Tugas Besar 1 IF2211 Strategi Algoritma

Pemanfaatan Algoritma Greedy dalam pembuatan bot permainan Diamonds

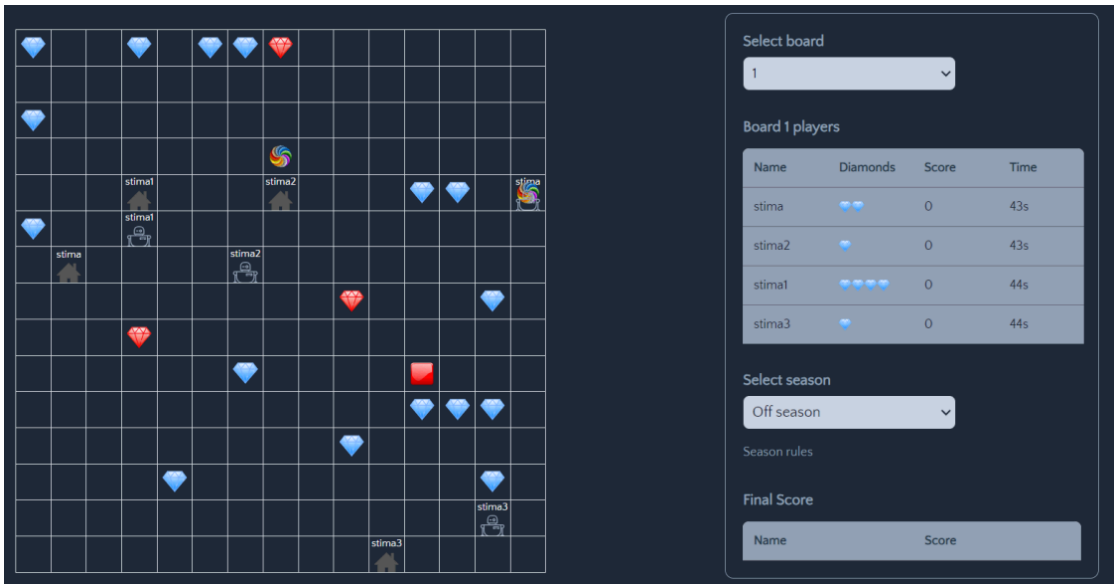
Tazkirah Amaliah (10023608)
Zaki Yudhistira Candra (13522031)
Dzaky Satrio Nugroho (13522059)

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2024

Daftar Isi

Bab 1: Deskripsi Masalah

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.



Komponen-komponen dari permainan Diamonds antara lain:

1. *Diamonds*

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*.

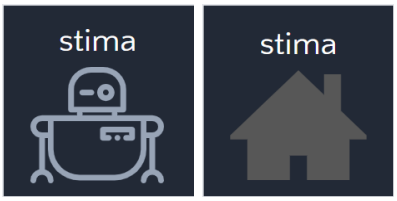
2. *Red Button/Diamond Button*

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*

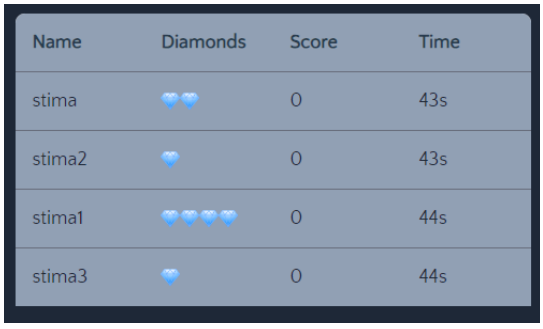
Terdapat 2 *teleporter* yang saling terhubung satu sama lain. Jika bot melewati sebuah *teleporter* maka bot akan berpindah menuju posisi *teleporter* yang lain.

4. *Bots and Bases*



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.

5. *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.

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

Bab 2: Landasan Teori

2.1. Algoritma *Greedy*

Algoritma *greedy* adalah metode yang cukup ampuh untuk melakukan optimasi tetapi tidak dapat menjamin akan mendapatkan hasil yang paling optimum. Optimasi hanya ada 2, yaitu maksimasi dan minimasi. Prinsip *greedy* adalah mengambil pilihan optimum pada tiap langkah dengan harapan akan mendapatkan hasil yang maksimal secara global.

2.2. Cara Kerja Program

Permainan ini merupakan permainan berbasis *web*, sehingga setiap aksi yang dilakukan mulai dari mendaftarkan bot hingga menjalankan aksi bot akan memerlukan HTTP *request* terhadap API *endpoint* tertentu yang disediakan oleh *backend*. Berikut adalah urutan *requests* yang terjadi dari awal mula permainan.

1. Program bot akan mengecek apakah bot sudah terdaftar atau belum, dengan mengirimkan *POST request* terhadap *endpoint* `/api/bots/recover` dengan *body* berisi *email* dan *password* bot. Jika bot sudah terdaftar, maka *backend* akan memberikan *response code* 200 dengan *body* berisi id dari bot tersebut. Jika tidak, *backend* akan memberikan *response code* 404.
2. Jika bot belum terdaftar, maka program bot akan mengirimkan *POST request* terhadap *endpoint* `/api/bots` dengan *body* berisi *email, name, password*, dan *team*. Jika berhasil, maka *backend* akan memberikan *response code* 200 dengan *body* berisi id dari bot tersebut.
3. Ketika id bot sudah diketahui, bot dapat bergabung ke *board* dengan mengirimkan *POST request* terhadap *endpoint* `/api/bots/{id}/join` dengan *body* berisi board id yang diinginkan (preferredBoardId). Apabila bot berhasil bergabung, maka *backend* akan memberikan *response code* 200 dengan *body* berisi informasi dari *board*.
4. Program bot akan mengkalkulasikan *move* selanjutnya secara berkala berdasarkan kondisi *board* yang diketahui, dan mengirimkan *POST request* terhadap *endpoint* `/api/bots/{id}/move` dengan *body* berisi *direction* yang akan ditempuh selanjutnya (“NORTH”, “SOUTH”, “EAST”, atau “WEST”). Apabila berhasil, maka *backend* akan memberikan *response code* 200 dengan *body* berisi kondisi *board* setelah *move* tersebut. Langkah ini dilakukan terus-menerus hingga waktu bot habis. Jika waktu bot habis, bot secara otomatis akan dikeluarkan dari *board*.

5. Program *frontend* secara periodik juga akan mengirimkan *GET request* terhadap *endpoint* `/api/boards/{id}` untuk mendapatkan kondisi *board* terbaru, sehingga tampilan *board* pada *frontend* akan selalu ter-*update*.

Bab 3: Aplikasi strategi *greedy*

3.1. Proses mapping persoalan Diamonds menjadi elemen-elemen algoritma Greedy

Proses mapping persoalan diamond ke elemen-elemen algoritma Greedy dapat melibatkan pengidentifikasian aspek-aspek yang dapat dimodelkan menggunakan pendekatan greedy. Dalam hal ini, algoritma Greedy dapat menjadi pendekatan yang berguna untuk memetakan persoalan ini ke dalam serangkaian langkah-langkah yang berfokus pada memaksimalkan pendapatan poin dari diamond di setiap tahap proses algoritmanya.

a. Himpunan kandidat : Semua potensi lokasi atau tindakan yang dapat diambil untuk mendapatkan diamond.
b. Himpunan solusi : Action yang telah dipilih untuk mendapatkan diamond berdasarkan strategi Greedy.
c. Fungsi solusi : Memeriksa apakah action yang dipilih telah berhasil dalam menambahkan jumlah diamond.
d. Fungsi seleksi : Memilih action yang paling efisien dalam menambahkan jumlah diamond yang diperoleh. Hal ini melibatkan action yang memiliki potensi untuk memberikan hasil yang terbesar dalam waktu yang paling efisien.
e. Fungsi kelayakan : Memeriksa apakah action yang dipilih adalah valid dan dapat dijalankan.
f. Fungsi objektif : Mencari output penambahan jumlah diamond yang maksimum dengan action ketika dijalankan akan memberikan jumlah diamond yang paling banyak.

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

Terdapat beberapa alternatif solusi Greedy yang dapat dipertimbangkan dalam persoalan Diamonds. Berikut adalah beberapa contoh alternatif solusi Greedy yang mungkin dipilih:

a. Greedy by Proximity (Ganjaran Terdekat)

Bot akan memprioritaskan pengambilan diamond yang terletak paling dekat dari posisi saat ini. Bot akan mencari diamond terdekat dan bergerak ke arahnya tanpa memperhatikan ukuran diamond atau faktor-faktor lainnya. Strategi ini bertujuan untuk

memaksimalkan jumlah diamond yang bisa diambil dalam waktu sesingkat mungkin dengan memanfaatkan diamond yang mudah dijangkau.

b. Greedy by Size (Ganjaran Berdasarkan Ukuran):

Bot akan memprioritaskan pengambilan diamond berdasarkan ukurannya atau nilai poinnya. Bot akan mencari diamond dengan nilai poin tertinggi dan berusaha untuk mengambilnya terlebih dahulu. Strategi ini bertujuan untuk memaksimalkan total poin diamond yang dikumpulkan, walaupun mungkin memerlukan waktu lebih lama untuk mencapai diamond dengan nilai poin yang lebih tinggi.

c. Greedy by Safety (Ganjaran dengan Keamanan):

Dalam strategi ini, bot akan mempertimbangkan faktor keamanan dalam pengambilan diamond. Bot akan cenderung mengambil diamond yang terletak di area yang lebih aman atau terhindar dari ancaman seperti bot musuh atau bahaya lainnya. Strategi ini bertujuan untuk meminimalkan risiko yang dihadapi bot saat mengambil diamond, meskipun mungkin mengorbankan potensi poin yang lebih tinggi.

d. Greedy by Resource Efficiency (Ganjaran dengan Efisiensi Sumber Daya):

Strategi ini bertujuan untuk memaksimalkan penggunaan teleporter saat mengumpulkan Diamond. Bot akan mencari diamond dengan pertimbangan efisiensi, misalnya memilih diamond yang dapat dijangkau dengan sedikit pergerakan atau mengambil diamond yang membutuhkan waktu lebih sedikit untuk mencapainya.

e. Greedy by Combination (Ganjaran Kombinasi):

Strategi ini menggabungkan beberapa kriteria seperti ukuran diamond, jarak, dan keamanan untuk menentukan diamond mana yang akan diambil lebih dulu. Bot akan menimbang berbagai faktor tersebut dan memilih Diamond yang memberikan hasil terbaik secara keseluruhan.

3.3. Analisis efisiensi dan efektivitas dari kumpulan alternatif solusi greedy yang dirumuskan

Pada program ini menggunakan pendekatan algoritma iteratif untuk mencari jarak terdekat antara bot dan objek diamond satu persatu atau reset button. Dengan menggunakan algoritma pengurutan, urutan pertama diamond yang ditemukan memiliki

jumlah poin satu dan dua dalam daftar diamond. Kemudian membandingkan jarak masing-masing diamond dengan posisi bot untuk menemukan diamond terdekat. Proses ini dilakukan dengan iterasi melalui daftar diamond dan membandingkan jarak masing-masing diamond dengan jarak diamond yang telah ditemukan sebelumnya. Namun, tidak ada pengurutan eksplisit yang dilakukan dalam implementasi program tersebut. Keseluruhan kompleksitas dari solusi greedy yang dirumuskan dalam program ini adalah :

T(n) = o(n) **(dengan n = jumlah diamond dan objek lainnya di board game)**

Strategi dari algoritma greedy yang dibuat akan memaksimalkan pendapatan poin diamond dengan mengambil diamond yang tersedia secara efisien dan dengan mempertimbangkan kondisi permainan. Meskipun tidak selalu menjadi strategi yang paling efektif dalam semua situasi, strategi greedy ini dapat memberikan hasil yang baik jika diterapkan dengan bijak dalam konteks permainan yang tepat.

Strategi efektif apabila:

a. Diamond tersedia dalam jumlah yang cukup. Strategi ini akan lebih efektif jika terdapat cukup banyak diamond yang tersebar di sekitar board permainan. Ketersediaan diamond yang cukup akan memungkinkan bot untuk terus memperoleh diamond.
b. Jarak antara bot relatif jauh. Ketika bot beroperasi dalam lingkungan di mana jarak antara bot sendiri dan bot lawan lainnya relatif jauh karena jarak yang jauh dapat memberikan kesempatan bagi bot untuk fokus pada pengumpulan diamond tanpa terganggu oleh interaksi dengan bot musuh.

Strategi tidak efektif apabila:

a. Bot lawan menggunakan strategi serangan. Jika bot musuh menggunakan strategi serangan yang mengarah dalam mempertemukan antar bot, maka bot sendiri akan dikirim ke *home base* dan semua *diamond* pada *inventory* bot sendiri akan hilang, dan diambil masuk ke *inventory* bot lawan.
b. Persediaan diamond yang terbatas. Jika diamond tersedia dalam jumlah yang terbatas atau tersebar di area yang jarang, strategi greedy mungkin tidak efektif. Bot akan menghabiskan terlalu banyak waktu untuk mencari diamond tanpa hasil yang memuaskan.
c. Persaingan yang intens dengan bot lain. Ketika terdapat banyak bot lawan yang juga menggunakan strategi greedy atau bersaing untuk mendapatkan diamond sebanyak-banyaknya, persaingan dapat menjadi sangat ketat. Hal ini dapat membuat bot kesulitan untuk memperoleh diamond dengan cepat atau tanpa terlibat dalam konflik dengan bot lain.

3.4. Strategi greedy yang dipilih

Program akan memilih langkah selanjutnya berdasarkan beberapa kasus dengan prioritas 1 paling tinggi:

1. Jarak ke base + 1 = sisa waktu, maka kembali ke *base*.
2. Inventori diamond = 5, maka kembali ke *base*.
3. Inventori diamond >= 3, maka menuju hal yang paling dekat diantara base, diamond terdekat, dan reset.
4. Jarak reset > jarak diamond, maka ke diamond terdekat.
5. Jarak diamond > jarak reset, maka ke reset button.

Algoritma yang digunakan tidak membedakan antara diamond merah dan biru, karena perbandingan diamond merah dan biru yang sangat kecil dan perbedaan poin hanya 1, tetapi jika inventori = 4 hanya mencari diamond biru saja.

Bab 4: Implementasi dan pengujian

4.1. Implementasi algoritma greedy pada program bot

Fungsi menghitung jarak dengan *manhattan distance*

```
return abs(posisi x objek1 - posisi x objek2) + abs(posisi y objek1
- posisi y objek2)
```

Fungsi pencarian diamond terdekat

```
if (inventory = 4) then
    array_diamond = [semua diamond biru]
    nearestDiamond = array_diamond[0]
    nearestDistance = jarak(array_diamond[0],bot)
    for i in range(1,len(array_diamond)
        if (jarak(array_diamond[i],bot) < nearestDistance) then
            nearestDistance = jarak(array_diamond[i],bot)
            nearestDiamond = array_diamond[i]
else
    array_diamond = [semua diamond]
    nearestDiamond = array_diamond[0]
    nearestDistance = jarak(array_diamond[0],bot)
    for i in range(1,len(array_diamond)
        if (jarak(array_diamond[i],bot) < nearestDistance) then
            nearestDistance = jarak(array_diamond[i],bot)
            nearestDiamond = array_diamond[i]
return nearestDiamond
```

Berikut adalah implementasi dari algoritma greedy kami dalam pseudocode, program akan dijalankan pada tiap iterasi.

```
if (WaktuTersisa > JarakKeBase) then
    if (inventory = 5) then
        Tujuan = base
    else if (inventory >= 3) then
        if (base lebih dekat dari button dan diamond terdekat) then
            Tujuan = base
        else if (diamond terdekat lebih jauh dari button) then
            Tujuan = button
        else
            Tujuan = diamond terdekat
    else
        if (diamond terdekat lebih jauh dari button) then
            Tujuan = button
        else
```

```
            Tujuan = diamond terdekat
else
    Tujuan = base
```

4.2. struktur data yang digunakan dalam program bot Diamonds

Pada pembuatan bot ini struktur data yang digunakan hanyalah larik. Larik digunakan untuk menyimpan diamond yang ada pada papan permainan yang akan diurutkan untuk mendapatkan langkah optimum lokal.

4.3. Pengujian

Hal yang diuji	Implementasi	Hasil
Pencarian diamond terdekat dan hanya mencari biru apabila inventory = 4.	Melakukan pengurutan dari larik diamond yang ada pada papan.	Bot menuju diamond terdekat sesuai dengan perhitungan.
Kembali ke base jika waktu sudah mau habis.	Apabila waktu tersisa + 1 <= jarak ke base maka atur tujuan ke base.	Bot kembali ke base apabila waktu tersisa + 1 <= jarak ke base.
Kembali ke base jika inventory = 5.	Cek apakah inventory = 5.	Bot kembali ke base apabila inventory sudah 5.
Kembali ke base jika inventory >= 3 dan base lebih dekat daripada diamond terdekat maupun button.	Cek inventory dan jarak bot ke diamond terdekat dan tombol reset.	Bot kembali ke base apabila inventory >= 3 dan base lebih dekat daripada diamond terdekat maupun button.
Ke reset button jika inventori < 3 dan lebih dekat daripada diamond.	Cek inventory dan jarak bot ke diamond terdekat dan tombol reset.	Bot ke reset button apabila inventori < 3 dan lebih dekat daripada diamond.
Apabila tujuan dihalangi oleh teleporter	Cek apakah ada teleporter yang di sumbu y yang sama dengan tujuan. Jika ada maka bot akan naik/turun hingga sumbu x nya sama dengan tujuan	Bot berhasil menghindari teleporter.

Bab 5: Kesimpulan dan saran

5.1. Kesimpulan

Kesimpulan yang dapat ditarik oleh kami adalah bahwa algoritma greedy dapat digunakan untuk mencapai solusi lokal yang optimal dari situasi saat ini. Namun, penting untuk dicatat bahwa solusi lokal ini tidak selalu menjadi solusi global optimal. Meskipun demikian, algoritma greedy tetap menjadi pilihan yang baik dalam pengambilan keputusan untuk pembuatan bot karena seringkali solusi lokal yang optimal sudah cukup mendekati solusi global yang optimal.

5.2. Saran

Saran untuk kami :

1. Lebih mendalami game engine.
2. Lebih teratur dalam menjadwalkan pengerjaan Tugas besar.

Lampiran

Tautan *Repository* GitHub

https://github.com/ZakiYudhistira/Tubes1_DevilTears

Tautan Video

https://youtu.be/bZICnREQens?si=j0IMG9EAUvZcQmxP

Daftar Pustaka

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf