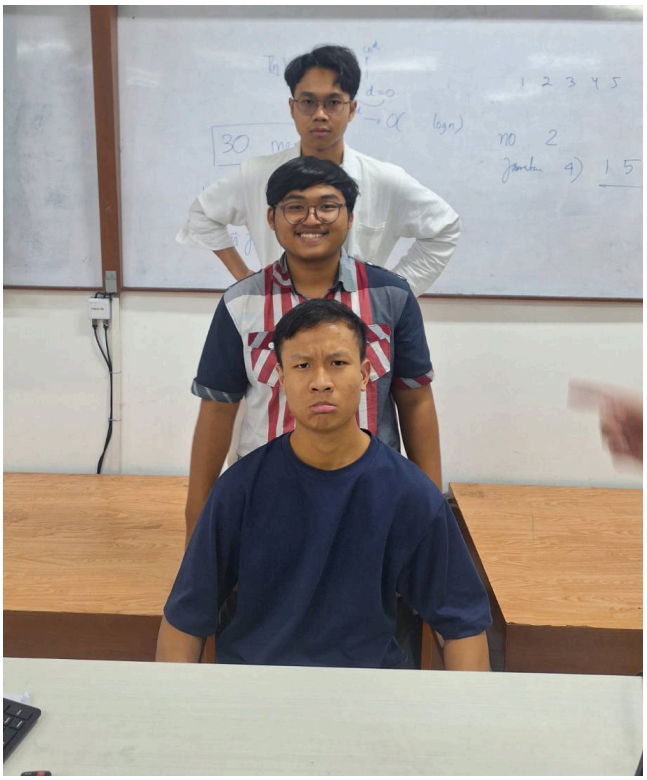
LAPORAN TUGAS BESAR 1

IF2211 STRATEGI ALGORITMA



Disusun oleh:

Tazkia Nizami	(13522032)
Dhidit Abdi Aziz	(13522040)
Muhammad Naufal Aulia	(13522074)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2024

DAFTAR ISI

BAB I
DESKRIPSI MASALAH

1.1 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. Kami menggunakan algoritma Greedy dalam membuat strategi bot permainan ini.

Gambar 1. Ilustrasi permainan Diamonds

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

Komponen-komponen dari permainan Diamonds antara lain:

1. Diamond

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.

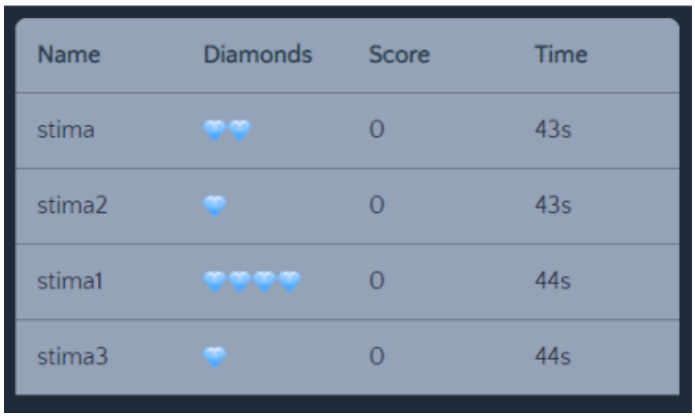
3. Teleporters

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 dibawah) 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.

Cara kerja permainan Diamond:

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 II
LANDASAN TEORI

2.1 Algoritma Greedy

Algoritma greedy adalah algoritma yang memecahkan persoalan secara langkah per langkah (step by step) sedemikian sehingga, pada setiap langkah:

a. mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsip "take what you can get now!")
b. dan "berharap" bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Sebagai contoh, pada masalah penukaran uang, strategi greedynya yaitu pada setiap langkah, pilihlah koin dengan nilai terbesar dari himpunan koin yang tersisa.

Terdapat enam elemen pada algoritma greedy:

a. Himpunan kandidat, C : berisi kandidat yang akan dipilih pada setiap Langkah (misal: simpul/sisi di dalam graf, job, task, koin, benda, karakter, dsb)
b. Himpunan solusi, S : berisi kandidat yang sudah dipilih
c. Fungsi solusi: menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi
d. Fungsi seleksi (selection function): memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik.
e. Fungsi kelayakan (feasible): memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak)
f. Fungsi obyektif : memaksimumkan atau meminimumkan

2.1 Cara Kerja Program

Pemenang dari permainan adalah bot yang berhasil mengumpulkan diamond terbanyak. Bot dapat diatur pergerakannya dan mampu bergerak ke atas, bawah, kiri, dan kanan pada papan untuk mengumpulkan diamond sebanyak-banyaknya serta menghindar dari tackle lawan. Perlu dicatat bahwa diamond baru akan dihitung apabila bot sudah kembali ke base-nya.Elemen-elemen lain pada papan, seperti bot lawan, diamond, portal, dan red button tidak dapat kita kendalikan tetapi hanya diketahui lokasinya. Maka dari itu, sudah sangat jelas bahwa algoritma greedy kita adalah berfokus untuk mengumpulkan diamond sebanyak-banyaknya dan menghindari elemen lain yang mengganggu pengumpulan diamond.

Pada permainan Diamonds, terdapat satu folder bernama “logic” yang berisikan file program python untuk mengatur jalannya bot. Secara default, sudah tersedia satu algoritma random untuk menjalankan bot. Class pada file random.py tersebut kami salin ke file baru untuk membuat class sejenis tetapi dengan algoritma yang berbeda. Penerapan algoritma greedy secara khususnya berada pada method next_move(). Kami juga menambah fungsi-fungsi sampingan untuk menjalankan program. Misalnya fungsi untuk mencari diamond terdekat, mencari jarak dari dua titik, defense dari musuh, dan lain-lain.

Langkah bot secara default bergerak penuh secara horizontal kemudian vertikal. Apabila bot kami juga berperilaku seperti itu, maka kami akan kesulitan untuk men-tackle ataupun menghindar dari lawan. Maka dari itu, kami mengubah aturan langkah bot kami agar bergantian antara horizontal dengan vertikal sehingga sulit ditebak lawan.

BAB III
APLIKASI STRATEGI GREEDY

3.1 *Mapping* Persoalan Diamonds

Persoalan Diamonds dapat dipecah menjadi elemen-elemen algoritma Greedy, yaitu

a. Himpunan kandidat, C :

   Himpunan posisi objek pada papan permainan

b. Himpunan solusi, S :

   Posisi objek yang terpilih untuk didatangi atau dihindari

c. Fungsi solusi:

   Memeriksa apakah total nilai diamond yang didapat sudah memenuhi kapasitas maksimal sehingga harus kembali ke *base*

d. Fungsi seleksi (selection function):

   Memilih objek bertipe diamond yang berjarak paling dekat dari bot

e. Fungsi kelayakan (feasible):

   Memeriksa apakah total diamond yang dibawa sudah memenuhi kapasitas maksimal serta masih tersisa waktu yang cukup untuk mencari diamond baru

f. Fungsi obyektif :

   Jumlah diamond yang didapatkan maksimum dalam waktu yang minimum

3.2 Eksplorasi Alternatif Solusi

3.2.1 Greedy by Distance

Greedy by distance merupakan pendekatan algoritma greedy pertama yang langsung terpintas untuk dieksplorasi, dimana pendekatan ini langsung memilih langkah yang paling dekat dengan tujuan yakni diamond terdekat. Diamond yang ada dalam board permainan akan dicari jaraknya yang paling dekat dengan posisi bot saat ini. Bot akan menetapkan tujuannya ke posisi diamond tersebut. Pendekatan ini sangat straightforward dengan objektif permainan, yakni mendapat poin sebanyak-banyaknya. Cara mendapat poin ialah dengan mengumpulkan diamond sebanyak-banyaknya. Dengan demikian, pendekatan ini sangat efektif dalam mengumpulkan diamond di daerah sendiri.

a. Analisis efisiensi

   Pendekatan ini sangat efisien karena membutuhkan waktu yang paling sedikit untuk dilakukan

b. Analisis efektivitas pendekatan Greedy by distance:
   1. Efektif jika:
      - Diamond terdekat berada di sekitar base awal
      - Diamond terdekat berada di sekitar posisi bot sendiri
   2. Tidak efektif jika:
      - Diamond terdekat berada di sekitar base lawan
      - Diamond terdekat berada di sekitar posisi bot lawan, sehingga memungkinkan untuk terjadi tackle

3.2.2 Greedy by Red Diamond

Greedy by red diamond merupakan pendekatan algoritma greedy yang memprioritaskan diamond merah terlebih dahulu. Diketahui bahwa diamond merah memiliki point dua kali lebih besar dibanding diamond biru. Dengan demikian, pendekatan ini sangat efektif dalam mengumpulkan poin sebanyak-banyaknya dalam sekali melangkahi diamond. Namun, perlu diingat bahwa rasio diamond merah hanya sedikit dan letak spawnnya pun tidak menentu, sehingga pendekatan ini tidak selalu efektif dalam mengumpulkan poin.

a. Analisis efisiensi
   Algoritma ini kurang efisien apabila jarak diamond merah terlalu jauh sehingga "membuang-buang waktu"

b. Analisis efektivitas pendekatan Greedy by red diamond:
   Kemunculan red diamond sangat jarang dan tersebar, maka pendekatan ini:
   1. Efektif jika:
      - Red diamond muncul di sekitar bot sendiri
      - Red diamond muncul di sekitar base awal
      - Red diamond tidak selalu menjadi prioritas utama jika jauh dari bot sendiri

   2. Tidak efektif jika:
      - Red diamond muncul di sekitar base lawan
      - Red diamond muncul di sekitar posisi bot lawan, sehingga memungkinkan untuk terjadi tackle
      - Red diamond terletak jauh dari bot sendiri, sehingga membutuhkan waktu dan langkah yang banyak untuk mencapainya

3.2.3 Greedy by Tackle

Greedy by tackle adalah pendekatan algoritma greedy dengan memfokuskan bot untuk melakukan tackle pada bot lawan, dengan begitu, diamond inventory milik lawan akan berpindah pada bot sendiri. Pendekatan ini menguntungkan karena memungkinkan untuk mendapat banyak diamond dalam waktu sangat singkat. Namun, terdapat beberapa pertimbangan yang perlu diperhatikan, yakni bot harus bergerak ke arah bot lawan, sehingga memungkinkan untuk terjebak

dalam situasi yang tidak diinginkan. Waktu eksak tiap bot bergerak pun berbeda sehingga niat untuk melakukan tackle bisa saja berbalik menjadi bot sendiri yang terkena tackle.

a. Analisis efisiensi:
   Algoritma ini tidak efisien karena waktu dihabiskan untuk mengejar-ngejar lawan

b. Analisis efektivitas pendekatan Greedy by tackle:
   Tackle efektif sebagai 'jalan pintas' mendapat diamond sebanyak-banyaknya. Namun, terdapat beberapa pertimbangan yang perlu diperhatikan, yakni:
   1. Efektif jika:
      - Timing tackle yang tepat
      - Bot lawan tidak melakukan gerakan yang menghindari tackle

   2. Tidak efektif jika:
      - Bot lawan melakukan gerakan yang menghindari tackle
      - Bot lawan sedang asyik bergerak namun bot sendiri terus mendeteksi keberadaannya, sehingga terjadi aksi kejar-kejaran yang tidak diinginkan
      - Timing tidak pas dan bot sendiri yang menjadi korban tackle

3.2.4 Greedy by Defense

Greedy by defense merupakan pendekatan yang mempertimbangkan masalah tackle dan di-tackle. Guna menjaga inventori diamond sendiri, bot harus bergerak ke arah yang aman, yakni tidak ada bot lawan yang berada di sekitar bot sendiri. Dengan demikian, pendekatan ini sangat efektif dalam menjaga diamond inventory, namun memiliki kekurangan yakni tidak memperhitungkan jarak antara bot sendiri dengan diamond terdekat, sehingga bisa saja bot bergerak ke arah yang salah. Selain itu, kembali pada masalah waktu eksak tiap bot, algoritma ini tidak selalu dapat mendeteksi keberadaan bot lawan di sekitar bot sendiri dalam waktu yang diinginkan. Sehingga tidak sepenuhnya menjaga diri dari tackle lawan melainkan hanya meminimalisir terjadinya hal tersebut.

a. Analisis efisiensi
   Algoritma ini kurang efisien karena waktu dihabiskan untuk menghindari lawan terus menerus alih-alih mengumpulkan diamond

b. Analisis efektivitas pendekatan Greedy by defense:
   Defense efektif dalam menjaga inventori diamond sendiri, namun memiliki pertimbangan yakni:
   1. Efektif jika:
      - Timing yang pas ketika mendeteksi kehadiran bot lawan di sekitar bot sendiri
      - Timing yang pas ketika bergerak menghindari bot lawan
   2. Tidak efektif jika:
      - Kehadiran bot lawan tidak sempat terdeteksi (perbedaan waktu eksak tiap bot)

- Arah menghindar ternyata menjauhi tujuan bot

3.2.5 Greedy by Avoiding Teleport

Greedy by avoiding teleport merupakan pendekatan algoritma greedy yang mempertimbangkan masalah teleporter. Niat awal kami adalah memanfaatkan teleport untuk efisiensi langkah dalam mencapai tujuan. Namun, berdasarkan observasi yang dilakukan selama permainan berlangsung, kemunculan teleporter sangat acak. Kedua teleport dapat muncul berjauhan maupun berdekatan, sehingga sangat banyak kemungkinan yang dapat terjadi termasuk loop bot keluar-masuk teleport. Dibandingkan dengan kemaslahatan yang didapat, risiko yang dihadapi jauh lebih besar. Sehingga, kami memutuskan untuk menghindari teleporter dalam pendekatan algoritma greedy ini. Dalam mencapai hal tersebut, digunakan sekuens gerakan untuk mengitari teleporter sehingga teleporter pasti terhindari.

a. Analisis efisiensi
   Algoritma ini cukup efisien karena bot dapat mencapai tujuan diamondnya dengan cepat tanpa terganggu akibat berpindah setelah menginjak teleport

b. Analisis efektivitas pendekatan Greedy by avoiding teleport:
   Menghindari teleporter efektif dalam menghindari risiko yang dihadapi, namun memiliki pertimbangan. Greedy in tidak efektif jika arah menghindar ternyata menjauhi tujuan bot.

3.2.6 Greedy by Reset Button

Greedy by reset button (diamond button) adalah pendekatan algoritma greedy yang mempertimbangkan masalah diamond button. Diamond button merupakan objek yang dapat diinjak oleh bot untuk mengacak ulang kemunculan diamond. Melalui observasi, didapat bahwa ketika diamond di sekitar bot sudah habis, akan dibutuhkan jarak yang jauh dan langkah yang banyak untuk bot menemukan diamond lainnya. Dengan memanfaatkan diamond button, bot dapat mengacak ulang kemunculan diamond sehingga dapat memperoleh diamond dengan jarak yang lebih dekat. Dalam mencapai hal tersebut, bot harus menginjak diamond button yang berada di sekitar bot. Sehingga, pendekatan ini sangat efektif dalam mengumpulkan poin sebanyak-banyaknya dalam waktu yang singkat.

a. Analisis efisiensi
   Algoritma ini efisien apabila diamond yang tersisa di sekitar bot sudah menipis sehingga perlu direset

b. Analisis efektivitas pendekatan Greedy by reset button:
   Memanfaatkan diamond button efektif dalam me-regenerate diamond untuk diambil sebanyak-banyaknya, namun memiliki pertimbangan yakni:
   1. Efektif jika:
      - Diamond button berada di sekitar bot sendiri
   2. Tidak efektif jika:

- Hasil regenerasi diamond tidak sesuai dengan harapan (makin jauh, berubah dari red ke blue diamond)
- Bot lawan mendahului menginjak diamond button

3.2.7 Greedy by Base Timing

Greedy by base timing adalah pendekatan algoritma greedy yang mempertimbangkan waktu untuk kembali ke base. Diketahui bahwa bot harus kembali ke base untuk menambahkan poin yang telah didapat. Dalam sekali sesi permainan, waktu yang ditetapkan adalah 60 detik. Yang menjadi masalah adalah ketika inventori belum penuh namun waktu sudah mendekati habis, bot tidak sempat mengumpulkan isi inventori sehingga terbuang begitu saja. Pendekatan base timing ini memperhatikan waktu akhir permainan yakni ketika waktu sudah menipis, berapapun inventori terisi saat ini harus segera disetorkan ke base (bot pulang ke base sebelum permainan berakhir). Dengan demikian, pendekatan ini efektif dalam memaksimalkan poin yang didapat dalam sekali sesi permainan.

a. Analisis efisiensi
   Algoritma ini cukup efisien apabila diamond yang telah dibawa cukup banyak sehingga sebuah keputusan yang tepat untuk kembali ke base

b. Analisis efektivitas pendekatan Greedy by base timing:
   Base timing efektif dalam memaksimalkan poin yang didapat dalam sekali sesi permainan, namun memiliki pertimbangan yakni:
   1. Efektif jika:
      - Timing pulang yang tepat
   2. Tidak efektif jika:
      - Timing pulang yang tidak tepat
      - Jarak bot sudah dekat base sehingga membuang waktu yang seharusnya dapat digunakan

3.3 Strategi Greedy yang Dipilih

Strategi greedy yang kami pilih utamanya merupakan **Greedy by Distance.** Hal ini disebabkan terdapatnya batasan waktu permainan. Untuk memaksimalkan penggunaan waktu, kami perlu mencari Diamond yang dapat dijangkau dalam waktu secepat-cepatnya, yaitu yang terdekat. Meskipun begitu, kami juga menggabungkannya dengan **Greedy by Red Diamonds**, yaitu ketika terdapat diamond biru dan diamond merah dengan jarak yang sama, maka diutamakan diamond merah. **Greedy by Base Timing** juga kami terapkan supaya ketika waktu permainan sudah mendekati habis, berapapun diamond yang sedang dibawa bot agar dikirimkan ke base sehingga dapat masuk ke perhitungan score. Selain itu, kami menghindari teleport dengan algoritma **Greedy by Avoiding Teleport** karena object tersebut cenderung mengacaukan perjalanan bot menuju diamond yang sudah menjadi tujuan. Terakhir, kami memanfaatkan object reset diamond button melalui **Greedy by Reset Button** untuk memperbanyak kembali diamond yang kemungkinan besar

sudah menipis setelah diperebutkan bot-bot lain. Perlu dicatat bahwa agar bot tidak terfokus untuk menginjak reset button, diberi batasan yaitu hanya akan mengincar reset button apabila jaraknya kurang dari sama dengan dua. Kami tidak menerapkan Greedy by Tackle maupun Greedy by Defense karena gerakan bot lawan selalu berubah-ubah tiap detiknya dan tidak dapat diprediksi. Maka dari itu, kami membuat pola gerakan bot kami menjadi "zig-zag" sehingga dapat membingungkan lawan yang ingin men-*tackle*.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi dalam Pseudocode

```
Class NanangBoneng extend public BaseBot {
  // class bot yang sudah dibuat
  Private:
    int timer_to_base = 0 // pengukur waktu permainan
    directions = [(1, 0), (0, 1), (-1, 0), (0, -1)] // list arah yang
diizinkan
    goal_position: Optional[Position] = None // penyimpan posisi tujuan
    current_direction = 0 // arah gerak
    sequenceMove = [] // sequence gerakan
  Public:
    // methods..
}
```

```
function distance(x2, x1, y2, y1 : int) -> float {
  // fungsi untuk mengukur jarak dari titik (x1, y1) ke (x2, y2)
      return ((x2 - x1) ** 2 + (y2 - y1) ** 2) ** 0.5
  }
```

```
function red_diamonds(board : Board) -> List[GameObject] {
    // fungsi untuk mendapatkan list diamond merah
      return [d for d in board.game_objects if d.properties.points == 2]
}
```

```
function get_teleporter(board : Board) -> List[GameObject] {
    // fungsi untuk mendapatkan list teleporter
      return [t for t in board.game_objects if t.type ==
"TeleportGameObject"]
}
```

```
function diamond_button_position(board : Board) -> Position {
    // fungsi untuk mendapatkan posisi dari diamond button
      for temp in board.game_objects {
        if (temp.type == "DiamondButtonGameObject") {
          return temp.position
        }
      }
      return None
}
```

```
function closest_diamond(Board : board, GameObject : board_bot) ->
Position {
// fungsi untuk mendapatkan posisi dari diamond terdekat
        if (board_bot.properties.diamonds == 4) {
            // Kalau inventory sudah terisi 4, maka hanya search diamond
biru
            goal_position = board_bot.properties.base
            int temp_distance = MAX_INT

            for (int i = 0; i < board.diamonds.size(); i++) {
                if (board.diamonds[i].properties.points == 1) {
                    int distance_to_diamond = distance(
                        board.diamonds[i].position.x,
                        board_bot.position.x,
                        board.diamonds[i].position.y,
                        board_bot.position.y
                    )
                    Position diamond_position =
board.diamonds[i].position;
                    if (distance_to_diamond < temp_distance) {
                        goal_position = diamond_position
                        temp_distance = distance_to_diamond
                    }
                }
            }
        } else {
            Position diamond_position = board.diamonds[0].position;
            goal_position = diamond_position
            int distance_to_diamond = distance(
                diamond_position.x,
                board_bot.position.x,
                diamond_position.y,
                board_bot.position.y
            )

            for (int i = 1; i < board.diamonds.size(); i++) {
                int distance_to_diamond_i = distance(
                    board.diamonds[i].position.x,
                    board_bot.position.x,
                    board.diamonds[i].position.y,
                    board_bot.position.y
                )

                if (distance_to_diamond_i < distance_to_diamond ||
                    (distance_to_diamond_i == distance_to_diamond &&
                    board.diamonds[i].properties.points >
board.diamonds[i - 1].properties.points)) {
                    diamond_position = board.diamonds[i].position
```

```
                    goal_position = diamond_position
                    distance_to_diamond = distance_to_diamond_i
                }
            }
        }

        return goal_position
    }
```

```
function defense_from_enemy(board : Board, board_bot : GameObject) ->
(int, int){
        // mendapatkan List of Position bot musuh
        bot_enemy_position : Position[]

        for (bot : bot  in board.bots) {
            if (bot.id != board_bot.id) {
                bot_enemy_position.append((bot.position.x,
bot.position.y));
            }
        }

        our_bot : Position = Position(board_bot.position.x,
board_bot.position.y);

        if (Position(board_bot.position.x + 1, board_bot.position.y) in
bot_enemy_position {
        // jika terdapat bot musuh di kanan posisi our_bot
            if (board_bot.position.x == 0) {
                if (board_bot.position.y == 0) {
                    // bergerak ke bawah Jika our_bot berada di pojok
kiri atas
                    return (0, 1);
                } else if (board_bot.position.y == board.height - 1) {
                   // bergerak ke atas jika our_bot berada di pojok kiri
bawah
                    return (0, -1);
                } else {
                    //bergerak ke bawah
                    return (0, 1);
                }
            } else { // jika tidak berada di paling kiri
                // bergerak ke kiri
                return (-1, 0);
            }
        }

        // Algoritma serupa untuk kemungkinan posisi bot lawan di kiri,
atas, dan bawah...
```

```
        return (0, 0);  // return (0,0) jika tidak ada musuh di dekat
our_bot
    }
```

```
function tackle_enemy(board : Board, board_bot : GameObject) -> (int,
int){
        // mendapatkan List of Position bot musuh
        bot_enemy_position : Position[]

        for (bot : bot  in board.bots) {
            if (bot.id != board_bot.id) {
                bot_enemy_position.append((bot.position.x,
bot.position.y))
            }
        }


        Position our_bot = Position{board_bot.position.x,
board_bot.position.y}

        // mendekati musuh sesuai dengan posisinya
        for (Position enemy_pos : bot_enemy_position) {
            if (Position{board_bot.position.x + 1, board_bot.position.y}
== enemy_pos) {
                return (1, 0)
            }

            if (Position{board_bot.position.x - 1, board_bot.position.y}
== enemy_pos) {
                return (-1, 0)
            }

            if (Position{board_bot.position.x, board_bot.position.y + 1}
== enemy_pos) {
                return (0, 1)
            }

            if (Position{board_bot.position.x, board_bot.position.y - 1}
== enemy_pos) {
                return (0, -1)
            }
        }

        return (0, 0) // return (0,0) jika tidak ada ada musuh di dekat
out_bot
    }
```

```
function get_direction_zigzag(current_x, current_y, dest_x, dest_y :
int) -> (int, int) {
        int delta_x = dest_x - current_x;
        int delta_y = dest_y - current_y;
        if (abs(delta_x) > abs(delta_y)) {
            if (delta_x > 0) {
                return (1, 0)
            }
            return (-1, 0)
        } else {
            if (delta_y > 0) {
                return (0, 1)
            }
            return (0, -1)
        }
    }
```

```
Position next_move(Bot board_bot, Board board) {
        BotProperties props = board_bot.properties;
        Position base = board_bot.properties.base;
        Position current_position = board_bot.position;

        // bergerak sesuai dengan sisa gerakan pada atribut sequenceMove
        if (!sequenceMove.empty()) {
            timer_to_base += 1
            return sequenceMove.pop(0) // return anggota pertama dari
sequenceMove dan menghapusnya dari sequenceMove
        }

        # Monitor jarak bot ke base
        distance_to_base = distance(base.x, board_bot.position.x,
base.y, board_bot.position.y)

        // pulang ke base ketika inventory sudah penuh
         if (props.diamonds == 5) {
            self.goal_position = base
         }

        // pulang sesuai dengan timing dari timer.
        else if (timer_to_base >= 45) {
             if (distance_to_base > 5) {
                 goal_position = base // pulang ke base
             } else if (timer_to_base >= 52) {
                  goal_position = base // pulang ke base
             } else {
                 goal_position = closest_diamond(board, board_bot) //
cari diamond terdekat
```

```
            }
        }

        // Injak reset diamond button jika beradap pada jarak 2 langkah
       else if (distance(current_position.x,
diamond_button_position(board).x, current_position.y,
diamond_button_position(board).y)) {
            diamond_button = diamond_button_position(board)
            goal_position = diamond_button
       }

       else {
           goal_position = closest_diamond(board, board_bot)
       }

       if (goal_position) {
       // jika goal_position sudah terdefinisi
           delta_x, delta_y = get_direction_zigzag(current_position.x,
current_position.y, goal_position.x, goal_position.y,) //melakukan
pencarian arah gerak sesuai dengan algoritma zigzag

           if (delta_x == delta_y)  {
           // Jika delta_x dan delta_y sama
                step = random.randint(-1, 1)
                delta_x = step
                if (delta_y == delta_x) {
                    if (delta_x == 0) {
                        delta_y = 1
                    } else {
                        delta_y = delta_x * -1
                   }
               }
          }

          // Sequence gerakan untuk menghindari teleport
         teleporter_position = []
         for (teleport in get_teleporter(board)) {
             teleporter_position.append((teleport.position.x,
teleport.position.y))
         }

         // mendapatkan posisi bot sendiri
         our_bot = (board_bot.position.x, board_bot.position.y)

        if ((current_position.x + delta_x, current_position.y) in
teleporter_position) {
            # ukur jarak y dari current_position ke goal_position
            y_distance_to_goal = self.goal_position.y -
```

```
current_position.y
            if (y_distance_to_goal > 0 or current_position.y == 0):  #
validasi agar tidak menabrak batas atas matrix
                delta_x = 0
                delta_y = 1
            } else {
                delta_x = 0
                delta_y = -1
            }
        }

        if ((current_position.x, current_position.y + delta_y) in
teleporter_position) {
            # ukur jarak x dari current_position ke goal_position
            if (delta_y == 1) {  # gerakan sedang turun
                if (current_position.x == 0) {
                    sequenceMove.append((1, 0))
                    sequenceMove.append((0, 1))
                    sequenceMove.append((0, 1))
                    sequenceMove.append((-1, 0))
                } else {
                    sequenceMove.append((-1, 0))
                    sequenceMove.append((0, 1))
                    sequenceMove.append((0, 1))
                    sequenceMove.append((1, 0))
                }
            } else {  # gerakan sedang naik
                if (current_position.x == 0) {
                    sequenceMove.append((1, 0))
                    sequenceMove.append((0, -1))
                    sequenceMove.append((0, -1))
                    sequenceMove.append((-1, 0))
                } else {
                    sequenceMove.append((-1, 0))
                    sequenceMove.append((0, -1))
                    sequenceMove.append((0, -1))
                    sequenceMove.append((1, 0))
                }
            }
        }

        # Periksa apakah terdapat gerakan di sequence
        if (sequenceMove) {
            delta_x, delta_y = self.sequenceMove.pop(0)
        }
        timer_to_base += 1
        return (delta_x, delta_y)
```

4.2 Struktur Data Program

Secara umum, struktur data program ini menggunakan pendekatan pemrograman berbasis objek yang terdiri dari beberapa kelas dengan atribut dan method yang saling berkaitan. Kelas-kelas ini ditempatkan dalam beberapa file dan folder terpisah. Berikut adalah struktur data program:

1. **Models**

   Pada bagian models, terdapat beberapa kelas yang digunakan untuk merepresentasikan objek-objek yang ada dalam permainan Diamonds. Kelas-kelas ini terdiri dari:

   a. GameObject

   Kelas GameObject berisi objek yang terdapat dalam permainan. Atribut properties pada kelas ini memiliki tipe data Optional[Properties] yang berarti atribut ini dapat memiliki nilai None. Selengkapnya adalah sebagai berikut:

   - id: int
   - position: Position
   - type: str
   - properties: Optional[Properties]

   b. Config

   Kelas Config berisi konfigurasi yang digunakan dalam permainan, yaitu

   - generation_ratio: Optional[float]
   - min_ratio_for_generation: Optional[float]
   - red_ratio: Optional[float]
   - seconds: Optional[int]
   - pairs: Optional[int]
   - inventory_size: Optional[int]
   - can_tackle: Optional[bool]

   c. Feature

   Kelas Feature berisi atribut sebagai berikut

   - name: str
   - config: Optional[Config]

   d. Board

   Kelas Board berisi apa-apa saja yang terkandung dalam papan permainan, termasuk objek dari kelas GameObject. Kelas ini memiliki beberapa atribut dan method, yaitu:

   - id: int
   - width: int
   - height: int
   - features: List[Feature]
   - minimum_delay_between_moves: int

- game_objects: Optional[List[GameObject]]
- bots: List[GameObject]
- diamonds: List[GameObject]
- get_bot(self, bot: Bot) -> Optional[GameObject]
- is_valid_move(self, current_position: Position, delta_x: int, delta_y: int) -> bool

e. Position

Kelas Position berisi koordinat posisi objek dalam permainan, yaitu:

- x: int
- y: int

f. Properties

Kelas Properties berisi atribut yang dimiliki oleh objek dalam permainan, yaitu:

- diamonds: int
- points: int
- base: Position
- inventory_size: int
- can_tackle: bool
- milliseconds_left: int
- time_joined: str
- name: str

2. Util

Pada bagian util, terdapat beberapa fungsi yang digunakan untuk mendukung jalannya permainan seperti mendapat arah untuk sekali gerakan dan memberikan status posisi antara dua objek. Fungsi-fungsi ini adalah:

- clamp
- get_direction
- position_equals

3. Logic

Pada bagian logic, terdapat beberapa kelas yang digunakan untuk mengatur logika permainan. Kelas-kelas ini terdiri dari:

a. BaseLogic

Kelas BaseLogic berisi metode next_move yang digunakan untuk mengatur logika permainan secara umum. Metode ini menerima dua parameter, yaitu board_bot dan board. Kelas ini merupakan kelas dasar yang digunakan sebagai parent class dari kelas-kelas lainnya.

b. NanangBoneng

Kelas NanangBoneng berisi metode next_move yang digunakan untuk mengatur logika permainan secara khusus. Kelas ini merupakan kelas turunan dari kelas BaseLogic yang menjadi kunci inti dalam pergerakan bot kami. Di sini kami mengimplementasikan algoritma greedy sebagai logika berjalannya bot di permainan. Dalam kelas ini, ada beberapa method yang digunakan untuk mengatur logika permainan, yaitu:

- distance
  Menghasilkan jarak antara dua titik.
- red_diamonds
  Mengembalikan list diamond merah dalam board permainan.
- get_teleporter
  Mengembalikan list teleporter dalam board permainan.
- diamond_button_position
  Mengembalikan posisi diamond button dalam board permainan.
- closest_diamond
  Mengembalikan posisi diamond terdekat dalam board permainan dengan memeriksa terlebih dahulu keadaan inventori.
- defense_from_enemy
  Mengembalikan arah gerakan yang valid untuk menghindari bot lawan.
- get_direction_zigzag
  Mengembalikan arah gerakan secara zig-zag menuju posisi tujuan.
- next_move
  Mengembalikan arah gerakan bot selanjutnya berdasarkan semua pendekatan greedy yang diimplementasikan.

4. Main

Terdapat file main.py yang digunakan untuk menjalankan permainan ketika dipanggil oleh executable. Bagian ini berisi program untuk memulai inisiasi bot ke dalam permainan dan menerjemahkan setiap gerakan bot dari implementasi logika yang telah dibuat ke bagian game engine.

4.3 Analisis dan Pengujian

4.3.1 Hasil Pengujian Alternatif Solusi Strategi

Nama yang Diuji	**Implementasi**	**Hasil Pengujian**
Mengambil diamond	Implementasi pertama mencari jarak diamond terdekat dari posisi bot saat ini. Jika dengan jarak yang sama terdapat diamond merah yang mempunyai poin lebih banyak,	**Sudah efektif** dalam menentukan jarak diamond terdekat dan menghampirinya. Ketika dihadapkan dalam situasi dimana terdapat diamond biru dan merah dengan jarak yang sama dari bot, bot berjalan ke posisi diamond merah terlebih dahulu.

	ambil diamond merah dahulu.	
Tackle bot lawan	Implementasi dibuat dengan mendeteksi sel sekitar bot dapat bergerak, yakni sel atas, bawah, kiri, dan kanan. Jika ada bot lawan melintas di sel tersebut, bot sendiri diperintahkan untuk bergerak ke sel serupa dalam rangka men-*tackle* lawan.	Strategi **berjalan** dan **berhasil** mendeteksi keberadaan lawan di sel sekitar bot sendiri. Bot sendiri juga berhasil melakukan gerakan yang menimpa posisi bot lawan. **Namun**, di beberapa kasus, yang terjadi adalah bot sendiri gagal menimpa bot lawan karena bot lawan sudah lebih dahulu melangkah sebelum bot sendiri menimpa, yang berimplikasi pada aksi kejar-kejaran. Hal ini terjadi karena waktu pergerakan kedua bot yang tidak selaras sehingga bot sendiri terus mendeteksi keberadaan bot namun tidak dapat mengejarnya.
Menghindari bot lawan	Sama seperti *tackle* implementasi dilakukan dengan mendeteksi sel sekitar bot dapat bergerak, yakni sel atas, bawah, kiri, dan kanan. Tetapi, jika terdeteksi bot lawan di sel-sel tersebut, bot sendiri akan langsung bergerak menghindar agar tak terkena lintasan pergerakan bot lawan.	Strategi **berjalan dengan baik**, bot sendiri berhasil mendeteksi keberadaan bot lawan dan segera menjauh pada gerakan selanjutnya. **Namun**, pada beberapa kasus, bot sendiri gagal menghindar karena belum sempat mendeteksi keberadaan musuh di sel sekitarnya. Lagi-lagi hal ini disebabkan oleh waktu pergerakan bot yang tidak selaras dalam permainan.
Menghindari teleport	Implementasi ini dibuat dengan mendeteksi keberadaan teleport. Jika posisi teleport sama dengan posisi gerakan bot selanjutnya, maka bot diperintahkan untuk mengitari teleport tersebut. Kenapa tidak hanya menghindar ke sel lain? Karena bot pada *get_direction* bot akan otomatis menyesuaikan kembali posisinya sehingga malah kembali berada di jalur teleport. Oleh karena itu dibuat sebuah sekuens gerakan yang membuat bot akan mengitari teleport. Implementasi ini juga sudah mempertimbangkan posisi dinding permainan sehingga tidak melakukan gerakan yang menabrak dinding.	**Berhasil** berjalan dengan baik. Ketika terdapat teleport di hadapan bot, bot langsung melakukan gerakan mengitari teleport. Ketika teleport terletak di dinding permainan, bot juga melakukan gerakan mengitar namun tidak menabrak dinding.

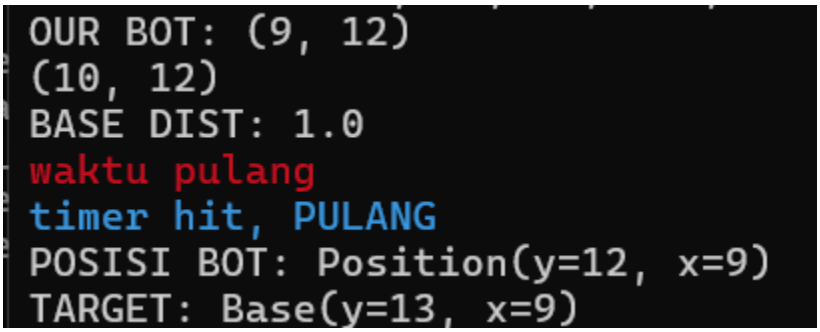
Menekan reset button (diamond button)	Ketika di sekitar bot sudah tidak ada diamond namun terdapat diamond button dalam jarak <= 2 sel dari bot, bot diperintahkan untuk menuju posisi button tersebut agar diamond dapat kembali ter-*regenerate*.	Dapat **berjalan dengan baik**, bot berhasil bergerak menuju posisi button ketika jaraknya dekat dengan posisi bot. Button dapat tertekan sehingga diamond ter-*regenerate* kembali untuk diambil oleh bot. **Namun**, terjadi sedikit kasus dimana diamond yang ter-*regenerate* tidak tersebar di sekitar diamond button.
Timing ke base	Implementasi dibuat dengan memerhatikan waktu berjalannya permainan. Karena delay setiap gerakan adalah 1 detik dan total waktu sesi permainan adalah 60 detik, maka dibuat *timer* untuk memantau waktu. Ketika *timer* sudah mencapai 46 detik, apabila posisi bot sedang jauh dari base, bot akan diperintahkan untuk segera kembali ke base. Bila ternyata posisi bot tidak jauh dari base, maka bot diberikan waktu tambahan untuk terus mencari diamond. Barulah ketika sudah menyentuh *timer* sebesar 52 detik bot akan dipaksa pulang ke base.	Strategi **berhasil** diterapkan. Bot dapat memantau waktu permainan dan pulang ketika *timer* yang ditentukan sudah tercapai guna menyetor diamond di inventori agar tidak sia-sia. Saat bot berada di posisi jauh dari base, bot segera menuju base. Saat posisinya dekat base, bot masih dapat mencari diamond. **Namun** terdapat permasalahan yakni ketika posisi bot berada di pertengahan batas antara jauh atau tidak dari base, bot menjadi bingung apakah perlu pulang atau tetap mencari diamond, sehingga di beberapa kasus terjadi gerakan bot yang tidak diperlukan.

4.3.2 Log Pengujian Alternatif Solusi Strategi

```
BOT Sendiri: GameObject(id=577, position=Position(y=13,
x=1), type='BotGameObject', properties=Properties(points
=None, pair_id=None, diamonds=0, score=1, name='nanangbo
n', inventory_size=5, can_tackle=True, milliseconds_left
=57885, time_joined='2024-03-09T13:45:55.073Z', base=Bas
e(y=14, x=1)))
ENEMY COUNT: 4
BOT ENEMY: [(3, 9), (14, 8), (1, 12)]
OUR BOT: (1, 13)
(2, 13)
ATAS ADA MUSUH
BERGERAK KE BAWAH
```

Pengujian Menghindari Musuh

Pengujian Menghindari Teleporter



Pengunjian *Timing* ke Base

4.3.3 Hasil Pertandingan Dengan Bot Lain

Ronde	Hasil	Keterangan
1		**nanangsad & nananf:** tanpa tackle tapi menggunakan *defense* **nanang22 & nanang4:** tanpa *tackle* dan *defense,* gerak zig-zag

4	Final Score Name / Score nanang22 14 nananf 11 nanang4 11 nanangsad 3	**nanangsad & nananf:** tanpa tackle tapi menggunakan *defense* **nanang22 & nanang4:** tanpa *tackle* dan *defense,* gerak zig-zag
7	Final Score Name / Score nanang4 12 nananf 11 nanang22 9 nanangsad 6	**nanangsad & nananf:** tanpa tackle tapi menggunakan *defense* **nanang22 & nanang4:** tanpa *tackle* dan *defense,* gerak zig-zag
10	Final Score Name / Score nanang22 16 nanang4 15 nananf 10 nanangsad 8	**nanangsad & nananf:** tanpa tackle tapi menggunakan *defense* **nanang22 & nanang4:** tanpa *tackle* dan *defense,* gerak zig-zag
Kesimpulan: dari percobaan tanding hingga 10 ronde, didapat posisi pertama dengan perolehan diamond terbanyak selalu dipegang oleh algoritma yang menerapkan semua alternatif solusi greedy namun tanpa strategi tackle dan defense. Variabel alternatif greedy yang lain juga sebetulnya sudah pernah diujikan, namun tidak ditampilkan karena hasilnya juga seperti ini.		

BAB V
PENUTUP

5.1 Kesimpulan

Algoritma Greedy memang tidak memberikan jaminan kesuksesan sebagaimana Bruteforce. Meskipun begitu, Greedy memiliki kelebihannya tersendiri, yaitu cepat dalam membuat keputusan sehingga mampu mencapai kondisi optimum lokal. Dengan menerapkan Greedy, bot kami berhasil mengambil langkah demi langkah yang optimal sehingga cukup mendekati optimal global. Solusi global akan berbeda jauh dengan kumpulan optimum lokal apabila terdapat kasus ekstrem, misalnya persebaran diamond yang tidak merata, dan lokasi bot lawan yang cenderung berdekatan. Berhubung hal tersebut di luar kendali kami, maka seluruh pertimbangan yang diimplementasikan dalam algoritma sudah merupakan yang terbaik.

5.2 Saran

Saran untuk kelompok kami diantaranya

a. Memperbanyak diskusi untuk menemukan alternatif solusi lain yang belum terpikirkan
b. Memperdalam game engine sehingga dapat menjadikan alternatif solusi baru
c. Memanajemen waktu dengan lebih baik

5.3 Refleksi

Refleksi dari kelompok kami diantaranya

a. Menjaga kesehatan dengan baik sehingga tidak masuk ke IGD selama pengerjaan tugas besar
b. Melakukan pembagian tugas kelompok dengan lebih baik sehingga tugas dapat lebih cepat terselesaikan
c. Mengembangkan kembali efektivitas algoritma Greedy yang telah dibuat

LAMPIRAN

Link repository : https://github.com/TazakiN/Tubes1_Nanang-Boneng/
Link video : https://youtu.be/XOUcY5OiNxc

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/2020-2021/Algoritma-Greedy-(2021)-Bag3.pdf