IF2211 STRATEGI ALGORITMA
TUGAS BESAR 1
PEMANFAATAN ALGORITMA GREEDY DALAM PEMBUATAN BOT PERMAINAN DIAMONDS

Dipersiapkan oleh:

Ahmad Naufal Ramadan - 13522005

Dewantoro Triatmojo - 13522011

Azmi Mahmud Bazeid - 13522109

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG 40132
2024

Daftar Isi

1. 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.

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.

Untuk mengetahui flow dari game ini, 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.

2. Landasan Teori

2.1. Dasar Teori

Algoritma Greedy adalah pendekatan dalam pemrograman yang memecahkan persoalan optimasi dengan cara yang tampaknya “rakus”. Pendekatan ini berfokus pada pengambilan keputusan sekarang dengan harapan bahwa setiap langkah akan membawa kita lebih dekat ke solusi optimum. Algoritma Greedy digunakan untuk menemukan solusi optimal dalam persoalan optimasi dengan cepat dan berguna dalam banyak kasus, seperti perencanaan jadwal, pengkodean data, manajemen sumber daya, dan lainnya.

Kelebihan algoritma greedy adalah sederhana dan cepat (dibandingkan dengan brute force) dan dapat membantu dalam memilih sejumlah kegiatan yang memiliki prioritas atau nilai tertinggi untuk dikerjakan terlebih dahulu. Kelemahannya adalah algoritma greedy tidak selalu memberikan solusi yang optimal. Pemilihan kriteria rakus yang kurang tepat dan tidak mempertimbangkan konsekuensi jangka panjang dari setiap langkah dapat menyebabkan solusi yang tidak optimal.

Berikut merupakan skema umum algoritma greedy untuk mengoptimalisasikan suatu nilai.

```
function greedy(C: himpunan_kandidat) -> himpunan_solusi
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy }

Deklarasi
  x: kandidat
  S: himpunan_solusi

Algoritma:
S ← {} { Inisialisasi S dengan kosong }
while (not SOLUSI(S)) and C != {}) do
  X ← SELEKSI(C) { Pilih kandidat dari himpunan kandidat }
```

```
  C ← C - {x} { Buang x dari C }
  If LAYAK(S U {x}) then { Jika x layak menjadi solusi }
    S ← S U {x} { Masukkan x ke dalam himpunan solusi }
  endif
endwhile

if SOLUSI(S) then { Solusi lengkap }
  return S
else
  print("Tidak ada solusi")
endif
```

2.2. Cara Kerja Program

Bot permainan diamonds ini merupakan permainan berbasis web. Script python yang dijalankan akan mengirim HTTP request terhadap API endpoint yang sudah disediakan oleh backend. Berikut merupakan proses yang terjadi:

1. Program bot akan mengirimkan `POST` request terhadap endpoint `/api/bots/recover` dengan body email dan password untuk mengecek apakah bot sudah terdaftar.
2. Jika bot belum terdaftar, maka bot akan mengirimkan `POST` request terhadap endpoint `/api/bots` dengan body email dan password untuk mendaftarkan bot.
3. Ketika ID bot sudah diketahui, bot bergabung ke permainan dengan mengirimkan `POST` Request terhadap endpoint `/api/bots/{id}/join` dengan body berisi board id yang diinginkan.
4. Untuk menjalankan bot pada board, program bot mengirimkan `POST` request terhadap endpoint `/api/bots/{id}/move` dengan body yang berisi direction ("NORTH", "SOUTH", "EAST", atau "WEST"). Langkah ini dilakukan terus menerus sampai waktu permainan habis.
5. Program frontend secara periodik juga mengirimkan `GET` request terhadap endpoint `/api/boards/{id}` untuk mendapatkan kondisi board yang terbaru.

Algoritma bot permainan diamonds dapat diimplementasikan dengan cara membuat kode python baru di directory `/src/game/logic`. Dalam file tersebut, buatlah sebuah class yang memiliki method `next_move` dengan parameter `self`, `board_bot` (bertipe `GameObject`), dan `board` (bertipe `Board`). Dari parameter `board_bot`, kita bisa mendapatkan state bot kita sekarang. Dari parameter `board`, kita bisa mendapatkan state board sekarang seperti lokasi diamonds, lokasi diamond button, teleporter, dll. Algoritma greedy diimplementasikan pada method next_move ini untuk menentukan arah gerak bot berdasarkan logika greedy dengan mengambil pilihan optimum lokal yang nantinya diharapkan akan menuju optimum global. Arah gerak yang telah ditentukan sebelumnya akan dikembalikan oleh method `next_move` untuk dikirim ke API endpoint. Setelah membuat logic bot dalam kode python, jangan lupa untuk mendaftarkan logic bot pada dictionary `CONTROLLERS` dalam file `/src/main.py` sehingga bisa dijalankan menggunakan script melalui CLI.

Bot permainan diamonds yang telah kami buat mengimplementasikan algoritma greedy untuk mendapatkan diamond paling banyak dalam waktu yang terbatas. Terdapat beberapa implementasi bot strategi greedy yang mengambil sudut pandang berbeda dalam strateginya:

1. Greedy (Logika greedy utama)
2. GreedyDense (Logika greedy by diamond density)
3. Chaser (Logika pergerakan bot mengejar bot lain)
4. Sandwich (Logika greedy dasar yang memilih preferensi terhadap teleporter dan red button secara acak)
5. DewoDT (Logika greedy dasar yang ditambah tambahan strategi portal, diamond button dan heuristic)
6. Aggresive (Logika greedy yang lebih menjaga jarak dari bot lain karena ada bug dari game Diamonds)

Kami juga telah membuat beberapa bot dengan tujuan untuk melakukan testing:

1. Wasd (Logika bot yang bisa dikontrol menggunakan tombol keyboard WASD)
2. Circular (Logika bot yang bergerak mengelilingi board)

Untuk menjalankan bot, lakukan langkah-langkah berikut.

1. Clone repository dan masuk ke root directory project tersebut.

```
git clone https://github.com/SandWithCheese/Tubes1_DNA.git
```

2. Masuk ke directory `/src`.

```
cd src
```

3. Untuk menjalankan 1 bot saja,

```
python main.py --logic <LOGIC_NAME> --email=<UNIQUE_EMAIL>
--name=<NAME> --password=<PASSWORD> --team <TEAM_NAME>
```

4. Untuk menjalankan lebih dari beberapa bot, gunakan script yang sudah dibuat
   Untuk Windows,

```
./run-bots.bat
```

   Untuk Linux / (possibly) macOS,
   Sebelum menjalankan script, pastikan untuk memberikan izin kepada shell script agar bisa dieksekusi

```
chmod +x run-bots.sh
```

   Jalankan script untuk banyak bot dengan command

```
./run-bots.sh
```

3. Aplikasi Strategi Greedy

3.1. Proses Mapping Persoalan Diamonds Menjadi Elemen-Elemen Algoritma Greedy

Dalam skema algoritma greedy umum, terdapat beberapa elemen penting seperti himpunan kandidat, himpunan solusi, fungsi solusi, fungsi seleksi, fungsi kelayakan, dan fungsi objektif.

Himpunan kandidat adalah multiset dimana elemen-elemennya merupakan kandidat solusi dari algoritma greedy. Himpunan kandidat meliputi seluruh objek dalam game yang dapat berinteraksi dengan bot kita seperti diamond (biru & merah), diamond button, teleporter, base, dan bot lawan.

Himpunan solusi adalah adalah multiset dimana elemen-elemennya merupakan solusi dari algoritma greedy sesuai dengan strategi greedy yang dipilih. Himpunan solusi meliputi seluruh objek dalam game yang telah berinteraksi/dilewati oleh bot kita. Misalnya diamond yang telah diambil, teleporter yang sudah dilewati, base yang sudah dilewati, diamond button yang sudah dilewati, dll.

Fungsi solusi adalah sebuah fungsi untuk menentukan apakah himpunan solusi yang terpilih sudah memenuhi solusi atau belum. Fungsi solusi mengembalikan true jika waktu game sudah selesai (game berhenti).

Fungsi seleksi adalah sebuah fungsi untuk menyeleksi kandidat berdasarkan kriteria tertentu. Fungsi seleksi ini berbeda-beda bergantung strategi greedy. Penjelasan fungsi seleksi lebih spesifik akan dijelaskan pada bab 3.2 untuk setiap strategi greedy yang kami design karena setiap strategi berbeda-beda fungsi seleksinya.

Fungsi kelayakan adalah sebuah fungsi untuk mengecek apakah kandidat yang akan dipilih jika digabungkan dengan himpunan solusi memenuhi solusi atau tidak. Secara umum, bot yang kami buat akan menghindari tackle sehingga jika ada bot yang berada atau mungking next possible movenya akan berada pada koordinat kandidat tersebut, kandidat tersebut menjadi tidak layak. Selain itu, jika waktu yang dibutuhkan untuk bot kembali ke base tidak cukup untuk mengambil diamond kandidat tesebut, maka bot akan tetap berada di base.

Fungsi objektif adalah sebuah fungsi untuk memaksimumkan diamond yang diambil dalam waktu yang terbatas. Penjelasan fungsi objektif lebih spesifik akan dijelaskan pada bab 3.2 untuk setiap strategi greedy yang kami design karena setiap strategi berbeda-beda fungsi objektifnya.

3.2. Eksplorasi Alternatif Solusi Greedy yang Mungkin Dipilih dalam Persoalan Diamonds

1. Greedy
   Strategi Greedy adalah strategi yang memanfaatkan algoritma greedy untuk mendapatkan diamond paling banyak dengan mencari jalur menuju goal dengan pemanfaatan BFS. Strategi Greedy sudah mempertimbangkan teleporter untuk menuju goal. Dengan memanfaatkan kedua hal tersebut, bot akan bergerak secara optimal menuju goal. Strategi ini tidak mempertimbangkan untuk menuju red button sama sekali.

Strategi bot ini mengimplementasikan fungsi seleksi & objektif dengan cara mentraversal seluruh list diamond yang tersedia dalam board dan menghitung jaraknya. Diamond yang akan dipilih adalah diamond dengan jarak terdekat.

2. GreedyDense
   Strategi GreedyDense adalah strategi yang memanfaatkan algoritma greedy untuk mendapatkan diamond paling banyak dengan mencari block terdekat yang memiliki diamond paling banyak. Strategi ini akan membagi board menjadi block lebih kecil berukuran 5x5 dan menghitung densitas diamond pada masing-masing block. Strategi ini sudah mempertimbangkan teleporter.

   Strategi bot ini mengimplementasikan fungsi seleksi & objektif dengan cara mentraversal seluruh list diamond yang tersedia dalam board dan menghitung densitas diamond di peta dengan ukuran 5x5 dibagi dan juga jaraknya lalu diberikan pembobotan. Peta dengan nilai parameter terbesar sebelumnya akan dipilih.

3. Chaser
   Strategi Chaser adalah strategi yang memanfaatkan fitur game yaitu tackle. Strategi ini akan terus mengejar bot lawan hingga terjadi tackle. Setelah terjadi tackle, bot ini akan mendapatkan seluruh diamond milik lawan lalu kembali ke base.

   Strategi bot ini mengimplementasikan fungsi seleksi & objektif dengan cara mentraversal seluruh list bot yang tersedia dalam board dan menghitung jaraknya. Bot yang akan dipilih adalah bot dengan jarak terdekat.

4. Sandwich
   Strategi Sandwich adalah strategi yang memanfaatkan algoritma greedy untuk mendapatkan diamond paling banyak dengan mencari diamond terdekat. Strategi ini sudah mempertimbangkan teleporter. Jika bot lebih dekat dengan teleporter daripada diamond terdekat, maka bot akan menuju teleporter. Strategi ini memiliki preferensi terhadap teleporter dan red button secara acak jika inventory bot belum penuh.

   Strategi bot ini mengimplementasikan fungsi seleksi & objektif dengan cara mentraversal seluruh list diamond yang tersedia dalam board dan menghitung jaraknya. Diamond yang akan dipilih adalah diamond dengan jarak terdekat.

5. DewoDT
   Strategi DewoDT adalah strategi yang didasari dari beberapa prinsip. Pertama, bot ini mengikuti prinsip stoicisme yaitu hindari sebisa mungkin hal-hal yang tidak bisa dikontrol/prediksi seperti tackle dan diamond button. Kedua, efisiensi yaitu ambil diamond dengan jarak terkecil dan berusaha untuk kembali ke base hanya jika inventory full atau waktu yang tersisa kurang dari waktu yang dibutuhkan untuk kembali ke base. Yang ketiga, diamond button hanya digunakan jumlah diamond <= 3 (karena akan berebut diamond). Yang terakhir, teleporter dapat dimanfaatkan untuk mendapatkan diamond atau kembali ke base.

Strategi bot ini mengimplementasikan fungsi seleksi & objektif dengan cara mentraversal seluruh list diamond yang tersedia dalam board dan menghitung jaraknya. Diamond yang akan dipilih adalah diamond dengan jarak terdekat.

6. Aggresive
   Strategi Aggresive adalah strategi yang memanfaatkan algoritma greedy untuk mendapatkan diamond paling banyak dengan mencari diamond terdekat, Ada sedikit tambahan logika di strategi ini yaitu bot ini lebih menjaga jarak dari bot lain. Hal ini karena adanya bug dalam game yang dapat mengakibatkan suatu bot dapat bergerak langsung melewati 2 cell.

   Strategi bot ini mengimplementasikan fungsi seleksi & objektif dengan cara mentraversal seluruh list diamond yang tersedia dalam board dan menghitung jaraknya. Diamond yang akan dipilih adalah diamond dengan jarak terdekat.

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

1. Greedy
   Strategi Greedy merupakan strategi yang sangat efisien dalam mencari jalur optimal untuk menuju goal. Pencarian jalur ini dilakukan dengan mempertimbangkan teleporter. Tanpa mempertimbangkan red button, strategi ini sudah cukup optimal dalam mengumpulkan diamond.

2. GreedyDense
   Strategi GreedyDense secara teori dapat mempercepat bot untuk mendapatkan diamond dengan menuju ke block yang memiliki densitas diamond tertinggi. Namun, pada kenyataannya strategi ini kurang efektif karena permainan dilakukan dengan bot lain sehingga densitas dari tiap block dapat berubah-ubah seiring berjalannya waktu. Karena hal ini, strategi ini memiliki kesulitan untuk menentukan tujuannya karena dapat berubah-ubah dengan cepat.

3. Chaser
   Strategi Chaser secara teori merupakan salah satu strategi yang dapat mengumpulkan diamond lebih banyak secara efisien. Namun, pada kenyataannya strategi ini kurang efektif dalam mendapatkan diamond. Dengan mekanisme game yang hanya memungkinkan bot untuk bergerak dalam selang 1 detik, sangat sulit bagi strategi ini untuk berhasil melakukan tackle pada bot lain. Jika bot lawan mengimplementasikan strategi untuk menjaga jarak dari bot lain, maka strategi Chaser akan sangat tidak efektif dan hanya akan mengejar bot lawan tersebut tanpa melakukan tackle.

4. Sandwich
   Strategi Sandwich merupakan strategi yang cukup efisien dalam mengumpulkan diamond. Dengan memanfaatkan teleporter dan red button, strategi ini dapat mengumpulkan diamond di keadaan yang lebih sulit. Namun, strategi ini masih belum memprioritaskan diamond merah untuk diambil. Selain itu, pilihan terhadap penggunaan teleporter dan red button masih dilakukan

secara acak, tanpa mempertimbangkan apakah lebih baik menuju teleporter atau red button pada saat tertentu.

5. DewoDT
   Strategi DewoDT merupakan strategi yang cukup efisien dalam mengumpulkan diamond. Strategi ini sudah memanfaatkan teleporter dan red button juga. Strategi ini juga menggunakan beberapa heuristic yang simple, yaitu jika jumlah diamond sudah sedikit bot ini akan menargetkan diamond button untuk mereset diamond kembali karena jika diamond sedikit, akan berebut saling berebut. Bot ini juga bisa menghindari bot lain agar tidak terjadi resiko tackle.

6. Aggresive
   Strategi Aggresive juga memiliki masalah yang sama seperti strategi Sandwich. Strategi ini lebih mementingkan untuk menjaga jarak dari bot lain. Strategi ini cocok untuk melakukan counter pada bot Chaser.

3.4. Strategi Greedy yang Dipilih

Strategi greedy yang kami pilih adalah strategi yang bernama Greedy. Pemilihan strategi ini sebagai strategi utama adalah algoritma yang digunakan lebih kompleks dan lebih optimal dibandingkan algoritma lain. Setelah dilakukan uji coba dengan bot lain, strategi Greedy juga terbukti lebih efektif dengan lebih sering memenangkan pertandingan dibandingkan dengan bot lain yang telah kami buat.

4. Implementasi dan Pengujian

4.1. Implementasi Algoritma Greedy pada Program Bot yang Digunakan

```
function next_move(self, board_bot: GameObject, board: Board) ->
direction
{ Menentukan langkah selanjutnya dari posisi bot saat ini }

Deklarasi
  cur_pos, base: Position
  closest_diamond: Position
  is_red: matrix

  goal: Position
  distance_matrix: matrix
  closest_cell, ideal_direction: integer

Inisialisasi
  self.board_bot ← board_bot
  self.board ← board

  cur_pos ← board_bot.position
```

```
  base ← board_bot.properties.base
  closest_diamond ← self.closest_diamond()
  is_red ← self.red_matrix()

Algoritma
{ Menentukan goal dari kondisi yang sudah ditentukan }
  if (

       {Jika jumlah diamond yang dimiliki sudah memenuhi inventory
maka balik ke base.}

     Board_bot.properties.diamonds =
board_bot.properties.inventory_size

     {Jika tidak cukup waktu untuk mengambil diamond terdekat maka
balik ke base.
        Diberi toleransi waktu RETREAT_DELAY untuk balik ke base.}
     or distance(cur_pos, closest_diamond) +
distance(closest_diamond, base) + RETREAT_DELAY
        > (board_bot.properties.milliseconds_left) / 1000.0

     {Jika jumlah diamond yang dimiliki sudah 4, namun diamond
terdekat adalah diamond
        merah maka balik ke base. Namun jika dalam perjalan balik ke
base diamond terdekat
        diamond biru, maka ambil diamond biru jika bisa.}
     or (
          board_bot.properties.diamonds =
board_bot.properties.inventory_size - 1
          and is_red[closest_diamond.x][closest_diamond.y]
     )
  ) then
     goal ← base
  else
     goal ← closest_diamond

  distance_matrix ← self.distance_matrix(goal)
  closest_cell ← 99999
  ideal_direction ← None

{ Mencari cell terdekat yang menuju goal }
  dx traversal [-1..1]
     dy traversal [-1..1]
          if (
          ((dx = 0) XOR (dy = 0))
```

```
            and 0 <= cur_pos.x + dx < 15
            and 0 <= cur_pos.y + dy < 15
            ) then
            if closest_cell > distance_matrix[cur_pos.x +
dx][cur_pos.y + dy] then
                closest_cell ← distance_matrix[cur_pos.x +
dx][cur_pos.y + dy]
                ideal_direction ← (dx, dy)
            endif
            endif
      endfor
  endfor

  → ideal_direction
```

4.2. Penjelasan Struktur Data yang Digunakan dalam Program Bot Diamonds

Program Bot Diamonds yang kami buat menggunakan beberapa struktur data untuk menyimpan informasi berkaitan dengan tentang papan permainan dan karakter dalam permainan. Kami menggunakan struktur data dari kit dan beberapa struktur data baru untuk memudahkan dalam penyelesaian persoalan. Struktur data yang kami gunakan adalah:

1. **GameObject**: Struktur data untuk menyimpan informasi tentang karakter dalam permainan, seperti posisi objek, tipe objek, dan properti objek (seperti base, diamonds, dan inventory_size).
2. **Board**: Struktur data untuk menyimpan informasi tentang papan permainan, seperti ukuran papan, larik berisi objek-objek dalam permainan, dan metode untuk mengakses dan memanipulasi informasi papan permainan.
3. **Position**: Struktur data untuk menyimpan informasi tentang posisi dalam permainan, seperti koordinat x dan y.
4. **Matrix**: Struktur data untuk menyimpan matriks nilai, yang digunakan dalam program ini untuk matriks yang mewakili peta permainan dan matriks lain yang mungkin diperlukan dalam perhitungan jarak dan penentuan langkah selanjutnya.
5. **Queue**: Struktur data queue digunakan untuk menyimpan posisi yang akan dieksplorasi selama algoritma BFS (Breadth-First Search) untuk pencarian jalur terpendek. Queue memungkinkan pengelolaan urutan tindakan yang akan dilakukan oleh karakter dalam permainan.

4.3. Analisis dari Desain Solusi Algoritma Greedy yang Diimplementasikan

Solusi algoritma greedy yang kami implementasikan telah mencakup hampir semua kasus yang kami anggap dapat mengembalikan diamond secara optimal. Algoritma greedy yang kami buat hanya akan menuju ke arah diamond terdekat tanpa memprioritaskan diamond biru atau merah. Pemilihan strategi ini dengan pertimbangan bahwa akan ada lebih banyak diamond yang terlewat dibandingkan dengan keuntungan yang didapat dari diamond merah. Dengan mempertimbangkan penggunaan

teleporter, bot yang kami buat dapat mengambil diamond yang terlihat jauh tetapi menjadi lebih dekat dengan menggunakan teleporter. Selain itu, dengan mempertimbangkan penggunaan teleporter, bot dapat kembali ke base lebih cepat dengan menggunakan teleporter.

Penentuan jalur menuju tujuan dilakukan dengan menggunakan algoritma BFS yang juga mempertimbangkan posisi teleporter. Kuncinya adalah bagaimana memodelkan sisi agar teleporter diperhitungkan dan semua bobotnya sama sehingga kita dapat menggunakan BFS. Graf yang digunakan dapat dijelaskan sebagai berikut:

1. Setiap cell dalam papan 15 x 15 adalah simpul.
2. Setiap dua cell yang bersebelahan yang bukan teleporter terhubung oleh 2 sisi berarah ke arah yang berlawanan (dengan bobot 1).
3. Jika sebuah cell (cell teleporter atau bukan) bersebelahan dengan cell lain yang merupakan cell teleporter, maka ada sisi berarah (dengan bobot 1) antara cell dan cell teleporter yang lainnya (cell teleporter yang tidak bersebelahan dengan cellnya).
4. Setiap cell teleporter terhubung dengan semua non-teleporter cell yang bersebelahan dengannya.

Karena ukuran papan sangat kecil, kita dapat secara efisien melakukan pengulangan pada setiap sel di papan untuk setiap gerakan.

Berikut adalah gambaran besar kode algoritma BFS:

```
# Dalam fungsi ini, goal sudah ditentukan di fungsi next_move di
atas.
def distance_matrix(self, goal: Position) -> list[list[int]]:

    visited = [[False for i in range(15)] for j in range(15)]
    distance = [[999 for i in range(15)] for j in range(15)]
    visited[goal.x][goal.y] = True
    distance[goal.x][goal.y] = 0

    q: Queue[Tuple[int, int]] = Queue()
    q.put((goal.x, goal.y))

    while not q.empty():
      v = q.get()
      x, y = v[0], v[1]

      edges: list[Tuple[int, int]] # edges dibuat sesuai penjelasan 1
- 4

      for edge in edges:
       if not visited[edge[0]][edge[1]]:
            visited[edge[0]][edge[1]] = True
            distance[edge[0]][edge[1]] = distance[x][y] + 1
            q.put(edge)
```

```
    return distance
```

Strategi yang kami buat tidak mempertimbangkan red button sama sekali. Hal ini merupakan kelebihan dan kekurangan dari strategi kami. Di satu sisi, dengan tidak mempertimbangkan red button, akan menjadi lebih sulit bagi bot lawan untuk mendapatkan diamond jika bot lawan jauh dari diamond. Namun, strategi ini kurang optimal jika terdapat red button yang jaraknya lebih dekat daripada diamond atau teleporter, sehingga bot akan berjalan lebih jauh untuk mendapatkan diamond.

5. Kesimpulan dan Saran

5.1. Kesimpulan

Melalui tugas besar ini, kami mengaplikasikan konsep algoritma greedy yang dipelajari dari kuliah IF2211 Strategi Algoritma dalam pembuatan bot game yang menyerupai permasalahan traveling salesman problem. Proyek ini memberikan pemahaman yang lebih mendalam mengenai algoritma greedy dan kemampuannya dalam menyelesaikan masalah yang memerlukan optimalisasi. Selain itu, kami juga memperoleh pemahaman yang lebih baik mengenai paradigma pemrograman OOP (Object-Oriented Programming) karena banyaknya konsep OOP yang digunakan dalam pembuatan kit bot. Proyek ini juga mendorong kami untuk mengeksplorasi algoritma-algoritma lain yang dapat digunakan dalam tugas besar ini.

5.2. Saran

1. Melakukan eksplorasi lebih lanjut terhadap berbagai solusi alternatif dan algoritma lain yang dapat digunakan untuk meningkatkan efisiensi dan performa bot.
2. Melakukan analisis lebih lanjut terhadap algoritma yang digunakan dalam bot untuk menemukan potensi optimasi dan peningkatan kinerja.
3. Melakukan pengujian lebih lanjut dengan berbagai skenario dan kondisi permainan untuk memastikan keandalan dan optimalitas bot dalam berbagai situasi..

Lampiran

Link Github: https://github.com/SandWithCheese/Tubes1_DNA
Link Video: https://youtu.be/46VSPWjHnvI

Daftar Pustaka

Munir, Rinaldi. 2024. “Algoritma Greedy (Bagian 1)”.
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf
(Diakses pada 8 Maret 2024).

Munir, Rinaldi. 2024. “Algoritma Greedy (Bagian 2)”.
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf
(Diakses pada 8 Maret 2024).

Munir, Rinaldi. 2024. “Algoritma Greedy (Bagian 3)”.
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf
(Diakses pada 8 Maret 2024).

Maintainer Python. “3.12.2 Documentation”.
https://docs.python.org/3/
(Diakses pada 8 Maret 2024).