LAPORAN TUGAS BESAR 1 IF2211 STRATEGI ALGORITMA

PEMANFAATAN ALGORITMA GREEDY DALAM PEMBUATAN BOT PERMAINAN DIAMONDS

Kelompok OkeGas

Anggota Kelompok:

1. **Samy Muhammad Haikal 13522151**
2. **Muhammad Roihan 13522152**
3. **Chelvadinda 13522154**

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023/2024

DAFTAR ISI

BAB I
DESKRIPSI TUGAS

Diamonds merupakan suatu *programming challenge* yang mempertandingkan bot yang dibuat 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. Penjelasan lebih lanjut mengenai aturan permainan akan dijelaskan di bawah.

Pada tugas pertama Strategi Algoritma ini, diminta untuk membuat sebuah bot yang nantinya akan dipertandingkan satu sama lain. Dalam membuat bot ini, harus menggunakan strategi *greedy.*

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

Para peserta akan ditantang untuk mengembangkan strategi algoritma untuk permainan Diamonds, yang melibatkan beberapa komponen sebagai berikut:

1. Diamonds

    Diamonds merupakan objek utama dalam permainan ini. Peserta bertugas untuk mengumpulkan diamonds sebanyak mungkin dengan melangkahinya. Terdapat dua jenis diamonds, yaitu diamond biru dan diamond merah. Diamond merah memiliki nilai dua poin, sementara diamond biru bernilai satu poin. Diamonds akan di-generate kembali secara berkala, dan rasio antara diamond merah dan biru akan berubah setiap kali proses regenerasi dilakukan.

2. Red Button/Diamond Button

    Red button atau Diamond button merupakan objek yang ketika dilewati atau dilangkahi, akan menghasilkan regenerasi kembali semua diamonds (termasuk red

diamonds) pada papan permainan dengan posisi acak. Posisi dari red button ini juga akan berubah secara acak setiap kali dilewati.

3. Teleporters

   Terdapat dua teleporter yang saling terhubung. Ketika bot melewati salah satu teleporter, bot akan dipindahkan ke posisi teleporter yang lainnya.

4. Bots and Bases

   Peserta akan mengendalikan bot untuk mengumpulkan diamond. Setiap bot memiliki sebuah base yang berfungsi untuk menyimpan diamond yang telah diambil. Ketika diamond disimpan ke base, skor bot akan bertambah sesuai dengan jumlah diamond yang disimpan, dan inventory bot akan menjadi kosong.

5. Inventory

   Setiap bot memiliki inventory yang berfungsi sebagai tempat penyimpanan sementara diamond yang telah diambil. Inventory ini memiliki kapasitas maksimum, dan bot harus kembali ke base untuk mengosongkan inventory agar dapat mengambil diamond lagi.

Setiap bot akan bergerak di papan permainan dan mengumpulkan diamonds sesuai dengan strategi yang telah dirancang. Masing-masing bot akan ditempatkan secara acak di papan permainan dengan home base, score, dan inventory awal yang bernilai nol. Objektif utama dari bot adalah mengumpulkan diamond sebanyak mungkin, dengan diamond merah bernilai dua poin dan diamond biru bernilai satu poin. Peserta juga harus memperhatikan untuk menghindari pertemuan dengan bot lawan, karena jika bot satu menimpa bot lainnya, bot yang ditimpa akan dikirim kembali ke home base dan diamond yang dibawanya akan diambil oleh bot lain. Selain itu, terdapat fitur tambahan seperti teleporter dan red button yang dapat dimanfaatkan untuk memperoleh keuntungan dalam permainan. Permainan akan berakhir ketika waktu seluruh bot telah habis dan *score* masing-masing pemain akan ditampilkan pada tabel *Final Score* di sisi kanan layar. Para peserta diharapkan dapat mengembangkan strategi yang efektif untuk mengoptimalkan pengumpulan diamond dan mencapai skor tertinggi dalam permainan Diamonds.

BAB II
LANDASAN TEORI

2.1 Algoritma Greedy

Algoritma Greedy adalah sebuah paradigma algoritma yang memilih pilihan terbaik pada setiap tahap tanpa mempertimbangkan konsekuensi ke depan, dengan harapan bahwa keputusan tersebut akan menghasilkan solusi global optimal untuk masalah yang diberikan. Algoritma ini tidak selalu menghasilkan solusi optimal untuk setiap masalah, namun seringkali memberikan solusi yang cukup baik dan efisien dalam waktu yang cepat.

Cara kerja algoritma greedy umumnya adalah dengan memilih opsi terbaik pada setiap langkah, tanpa mempertimbangkan konsekuensi pilihan tersebut pada langkah-langkah berikutnya. Algoritma ini cocok digunakan untuk masalah-masalah optimasi di mana setiap langkah dapat diambil secara independen tanpa memperhitungkan solusi keseluruhan.

Kelemahan dari algoritma greedy adalah bahwa pilihan yang diambil pada setiap langkah tidak dapat dibatalkan, sehingga jika sebuah pilihan diambil pada tahap awal yang sebenarnya tidak optimal, maka solusi keseluruhan yang dihasilkan juga mungkin tidak optimal. Oleh karena itu, algoritma greedy harus digunakan dengan hati-hati dan hanya pada masalah-masalah di mana sifat greedy dapat menghasilkan solusi yang cukup baik.

2.2 Game Engine Etimo Diamonds

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.

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

Untuk mengimplementasikan algoritma pada bot tersebut, dapat menggunakan *game engine* dan membuat bot dari *bot starter pack* yang telah tersedia pada pranala berikut.

- ***Game engine*** **:** https://github.com/haziqam/tubes1-IF2211-game-engine/releases/tag/v1.1.0
- ***Bot*** **:** https://github.com/mroihn/Tubes1_OkeGas

2.3 Cara menjalankan bot

Berikut langkah-langkah untuk menjalankan bot:

1. Pastikan python telah terinstall
2. Download source code pada bagian sebelumnya
3. Install dependencies dengan menggunakan “pip install -r requirements.txt”
4. Atur konfigurasi pada file run-bots.bat (untuk Windows) dan run-bots.sh (untuk linux)
5. Jika ingin menambah atau mengurangi jumlah bot yang ingin dimainkan copy command atau hapus command pada file run-bots sesuai dengan jumlah bot yang ingin dimainkan
6. Jalankan file run-bots pada terminal di folder src

2.4 Algoritma bot

Bot yang kami buat menggunakan algoritma Greedy dalam operasinya. Algoritma Greedy bekerja dengan memecahkan masalah langkah demi langkah, mengikuti prinsip "*take what you can get now*", di mana ia memilih opsi terbaik yang tersedia saat itu dengan harapan mencapai hasil optimal. Ketika permainan dimulai, bot akan melakukan pemindaian papan untuk menemukan langkah terbaik. Langkah terbaik menurut bot kami adalah yang memiliki poin per jarak terbesar. Bot juga akan memutuskan apakah menggunakan *red button* atau tidak. Bot akan menggunakan *red button* jika posisi *red button* lebih dekat dengan posisi bot dibandingkan dengan posisi *diamond*. Kondisi

inventory bot juga mempengaruhi langkah bot. Jika inventory bot penuh, maka bot akan kembali ke *base*. Jika inventory bot sudah berisi lebih dari dua *diamond*, bot akan memutuskan apakah akan kembali ke *base* atau melanjutkan pencarian *diamond*. Bot akan kembali ke *base* jika posisi *base* lebih dekat daripada posisi *diamond*. Bot juga akan kembali ke base jika waktu yang tersisa dikurangi waktu bot ke base (asumsi tiap langkah 1 detik) < 2.5 detik.

BAB III
APLIKASI STRATEGI GREEDY

3.1 Alternatif Solusi Greedy

3.1.1 Greedy by Distance

Greedy by distance adalah pendekatan algoritma greedy dengan memprioritaskan jarak ke diamond terdekat. Strategi hanya menghitung jarak dari diamond game object ke current position, tidak ada perbedaan perlakuan antara diamond biru dan diamond merah.

a) Mapping Elemen Greedy
   I. Himpunan Kandidat : Semua Tujuan
   II. Himpunan Solusi : Tujuan yang terpilih
   III. Fungsi Solusi : Memeriksa apakah diamond yang dituju memiliki jarak terkecil
   IV. Fungsi Seleksi : Pilih diamond dengan jarak terkecil
   V. Fungsi Kelayakan : Memeriksa apakah bot masih bisa mengambil diamond (inventory tidak penuh)
   VI. Fungsi Objektif : Mencari diamond terdekat dengan posisi bot saat ini

b) Analisis Efisiensi Solusi

   Algoritma ini cenderung efisien karena hanya perlu menghitung jarak dari posisi saat ini ke setiap diamond yang tersedia, tanpa perlu mempertimbangkan faktor-faktor lain seperti jenis diamond, red button atau teleporter. Dengan demikian, kompleksitas waktu algoritma ini dapat relatif rendah.

c) Analisis Efektivitas Solusi

   Dalam konteks tertentu, algoritma Greedy by Distance bisa menjadi pilihan yang baik karena sederhana dan mudah diimplementasikan. Namun, untuk meningkatkan efektivitas, bisa dipertimbangkan untuk memperhitungkan faktor-faktor tambahan seperti nilai atau jenis diamond untuk menyesuaikan algoritma agar lebih sesuai dengan tujuan permainan.

3.1.2 Greedy by Density

Greedy by Density adalah pendekatan algoritma greedy dengan memprioritaskan goal position dengan density poin terbesar. Penghitungan density dilakukan dengan cara membagi poin yang didapat dengan jarak yang harus ditempuh untuk mencapai tujuan.

a) Mapping Elemen Greedy
   I. Himpunan Kandidat : Semua Tujuan
   II. Himpunan Solusi : Tujuan yang terpilih

III. Fungsi Solusi : Memeriksa apakah diamond yang dituju memiliki nilai density terbesar
IV. Fungsi Seleksi : Pilih diamond dengan nilai poin/jarak terbesar
V. Fungsi Kelayakan : Memeriksa apakah bot masih bisa mengambil diamond (inventory tidak penuh)
VI. Fungsi Objektif : Mencari diamond terdekat dengan posisi bot saat ini

b) Analisis Efisiensi Solusi

Strategi ini memprioritaskan poin per langkah daripada hanya memprioritaskan jarak terdekat. Dengan memprioritaskan goal position yang memiliki density poin terbesar, algoritma ini dapat mengarahkan pemain untuk mendapatkan poin sebanyak mungkin dalam jarak tempuh yang minimal.

c) Analisis Efektivitas Solusi

Jika density poin dihitung dengan benar dan memperhitungkan semua faktor yang relevan, algoritma ini dapat memberikan solusi yang efektif dengan mengoptimalkan jumlah poin yang didapat dalam jarak tempuh yang minimal. Namun, seperti halnya dengan algoritma Greedy lainnya, algoritma ini tidak selalu menghasilkan solusi yang optimal dalam semua situasi, tergantung pada kondisi permainan dan faktor-faktor lainnya.

3.2 Solusi Greedy yang diterapkan dan Pertimbangannya

Greedy by Distance cenderung memilih diamond terdekat dari posisi pemain pada setiap langkah. Keuntungan dari pendekatan ini adalah kemampuannya untuk segera mendapatkan diamond tanpa harus melakukan perjalanan jauh. Ini dapat menghemat waktu dan upaya, serta memungkinkan pemain untuk segera mengumpulkan sejumlah diamond dalam waktu yang singkat. Namun, kelemahan dari Greedy by Distance adalah bahwa ia tidak mempertimbangkan nilai diamond. Dalam beberapa kasus, diamond yang terdekat mungkin memiliki nilai yang rendah, sementara diamond yang lebih jauh mungkin memiliki nilai yang lebih tinggi. Dengan demikian, pendekatan ini tidak selalu menghasilkan solusi yang optimal dalam mengoptimalkan jumlah nilai poin yang diperoleh.

Greedy by Density memprioritaskan goal position dengan density poin terbesar, di mana density poin dihitung dengan membagi jumlah poin yang didapat dengan jarak yang harus ditempuh untuk mencapai tujuan. Keuntungan dari pendekatan ini adalah kemampuannya untuk mengoptimalkan jumlah poin yang didapat dalam jarak tempuh yang minimal. Dengan memilih goal position yang memiliki density poin terbesar, pemain dapat meningkatkan akumulasi nilai poin mereka dalam waktu yang relatif singkat.

Berdasarkan pertimbangan optimasi poin, maka kami memilih untuk menggunakan algoritma greedy by density dikarenakan kemampuannya untuk mengoptimalkan jumlah poin dengan jarak tempuh yang minimal.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Dalam *Pseudocode*

Function _init_

Metode untuk menginisialisasi variabel-variabel yang akan digunakan oleh bot selama permainan.

```
function __init__(self):
    self.directions ⟵ [(1, 0), (0, 1), (-1, 0), (0, -1)]
    self.goal_position ⟵ null
    self.current_direction ⟵ 0
    self.button ⟵ null
```

Function next_move

Metode untuk menentukan langkah selanjutnya dari bot berdasarkan kondisi permainan saat ini, termasuk mencari diamond berdasarkan kepadatan, memutuskan menggunakan button atau tidak, kembali ke base jika inventory penuh, serta menentukan apakah ada bot lawan disekitarnya.

```
function next_move(self, board_bot: GameObject, board: Board):
    props ⟵ board_bot.properties
    current_position ⟵ board_bot.position
    base ⟵ board_bot.properties.base

    {Cari diamond berdasarkan density}
    self.findByDensityDiamond(board_bot, board)

    { Menggunakan button atau tidak}
    self.useButtonOrNot(board_bot, board)

    {Jika inventory sudah full kembali ke base}
    if props.diamonds == 5 then
        base ⟵  board_bot.properties.base
        self.goal_position ⟵ base

    {Jika inventory sudah terisi >= 3 putuskan kembali ke base atau lanjut mengambil diamond}
    elif props.diamonds >= 3 then
        self.cekBaseOrDir(board_bot, board)

    {Jika waktu yang tersisa dikurangi waktu posisi sekarang ke base < 2.5 detik, bot kembali ke base}
```

```
  distBase ⟵ abs(board_bot.properties.base.x - current_position.x) + abs( board_bot.properties.base.y - current_position.y )
    if((props.milliseconds_left - distBase*1000)<2500 and distBase ≠ 0):
      self.goal_position ⟵ base

  { Hitung delta_x dan delta_y untuk bergerak menuju goal_position}
  if self.goal_position then
    delta_x, delta_y ⟵ get_direction(
      current_position.x,
      current_position.y,
      self.goal_position.x,
      Self.goal_position.y
  else
  {move to base}
    base ⟵ board_bot.properties.base
     self.goal_position = base

  ⟶ delta_x, delta_y
```

Function findByDensityDiamond

Metode ini mencari diamond berdasarkan kepadatan tertinggi dengan mempertimbangkan jaraknya.

```
function findByDensityDiamond(board_bot: GameObject, board: Board):
    current_position ⟵ board_bot.position
    props ⟵ board_bot.properties
    minDen ⟵ -1
    goal ⟵ current_position
    {Iterasi seluruh game object}
    for i ⟵ board.game_objects do
      {Tentukan jarak dengan objek}
      dist ⟵ abs(i.position.x - current_position.x) + abs(i.position.y - current_position.y)
      {Jika ditemukan diamond cek apakah diamond merupakan diamond yang memiliki point/jarak terdekat}
      if i.type == "DiamondGameObject" and dist ≠ 0 then
        tmp ⟵ i.properties.points / dist
        if tmp ≠ 0 then
          if props.diamonds == 4 then
            if i.properties.points == 1 then
              if minDen == -1 then
                minDen ⟵ tmp
                goal ⟵ i.position
              elif tmp > minDen then
                minDen ⟵ tmp
```

```
                goal ⟵ i.position
            else
                if minDen == -1 then
                    minDen ⟵ tmp
                    goal ⟵ i.position
                elif tmp > minDen then
                    minDen ⟵ tmp
                    goal ⟵ i.position
        {Jika ketemu diamond button simpan di self.button}
        if i.type == "DiamondButtonGameObject" then
            self.button ⟵ i

    self.goal_position ⟵ goal
```

Function cekBaseOrDir

Metode ini memutuskan apakah akan kembali ke base atau melanjutkan mengambil diamond tergantung pada jarak antara bot dengan base dan jarak antara bot dengan diamond saat ini.

```
function cekBaseOrDir(self, board_bot: GameObject, board: Board):
    current_position ⟵ board_bot.position

    distBase ⟵ abs(board_bot.properties.base.x - current_position.x) + abs(board_bot.properties.base.y
- current_position.y)

    distNow ⟵ abs(self.goal_position.x - current_position.x) + abs(self.goal_position.y -
current_position.y)

    output (board_bot.properties.base)
    if distBase ≠ 0 and distNow ≠ 0 then
        if distBase < distNow then
            self.goal_position ⟵ board_bot.properties.base
```

Function useButtonOrNot

Metode ini menentukan apakah bot harus menggunakan button atau tidak tergantung jarak bot dengan button dan jarak bot dengan diamond.

```
function useButtonOrNot(self, board_bot: GameObject, board: Board):
    current_position ⟵ board_bot.position
    distBase ⟵ abs(self.button.position.x - current_position.x) + abs(self.button.position.y -
current_position.y)
    distNow ⟵ abs(self.goal_position.x - current_position.x) + abs(self.goal_position.y -
```

```
current_position.y)

  if distBase < distNow then
    self.goal_position ⟵ self.button.position
```

4.2 Struktur Data Program

Pada program bot Diamonds yang dibuat dengan bahasa *python*, terdapat struktur data yang digunakan untuk memfasilitasi permainan. Berikut penjelasan tentang struktur data yang digunakan:

a. Models

Dalam models, terdapat kelas-kelas berisi entitas yang digunakan dalam permainan beserta dengan atribut yang dimilikinya. Kelas - kelas tersebut antara lain:

- Bot: Data kelas yang merepresentasikan bot dalam permainan. Setiap bot memiliki atribut seperti nama, email, dan id.
- Position: Data kelas yang merepresentasikan posisi dalam permainan. Ini digunakan untuk menentukan posisi objek di peta permainan.
- Base: Turunan dari kelas Position yang merepresentasikan basis dalam permainan.
- Properties: Data kelas yang menyimpan informasi tambahan tentang objek permainan, seperti jumlah poin, jumlah *diamond*, *inventory*, *score*, dll. Ini digunakan untuk mengidentifikasi properti objek dalam permainan.
- GameObject: Data kelas yang merepresentasikan objek dalam permainan. Setiap GameObject memiliki ID, posisi, tipe, dan properti-properti yang terkait.
- Config: Data kelas yang menyimpan konfigurasi permainan seperti rasio generasi, ukuran inventaris, dll.
- Feature: Data kelas yang merepresentasikan fitur dalam permainan, seperti jenis permainan atau mode permainan tertentu. Ini berisi informasi tentang nama fitur dan konfigurasi terkait.
- Board: Data kelas yang merepresentasikan peta permainan. Ini memiliki atribut seperti ID, lebar, tinggi, fitur-fitur, delay minimum antar gerakan, dan objek-objek permainan yang terkait. Ini juga memiliki beberapa metode yang berguna, seperti mendapatkan semua bot atau berlian dari papan, memeriksa kevalidan gerakan, dan mendapatkan bot tertentu dari papan berdasarkan nama.

Struktur data ini digunakan untuk merepresentasikan objek-objek dalam permainan dan memfasilitasi interaksi antara bot dan sistem permainan. Dengan menggunakan struktur data ini, bot dapat memahami dan bereaksi terhadap keadaan dalam permainan dengan lebih baik.

4.3 Analisis dan Pengujian

Tujuan utama dari bot adalah mengumpulkan *diamond* yang tersebar di board permainan. Bot akan terus melakukan pergerakan untuk mencapai dan mengumpulkan *diamond* tersebut. Bot memiliki sebuah *inventory* untuk menyimpan *diamond* yang telah dikumpulkan selama perjalanan. Inventory ini memiliki kapasitas maksimum yang dapat menampung maksimal 5 *diamond*. Ketika *inventory* penuh, bot akan kembali ke base. Dengan kembali ke base akan memungkinkan bot untuk meletakkan *diamond* yang telah dikumpulkan sehingga inventory bot kosong kembali dan dapat mengumpulkan *diamond* kembali.

Bot Kembali Ke Base Ketika Inventory Full

Bot melakukan evaluasi jarak antara posisinya saat ini dengan posisi *button* dan posisi *diamond* terdekat. Jika jarak ke *button* lebih dekat daripada jarak ke *diamond* terdekat, maka bot akan memilih untuk menggunakan *button* terlebih dahulu. Dengan memilih untuk menggunakan *button* ketika jaraknya lebih dekat, bot dapat mencapai tujuan secara lebih efisien.

Bot Menggunakan Red Button

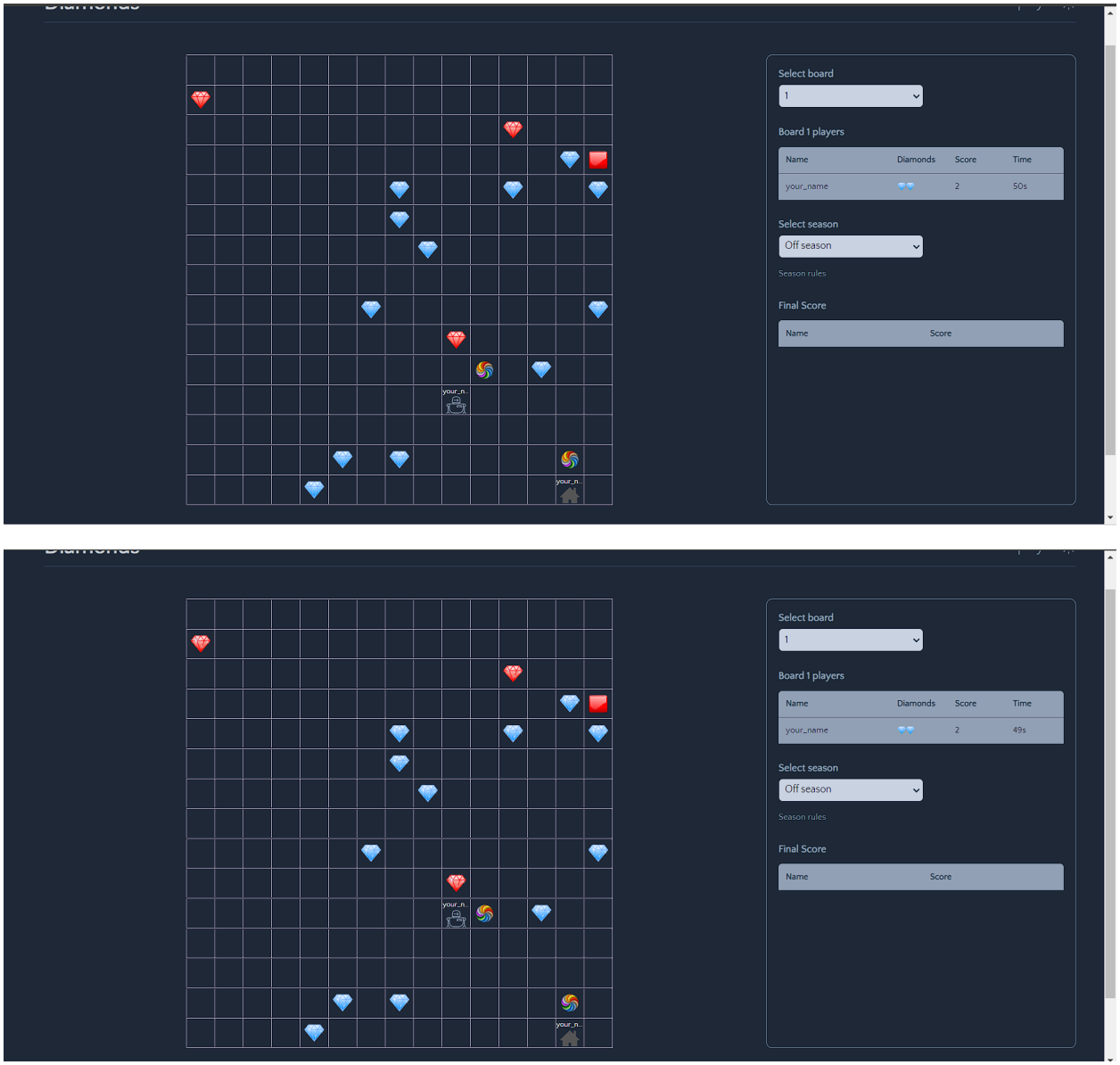
Ketika inventory terisi lebih dari dua diamond, bot melakukan evaluasi jarak antara posisinya saat ini dengan posisi *base* dan *diamond* terdekat. Jika jarak ke base lebih dekat

daripada jarak ke *diamond* terdekat, bot akan memilih untuk kembali ke *base*. Strategi ini memungkinkan bot untuk mengoptimalkan penggunaan waktu dengan efisien.

Bot kembali ke base ketika inventory terisi > 2 diamond dan jarak ke base lebih dekat

Dalam permainan, bot akan bergerak ke diamond merah karena memiliki nilai (poin / jarak) terbesar. Hal ini akan memberikan kontribusi besar terhadap peningkatan skor bot. Dengan mencari diamond yang memiliki nilai (poin dibagi jarak) terbesar, bot memiliki peluang yang lebih besar untuk mendapatkan skor yang lebih tinggi secara keseluruhan dalam permainan. Ini

karena bot dapat mengumpulkan lebih banyak poin dengan jumlah gerakan yang lebih sedikit.



Bot memilih langkah berdasarkan point per jarak terbesar

Ketika waktu yang tersisa dikurangi waktu bot ke base (asumsi tiap langkah 1 detik) < 2.5 detik bot akan kembali ke base untuk mendapatkan poin bedasarkan jumlah diamond di inventory.

Bot kembali ke base ketika waktu yang tersisa dikurangi waktu bot ke base < 2.5 detik

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dengan ini, kami berhasil membuat strategi bot untuk permainan bot *diamonds* dengan pendekatan algoritma greedy. Program akhir yang kami buat merupakan hasil dari penggabungan beberapa solusi greedy yang tersedia dan sudah kami uji untuk menghasilkan strategi terbaik versi penulis.

5.2 Saran

Saran untuk pengembangan bot *diamonds* yaitu ujikan solusi-solusi algoritma greedy yang tersedia agar mampu menciptakan strategi bot yang lebih efektif.

LAMPIRAN

Tautan *Repository* Github:

https://github.com/mroihn/Tubes1_OkeGas

Tautan Video :

https://youtu.be/O-_tmSYwVrY?feature=shared

DAFTAR PUSTAKA

Rinaldi, M. (n.d.). *Algoritma Greedy* (2021) Bag1. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

Rinaldi, M. (n.d.). *Algoritma Greedy* (2021) Bag2. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf

Rinaldi, M. (n.d.). *Algoritma Greedy* (2021) Bag3. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag3.pdf