diamond yang memiliki nilai lebih besar tidak terlalu jauh dari posisi diamond dengan nilai lebih kecil yang diambil

3.3.3. Greedy by Density

Efisiensi : Berpotensi memiliki tingkat keefisienan yang tinggi, karena bot akan selalu menghitung antara jarak dan nilai dari diamond

Efektivitas : Berpotensi memiliki keefektivitasan yang tinggi, karena bot tidak akan sembarangan mengambil langkah sebelum memperhitungkan antara jarak dan nilai

3.4 Strategi Greedy yang Dipilih

Setelah melakukan eksplorasi terhadap beberapa alternatif solusi greedy dan menganalisis efisiensi serta efektivitas masing-masing, strategi yang akhirnya dipilih untuk diimplementasikan adalah Solusi Greedy by Density.

Strategi ini dianggap lebih unggul dibandingkan dua alternatif lainnya karena:

1. Menghindari potensi inefisiensi dari Greedy by Nilai Diamond, di mana bot bisa menghabiskan terlalu banyak waktu atau langkah untuk mengejar diamond bernilai tinggi yang lokasinya sangat jauh, sehingga mengabaikan peluang lain yang lebih cepat dijangkau.
2. Mengatasi kelemahan Greedy by Jarak Diamond, yang meskipun efisien dalam pergerakan, seringkali kurang efektif dalam memaksimalkan skor total karena cenderung mengumpulkan diamond bernilai rendah hanya karena lokasinya dekat.

Dengan demikian, strategi Greedy by Density diharapkan dapat menghasilkan performa bot yang lebih optimal secara keseluruhan, karena menyeimbangkan antara seberapa berharga sebuah diamond dan seberapa sulit atau jauh untuk mendapatkannya, sehingga cocok untuk kondisi permainan dengan waktu atau energi yang terbatas.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma Greedy

4.1.1 Pseudocode

CLASS Pesemka:

INIT:

```
goal_position ← None
teleporters ← kosong (dictionary pasangan teleport)
target_after_teleport ← None
```

METHOD get_diamonds(board):

```
RETURN semua objek bertipe Diamond dari papan
```

METHOD get_teleporters(board):

```
RETURN semua objek bertipe Teleporter dari papan
```

METHOD get_reset_button(board):

```
RETURN semua objek bertipe Reset Button dari papan
```

METHOD get_distance_without_teleport(pos1, pos2):

```
RETURN jarak antara pos1 dan pos2
```

METHOD get_distance_with_teleport(pos1, pos2, board):

```
IF tidak ada teleporters:
    RETURN jarak langsung pos1 ke pos2
```

```
best_distance ← jarak langsung
```

UNTUK setiap pasangan teleporter:

```
HITUNG dua kemungkinan:
    - Masuk ke tele1 → keluar di tele2
    - Masuk ke tele2 → keluar di tele1
```

```
    Bandingkan total jaraknya
```

```
    SIMPAN jarak minimum
```

```
RETURN best_distance
```

```

METHOD is_position_equals(pos1, pos2):
    RETURN True jika pos1 dan pos2 sama, False jika tidak

METHOD next_move(bot, board):
    Ambil posisi sekarang, base, dan jumlah diamond

    JIKA diamond penuh (5):
        HITUNG jarak langsung dan via teleport ke base

        JIKA lebih cepat via teleport:
            Cari teleporter terdekat
            SET goal_position ke teleporter
        ELSE:
            SET goal_position ke base

    ELSE:
        IF sudah berada di posisi teleporter:
            Ubah goal_position ke target setelah teleport

        Ambil semua objek penting: diamond, teleporter, reset button, bot

        Kelompokkan teleporters berdasarkan pair_id

        Buat list kandidat (diamond, tombol) beserta jaraknya

        Cari kandidat terbaik berdasarkan nilai (jarak / bobot poin)

        JIKA jarak via teleport lebih kecil:
            Cari jalur teleportasi tercepat
            SET goal_position ke teleporter
            SET target_after_teleport ke tujuan akhir
        ELSE:
            SET goal_position ke posisi kandidat

    IF sudah berada di goal_position:
        SET goal_position ke base

    RETURN arah gerakan dari posisi sekarang ke goal_position

```

4.2 Struktur Data yang Digunakan

Dalam pembuatan bot Pesemka ini terdapat beberapa jenis struktur data yang digunakan yaitu:

- List

Tipe data list digunakan untuk menyimpan sekumpulan objek permainan, seperti berlian, teleporter, dan buttont. List dipilih karena mendukung proses iterasi secara sekuensial yang dibutuhkan dalam proses pencarian kandidat terbaik.

- Dictionary

Tipe data dictionary digunakan untuk mengelompokkan pasangan teleporter berdasarkan `pair_id`. Setiap pasangan disimpan dengan key berupa ID unik dan value berupa list dua teleporter.

- Tuple

Tuple digunakan untuk menyimpan pasangan nilai tetap, seperti koordinat pergerakan bot dan pasangan objek dengan nilai evaluasi tertentu (misalnya jarak atau density). Karena bersifat immutable, tuple aman digunakan dalam proses komputasi yang tidak membutuhkan modifikasi data.

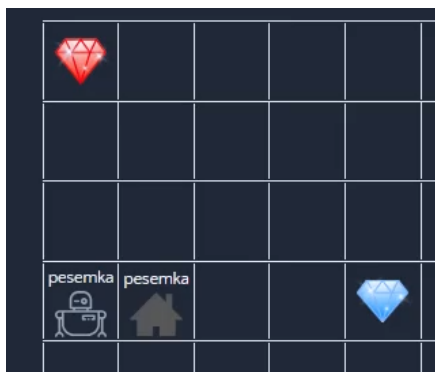
- Optional

Struktur data Optional digunakan pada variabel yang nilainya bisa berubah atau tidak tersedia pada waktu tertentu. Hal ini penting untuk mencegah kesalahan akses ketika variabel belum diinisialisasi.

4.3 Pengujian Program

4.3.1 Skenario Pengujian

Skenario 1:



Skenario 2:



Skenario 3:



4.3.2 Hasil Pengujian dan Analisis

Skenario 1:

Diamond Merah: $w=4$, $p=3$, $d=0,75$. Diamond Biru: $w=2$, $p=4$, $d=2$.

Pada kasus skenario 1, bot akan memilih Diamond Merah dengan nilai d (densitas) 0.75.. Karena nilai d diamond merah lebih kecil dibandingkan dengan diamond biru.

Skenario 2:

Diamond Merah: $w=4$, $p=2$, $d=0.5$. Red Button: $w=1$, $p=4$, $d=4$. Diamond Biru: $w=2$, $p=5$, $d=2,5$.

Pada kasus skenario 2, bot akan memilih diamond merah dengan nilai $d=2$, lalu diamond biru dengan nilai $d=2,5$, dan red button dengan nilai $d=4$. Asumsikan inventory bot hanya dapat menampung 4 bobot lagi. Maka bot akan mengambil diamond merah, lalu pergi ke red button untuk mereset lokasi diamond.

Skenario 3:

Diamond Merah: $w=4$, $p=2$, $d=0.5$. Diamond Biru Atas: $w=2$, $p=1$, $d=0.5$. Diamond Biru Bawah: $w=2$, $p=2$, $d=1$.

Pada kasus skenario 3, nilai densitas Diamond Merah sama dengan nilai densitas Diamond Biru Atas. Dalam kasus ini bot akan memilih nilai bobot(2) yang lebih besar yaitu Diamond Merah.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan implementasi dan pengujian yang telah dilakukan pada bot dalam permainan Diamonds kami menyimpulkan bahwa strategi yang paling efektif adalah strategi Greedy by Density, strategi ini dinilai lebih efektif dibandingkan dua strategi lainnya. Bot kami juga mampu memilih target diamond atau reset button yang memberikan hasil terbaik per satuan jarak. Hasil pengujian juga menunjukkan bahwa keputusan bot selaras dengan strategi yang diharapkan. Oleh karena itu implementasi bot Pesemka berhasil menunjukkan performa yang optimal dalam konteks strategi algoritma greedy dengan keputusan yang terukur dan efisien berdasarkan kondisi papan permainan.

LAMPIRAN

A. Repository Github

[Tubes 1 Stima Pesemka](#)

B. Video Penjelasan

[YouTube](#)

DAFTAR PUSTAKA

- [1] R. Munir, "Algoritma Greedy (Bagian 1)," IF2211 Strategi Algoritma. Sekolah Teknik Elektro dan Informatika ITB, Bandung, Indonesia. [Materi Kuliah]. Tersedia: [[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-\(2025\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf)]. [Diakses: 1 Juni 2025].