

Laporan Tugas Besar I

IF2211 Strategi Algoritma

Pemanfaatan Algoritma *Greedy* dalam Aplikasi Permainan “Overdrive”



oleh :

- | | |
|----------|---------------------------|
| 13520093 | Vincent Ho |
| 13520136 | Vincent Christian Siregar |
| 13520165 | Ghazian Tsabit Alkamil |

Sekolah Teknik Elektro dan Informatika
Program Studi Teknik Informatika
Institut Teknologi Bandung
Tahun Ajaran 2021/2022

Daftar Isi

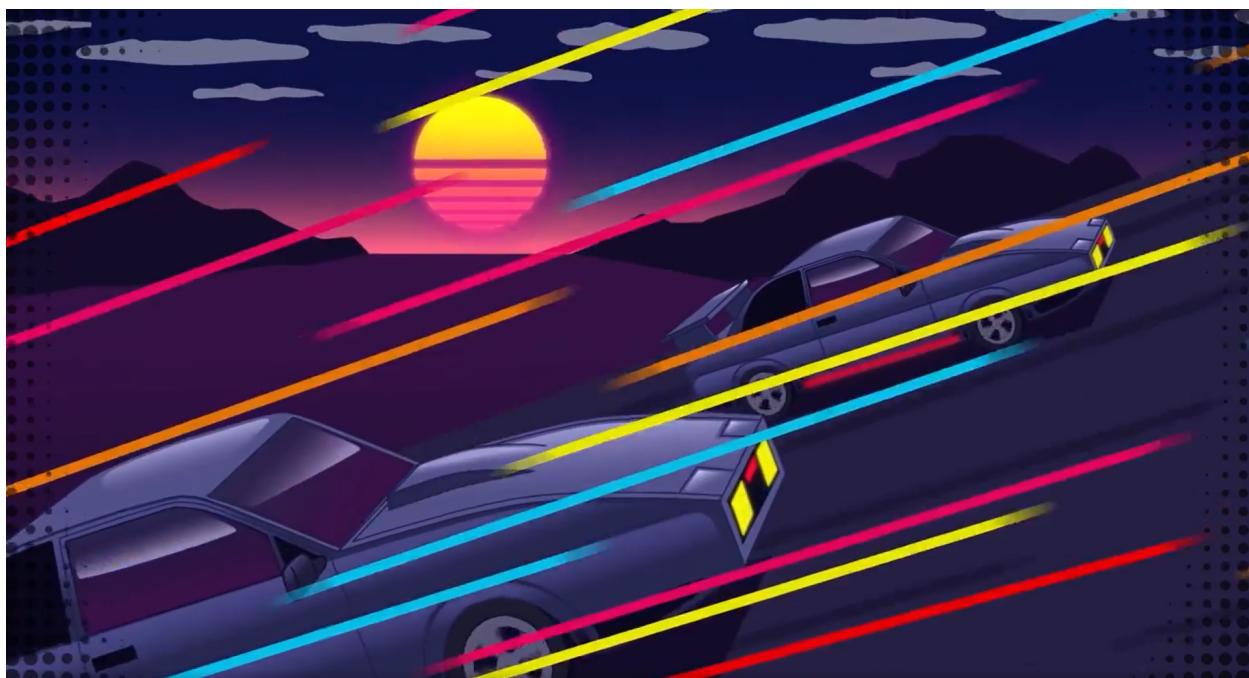
BAB I	3
Pendahuluan	3
1. 1. Deskripsi Tugas	3
BAB II	6
Landasan Teori	6
2.1. Dasar Teori	6
2.2. Pengaturan Game Engine	7
2.3. Menambahkan Strategi Greedy pada Bot	8
2.4. Menjalankan Game Engine	9
BAB III	11
Aplikasi Strategi Greedy	11
3.1. Mapping Persoalan	11
3.2. Eksplorasi Alternatif Solusi Greedy	12
BAB IV	14
Implementasi dan Pengujian	14
4.1. Implementasi Program dalam Game Engine	14
4.2. Penjelasan Struktur Data	16
4.3. Analisis Desain Solusi	17
BAB V	20
Kesimpulan dan Saran	20
5.1. Kesimpulan	20
5.2. Saran	20
Daftar Pustaka	21

BAB I

Pendahuluan

1. 1. Deskripsi Tugas

Overdrive adalah sebuah game yang mempertandingan 2 bot mobil dalam sebuah ajang balapan. Setiap pemain akan memiliki sebuah bot mobil dan masing-masing bot akan saling bertanding untuk mencapai garis finish dan memenangkan pertandingan. Agar dapat memenangkan pertandingan, setiap pemain harus mengimplementasikan strategi tertentu untuk dapat mengalahkan lawannya.



Gambar 1.1. Ilustrasi Permainan Overdrive

Pada tugas besar pertama Strategi Algoritma ini, gunakanlah sebuah game engine yang mengimplementasikan permainan Overdrive. Game engine dapat diperoleh pada laman berikut:

<https://github.com/EntelectChallenge/2020-Overdrive>.

Tugas mahasiswa adalah mengimplementasikan bot mobil dalam permainan Overdrive dengan menggunakan strategi greedy untuk memenangkan permainan. Untuk mengimplementasikan bot tersebut, mahasiswa disarankan melanjutkan program yang terdapat pada starter-bots di dalam starter-pack pada laman berikut ini:

<https://github.com/EntelectChallenge/2020-Overdrive/releases/tag/2020.3.4>

Spesifikasi permainan yang digunakan pada tugas besar ini disesuaikan dengan spesifikasi yang disediakan oleh game engine Overdrive pada tautan di atas. Beberapa aturan umum adalah sebagai berikut.

1. Peta permainan memiliki bentuk array 2 dimensi yang memiliki 4 jalur lurus. Setiap jalur dibentuk oleh block yang saling berurutan, panjang peta terdiri atas 1500 block. Terdapat 5 tipe block, yaitu Empty, Mud, Oil Spill, Flimsy Wall, dan Finish Line yang masing-masing karakteristik dan efek berbeda. Block dapat memuat powerups yang bisa diambil oleh mobil yang melewati block tersebut.
2. Beberapa powerups yang tersedia adalah:
 - a. Oil item, dapat menumpahkan oli di bawah mobil anda berada.
 - b. Boost, dapat mempercepat kecepatan mobil anda secara drastis.
 - c. Lizard, berguna untuk menghindari lizard yang mengganggu jalan mobil anda.
 - d. Tweet, dapat menjatuhkan truk di block spesifik yang anda inginkan.
 - e. EMP, dapat menembakkan EMP ke depan jalur dari mobil anda dan membuat mobil musuh (jika sedang dalam 1 lane yang sama) akan terus berada di lane yang sama sampai akhir pertandingan. Kecepatan mobil musuh juga dikurangi 3.
3. Bot mobil akan memiliki kecepatan awal sebesar 5 dan akan maju sebanyak 5 block untuk setiap round. Game state akan memberikan jarak pandang hingga 20 block di depan dan 5 block di belakang bot sehingga setiap bot dapat mengetahui kondisi peta permainan pada jarak pandang tersebut.
4. Terdapat command yang memungkinkan bot mobil untuk mengubah jalur, mempercepat, memperlambat, serta menggunakan powerups. Pada setiap round, masing-masing pemain dapat memberikan satu buah command untuk mobil mereka. Berikut jenis-jenis command yang ada pada permainan:
 - a. NOTHING
 - b. ACCELERATE
 - c. DECELERATE
 - d. TURN_LEFT
 - e. TURN_RIGHT
 - f. USE_BOOST
 - g. USE_OIL
 - h. USE_LIZARD
 - i. USE_TWEET
 - j. USE_EMP
 - k. FIX

5. Command dari kedua pemain akan dieksekusi secara bersamaan (bukan sekuensial) dan akan divalidasi terlebih dahulu. Jika command tidak valid, bot mobil tidak akan melakukan apa-apa dan akan mendapatkan pengurangan skor.
6. Bot pemain yang pertama kali mencapai garis finish akan memenangkan pertandingan. Jika kedua bot mencapai garis finish secara bersamaan, bot yang akan memenangkan pertandingan adalah yang memiliki kecepatan tercepat, dan jika kecepatannya sama, bot yang memenangkan pertandingan adalah yang memiliki skor terbesar.

Adapun peraturan yang lebih lengkap dari permainan Overdrive, dapat dilihat pada laman :

<https://github.com/EntelectChallenge/2020-Overdrive/blob/develop/game-engine/game-rules.md>

Hasil akhir yang diharapkan adalah terbentuknya sebuah *bot* untuk permainan *Overdrive* yang menggunakan strategi algoritma *greedy* dalam pemrogramannya untuk memenangkan permainan.

BAB II

Landasan Teori

2.1. Dasar Teori

Algoritma *greedy* merupakan metode yang populer dan sederhana untuk memecahkan persoalan optimasi. Algoritma ini memecahkan persoalan secara langkah per langkah (*step by step*) sedemikian sehingga, pada setiap langkah dilakukan pengambilan pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensinya ke depan (prinsip “take what you can get now!”) dan “berharap” bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global. Optimum lokal ini dapat ditentukan dari banyak faktor tergantung persoalan, seperti contohnya selalu mengambil pilihan dengan nilai variabel tertentu yang terbesar.

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 objektif : 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 dioptimasi oleh fungsi objektif.

Skema Umum Algoritma Greedy adalah sebagai berikut:

```
function greedy(C : himpunan_kandidat ) → himpunan_solusi
{Mengembalikan solusi dari persoalan optimasi dengan algoritma
greedy}
Deklarasi
x : kandidat
S : himpunan_solusi
Algoritma:
    S ← {} {inisialisasi S dengan kosong}
    while (not SOLUSI(S)) and (C != {}) do
        x ← SELEKSI(C) {pilih sebuah kandidat dari C}
```

```
C ← C - {x} {buang x dari C karena sudah dipilih}
if LAYAK(S ∪ {x}) then x {memenuhi kelayakan untuk
dimasukkan ke dalam himpunan solusi}
    S ← S ∪ {x} {masukkan x ke dalam himpunan solusi}
endif
endwhile
{SOLUSI(S) or C = {} }
if SOLUSI(S) then {solusi sudah lengkap}
    return S
else
    writeln('tidak ada solusi')
endif
```

- Pada akhir setiap iterasi, solusi yang terbentuk adalah optimum lokal
- Pada akhir **while-do** diperoleh optimum global (jika ada)

Beberapa catatan untuk algoritma *greedy* :

1. Optimum global belum tentu merupakan solusi optimum (terbaik), bisa jadi merupakan solusi *sub-optimum* atau *pseudo-optimum*.
2. Alasan :
 - a. Algoritma *greedy* tidak beroperasi secara menyeluruh terhadap semua kemungkinan solusi yang ada (sebagaimana pada metode exhaustive search).
 - b. Terdapat beberapa fungsi seleksi yang berbeda, sehingga kita harus memilih fungsi yang tepat jika kita ingin algoritma menghasilkan solusi optimal.
3. Jadi, pada sebagian persoalan, algoritma *greedy* tidak selalu berhasil memberikan solusi yang optimal, namun sub-optimal.
4. Jika solusi terbaik mutlak tidak terlalu diperlukan, maka algoritma *greedy* dapat digunakan untuk menghasilkan solusi hampiran (*approximation*), daripada menggunakan algoritma yang kebutuhan waktunya eksponensial untuk menghasilkan solusi yang eksak.
5. Algoritma *greedy* yang dapat menghasilkan solusi optimal, maka keoptimalannya itu harus dibuktikan secara matematis.

2.2. Pengaturan Game Engine

Pada *game engine Overdrive*, terdapat dua player yang akan bertarung, masing-masing *player* berada di *lane* yang berbeda ketika awal permainan. Penambahan *bot* dilakukan dengan menyesuaikan *bot* yang sudah di build melalui maven project yang akan dibangun *dependencies*-nya melalui pom.xml dan bot.json. Jadi, setiap perintah yang kita rancang di kelas java akan dieksekusi sesuai dengan prinsip pemrograman berorientasi objek. Cara membangun *game engine* adalah sebagai berikut :

1. Masuk ke dalam folder game-runner
2. Masukkan game-engine yang ada pada starter pack

3. Sesuaikan game-config-runner.json dan masukkan folder yang ditunjuk sebagai player A maupun player B
4. Klik build yang ada pada folder untuk menginstall game engine

2.3. Menambahkan Strategi Greedy pada Bot

Menambahkan strategi *greedy* ke dalam engine bisa dilakukan dengan cara pertama masuk ke folder starter-pack kemudian edit file-file yang terdapat pada folder .\starter-bots\java\src\main\java\za\co\entelect\challenge khususnya file Bot.java.

Algoritma *greedy* akan dimasukkan ke dalam fungsi *private Command run()*. Terdapat beberapa *command* yang dapat dimasukkan ke bot untuk menentukan langkah yang akan dilakukan oleh mobil yang bersangkutan. *Command* yang ada yaitu:

1. NOTHING : berfungsi untuk membuat mobil tidak melakukan tindakan apa-apap.
2. ACCELERATE : berfungsi untuk membuat kecepatan mobil meningkat ke kondisi kecepatan selanjutnya, bisa sampai ke *maximum speed* dan mobil tetap berada pada *lane* yang sama.
3. DECELERATE : berfungsi untuk membuat kecepatan mobil menurun ke kondisi kecepatan sebelumnya, bisa sampai ke *minimum speed* dan mobil tetap berada pada *lane* yang sama.
4. TURN_LEFT : berfungsi untuk membuat mobil berpindah ke *lane* yang berada di sebelah kiri posisi mobil pada saat itu, dengan menggunakan *command* ini maka kecepatan mobil pada saat itu berkurang satu poin.
5. TURN_RIGHT : berfungsi untuk membuat mobil berpindah ke *lane* yang berada di sebelah kanan posisi mobil pada saat itu, dengan menggunakan *command* ini maka kecepatan mobil pada saat itu berkurang satu poin.
6. USE_BOOST : berfungsi untuk meningkatkan kecepatan mobil menjadi 15 poin selama lima putaran, ketika mobil menabrak obstacle atau melakukan *decelerate* maka kecepatan mobil akan kembali ke kecepatan semula, *command* ini hanya dapat dilakukan ketika mobil memiliki *power up BOOST*.
7. USE_OIL : berfungsi untuk menumpahkan oli dibawah mobil *player*, yang apabila mobil lawan melewati tumpahan oli tersebut maka kecepatan mobil lawan akan berkurang ke level kecepatan sebelumnya.
8. USE_LIZARD : berfungsi untuk membuat mobil *player* melompat ketika terdapat *lizard* berlari di jalan.
9. USE_TWEET : berfungsi untuk menaruh obstacle berupa *cyber truck* pada posisi yang kita inginkan.
10. USE_EMP : berfungsi untuk menembakkan *EMP* ke depan mobil kita dan juga *lane* kanan dan kiri, lawan yang terkena *EMP* akan berhenti selama sisa putaran, dan untuk putaran berikutnya kecepatan akan menjadi tiga poin.

11. FIX : berfungsi untuk memperbaiki mobil kita dengan cara mengurangi *damage* sebesar dua poin.

Untuk menyusun identitas *bot*, kita harus mengurnya sesuai dengan pengaturan yang tertera pada file *bot.json* yang ada pada folder `.\starter-bots\java`. Pengaturan pada *bot.json* adalah sebagai berikut:

```
{  
    "author": <nama pembuat bot>,  
    "email": <email pembuat bot>,  
    "nickName": <Tampilan nama pada visualizer>,  
    "botLocation": <lokasi bot di build pada kelas tertenu>,  
    "botFileName": <format nama jar untuk menjalankan bot >,  
    "botLanguage": <bahasa pemrograman yang dipilih>  
}
```

Ketika strategi *greedy* dan identitas *bot* telah selesai diimplementasikan maka langkah selanjutnya adalah kita perlu melakukan *build* pada maven project yang ada pada folder *starter-bots*. Ketika proses *build* selesai maka file *executable* yang dihasilkan akan tersimpan dalam folder *target* yang terletak pada `.\starter-bots\java` dengan nama file yang sesuai dengan yang sebelumnya sudah diatur di file *bot.json*.

2.4. Menjalankan Game Engine

Kita dapat menjalankan *game* dan *bot* yang telah kita implementasikan dengan cara menjalankan file *run.bat* yang terdapat pada folder *starter-pack*. Ketika file *run.bat* dijalankan maka *cmd* akan muncul dan menampilkan *state* setiap *round* yang ada hasil dari menjalankan setiap *command* yang sebelumnya telah diimplementasikan di *bot*. Hasil dari menjalankan file *run.bat* adalah folder yang memiliki format nama *timestamp* yang berisi history setiap round beserta detail dari permainan seperti *command* yang digunakan oleh kedua pemain, kondisi *health* kedua pemain, dan lain lain, yang mana folder tersebut langsung tersimpan pada folder *match-logs*.

```
Starting game
=====
Starting round: 1
Player A - Coffee: Map View
=====
round:1
player: id:1 position: y:1 x:1 speed:5 state:READY statesThatOccurredThisRound:READY boosting:false boost-counter:0 damage:0 score:0 powerups:
opponent: id:2 position: y:4 x:1 speed:5

[1      #      ]
[      |      B  ]
[      |      ]
[2      ]      =====

Received command C;1;ACCELERATE
Player B - CoffeeRef: Map View
=====
round:1
player: id:2 position: y:4 x:1 speed:5 state:READY statesThatOccurredThisRound:READY boosting:false boost-counter:0 damage:0 score:0 powerups:
opponent: id:1 position: y:1 x:1 speed:5

[1      #      ]
[      |      B  ]
[      |      ]
[2      ]      =====

Received command C;1;ACCELERATE
Completed round: 1
=====
```

Gambar 2.4 Contoh tampilan CLI ketika file run.bat dijalankan

BAB III

Aplikasi Strategi *Greedy*

3.1. Mapping Persoalan

Asumsi yang digunakan untuk perancangan strategi *greedy* pada permainan ini adalah:

1. Setiap mobil akan bergiliran melakukan aksi dimulai dari *id* pertama kemudian *id* kedua
2. Pemain yang menabrak obstacle maka akan dikenai pengurangan *score* dan kecepatan pemain
3. Penggunaan dan pengambilan *power up* akan menambah *score* dan kecepatan pemain
4. Mobil yang dapat mencapai garis *finish* pertama adalah pemenangnya
5. Apabila kedua mobil sampai di garis *finish* secara bersamaan, maka pemenangnya ditentukan oleh perolehan *score*

Proses *mapping* persoalan *Overdrive* menjadi elemen-elemen algoritma *Greedy* (himpunan kandidat, himpunan solusi, fungsi seleksi, fungsi kelayakan, fungsi objektif). Adapun penjelasan dari *mapping* persoalan *Overdrive* adalah :

Himpunan kandidat :

{*TURN_RIGHT*, *TURN_LEFT*, *ACCELERATE*, *FIX*, *USE_BOOST*, *USE_TWEET*, *USE_OIL*,
USE_LIZARD, *USE_EMP* }

Himpunan solusi :

Command yang valid untuk setiap rondenya dan mobil pemain mendapatkan perintah yang sesuai dengan aturan permainan dan untuk memenangkan permainan.

Fungsi seleksi :

Pilihlah aksi yang memberikan jangkauan *block* paling jauh sesuai dengan *max speed* bagi mobil pemain pada saat itu, apabila posisi pemain pada saat itu tidak memberikan jangkauan *block* terjauh sesuai dengan *max speed* maka berbelok ke arah kanan atau kiri sesuai dengan keadaan mobil pemain pada saat itu dengan begitu juga maka secara otomatis mobil dapat mempertahankan nilai *max speed* yang paling besar.

Fungsi kelayakan :

Command valid seperti *power up LIZARD* masih dimiliki oleh mobil pemain, kemudian posisi mobil pemain valid ketika ingin berbelok ke arah kanan atau kiri.

Fungsi objektif :

Menangkan permainan dengan menjadi pemain pertama yang sampai di garis *finish* atau apabila pemain dan lawan sampai di garis *finish* bersamaan maka pemenang ditentukan dengan jumlah total *score* yang diperoleh.

3.2. Eksplorasi Alternatif Solusi Greedy

1. Greedy by avoiding obstacle

Sebuah strategi penerapan algoritma greedy yang mengutamakan pemilihan lane yang paling optimal. Optimal dalam hal ini berarti menghindari obstacle agar mobil dapat melaju dengan jarak tempuh yang paling jauh dan tidak menerima damage yang dapat melambatkan mobil dan menurunkan max speed. Hal ini dilakukan dengan cara melihat lane mana yang memiliki obstacle paling sedikit sehingga akan dilakukan command TURN_LEFT jika lane kiri lebih optimal, TURN_RIGHT jika lane kanan lebih optimal, atau ACCELERATE (tidak berbelok) jika lane yang sedang ditempati bot adalah lane yang paling optimal. Command akan divalidasi terlebih dahulu sebelum dijalankan, misalnya jika mobil berada di lane 1 (lane paling kiri) maka command TURN_LEFT tidak akan menjadi opsi command yang dapat dijalankan. Hal ini dilakukan untuk menghindari error miss input command yang dapat mengurangi skor dan menghilangkan satu round permainan tanpa melakukan apa-apa.

Analisis efisiensi dan efektivitas:

Algoritma greedy by lane dari sisi efisiensi memiliki efisiensi yang sama dengan algoritma greedy yang lainnya yaitu $O(n)$ dengan n sebagai jumlah ronde permainan. Hal ini dikarenakan terdapat iterasi algoritma sebanyak n kali. Jika tidak menghitung loop / iterasi dari tiap ronde maka kompleksitas waktu komputasi algoritma greedy ini hanya $O(1)$ hal ini dikarenakan algoritma hanya tersusun dari kumpulan conditional if dan else yang digunakan untuk pengambilan keputusan dalam setiap ronde permainan.

Algoritma greedy by lane merupakan algoritma yang cukup efektif untuk memenangkan permainan jika dibandingkan dengan algoritma greedy by speed. Hal ini dikarenakan bot akan memilih lane paling optimal untuk ditelusuri setiap round sehingga akan lebih sedikit menerima damage yang dapat menyebabkan menurunnya speed dan max speed mobil.

2. Greedy by speed

Sebuah strategi penerapan algoritma greedy yang mengutamakan kecepatan mobil berjalan diatas yang lain. Hal ini menyebabkan mobil akan memprioritaskan menggunakan power up BOOST (jika ada) karena dapat membuat speed mobil menjadi 15. Kemudian command berikutnya adalah ACCELERATE untuk menambahkan kecepatan mobil sebanyak 1. Algoritma greedy by speed juga harus melakukan command FIX yang banyak karena lebih mengutamakan kecepatan dibandingkan dengan menghindari obstacle. Hal ini perlu dilakukan karena setiap kali mobil terkena damage

maka max speed akan turun dan bahkan bisa berhenti apabila damage mencapai 5. Setelah mengutamakan semua hal diatas barulah bot akan melakukan command untuk menghindar dari obstacle.

Analisis efisiensi dan efektivitas:

Algoritma greedy by speed dari sisi efisiensi memiliki efisiensi yang sama dengan algoritma greedy yang lainnya yaitu $O(n)$ dengan n sebagai jumlah ronde permainan. Hal ini dikarenakan terdapat iterasi algoritma sebanyak n kali. Jika tidak menghitung loop / iterasi dari tiap ronde maka kompleksitas waktu komputasi algoritma greedy ini hanya $O(1)$ hal ini dikarenakan algoritma hanya tersusun dari kumpulan conditional if dan else yang digunakan untuk pengambilan keputusan dalam setiap ronde permainan.

Algoritma greedy by speed merupakan algoritma yang kurang efektif untuk memenangkan permainan. Hal ini dikarenakan walaupun secara logika mobil tercepat yang menang namun jika bot hanya berfokus untuk meningkatkan kecepatan tanpa menghindari dari obstacle maka bot akan menabrak dan speednya berkurang. Saat menabrak obstacle mobil juga akan mendapatkan damage yang dapat mengurangi max speed yang dapat dicapai oleh mobil tersebut, hal ini akan berdampak besar dalam jangka panjang. Sehingga penggunaan Algoritma greedy by speed ini kurang disarankan untuk membangun bot permainan Overdrive.

3. Greedy by power up

Sebuah strategi penerapan algoritma greedy yang mengutamakan pengambilan power up dan kemudian penggunaannya. Hal ini menyebabkan bot akan memilih lane yang memiliki power up dan bila power up sudah terkumpul sampai ke jumlah tertentu maka bot akan menggunakan power up sesuai dengan kebutuhan. Bot yang menggunakan algoritma ini akan bergerak dengan kecepatan yang lebih pelan karena lebih mengutamakan pengambilan power up daripada mempercepat diri dengan command ACCELERATE atau menghindar dari obstacle yang dapat menyebabkan mobil semakin pelan.

Analisis efisiensi dan efektivitas:

Algoritma greedy by power up dari sisi efisiensi memiliki efisiensi yang sama dengan algoritma greedy yang lainnya yaitu $O(n)$ dengan n sebagai jumlah ronde permainan. Hal ini dikarenakan terdapat iterasi algoritma sebanyak n kali. Jika tidak menghitung loop / iterasi dari tiap ronde maka kompleksitas waktu komputasi algoritma greedy ini hanya $O(1)$ hal ini dikarenakan algoritma hanya tersusun dari kumpulan conditional if dan else yang digunakan untuk pengambilan keputusan dalam setiap ronde permainan.

Algoritma greedy by power up merupakan algoritma yang kurang efektif untuk memenangkan permainan. Hal ini dikarenakan algoritma hanya fokus mengambil power up yang belum tentu berguna juga apabila digunakan, misalnya jika mobil kita dibelakang mobil musuh maka power up OIL tidak berguna atau jika kita di depan mobil musuh maka power EMP tidak berguna. Ditambah lagi dengan kecepatan mobil yang cenderung tidak bertambah dan sering bertabrakan dengan obstacle maka algoritma ini kurang efektif dalam memenangkan permainan.

BAB IV

Implementasi dan Pengujian

4.1. Implementasi Program dalam Game Engine

```
// Perbaiki mobil jika terlalu rusak

IF (damage lebih dari 5) then
    Gunakan Command FIX

// Percepat mobil jika mobil berhenti
IF (kecepatan kurang dari 3) then
    Gunakan command ACCELERATE

// Algoritma menghindar dari rintangan
IF (Terdapat rintangan di depan mobil) then
    IF (Mobil pada ordinat 4 atau 1) then
        IF (Terdapat rintangan di depan kanan mobil)then
            Gunakan command TURN_LEFT
        ELSE IF (Terdapat rintangan di depan kiri mobil) then
            Gunakan command TURN_RIGHT
        ELSE IF (Terdapat rintangan di depan kanan mobil dan depan
kiri mobil) then
            IF (Memiliki lizard) then
                Gunakan command LIZARD
            IF (Terdapat truck di depan) then
                IF (Terdapat truck di depan kiri)
                    Gunakan command TURN_RIGHT
                ELSE IF (Terdapat truck di depan kanan)
                    Gunakan command TURN_LEFT
                ELSE IF (Terdapat truck di depan kanan dan depan
kiri)
                    Gunakan command TURN_RIGHT
            ELSE
                IF (Terdapat truck di depan kanan) THEN
                    IF (Tidak Terdapat truck di depan kiri) THEN
                        IF (damage depan kiri < damage depan) THEN
                            Gunakan command TURN_LEFT
                    ELSE
                        IF (Terdapat truck di depan kiri)
                            IF(damage depan kanan < damage depan)THEN
                                Gunakan command TURN_RIGHT
                        ELSE
                            Mobil memilih jalan yang memberikan damage
                            lebih sedikit
                    ELSE IF (Tidak terdapat rintangan di depan kanan dan tidak
terdapat rintangan di depan kiri) then
                        Gunakan command TURN_RIGHT
```

```
ELSE IF (Mobil pada ordinat 1) then
    IF (Terdapat rintangan di depan kanan mobil) THEN
        IF (Memiliki lizard) then
            Gunakan command LIZARD
        ELSE IF (Terdapat truk di depan mobil) THEN
            Gunakan command TURN_RIGHT
        ELSE
            IF (Tidak terdapat truk di depan kanan) THEN
                IF (damage depan > damage kanan) THEN
                    Gunakan command TURN_RIGHT
            ELSE
                Gunakan command TURN_RIGHT
    ELSE IF (Mobil pada ordinat 4) then
        IF (Terdapat rintangan di depan kiri mobil) then
            IF (Memiliki lizard) then
                Gunakan command LIZARD
            ELSE IF (Terdapat truk di depan mobil) then
                Gunakan command TURN_LEFT
            ELSE
                IF (Tidak terdapat truk di depan kiri) THEN
                    IF (damage depan > damage kiri) THEN
                        Gunakan command TURN_LEFT
                ELSE
                    Gunakan command TURN_LEFT

// implementasi penggunaan power up
IF (Player memiliki power up) THEN
    IF (Player memiliki boost) THEN
        IF (damage mobil == 0) THEN
            IF (Tidak terdapat rintangan pada 15 block di depan) THEN
                Gunakan command BOOST
        IF (Absis player di belakang musuh) THEN
            IF (Player memiliki emp) THEN
                Gunakan command EMP
        IF (Player memiliki tweet) THEN
            Gunakan command TWEET
        IF (Absis player di depan musuh) THEN
            IF (Player mencapai max speed) THEN
                IF (Player memiliki oil) THEN
                    Gunakan command OIL

    IF (Player memiliki boost) THEN
        IF (damage lebih dari sama dengan 1) THEN
            Gunakan command FIX
        IF (damage lebih dari sama dengan 2) THEN
            Gunakan command FIX

ELSE
    Gunakan command ACCELERATE
```

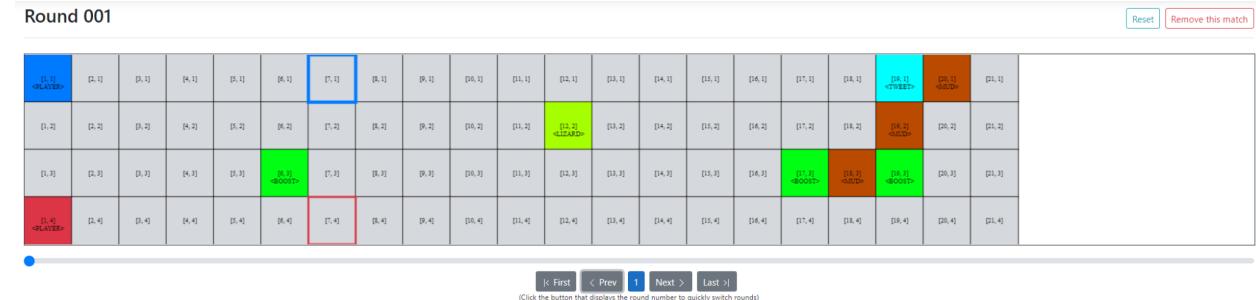
4.2. Penjelasan Struktur Data

Adapun struktur data yang kami gunakan terbagi menjadi tiga bagian besar yaitu command, entities, enum yang disertakan dengan *bot.java* dan *main.java*.

1. *Command* berfungsi untuk menyimpan kelas-kelas yang berhubungan dengan prosedur game-engine yang dijelaskan pada Bab 2
 - a. *AccelerateCommand* : berfungsi untuk melakukan akselerasi atau menambah kecepatan mobil pemain
 - b. *BoostCommand* : berfungsi untuk menggunakan *power up BOOST* apabila pemain memilikinya
 - c. *ChangeLaneCommand* : berfungsi untuk berpindah *lane*, ke kanan atau ke kiri
 - d. *DecelerateCommand* : berfungsi untuk melakukan dekselerasi atau mengurangi kecepatan mobil pemain
 - e. *DoNothingCommand* : berfungsi untuk membuat mobil pemain tidak melakukan kegiatan apapun
 - f. *EmpCommand* : berfungsi untuk menggunakan *power up EMP* apabila pemain memilikinya
 - g. *FixCommand* : berfungsi untuk memperbaiki mobil atau mengurangi damage pada mobil pemain
 - h. *LizardCommand* : berfungsi untuk menggunakan *power up LIZARD* apabila pemain memilikinya
 - i. *OilCommand* : berfungsi untuk menggunakan *power up OIL* apabila pemain memilikinya
 - j. *TweetCommand Y X* : berfungsi untuk menggunakan *power up TWEET* pada lane Y dan block X.
2. Entities berguna sebagai objek yang ada pada *gamestate*
 - a. *Car* : objek yang menyimpan state mobil pada ronde tertentu.
 - b. *GameState* : objek yang menyimpan *state* tertentu mulai dari health, id, damage, dan infor
 - c. *Lane* : objek yang menyimpan posisi, medan, dan pemain yang berada pada lane tersebut sepanjang permainan *Overdrive*
 - d. *Position* : objek posisi berupa koordinat
3. Enums berguna sebagai alat iterasi yang dibuat untuk mencocokan setiap kemungkinan yang ada
 - a. *Direction* : arahan mobil pemain untuk bergerak
 - b. *PowerUps* : berisi kemungkinan *power ups* yang akan dimiliki oleh mobil pemain
 - c. *State* : Keadaan mobil pada ronde saat ini
 - d. *Terrain* : berisi block yang terdapat pada map
4. *Bot.java* berisi algoritma bot yang dibuat oleh pemain

5. Main.java berisi program utama untuk menjalankan bot selama permainan berlangsung.

4.3. Analisis Desain Solusi



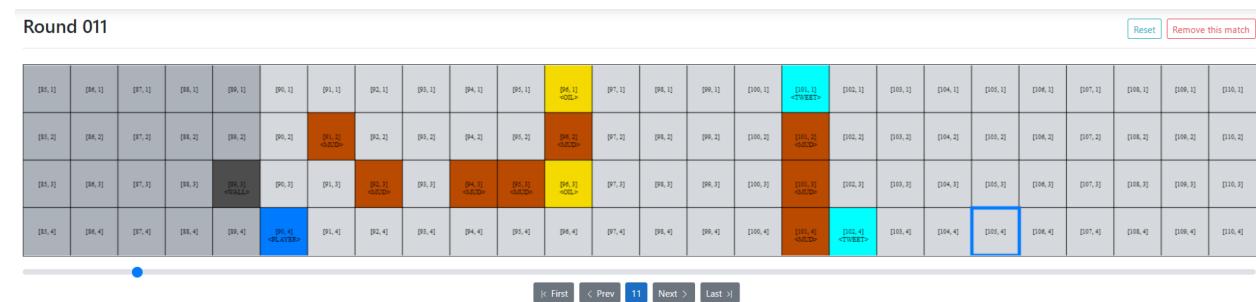
Gambar 4.3.1 Awal permainan



Gambar 4.3.2 Bot berbelok ketika menemukan rintangan



Gambar 4.3.3 Bot berbelok untuk mengambil boost



Gambar 4.3.4 Bot memilih jalur dengan damage terendah

Round 038

[Reset](#) [Remove this match](#)

[426, 1]	[427, 1]	[428, 1]	[429, 1]	[430, 1]	[431, 1] CHARTER	[432, 1]	[433, 1]	[434, 1]	[435, 1]	[436, 1]	[437, 1]	[438, 1]	[439, 1] CHARTER	[440, 1]	[441, 1]	[442, 1]	[443, 1]	[444, 1]	[445, 1] CHARTER	[446, 1] CHARTER	[447, 1]	[448, 1]	[449, 1]	[450, 1]	[451, 1]
[426, 2]	[427, 2]	[428, 2]	[429, 2]	[430, 2]	[431, 2]	[432, 2]	[433, 2]	[434, 2]	[435, 2]	[436, 2]	[437, 2]	[438, 2]	[439, 2] CHARTER	[440, 2]	[441, 2]	[442, 2]	[443, 2]	[444, 2] CHARTER	[445, 2] CHARTER	[446, 2]	[447, 2]	[448, 2]	[449, 2]	[450, 2]	[451, 2]
[426, 3]	[427, 3]	[428, 3]	[429, 3]	[430, 3]	[431, 3]	[432, 3] CHARTER	[433, 3]	[434, 3]	[435, 3]	[436, 3]	[437, 3]	[438, 3]	[439, 3] CHARTER	[440, 3]	[441, 3] CHARTER	[442, 3]	[443, 3]	[444, 3]	[445, 3] CHARTER	[446, 3]	[447, 3]	[448, 3]	[449, 3]	[450, 3]	[451, 3]
[426, 4]	[427, 4]	[428, 4]	[429, 4]	[430, 4]	[431, 4]	[432, 4]	[433, 4]	[434, 4]	[435, 4]	[436, 4]	[437, 4]	[438, 4]	[439, 4] CHARTER	[440, 4]	[441, 4]	[442, 4]	[443, 4]	[444, 4]	[445, 4] CHARTER	[446, 4]	[447, 4]	[448, 4]	[449, 4]	[450, 4]	[451, 4]

Bot Command

Command: USE_LIZARD

Execution time: 6ms

Exception: null

Gambar 4.3.5 Bot menggunakan lizard jika terhalang rintangan

Round 113

[Reset](#) [Remove this match](#)

[1421, 1]	[1433, 1]	[1484, 1]	[1445, 1] «ФИЛ»	[1486, 1]	[1487, 1]	[1488, 1]	[1489, 1]	[1490, 1]	[1491, 1]	[1492, 1]	[1493, 1]	[1494, 1]	[1495, 1]	[1496, 1]	[1497, 1] «ФИЛ»	[1498, 1] «ФИЛ»	[1499, 1]	[1500, 1] «ФИЛ»
[1421, 2]	[1433, 2]	[1484, 2]	[1445, 2] «ФИЛ»	[1486, 2]	[1487, 2] «ФИЛ»	[1488, 2]	[1489, 2]	[1490, 2]	[1491, 2] «ФИЛ»	[1492, 2]	[1493, 2]	[1494, 2]	[1495, 2]	[1496, 2]	[1497, 2]	[1498, 2]	[1499, 2]	[1500, 2] «ФИЛ»
[1421, 3]	[1433, 3]	[1484, 3] «ФИЛ»	[1445, 3] «ФИЛ»	[1486, 3] «ФИЛ»	[1487, 3] «ФИЛ»	[1488, 3]	[1489, 3]	[1490, 3]	[1491, 3]	[1492, 3]	[1493, 3]	[1494, 3]	[1495, 3]	[1496, 3]	[1497, 3]	[1498, 3]	[1499, 3] «ФИЛ»	[1500, 3] «ФИЛ»
[1421, 4]	[1433, 4]	[1484, 4]	[1445, 4]	[1486, 4]	[1487, 4]	[1488, 4]	[1489, 4]	[1490, 4]	[1491, 4]	[1492, 4]	[1493, 4]	[1494, 4]	[1495, 4]	[1496, 4]	[1497, 4]	[1498, 4]	[1499, 4]	[1500, 4] «ФИЛ»

Gambar 4.3.6 Bot mencapai garis finish

Gambar diatas merupakan hasil pertandingan bot Algoritma Tamak melawan bot *reference*. Bot Algoritma Tamak berwarna biru dan mulai dari posisi lane 1. Bot *reference* berada pada block yang sama namun pada posisi lane 4. Pada gambar 4.3.2 bot mendeteksi adanya rintangan di hadapan bot, sehingga bot memilih untuk berbelok ke kanan dan tidak mengalami kerusakan serta perlambatan. Pada Gambar 4.3.3 bot Algoritma Tamak berbelok untuk mendapatkan boost. Hal ini dilakukan karena mobil akan berjalan jauh lebih cepat ketika melakukan boost. Ketika setiap jalur yang bisa dilalui bot dihalangi oleh rintangan, bot akan memilih jalur dengan total damage terendak. Pada gambar 4.3.4, terlihat rintangan menutupi semua jalur yang dapat dilalui bot, bot memilih untuk tetap maju karena rintangan di kiri akan memberikan damage yang lebih besar. Jika bot mempunyai lizard, bot akan menggunakan lizard seperti yang terlihat pada gambar 4.3.5. Pada gambar 4.3.6, terlihat bot kami mencapai garis finish pada ronde 113. Menurut kami, performa bot Algoritma Tamak cukup memuaskan karena memiliki rata-rata waktu menyelesaikan pertandingan yang cukup cepat dibanding dengan alternatif bot kami yang lain. Menurut analisis kami, mobil dapat berjalan dengan baik jika tidak terlalu banyak menabrak rintangan.

BAB V

Kesimpulan dan Saran

5.1. Kesimpulan

Kesimpulan yang dapat diambil dari implementasi algoritma *greedy* pada pembuatan *bot* permainan *Overdrive* adalah algoritma *greedy* dapat diimplementasikan dengan baik dan dapat mencapai objektif dari permainan ini sendiri yaitu memenangkan permainan dengan mencapai garis *finish* pertama atau memperoleh *score* yang tinggi. Strategi *greedy* yang diimplementasikan pada program kali ini adalah *greedy by avoiding obstacle* yang berfokus pada gerakan menghindari *obstacle* yang ada pada *map* permainan, dengan begitu maka kecepatan mobil dari pemain akan stabil atau selalu berada pada kecepatan yang optimal dan mobil pemain tidak mengalami pengurangan *score*.

Video demo program implementasi algoritma *greedy* pada permainan *Overdrive* dapat diakses pada link berikut : <https://youtu.be/OA1xUbCkm3Q>

Source code program implementasi algoritma *greedy* pada permainan *Overdrive* dapat diakses pada link berikut : <https://github.com/vincen-tho/Greedy-Algorithm-Bot>

5.2. Saran

Berdasarkan hasil kerja kelompok kami, terdapat beberapa saran yang dapat digunakan untuk membantu pengimplementasian algoritma *greedy* pada permainan *Overdrive* :

1. Diperlukan pemahaman yang baik pada konsep pemrograman berorientasi objek, karena pada implementasi algoritma *greedy* kali ini menggunakan bahasa pemrograman java.
2. Diperlukan pemahaman yang baik pada konsep algoritma *greedy*, karena dengan pemahaman yang baik tersebut algoritma yang dihasilkan diharapkan menjadi algoritma yang paling efektif dan efisien.
3. Menurut kami tahap yang paling sulit dari perancangan algoritma *greedy* ini adalah proses mapping persoalan, seperti penentuan himpunan kandidat, himpunan solusi, fungsi seleksi, fungsi kelayakan, dan fungsi objektif.
4. Menurut kelompok kami proses debugging pada perancangan algoritma *greedy* ini juga lumayan sulit, karena kita diharuskan melakukan *build* pada *maven project* setiap ingin melakukan evaluasi terhadap program yang telah dibuat.

Daftar Pustaka

1. [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)-Bag1.pdf)
2. <https://github.com/EntelectChallenge/2020-Overdrive>