

Tugas Besar 1 IF2211 Strategi Algoritma
Semester II tahun 2023/2024
Pemanfaatan Algoritma Greedy dalam pembuatan bot permainan
Diamonds



KELOMPOK : 36 - pinjamdulu100v2
Hugo Sabam Augusto Sianturi 13522129
Muhammad Zaki 13522136
Andi Marihot Sitorus 13522138

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI
BANDUNG TAHUN AJARAN 2023/2024

BAB 1

DESKRIPSI MASALAH

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

Pada tugas Strategi Algoritma ini, mahasiswa diminta untuk membuat sebuah bot yang nantinya akan dipertandingkan satu sama lain. Tentunya mahasiswa harus menggunakan strategi greedy dalam membuat bot ini.

Komponen-komponen dari permainan Diamonds antara lain:

1. Diamonds

Untuk memenangkan pertandingan, kita harus mengumpulkan diamond ini sebanyak-banyaknya dengan melewati/melangkahnya. Terdapat 2 jenis diamond yaitu diamond biru dan diamond merah. Diamond merah bernilai 2 poin, sedangkan yang biru bernilai 1 poin. Diamond akan di-regenerate secara berkala dan rasio antara diamond merah dan biru ini akan berubah setiap regeneration.

2. Red 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 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 menempa posisi bot B, bot B akan dikirim ke home base dan semua diamond pada inventory bot B akan hilang, diambil masuk ke inventory bot A (istilahnya tackle). 7. Selain itu, terdapat beberapa fitur tambahan seperti teleporter dan red button yang dapat digunakan apabila anda menuju posisi objek tersebut. 8. Apabila waktu seluruh bot telah berakhir, maka permainan berakhir. Score masing-masing pemain akan ditampilkan pada tabel Final Score di sisi kanan layar.

BAB 2

LANDASAN TEORI

2.1 Algoritma Greedy

Algoritma greedy adalah algoritma yang memecahkan persoalan secara langkah per langkah (step by step) sedemikian sehingga, pada setiap langkah kita mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsip “take what you can get now!”) dan “berharap” bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Elemen-elemen algoritma greedy:

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

Dengan menggunakan elemen-elemen di atas, maka dapat dikatakan bahwa: Algoritma greedy melibatkan pencarian sebuah himpunan bagian, S , dari himpunan kandidat, C ; yang dalam hal ini, S harus memenuhi beberapa kriteria yang ditentukan, yaitu S menyatakan suatu solusi dan S di optimisasi oleh fungsi obyektif.

2.2 Cara Kerja

Program ini terfokus pada bot yang ditujukan untuk mendapatkan diamond yang tersebar di board sebanyak mungkin. Untuk mencapai tujuan tersebut digunakan algoritma greedy

untuk menentukan arah gerak bot dan diharapkan bot akan mendapatkan diamond sebanyak-banyaknya. Untuk menjalankan bot, diperlukan posisi dari objek yang dituju untuk mendapatkan arah gerak selanjutnya sehingga dapat dikembalikan oleh fungsi next move. Untuk itu, harus dibuat prioritas objek yang dituju pada kondisi yang dapat terjadi lalu mengambil posisi objek yang diinginkan pada kondisi tersebut. Untuk itu, setiap langkah bot akan ditentukan dengan menilai objek paling menguntungkan untuk dituju sesuai dengan keinginan pembuat.

BAB 3

Aplikasi Strategi Greedy

3.1 Mapping Persoalan Diamonds menjadi Elemen Greedy

3.1.1 Mapping Persoalan Pengambilan Diamonds Ketika waktu diatas 15 detik

Karena objektif dari game ini adalah mengumpulkan diamonds sebanyak banyaknya maka diperlukan strategi untuk mengumpulkan diamonds seefektif mungkin

Nama Elemen	Definisi Elemen
Himpunan Kandidat	Himpunan state sub persoalan dengan pencarian berbagai jarak diamonds relatif terhadap player
Himpunan Solusi	Jarak diamonds terdekat atau terjauh
Fungsi Solusi	Memeriksa apakah pilihan sudah berupa pilihan state dengan aksi yang layak
Fungsi Seleksi	Pilih state sub persoalan dengan diamonds terbanyak
Fungsi Kelayakan	Memeriksa apakah diamonds yang ingin dicapai sesuai dengan kondisi sehingga layak untuk diambil
Fungsi Objektif	Meraup diamond sebanyak-banyaknya dengan jarak terdekat

3.1.2 Mapping Persoalan Pengambilan Diamonds ketika waktu dibawah 15 detik

Nama Elemen	Definisi Elemen
Himpunan Kandidat	Himpunan aksi kembali ke <i>base</i> atau mencari diamond terdekat dengan poin terbesar
Himpunan Solusi	Kembali ke base atau tetap mencari diamond terdekat dengan poin terbesar
Fungsi Solusi	Memeriksa apakah layak untuk kembali ke <i>base</i> (memiliki diamond yang cukup) atau tetap mencari diamond terdekat dengan poin terbesar
Fungsi Seleksi	Memilih aksi balik ke <i>base</i> apabila memiliki <i>diamond</i> lebih besar daripada nol / tidak kosong.
Fungsi Kelayakan	Memeriksa apakah bot memiliki memiliki diamond sehingga layak untuk kembali ke <i>base</i>
Fungsi Objektif	Bot kembali ke base dengan diamond yang ada.

3.1.3 Mapping Persoalan Teleportasi

Nama Elemen	Definisi Elemen
-------------	-----------------

Himpunan Kandidat	Himpunan aksi melakukan teleportasi atau tidak.
Himpunan Solusi	Pilihan untuk melakukan teleportasi atau tidak.
Fungsi Solusi	Memeriksa apakah solusi berupa pilihan untuk melakukan teleportasi atau tidak melakukan teleportasi.
Fungsi Seleksi	Pilih state sub persoalan dengan waktu tersingkat
Fungsi Kelayakan	Mengecek apakah layak untuk melakukan teleportasi atau tidak(menghitung jarak terpendek baik ke diamond maupun ke base)
Fungsi Objektif	Menggunakan teleporter agar bot mendapat jalan paling efisien

3.1.4 Mapping Persoalan bot player

Nama Elemen	Definisi Elemen
Himpunan Kandidat	Himpunan aksi mengejar player lain atau tidak
Himpunan Solusi	Pilihan untuk melakukan pengejaran atau tidak.

Fungsi Solusi	Memeriksa apakah solusi berupa pilihan untuk melakukan pengejaran atau tidak.
Fungsi Seleksi	Pilih state sub persoalan dengan pengaruh pendapatan diamonds bot.
Fungsi Kelayakan	Cek apakah aksi pada state yang dipilih layak (bot yang dikejar memiliki diamond banyak dan dekat dengan bot_player).
Fungsi Objektif	Men-tackle bot lawan dengan diamond banyak dan dekat

3.1.5 Mapping Persoalan Diamond reset button

Nama Elemen	Definisi Elemen
Himpunan Kandidat	Himpunan aksi menuju reset button atau tidak
Himpunan Solusi	Pilihan untuk menuju reset button atau tidak.
Fungsi Solusi	Memeriksa apakah posisi bot valid untuk melakukan pilihan menuju reset button.
Fungsi Seleksi	Memilih aksi bergerak menuju reset button apabila posisi bot dalam radius 1 balok
Fungsi Kelayakan	Jika ketika bot bergerak, terdapat button reset diamond di sekitar 1 radius balok dari bot
Fungsi Objektif	Memencet tombol reset diamond

3.2 Eksplorasi Alternatif Solusi serta Analisis Efisiensi dan Efektivitas

3.2.1 Eksplorasi Alternatif Solusi serta Analisis Efisiensi dan Efektivitas untuk Persoalan pengambilan diamonds ketika waktu lebih dari 15 detik

Alternatif Solusi	Analisis Efisiensi dan Efektivitas
Memilih State dengan diamonds terdekat dengan player	Efektif karena kita hanya mengambil yang terdekat dan tidak terlalu membuang banyak waktu
Memilih state dengan diamonds terjauh terlebih dahulu dengan player	Tidak efektif dikarenakan bisa membuang waktu untuk mencapai diamonds terjauh terlebih dahulu
Memilih state dengan wilayah diamonds terbanyak	Tidak efektif dikarenakan diamonds yang ada di boards itu dinamis jadi akan ada kemungkinan bot akan bolak balik karena kondisi yang selalu berubah
Memilih State dengan diamonds terdekat dengan player dengan prioritas red diamonds	Efektif dan juga efisien dikarenakan kita memprioritaskan diamonds merah yang punya points lebih banyak dan juga dekat

3.2.2 Eksplorasi Alternatif Solusi serta Analisis Efisiensi dan Efektivitas untuk Persoalan pengambilan diamonds ketika waktu kurang dari 15 detik

Alternatif Solusi	Analisis Efisiensi dan Efektivitas
Langsung kembali ke base	Kurang efektif dikarenakan jika langsung kembali ke base maka akan ada diamonds yang terlewat
Mengambil diamonds terdekat dengan base	Efektif karena jika ada diamonds yang dekat dengan base maka akan terambil dan akan menambah point

3.2.3 Eksplorasi Alternatif Solusi serta Analisis Efisiensi dan Efektivitas Teleportasi untuk kembali ke base

Alternatif Solusi	Analisis Efisiensi dan Efektivitas
Memakai teleportasi untuk kembali ke base	Efektif dikarenakan akan memangkas waktu perjalanan, dan bisa menambah waktu untuk mengambil diamonds lainnya
Tidak memakai teleportasi untuk ke base	Tidak efektif karena akan membuang buang waktu dan juga kesempatan untuk mendapatkan lebih banyak diamond
Memakai teleportasi untuk kembali ke base hanya jika lebih dekat daripada jalan	Efektif dan juga efisien karena kita hanya akan mengambil portal apabila waktu tempuh nya lebih sedikit daripada jalan seperti biasa.

3.2.4 Eksplorasi Alternatif Solusi serta Analisis Efisiensi dan Efektivitas pengambilan Diamond Reset Button

Alternatif Solusi	Analisis Efisiensi dan Efektivitas
Mengincar reset button	Tidak efektif dikarenakan jika kita mengincar reset button tanpa mempedulikan apapun akan membuang kesempatan untuk mendapatkan point di diamonds
Tidak mengincar reset button	Efektif dikarenakan kita hanya mengincar diamonds, karena itu yang dibutuhkan untuk memenangkan permainan
Hanya menekan reset button dalam radius 1	Efektif dan efisien dikarenakan bisa memperbesar kemungkinan kita untuk mendapatkan diamonds yang lebih menguntungkan untuk kita

3.2.5 Eksplorasi Alternatif Solusi serta Analisis Efisiensi dan Efektivitas Konfrontasi dengan Bot Lain

Alternatif Solusi	Analisis Efisiensi dan Efektivitas
Mengincar bot dengan diamond terbanyak	Tidak efektif karena sulitnya melakukan pengejaran terhadap bot lain, selain itu bot dengan diamond terbanyak juga bisa

	berganti-ganti
Mengincar Bot terdekat	Tidak efektif dan tidak efisien karena bot terdekat bisa tidak memiliki diamond dan hanya menyia-nyiakan waktu
Tidak mengincar bot lain	Efektif dan efisien dikarenakan bisa membuat bot kita fokus untuk mendapatkan diamonds
Menghindari serangan bot lain	Tidak efektif karena dapat menimbulkan kejar-kejaran yang membuang-buang waktu

3.3 Strategi Greedy yang Dipilih dan Pertimbangan Pemilihan

3.3.1 Strategi Greedy yang Dipilih untuk Persoalan pengambilan diamonds ketika waktu lebih dari 15 detik

Strategi Greedy yang dipilih untuk sub persoalan pengambilan diamonds ketika waktu lebih dari 15 detik adalah memilih State dengan diamonds terdekat dengan player dengan prioritas red diamonds. Hal ini dipilih karena melalui percobaan yang kita lakukan dengan memilih state tersebut kita bisa mendapatkan lebih banyak diamonds dibandingkan dengan hanya mengambil yang terdekat tanpa ada prioritas diamonds yang diambil

3.3.2 Strategi Greedy yang Dipilih untuk Persoalan Pengambilan Diamonds ketika waktu dibawah 15 detik

3.3.3 Strategi Greedy yang Dipilih untuk Persoalan pengambilan Teleportasi untuk kembali ke base

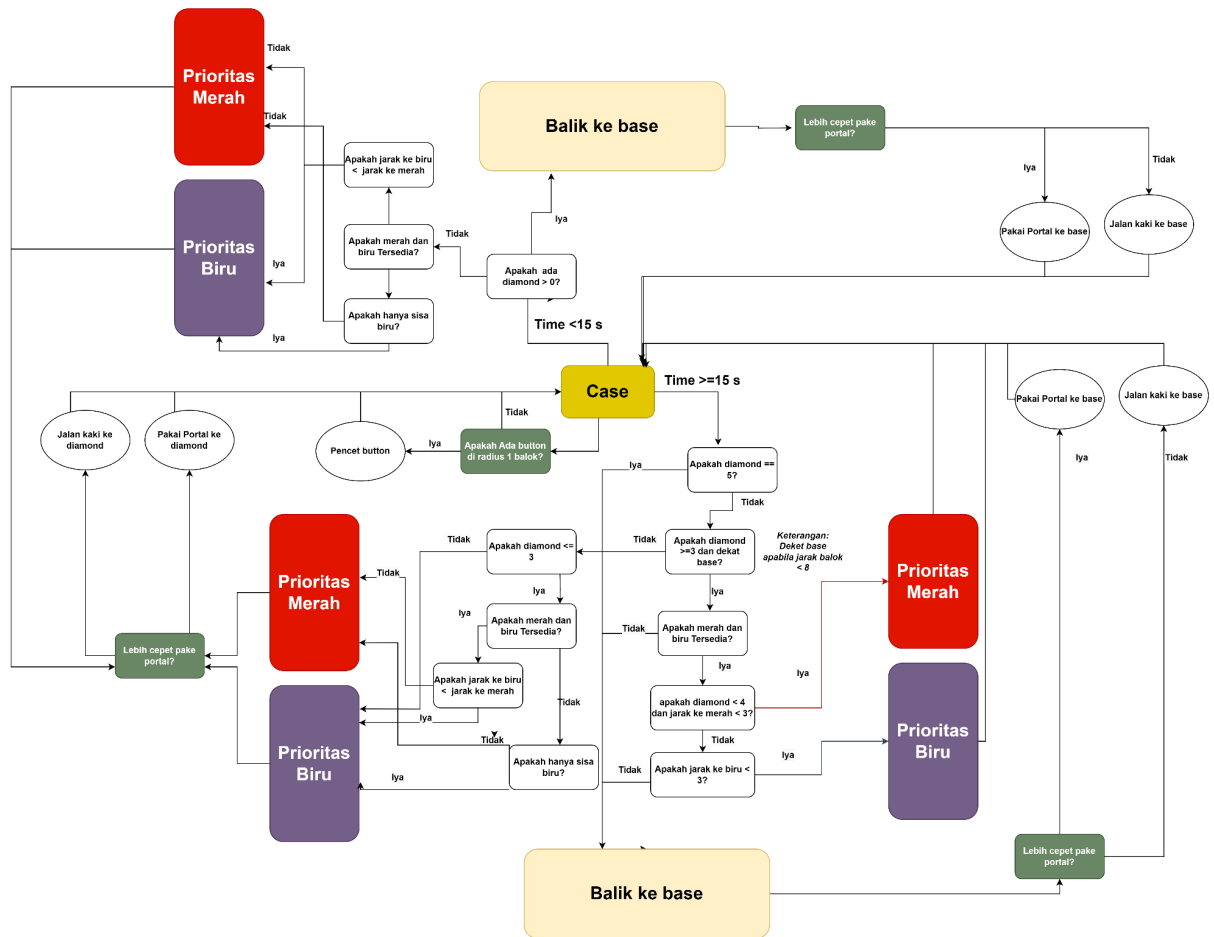
Strategy Greedy yang dipilih untuk sub persoalan pengambilan teleportasi adalah memakai teleportasi untuk kembali ke base hanya jika lebih dekat daripada jalan, dikarenakan akan lebih banyak menghemat waktu dibandingkan jika kita hanya mengincar teleport tanpa ada pertimbangan ataupun hanya berjalan kaki yang mengakibatkan kita mempunyai lebih banyak waktu untuk mengambil diamonds.

3.3.4 Strategi Greedy yang Dipilih untuk Persoalan pengambilan Diamond Reset Button

Strategy greedy yang dipilih untuk persoalan pengambilan diamond reset button adalah hanya menekan apabila dalam radius 1, hal ini kita pilih agar mengefisienkan waktu daripada kita memprioritaskan untuk mengambil diamond reset button, pengambilan diamond reset button juga dapat meningkatkan kesempatan mendapat diamonds yang lebih dekat dengan base kita.

BAB 4

IMPLEMENTASI DAN PENGUJIAN



Gambar 4.1 : Pohon Keputusan Bot

4.1 Pseudocode Implementasi Algoritma Greedy pada Program Bot

4.1.1 Pseudocode pengambilan diamonds ketika waktu lebih dari 15 detik

```

if board_bot.properties.diamonds = 5 then
    if self.isHomeWithPortal(board, board_bot) and usPos != tPos1 then
        (delta_x, delta_y) ← self.goTo(usPos, tPos1)
    else
        (delta_x, delta_y) ← self.goTo(usPos, base)
elif board_bot.properties.diamonds >= 3 and self.isCloseBase(usPos,
base) then
    if self.isRedAvailable(board) and self.isDiamondAvailable(board)
then
        if self.countDistance(usPos, redPos) < 3 and
board_bot.properties.diamonds < 4 then
            (delta_x, delta_y) ← self.goTo(usPos, redPos)
        elif self.countDistance(usPos, bluePos) < 3 then
            (delta_x, delta_y) ← self.goTo(usPos, bluePos)
        else
            if self.isHomeWithPortal(board, board_bot) and usPos !=
tPos1 then
                (delta_x, delta_y) ← self.goTo(usPos, tPos1)
            else
                (delta_x, delta_y) ← self.goTo(usPos, base)
        else
            if self.isHomeWithPortal(board, board_bot) and usPos != tPos1
then
                (delta_x, delta_y) ← self.goTo(usPos, tPos1)
            else
                (delta_x, delta_y) ← self.goTo(usPos, base)
    elif board_bot.properties.diamonds <= 3 then
        If self.isDiamondAvailable(board) and self.isRedAvailable(board)
then
            if self.isBetterPortalDiamond(usPos, board, board_bot) and
usPos != tPos1 then
                (delta_x, delta_y) ← self.goTo(usPos, tPos1)

```



```

        elif self.countDistance(usPos, bluePos) <
self.countDistance(usPos, redPos) then
            (delta_x, delta_y) ← self.goTo(usPos, bluePos)
        else
            (delta_x, delta_y) ← self.goTo(usPos, redPos)
        elif self.isDiamondAvailable(board) and self.isRedAvailable(board)
== False then
            (delta_x, delta_y) ← self.goTo(usPos, bluePos)
        else
            (delta_x, delta_y) ← self.goTo(usPos, redPos)
else
        if self.isDiamondAvailable(board) then
            (delta_x, delta_y) ← self.goTo(usPos, bluePos)
        else
            if self.isHomeWithPortal(board, board_bot) and usPos != tPos1
then
                (delta_x, delta_y) ← self.goTo(usPos, tPos1)
            else
                (delta_x, delta_y) ← self.goTo(usPos, base)

```

4.1.2 Pseudocode pengambilan diamonds ketika waktu kurang dari 15 detik

```

if board_bot.properties.diamonds > 0 then
    if self.isHomeWithPortal(board, board_bot) and usPos != tPos1 then
        (delta_x, delta_y) ← self.goTo(usPos, tPos1)
    else
        (delta_x, delta_y) ← self.goTo(usPos, base)

```

```

else
    if self.isDiamondAvailable(board) and self.isRedAvailable(board)
then
    if self.countDistance(usPos, bluePos) <
self.countDistance(usPos, redPos) then
        (delta_x, delta_y) ← self.goTo(usPos, bluePos)
    else
        (delta_x, delta_y) ← self.goTo(usPos, redPos)
    elif self.isDiamondAvailable(board) and self.isRedAvailable(board)
== False then
        (delta_x, delta_y) ← self.goTo(usPos, bluePos)
    else
        (delta_x, delta_y) ← self.goTo(usPos, redPos)

```

4.1.3 Pseudocode Pengambilan Teleportasi untuk kembali ke base

```

Function isHomeWithPortal(self, board: Board , me : GameObject):
    base = me.properties.base
    usPos = me.position
    temp = []
    for i in range(len(board.game_objects)):
        if(board.game_objects[i].type == "TeleportGameObject")
then:
        temp.append(board.game_objects[i])
        if(self.countDistance(me.position, temp[0].position) >
self.countDistance(me.position, temp[1].position)) then:
            tPos1 = temp[1].position
            tGo1 = temp[1]

```

```

        tPos2 = temp[0].position
        tGo2 = temp[0]
    else:
        tPos1 = temp[0].position
        tGo1 = temp[0]
        tPos2 = temp[1].position
        tGo2 = temp[1]

    #tPos1 itu selalu posisi teleporter yang paling dekat sama
    botplayer

    j_us_to_tPos1 = self.countDistance(usPos,tPos1)
    j_tPos2_to_base = self.countDistance(tPos2,base)
    j_us_to_base = self.countDistance(usPos,base)

    if(j_us_to_tPos1 + j_tPos2_to_base < j_us_to_base):
        return True
    else:
        return False

```

4.1.4 Strategi Greedy untuk Pengambilan Diamond Reset Button

```

Function checkDiamondReset(self,mePosition:Position,board:Board):
    temp = []
    for i in range(len(board.game_objects)):
        if(board.game_objects[i].type ==
"DiamondButtonGameObject") then:
            temp.append(board.game_objects[i])
        if(mePosition.x + 1 == temp[0].position.x and mePosition.y ==

```

```

temp[0].position.y) then:
    return temp[0].position
    elif(mePosition.x - 1 == temp[0].position.x and mePosition.y
== temp[0].position.y) then:
    return temp[0].position
    elif(mePosition.x == temp[0].position.x and mePosition.y + 1
== temp[0].position.y) then:
    return temp[0].position
    elif(mePosition.x == temp[0].position.x and mePosition.y - 1
== temp[0].position.y) then:
    return temp[0].position
    else:
        return mePosition

```

4.2 Struktur Data yang Digunakan pada Program Bot

4.2.1 Struktur data pada *Class* Mybot

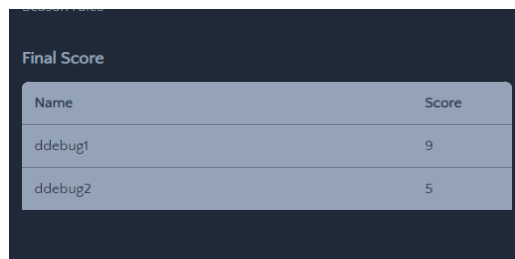
Atribut	Penjelasan
List directions	List of tuple dari integer yang berfungsi untuk memberikan arah (delta_x dan delta_y)
Metode	Penjelasan
def isDiamondAvailable(self,board: Board) -> boolean	Metode ini mengembalikan boolean True apabila dalam board ada diamond biru(poin = 1)

def next_move(self, board_bot: GameObject, board: Board) -> directions	Metode ini digunakan untuk menentukan akan ke arah mana bot kita berjalan dengan mempertimbangkan berbagai kondisi.
def isRedAvailable(self, board: Board) -> boolean	Metode ini mengembalikan boolean True apabila dalam board ada diamond merah (poin = 2)
def goTo(self, a : Position, b: Position) -> directions	Mengembalikan tuple directions dari kedua posisi, digunakan untuk method get_directions
def checkSurrounding(self, nextPosition : Position, mePosition: Position, board: Board) -> string	Metode ini digunakan untuk check disekitar kita apakah ada reset button atau tidak.
def isBetterPortalDiamond(self, usPos : Position, board: Board, me : GameObject) -> boolean	Metode ini digunakan untuk mengecek apakah lebih baik menggunakan portal atau tidak untuk mengambil diamonds
def isHomeWithPortal(self, board: Board, me : GameObject) -> boolean	Metode ini digunakan untuk mengecek apakah bila menggunakan portal untuk pulang lebih efisien dibandingkan dengan berjalan
def countDistance(self, a: Position, b: Position) -> integer	Metode ini mengembalikan nilai jarak posisi a ke posisi b, dalam bentuk $\Delta X + \Delta Y$
def getClosestTeleportPos(self, board: Board, me : GameObject) -> Position	Metode ini digunakan untuk melihat portal manakah yang terdekat dengan kita


def getClosestDiamondPos(self,board:Board, me : GameObject) -> Position	Metode ini mengembalikan posisi <i>Game Object</i> diamond biru yang terdekat dengan bot.
def getClosestRedPos(self,board:Board, me : GameObject) -> Position	Metode ini mengembalikan posisi <i>Game object</i> diamond merah yang terdekat dengan bot.
def checkDiamondReset(self,mePosition:Position,board:Board)	Metode ini digunakan untuk mencari tau apakah ada diamond reset button pada radius 1 blok terhadap player.

4.3 Analisis Desain Solusi pada Setiap Pengujian

4.3.1 Analisis Desain Solusi pengambilan diamonds ketika waktu lebih dari 15 detik



Name	Score
ddebug1	9
ddebug2	5



Name	Score
ddebug1	7
ddebug2	5

Pengujian	Prioritas Jenis Diamond (Merah > biru)(diamond)	Tanpa prioritas (diamonds)
1	9	5
2	6	5
3	10	8
4	7	5
5	9	8

Berdasarkan uji coba di atas., Saya membandingkan kedua faktor dalam kondisi ketika waktu lebih besar dari 15 detik, yaitu Pertama diuji menggunakan sistem prioritas pengambilan diamond merah apabila memiliki jarak lebih dekat dibanding diamond biru. Kedua tanpa prioritas, jadi dicari diamond jenis apapun dengan jarak terdekatnya. Dari data di atas, bisa disimpulkan bahwa dengan sistem prioritas menghasilkan diamond yang cenderung lebih banyak daripada tanpa prioritas. Hal ini membuktikan bahwa solusi ini dapat diambil dan diimplementasikan di bot kami.

4.3.2 Analisis Desain Solusi pengambilan diamonds ketika waktu kurang dari 15 detik

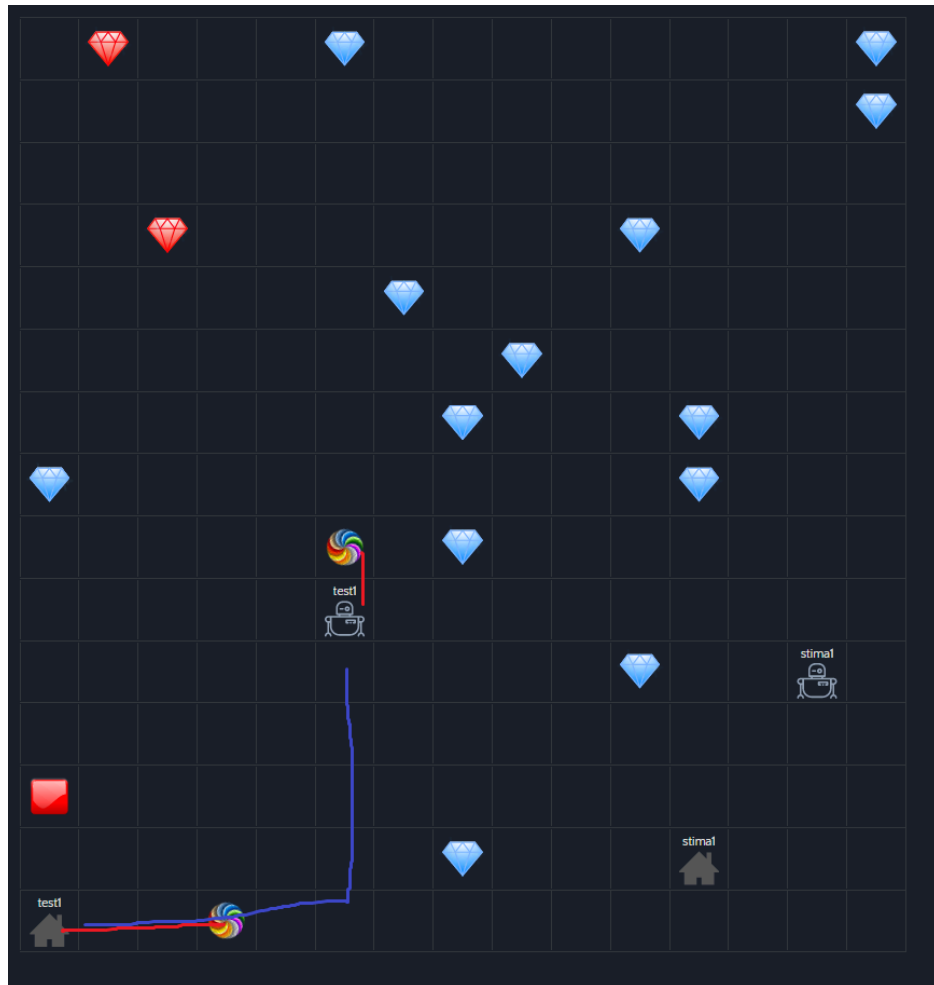
Final Score	
Name	Score
ddebug1	11
ddebug2	8

Final Score	
Name	Score
ddebug1	10
ddebug2	8

Pengujian	Prioritas terdekat (diamond)	Tanpa prioritas (diamonds)
1	12	10
2	13	15
3	11	8
4	12	9
5	10	8

Dari pengujian yang telah kami lakukan dapat dibuktikan bahwa strategi greedy dengan prioritas mengambil diamonds terdekat dan juga jika sudah mendapat lebih besar daripada satu diamonds langsung kembali ke base. Hal ini terjadi karena jika tanpa ada prioritas seperti itu bisa saja kita tidak punya waktu kembali untuk mengembalikan waktu ke base dan itu bisa merugikan karena diamonds yang sudah diambil tidak dihitung ke papan skor. Bisa dilihat dari data tabel diatas bahwa rata rata jika kami mengambil prioritas terdekat bisa mengambil lebih banyak diamond di akhir. Dari 5 pengujian yang kami lakukan rata rata yang didapat oleh bot yang memprioritaskan terdekat yaitu sekitar 11,4 diamond sedangkan yang tanpa prioritas sebanyak 10,2 diamond, lebih banyak sekitar 1,2 diamond atau 1 diamond

4.3.3 Analisis Desain Solusi pengambilan teleportasi untuk kembali ke base



Kita mengambil teleportasi dikarenakan seperti pada pengujian diatas jika kita mengambil rute dengan teleport hanya membutuhkan 4 langkah saja sedangkan jika kita hanya mengambil rute dengan langkah kaki bisa mengakibatkan *infinite loop* dan jika bot menghindari teleport tetap saja tidak seefektif menggunakan teleportasi. Dengan kita menghemat jumlah langkah kita bisa mengkonversikan waktu tersebut untuk mencari diamond lainnya yang bisa menjadi salah satu faktor yang membuat bot ini memenangkan pertandingan.

4.3.4 Analisis Desain Solusi pengambilan diamond reset button

Pengujian diamond reset button menurut kami apabila dilakukan metode pengujian dengan cara yang di atas tidaklah akurat. Karena diamond yang dihasilkan reset button ini memiliki hasil yang terlalu acak, tergantung dengan posisi diamond yang dihasilkan

BAB 5

Kesimpulan

Berdasarkan penjabaran di atas, dapat disimpulkan bahwa program bot untuk menyelesaikan persoalan permainan Etimo Diamonds kami sudah cukup efektif dalam mencapai tujuan optimasi strategy greedy yaitu memaksimalkan skor pada pertandingan. Dalam menyelesaikan persoalan ini, kami melakukan dekomposisi persoalan Etimo Diamonds menjadi beberapa sub persoalan. Diantaranya sub persoalan mengambil diamond ketika waktu di atas 15 detik, mengambil diamonds ketika waktu di bawah 15 detik, decision pengambilan portal atau teleport untuk kembali ke base, menekan tombol reset diamond.

Untuk sub persoalan mengambil diamond ketika waktu di atas 15 detik, kami memilih State dengan diamonds terdekat dengan player dengan prioritas red diamonds karena efektif dan juga efisien dikarenakan kita memprioritaskan diamonds merah yang punya points lebih banyak dan juga dekat, sedangkan untuk sub persoalan mengambil diamond ketika waktu di bawah 15 detik kami memilih mengambil diamonds terdekat dengan base karena efektif jika ada diamonds yang dekat dengan base maka akan terambil dan akan menambah point. Untuk sub persoalan pengambilan portal atau teleport kembali ke base kami memilih memakai teleportasi untuk kembali ke base hanya jika lebih dekat daripada jalan karena efektif dan juga efisien karena kita hanya akan mengambil portal apabila waktu tempuh nya lebih sedikit daripada jalan seperti biasa. Lalu untuk menekan tombol reset diamond kami memilih hanya menekan reset button dalam radius 1 dikarenakan bisa memperbesar kemungkinan kita untuk mendapatkan diamonds yang lebih menguntungkan untuk kita.

DAFTAR PUSTAKA

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Tubes1-Stima-2024.pdf>

<https://github.com/haziqam/tubes1-IF2211-bot-starter-pack>

<https://github.com/haziqam/tubes1-IF2211-game-engine>

LAMPIRAN

Link Repository :

https://github.com/mzaki9/Tubes1_pinjamdulu100v2

Link Video :

https://youtu.be/iSh_AtCPGS8

Link Foto Decision Tree :

https://drive.google.com/file/d/1-HRrGdn0qq618DJHjX27_jOjL4GeTSIg/view?usp=sharing