

**LAPORAN TUGAS BESAR**

**Pemanfaatan Algoritma Greedy dalam Aplikasi  
Permainan “Overdrive”**

Ditujukan untuk memenuhi salah satu tugas besar mata kuliah IF2211 Strategi Algoritma  
pada Semester II Tahun Akademik 2021/2022

Disusun oleh:

<b>Saul Sayers (K1)</b>	<b>13520094</b>
<b>Patrick Amadeus Irawan (K1)</b>	<b>13520109</b>
<b>Rania Dwi Fadhilah (K1)</b>	<b>13520142</b>



**PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
BANDUNG  
2021**

## DAFTAR ISI

DAFTAR ISI .....	i
BAB I DESKRIPSI TUGAS .....	1
BAB II LANDASAN TEORI .....	3
BAB III APLIKASI STRATEGI <i>GREEDY</i> .....	10
BAB IV IMPLEMENTASI DAN PENGUJIAN .....	14
BAB V KESIMPULAN DAN SARAN .....	24
LINK PENTING & DAFTAR PUSTAKA .....	ii

## BAB I

### 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. 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* <*lane*> <*block*>
  - 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>

## BAB II

### LANDASAN TEORI

#### 2. 1. Algoritma *Greedy*

Algoritma *greedy* adalah algoritma yang mengikuti pemecahan masalah heuristik yang dilakukan dengan cara membangun sebuah solusi langkah per langkah. Pada setiap langkahnya, diambil keputusan yang terlihat paling menguntungkan dan optimal tanpa melihat konsekuensi ke depannya. Strategi ini mengambil nilai optimum lokal sementara, tanpa mempertimbangkan gambaran besar permasalahan. Seluruh nilai optimum lokal kemudian disatukan dengan harapan dapat memperoleh solusi optimum global.

Selaras dengan namanya yang dapat diartikan sebagai rakus, prinsip utama dari algoritma ini adalah “*take what you can get now!*”. Kalimat ini menggambarkan bahwa ketika menerapkan algoritma *greedy*, maka keputusan yang diambil adalah keputusan yang paling optimal dan menguntungkan pada setiap langkahnya dengan harapan bahwa dapat tercapai nilai optimum global. Dalam berbagai pemecahan masalah, strategi algoritma ini kerap tidak menghasilkan solusi yang paling optimal. Namun, melalui algoritma ini, didapatkan solusi yang paling mendekati solusi optimal global dalam waktu yang wajar.

Algoritma ini memiliki beberapa elemen umum, yaitu :

1. Himpunan kandidat (C), adalah himpunan yang berisi elemen kandidat solusi (dapat berupa simpul/sisi dalam graf, job, task, koin, benda, dll).
2. Himpunan solusi (S), adalah himpunan yang berisi kandidat solusi terpilih.
3. Fungsi solusi, adalah fungsi yang menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi (nilainya *boolean*).
4. Fungsi seleksi, adalah fungsi yang memilih kandidat berdasarkan strategi *greedy* tertentu (bersifat heuristik).
5. Fungsi kelayakan, adalah fungsi yang mempertimbangkan kelayakan kandidat yang dipilih untuk dapat dimasukkan ke dalam himpunan solusi.
6. Fungsi obyektif, adalah fungsi yang mengoptimalkan solusi.

Skema umum dari 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
    write('tidak ada solusi')
endif
```

Dalam tugas Overdrive ini, akan diimplementasikan algoritma *greedy* untuk dapat menyelesaikan permasalahan optimasi “mengalahkan lawan secara lebih cepat”. Algoritma permainan ini akan diimplementasikan pada *game engine* yang telah disediakan Entelect dan akan dijelaskan secara lebih detail dalam laporan ini.

## 2.2. Game Engine Permainan Overdrive

*Game engine* adalah perangkat lunak yang berfungsi untuk membuat proses pembuatan dan pengembangan permainan lebih ekonomis. Pada sub-bab ini, akan dijelaskan mengenai *game engine* yang digunakan untuk permainan Overdrive, pemain, cara mengimplementasikan algoritma *greedy*, dan cara menjalankan *game engine*.

### 2.2.1 Prerequisites

Permainan Overdrive dalam tugas ini dijalankan menggunakan *game engine* yang sudah tersedia dan dapat diunduh pada laman <https://github.com/EntelectChallenge/2020-Overdrive/releases/tag/2020.3.4>. Selain *game engine*-nya sendiri, kita juga harus memenuhi beberapa *requirement* dasar, yaitu :

1. Bahasa Pemrograman Java (minimal Java 8) :  
<https://www.oracle.com/java/technologies/downloads/#java8>
2. IDE IntelliJ / VS Code + Maven :  
<https://www.jetbrains.com/idea/>  
<https://code.visualstudio.com/download> + <https://maven.apache.org/download.cgi>
3. NodeJS : <https://nodejs.org/en/download/>

### 2.2.2 Starter-pack

Pada laman Github Overdrive, file yang perlu diunduh adalah file starter-pack.zip. *Game engine* terdapat pada zip ini. *Game engine* tersedia dalam bentuk Executable Jar File. File inilah yang kemudian akan digunakan untuk menjalankan permainan. Aturan dan cara kerja permainan Overdrive sendiri dapat diakses melalui laman berikut : <https://github.com/EntelectChallenge/2020-Overdrive/blob/master/game-engine/game-rules.md#structure>

Struktur file dari folder *starter-pack* ini adalah sebagai berikut :

```
|--- match-logs
|--- reference-bot
|--- starter-bots
|   |--- cplusplus
|   |--- dotnetcore
|   |--- golang
```

```
|--- java
|   |--- src
|   |   |--- run
|   |   |   |--- main
|   |   |       |--- {package_directories}
|   |   |           |--- Bot.java
|   |   |           |--- Main.java
|   |   |           |--- entities
|   |   |               |--- Car.java
|   |   |               |--- GameState.java
|   |   |               |--- Lane.java
|   |   |               |--- Position.java
|   |   |           |--- enums
|   |   |               |--- Powerups.java
|   |   |               |--- Direction.java
|   |   |                   |--- State.java
|   |   |                   |--- Terrain.java
|   |   |           |--- command
|   |   |               |--- AccelerateCommand.java
|   |   |               |--- BoostCommand.java
|   |   |               |--- ChangeLaneCommand.java
|   |   |               |--- Command.java
|   |   |               |--- DecelerateCommand.java
|   |   |               |--- DoNothingCommand.java
|   |   |               |--- OilCommand.java
|   |   |--- target
|   |   |--- bot.json
|   |   |--- pom
|   |--- javascript
|   |--- lisp
|   |--- python3
|   |--- rust
|   |--- README.md
|--- game-config.json
|--- game-engine
|--- game-runner-config.json
|--- game-runner-jar-with-dependencies
|--- makefile
```

### 2.2.3 Pemain

Permainan Overdrive adalah permainan yang mempertandingkan 2 bot mobil. Oleh karena itu, pada *starter-pack* terdapat 2 bot yang dapat digunakan sebagai pemain. Masing-masing bot dilengkapi oleh berbagai jenis kode yang dapat digunakan dan diatur sedemikian rupa untuk menjalankan perintah yang sesuai dengan bawaan program. Bot pertama adalah *reference-bot*, yaitu bot bawaan yang disediakan oleh Entelect sebagai bot referensi. Bahasa yang digunakan oleh bot ini adalah Java. Bot kedua adalah *starter-bots*, yaitu bot yang dapat dimodifikasi sesuai strategi permainan. Bahasa yang disediakan oleh bot kedua ada banyak, antara lain C++, .NET Core, Golang, Java, JavaScript, Lisp, Python, dan Rust. Pada tugas ini, bahasa yang akan kita gunakan adalah Java. Isi/struktur dari folder Java terdapat pada struktur file *starter-pack* yang ada pada subbab sebelumnya.

Dengan memilih bahasa permainan yang tidak sesuai dengan *default*, maka *programmer* harus mengubah file `game-runner-config.json` yang ada pada direktori `starter-pack`. Bagian yang diubah adalah “player-a”, dari bahasa sebelumnya menjadi bahasa yang ingin diimplementasikan.

```
1 {
2   "round-state-output-location": "./match-logs",
3   "game-config-file-location": "game-config.json",
4   "game-engine-jar": "game-engine.jar",
5   "verbose-mode": true,
6   "max-runtime-ms": 1000,
7   "player-a": "./starter-bots/java",
8   "player-b": "./reference-bot/java",
9   "max-request-retries": 10,
10  "request-timeout-ms": 5000,
11  "is-tournament-mode": false,
12  "tournament": {
13    "connection-string": "",
14    "bots-container": "",
15    "match-logs-container": "",
16    "game-engine-container": "",
17    "api-endpoint": "http://localhost"
18  }
19 }
```

Terdapat opsi opsional untuk mengganti pengaturan nama pemain, yaitu dengan cara mengedit file `bot.json` yang tersedia pada folder `Java`. Kolom “`nickName`” dapat diubah sesuai preferensi. Pengubahan ini akan mengubah tampilan nama bot pada permainan.

```
1 {
2   "author": "John Doe",
3   "email": "John.Doe@example.com",
4   "nickName": "s4Rap",
5   "botLocation": "/target",
6   "botFileName": "java-starter-bot-jar-with-dependencies.jar",
7   "botLanguage": "java"
8 }
9
```

## 2.2.4 Implementasi Algoritma *Greedy* pada Bot

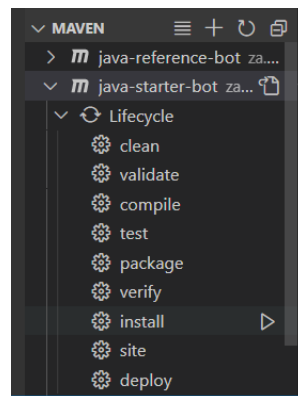
Berdasarkan spesifikasi tugas, bahasa pemrograman yang digunakan adalah `Java`. Oleh karena itu, algoritma *greedy* akan diimplementasikan pada direktori `java`, lebih tepatnya file `bot.java`. Pada file `bot.java`, terdapat `public command run ()`. Strategi permainan menggunakan algoritma *greedy* akan dimasukkan pada bagian `command` tersebut. `Command` yang dapat digunakan pada permainan ini antara lain adalah :

1. NOTHING = tidak melakukan apa-apa
2. ACCELERATE = meningkatkan kecepatan mobil ke *state* kecepatan selanjutnya (hanya bisa sampai kecepatan maksimal atau 9)
3. DECELERATE = menurunkan kecepatan mobil ke *state* kecepatan sebelumnya
4. TURN\_LEFT = pindah ke lane kiri (Apabila `speed` = 9, maka akan pindah ke lane kiri sebanyak 1 kemudian 8 block ke depan)
5. TURN\_RIGHT = pindah ke lane kanan
6. USE\_BOOST = apabila mempunyai *powerup boost* mengubah kecepatan menjadi 15 selama 5 putaran



7. USE\_OIL = apabila mempunyai *powerup oil*, maka akan membuat *block oil* di bawah posisi bot mobil kita yang kemudian akan berimbas pada bot mobil lawan apabila melintasi *block* tersebut
8. USE\_TWEET X Y = X dan Y diisi dengan nomor *lane* dan *block*. Apabila mempunyai *powerup tweet*, maka akan menaruh cybertruck pada *lane* dan *block* yang diminta. Cybertruck akan membuat bot mobil terjebak di belakangnya (apabila berada di *lane* yang sama) dan kecepatan akan berubah menjadi 3
9. USE\_LIZARD = apabila mempunyai *powerup lizard*, maka semua *powerup pickups*, hambatan, juga pemain lain akan diabaikan
10. USE\_EMP = apabila mempunyai *powerup emp*, maka dapat menembak EMP ke seluruh *lane* di depannya dan akan membuat bot mobil lawan berhenti dan kecepatan berubah menjadi 3.
11. FIX = mengurangi poin *damage* sebanyak 2

Setelah mengimplementasikan algoritma, maka perlu dilakukan *build* menggunakan Maven agar terbentuk *executable jar file* dari bot tersebut. File *.jar* yang dihasilkan setelah proses ini akan dibaca oleh *game engine*. Pada VSCode, *build* dilakukan dengan cara menekan compile kemudian install pada tombol berikut :



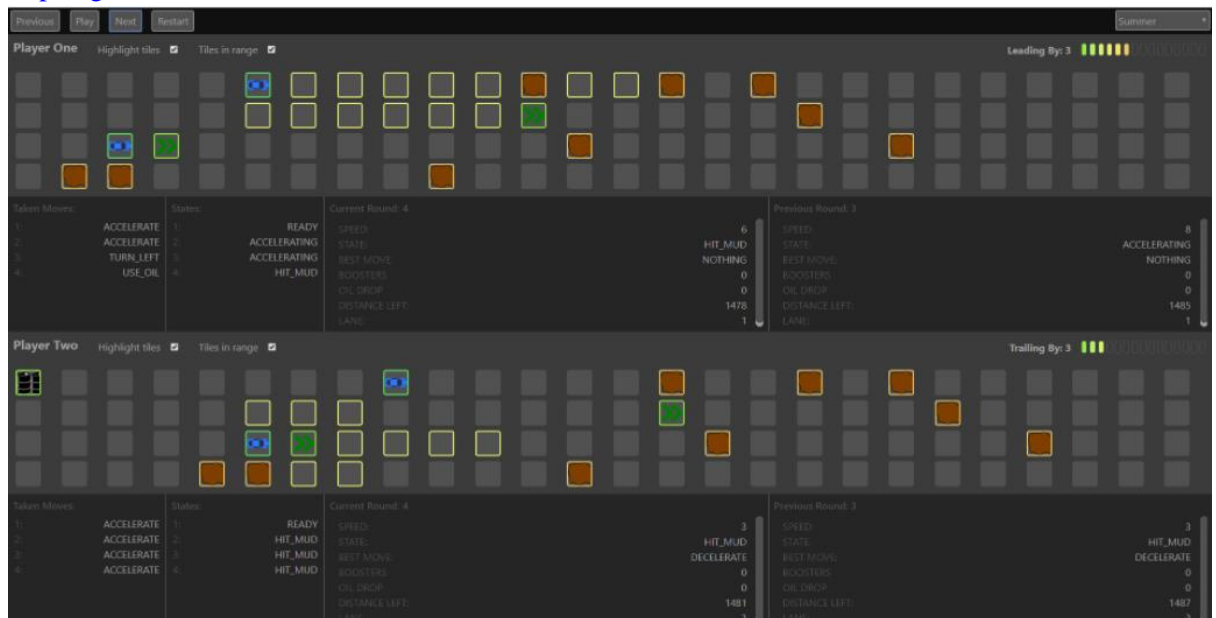
### 2.2.5 Menjalankan *game engine*

Permainan dapat dijalankan dengan membuka file *run.bat* yang tersedia pada direktori *starter-pack*. Setelah dibuka, file akan berjalan secara otomatis dalam jangka waktu yang cukup lama. Detail tiap *round* dan pemenang akan terlihat pada terminal. Selain itu, *history* pertandingan juga akan secara otomatis tersimpan pada direktori *match-logs*. Berikut adalah contoh tampilan permainan Overdrive pada command line.

```
*****
Starting round: 228
Player A - s4Rap: Map View
*****
round:228
Player: id:1 position: y:3 x:1496 speed:8 state:ACCELERATING statesThatOccurredThisRound:ACCELERATING boosting:false boost-counter:0 damage:2 score:201 powerups: OIL:1, LIZARD:1, EMP:2, TWEET:12
opponent: id:2 position: y:4 x:797 speed:8
*****
[Map View]
*****
Received command C;228;ACCELERATE
Received command S; avg speed = 6.000000
Player B - CoffeeRef: Map View
*****
round:228
Player: id:2 position: y:4 x:797 speed:0 state:HIT_WALL statesThatOccurredThisRound:ACCELERATING, HIT_MUD, HIT_WALL boosting:false boost-counter:0 damage:5 score:-46 powerups: OIL:11, LIZARD:1, EMP:9, TWEET:6
opponent: id:1 position: y:3 x:1496 speed:8
*****
[Map View]
*****
Received command C;228;FIX
Completed round: 228
*****
Game Complete
Checking if match is valid
*****
The winner is: A - s4Rap
*****
A - s4Rap - score:201 health:0
B - CoffeeRef - score:-46 health:0
*****
```

Tampilan pada terminal cenderung sulit untuk dibaca dan cukup membosankan. Oleh karena itu, terdapat beberapa *visualizer* yang dapat digunakan untuk melihat detail permainan, yaitu :

1. <https://github.com/Affuta/overdrive-round-runner>



2. <https://entelect-replay.raezor.co.za/>

### Round 001

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

[1, 1] s4Rap	[2, 1]	[3, 1]	[4, 1]	[5, 1]	[6, 1]	[7, 1]	[8, 1]	[9, 1]	[10, 1]	[11, 1]	[12, 1]	[13, 1]	[14, 1]	[15, 1]	[16, 1]	[17, 1]	[18, 1]	[19, 1]	[20, 1]	[21, 1]	
[1, 2]	[2, 2]	[3, 2]	[4, 2]	[5, 2]	[6, 2]	[7, 2]	[8, 2]	[9, 2]	[10, 2]	[11, 2]	[12, 2]	[13, 2]	[14, 2]	[15, 2]	[16, 2]	[17, 2]	[18, 2]	[19, 2]	[20, 2]	[21, 2]	
[1, 3]	[2, 3]	[3, 3]	[4, 3]	[5, 3]	[6, 3]	[7, 3]	[8, 3]	[9, 3]	[10, 3] s4Rap	[11, 3]	[12, 3]	[13, 3]	[14, 3]	[15, 3]	[16, 3]	[17, 3]	[18, 3]	[19, 3] CoffeeRef	[20, 3]	[21, 3]	
[1, 4] s4Rap	[2, 4]	[3, 4]	[4, 4]	[5, 4]	[6, 4]	[7, 4]	[8, 4]	[9, 4]	[10, 4] s4Rap	[11, 4]	[12, 4]	[13, 4]	[14, 4]	[15, 4]	[16, 4]	[17, 4]	[18, 4]	[19, 4] CoffeeRef	[20, 4]	[21, 4]	

[First](#) [Prev](#) [1](#) [Next](#) [Last](#)

(Click the button that displays the round number to quickly switch rounds)








### Round Details

Max Rounds: 600

Current Round: 1

#### A - s4Rap

[\(select\)](#)

						
Position <b>1</b> [x: 1, y: 1]	Speed <b>5</b>	Lane <b>1</b>	Distance <b>1</b>	Boosts <b>0</b>	Boosting <b>No</b>	Powerups

#### Bot Command

Command: ACCELERATE

Execution time: 751ms

Exception: null

#### B - CoffeeRef

[\(click to select\)](#)

#### End Game Result

Match seed: 53790

The winner is: A - s4Rap

A - s4Rap - score:201 health:0

B - CoffeeRef - score:-46 health:0

## BAB III

### APLIKASI STRATEGI *GREEDY*

#### 3.1 Mapping persoalan Overdrive Menjadi Elemen-elemen Algoritma Greedy

##### 3.1.1 Himpunan Kandidat

Semua jenis *command* yang dapat dipilih di dalam permainan, yakni :

NOTHING, ACCELERATE, DECELERATE, TURN\_LEFT, TURN\_RIGHT, USE\_BOOST, USE\_OIL, USE\_LIZARD, USE\_TWEET, USE\_EMP, FIX

##### 3.1.2 Himpunan Solusi

Runtutan pemakaian *command* yang diatur sedemikian rupa untuk memenangkan pertandingan balap *Overdrive* melawan bot lawan.

##### 3.1.3 Fungsi Solusi

Fungsi untuk menentukan apakah pilihan *command* meminimalkan margin kekalahan / memaksimalkan margin kemenangan.

##### 3.1.4 Fungsi Seleksi

Jika kondisi *damage* terlalu tinggi, prioritas untuk seleksi *fix* terlebih dahulu. Prioritas seleksi selanjutnya adalah menghindari *obstacle* dengan tingkatan **Cyber Truck, Wall, Oil, Mud** apabila dihadapkan dengan *obstacle*. Penghindaran ini diprioritaskan dengan menggunakan **Lizard** terlebih dahulu, baru kemudian berpindah jalur. Apabila bebas dari *obstacles* atau perpindahan ditinjau dari keadaan *obstacles* tidak memungkinkan, maka perpindahan akan ditinjau dari perbandingan keberadaan power-up. Jika tetap tidak memungkinkan, mobil akan bergerak ke arah jalur tengah (Lane 2 / 3).

##### 3.1.5 Fungsi Kelayakan

Fungsi yang memeriksa kelayakan suatu *command* untuk dilakukan atau tidak. *Command* penggunaan *power-up* tertentu akan melakukan pengujian kelayakan apakah *power-up* bersangkutan dimiliki atau tidak. Begitu juga dengan *command* perpindahan, dilakukan uji kelayakan apakah perpindahan dapat dilakukan atau tidak dengan parameter tertentu.

##### 3.1.6 Fungsi Objektif

Fungsi yang memeriksa bot berhasil memenangkan pertandingan dengan kondisi berikut :

- Sampai lebih dahulu di garis finis
- Kecepatan bot lebih tinggi apabila bersamaan sampai di garis finis
- Skor bot lebih tinggi apabila bersamaan sampai di garis finis dengan kecepatan yang sama

## 3.2 Alternatif solusi *greedy*

### 3.2.1 *Greedy by Obstacles* (Alternatif Solusi 1)

Bot akan berusaha melakukan penghindaran *obstacles* dengan prioritas penghindaran yakni **Cyber Truck, Wall, Oil, Mud**, dengan prioritas pemakaian **Lizard** untuk menembus *obstacles* atau dengan perpindahan jalur balap. Apabila ditemukan *obstacles* dengan jenis serupa di jalur lainnya, bot akan berpindah ke jalur dengan *obstacles* yang relatif lebih sedikit, selanjutnya apabila kondisi perbandingan jumlah *obstacles* masih sama / jalur perpindahan bebas dari *obstacles*, bot akan berpindah ke jalur dengan jumlah *power-up* yang lebih banyak. Kasus dimana fungsi seleksi terakhir (perbandingan *power-up*) tidak menemukan solusi akan menyebabkan bot berpindah ke jalur tengah (*lane 2/3*).

### 3.2.2 *Greedy by Speed* (Alternatif Solusi 2)

Bot akan berusaha mencapai kemungkinan kecepatan tertinggi dengan memanfaatkan command **ACCELERATE** setiap kondisi memungkinkan. Kondisi memungkinkan yang dimaksud adalah ketika jumlah *obstacles* minimum, maka akan dilakukan **ACCELERATE** tanpa mempertimbangkan keberadaan *powerups*, kecuali *powerups* **BOOST** untuk membantu objektif greedy method yakni mencapai kecepatan maksimal. Runtutan prioritas **ACCELERATE** ini akan berakhir apabila kondisi damage bot melebihi batas yang telah ditentukan untuk mencegah *boosting* yang tidak efektif, sehingga bot akan masuk ke fase **FIX** terlebih dahulu, kemudian melanjutkan runutan percepatan lagi.

### 3.2.3 *Greedy by Power-ups*

Bot akan berusaha untuk mencari posisi yang paling menguntungkan untuk mendapatkan *power-ups* sebanyak-banyaknya, kemudian melakukan serangan sebisa mungkin kepada lawan. Strategi ini diawali dengan mengecek kuantitas *powerups* dari berbagai lane kemudian memilih lane dengan jumlah *powerups* tertinggi. Setelah terkumpul *powerups* selanjutnya bot akan mempertimbangkan posisi lawan balap dan bergerak ke posisi terbaik untuk melakukan serangan dengan *powerups* yang sudah dikumpulkan tadi. Apabila dalam posisi kalah, *powerups* yang diprioritaskan untuk digunakan adalah EMP untuk memberhentikan lawan untuk fungsi penyerangan, sedangkan pada posisi menang *powerups* **OIL\_POWER** dan **TWEET** dengan tujuan untuk menghadang lawan dan berpotensi mengakibatkan damage bagi lawan. Kedua strategi tersebut dibarengi dengan penggunaan **LIZARD** untuk menghindari *obstacles* yang ada, barulah mengimplementasikan algoritma penghindaran.

### 3.2.4 *Greedy by benefit*

Strategi ini merupakan gabungan implementasi antara penggunaan strategi penghindaran *greedy by obstacles* dan juga strategi *greedy by Powerups* (pada bagian pengambilannya). Definisi dari *benefit* ini adalah selisih antara jumlah *powerups* dan jumlah *obstacles* yang dideteksi. Strategi *greedy by benefit* ini memungkinkan Bot untuk berusaha mengambil jalur dengan tingkat *benefit* paling tinggi dengan pemikiran untuk meminimalkan jumlah *obstacles* yang mungkin dihadapi sembari mengumpulkan *powerups* yang dapat dipakai pada *round* selanjutnya. Perbandingan selanjutnya yang dilakukan adalah perbandingan penyerangan untuk memastikan *powerups* serangan yang dipakai mengenai lawan.

### 3.3 Analisis Efisiensi dan Efektivitas Solusi *Greedy*

#### 3.3.1 Alternatif Solusi 1

Tujuan utama dari alternatif metode *greedy* ini adalah meminimalisir damage dan pengurangan score yang diterima oleh Bot dari obstacles yang ada. Kelebihannya adalah damage berbanding terbalik dengan maximum speed yang dimiliki oleh bot sehingga dengan meminimalisir damage, secara tidak langsung Bot akan tetap berhasil me-maintain kecepatan tinggi yang dia miliki. Alternatif ini juga imbang dalamantisipasi pengurangan skor sehingga selain dapat menjaga kecepatan maksimum, akumulasi skor yang didapat di akhir permainan juga relatif cukup tinggi. Adapun kekurangan pada alternatif ini adalah seringkali menyia-nyiakan kemampuan untuk menyerang atau menggunakan powerup. Misalnya ada situasi yang cocok untuk menggunakan EMP dan mengeluarkan Cybertweet untuk menyerang lawan, maka bot akan lebih memilih untuk menghindari obstacle di depan apabila ada meskipun hanya sebuah mud atau oil. Dengan demikian, mengorbankan kemampuan menyerang demi menghindari obstacle tanpa mempertimbangkan apakah menghindar lebih layak daripada menyerang pada situasi tersebut.

#### 3.3.2 Alternatif Solusi 2

Tujuan utama dari alternatif metode *greedy* ini adalah memastikan Bot tetap pada kecepatan tinggi dengan harapan mencapai garis finis terlebih dahulu. Kelebihannya adalah Bot tetap me-maintain speed untuk selalu fokus pada kecepatan tinggi, tetapi alternatif ini memiliki banyak kendala terutama pada bagian *powerups*. Bot seringkali menyia-nyiakan kesempatan untuk mengambil powerups yang memang cukup dibutuhkan seperti **TWEET** dan **EMP** yang seringkali memutarbalikkan kondisi pertandingan. Selain itu, seringkali **ACCELERATE** command berujung pada tabrakan *obstacles* apabila dari strateginya kurang mempertimbangkan dan memprediksi apakah pada *range* blocks kecepatan baru terdapat *obstacles* atau tidak.

#### 3.3.3. Alternatif Solusi 3

Tujuan utama dari alternatif metode *greedy* ini adalah memanfaatkan keberadaan *powerups* semaksimal mungkin, sembari memaksimalkan penyerangan terhadap lawan guna mengakibatkan lawan semakin tertinggal di belakang karena *constraint damage* dan *obstacles*. Metode ini memiliki keuntungan yaitu memastikan lawan terserang habis-habisin sehingga menghalau gerakan lawan untuk menyalip posisi kemenangan Bot atau di sisi lain menghambat lawan untuk memperbesar jarak kemenangan dengan menggunakan **EMP** untuk menghentikan pergerakan lawan sementara. Namun, karena implementasinya yang fokus dengan keberadaan *powerups* dan orientasi posisi lawan, seringkali keberadaan *obstacles* menjadi prioritas kedua sehingga mengakibatkan Bot mengalami tabrakan dengan *obstacles* yang ada. Selain itu, pada kondisi tertinggal jauh, Bot juga berusaha untuk mencari *powerups* di saat kondisi berkaitan Bot seharusnya fokus untuk mengejar ketertinggalan terlebih dahulu, kondisi ini seringkali mengakibatkan *margin* kekalahan semakin besar untuk periode tertentu.

#### 3.3.4. Alternatif Solusi 4

Tujuan utama dari alternatif metode *greedy* ini adalah melakukan penggabungan strategi 1 dan 3, yakni approach penghindaran *obstacles* dan utilisasi penggunaan *powerups* dengan harapan didapat pilihan yang memaksimalkan jumlah *powerups* yang dapat diperoleh dan meminimalkan jumlah *obstacles* yang terkena. Secara umum, strategi ini bekerja dengan baik pada berbagai kasus dan berhasil menciptakan pergerakan yang serupa dengan alternatif solusi pertama, tetapi terdapat beberapa kasus pojok (dan di

beberapa pertandingan sering terjadi) dimana terjadi bias pada penjumlahan *benefit* (*powerups* – *obstacles*) yang dicanangkan dari awal. Misalkan pada kasus dengan lane dengan jumlah **MUD** sebanyak 4 dan jumlah *powerups*, maka pemilihan akan menjadi bias karena *benefit* dari lane bersangkutan adalah 0, padahal damage dari tabrakan *obstacles* yang terjadi dapat memberikan efek yang lebih merugikan dibanding jumlah *powerups* yang tersedia. Seringkali, kasus seperti ini mengakibatkan posisi kemenangan yang sudah diraih hilang karena Bot terpaksa melakukan pemberhentian untuk command **FIX** agar aman untuk melaju ke babak selanjutnya.

### 3.4 Strategi Greedy yang dipilih

Dari seluruh solusi *greedy* yang tersedia, kelompok kami akhirnya memutuskan untuk memilih algoritma *greedy by obstacle* sebagai fokus utama. Namun, karena implementasinya yang menurut kami kurang efisien dan efektif, maka kami memutuskan untuk menggabungkannya dengan *greedy by powerups*.

Pada strategi ini, bot mobil akan terlebih dahulu mengecek nilai *damage* dari mobil. Apabila melebihi 2, maka mobil akan melakukan *fix*. Hal ini dilakukan agar kecepatan maksimum dari mobil tetap tinggi sehingga mobil dapat berjalan, *boost* dapat digunakan, dan dapat dilakukan akselerasi. Kemudian, bot akan menggunakan *boost* apabila punya dan lajur aman. *Lizard* juga digunakan untuk mengurangi adanya *damage* dari *obstacle*. Selain itu, ada EMP juga untuk menyerang lawan. Kemudian, baru bot akan mulai menghindari keberadaan *powerups* dengan menggunakan command **TURN\_LEFT**, **TURN\_RIGHT**, dan **LIZARD**. Ketika menghindari *obstacle*, algoritma akan mengarahkan bot untuk memilih jalur yang lebih aman dan memiliki *powerups* lebih banyak untuk diambil. Hal ini dilakukan agar bot dapat terhindar dari *damage* dan mendapatkan skor yang lebih besar. Kemudian, setelah menghindari *obstacle*, baru algoritma melakukan *approach* terhadap penggunaan *tweet* dan *oil*. Lalu, baru algoritma akan mempertimbangkan jalur yang lebih memberikan *benefit* dan memprioritaskan jalur tengah agar memiliki lebih banyak opsi pindah jalur. Setelah itu, baru algoritma fokus pada penggunaan command **ACCELERATE** apabila tidak terdapat hambatan dalam jangkauan mobil hingga kecepatan baru.

Pertimbangan dan konsiderasi dalam memilih solusi ini adalah keefektivitasan dan keefisiensiannya. Solusi ini cukup efektif dalam membawa kita memenangkan permainan yang memiliki objektif utama mencapai garis *finish* lebih cepat. Kemudian, apabila tiba bersamaan, maka kecepatan yang akan dibandingkan. Lalu, jika kecepatan sama, baru mengecek skor. Dengan mengimplementasikan *boost* dan *greedy by obstacle*, mobil diharapkan dapat mencapai garis *finish* dengan lebih cepat. Namun, apabila bot mencapai garis *finish* secara bersamaan dengan kecepatan yang sama, maka penyelamatnya adalah *greedy by powerups* yang berfungsi untuk memperbanyak skor permainan. Kompleksitas algoritma ini dapat dikatakan cukup besar. Namun, pada permainan Overdrive ini, kompleksitas algoritma tidak terlalu berpengaruh.

## BAB IV

### IMPLEMENTASI DAN PENGUJIAN

#### 4.1 Implementasi Program dalam *Game Engine*

Implementasi algoritma greedy pada program kami terdapat pada bagian file bot.java yang tersedia pada folder Java dalam folder starter-bot. Di dalam file tersebut, terdapat sebuah fungsi pemanggilan utama yaitu fungsi run. Kami menuliskan implementasi algoritma greedy kami dalam fungsi run tersebut. Baris yang dimulai dengan “//” adalah komentar yang bukan bagian dari pseudocode, hanya untuk memperjelas.

```
function run(input gamestate : gamestate) → Command

Kamus Lokal
myCar, opponent : Car
Tweet : Command
Lanepos, rlane, l lane : int
rBlocks, lBlocks, blocks : array of objects
countPowerUp, countPowerUpLeft, countPowerUpRight : int
isCT, isCTLeft, isCTRight : boolean
Lane : Lane
countObstacle, countObstacleLeft, countObstacleRight : int
LeftBen, SelfBen, RightBen : int
kiriaman, kananaman : boolean
kiriadapower, kananadapower : boolean

Algoritma
// Mengambil data player dan opponent
myCar ← gameState.player
opponent ← gameState.opponent

// 1. LOGIKA UTAMA
// Mengambil data blocks di depan (di lane sebelah kiri dan kanan)
lanepos ← myCar.position.lane
rlane ← 0
llane ← 0
if (lanepos > 1 and lanepos < 4) then
    rlane ← lanepos + 1
    llane ← lanepos - 1
else if (lanepos == 1) then
    rlane ← lanepos + 1
    llane ← lanepos
else
    llane ← lanepos - 1
    rlane ← lanepos

// Menghitung jumlah powerup dari tiap lane
countPowerUp ← 0
countPowerUpLeft ← 0
countPowerUpRight ← 0
for (tiap block kedepan dalam jangkauan speed myCar) do
    if (blocks memiliki power up) then
        countPowerUp ← countPowerUp + 1
    if (rBlocks memiliki power up) then
```



```
        countPowerUpRight ← countPowerUpRight + 1
        if (lBlocks memiliki power up) then
            countPowerUpLeft ← countPowerUpLeft + 1

// Mendeteksi CyberTruck dari tiap lane
isCT ← false
isCTLeft ← false
isCTRight ← false
for (tiap block dalam lane kedepan dalam jangkauan maxspeed) do
    if (lane kosong atau mencapai finish lane) then
        break loop
    if (ada cyberTruck dalam lane) then
        isCT ← true

// Pengecekan CT pada lane kiri dan kanan
for (tiap block kanan dalam lane kedepan dalam jangkauan maxspeed) do
    if (lane kanan kosong atau mencapai finish lane) then
        break loop
    if (ada cyberTruck dalam lane kanan) then
        isCTRight ← true

for (tiap block kiri dalam lane kedepan dalam jangkauan maxspeed) do
    if (lane kiri kosong atau mencapai finish lane) then
        break loop
    if (ada cyberTruck dalam lane kiri) then
        isCTLeft ← true

// Menghitung jumlah obstacle dari tiap lane
countObstacle ← 0
countObstacleLeft ← 0
countObstacleRight ← 0
for (tiap blocks kedepan dalam jangkauan speed myCar) do
    if (block depan memiliki obstacle) then
        if (obstacle berupa wall) then
            countObstacle ← countObstacle + 2
        else
            countObstacle ← countObstacle + 1
    if (block kanan memiliki obstacle) then
        if (obstacle berupa wall) then
            countObstacleRight ← countObstacleRight + 2
        else
            countObstacleRight ← countObstacleRight + 1
    if (block kiri memiliki obstacle) then
        if (obstacle berupa wall) then
            countObstacleLeft ← countObstacleLeft + 2
        else
            countObstacleLeft ← countObstacleLeft + 1
if (isCT) then
    countObstacle ← countObstacle + 2
if (isCTLeft) then
    countObstacleLeft ← countObstacleLeft + 2
if (isCTRight) then
    countObstacleRight ← countObstacleRight + 2

// Hitung benefit dari tiap lane
LeftBen ← countPowerUpLeft - countObstacleLeft
```

```
SelfBen ← countPowerUp - countObstacle

RightBen ← countPowerUp - countObstacleRight

// 2. MEMBENARKAN MOBIL APABILA RUSAK
if (damage myCar ≥ 2) then
    → FIX

// 3. MENGGUNAKAN POWERUPS BOOST, LIZARD, DAN EMP
if (punya boost and tidak ada wall and tidak ada CT and damage myCar < 2 and
sedang state myCar tidak sedang ngeboost) then
    → BOOST

if (ada obstacle di block depan) then
    → LIZARD

if ((opponent.lane = lanepos or opponent.lane = llane or opponent.lane = rrlane)
and myCar dibelakang opponent) then
    → EMP

// Mencari Jalan dengan obstacle tersedikit ketika jalan menggunakan boost
if (state myCar sedang ngeboost) then
    if (lanepos = 1) then
        if (countObstacleRight < countObstacle and not isCTRight and tidak ada
wall di kanan) then
            → TURN_RIGHT
        if (lanepos = 2 or lanepos = 3) then
            if (countObstacleLeft < countObstacle) then
                if (countObstacleRight < countObstacle and not isCTRight and
tidak ada wall di kanan) then
                    → TURN_RIGHT
                if (countObstacleRight < countObstacle) then
                    if (countObstacleLeft < countObstacle and not isCTLeft and tidak
ada wall di kiri) then
                        → TURN_LEFT
                    if (lanepos = 4) then
                        if (countObstacleLeft < countObstacle and not isCTLeft and tidak ada
wall di kiri) then
                            → TURN_LEFT

// 4. MENGHINDARI OBSTACLE (HANYA BERLAKU APABILA SPEED != 0
// URUTAN PRIORITAS MENGHINDARI : CT, WALL, OIL, MUD
if (speed myCar > 0) then
    if (isCT) then
        // kasus 1
        if (lanepos = 1) then
            if (punya powerup lizard) then
                → LIZARD
            else
                → TURN_RIGHT

        // kasus 2
        if (lanepos = 2 or lanepos = 3) then
            // kasus 2.1. ada wall di kanan dan kiri
            if (ada wall di kiri and ada wall di kanan) then
                if (punya power up lizard) then
```

```
        → LIZARD
    else if (RightBen ≥ LeftBen) then
        → TURN_RIGHT
    else
        → TURN_LEFT

// Kasus 2.2 kanan saja yang ada wall
if (rBlocks ada wall) then
    if (punya powerup Lizard) then
        → LIZARD
    else
        → TURN_LEFT

// Kasus 2.3 kiri saja yang ada wall
if (lBlocks ada wall) then
    if (punya powerup Lizard) then
        → LIZARD
    else
        → TURN_RIGHT

// Kasus 2.4 kiri dan kanan tidak ada wall
if (lBlocks tidak ada wall and rBlocks tidak ada wall) then
    if (LeftBen ≥ RightBen) then
        → TURN_LEFT
    else
        → TURN_RIGHT

// kasus 4
if (lanepos = 4) then
    if (punya powerup lizard) then
        → LIZARD
    else
        → TURN_LEFT

// KENA WALL
if (blocks depan dalam jangkauan speed ada wall) then
    // SEGMENT LURUS, PAKAI LIZARD
    if (punya powerup lizard) then
        → LIZARD

    if (lanepos = 1) then
        // 1. TIDAK ADA CT ATAUPUN WALL DI KANAN
        if (wall kanan tidak ada wall ataupun CT) then
            if (countObstacleRight < countObstacle) then
                → TURN_RIGHT
        // 2. ADA WALL DI LANE KANAN
        if (ada wall di kanan) then
            if (countObstacle > countObstacleRight) then
                → TURN_RIGHT
            else if (countObstacle = countObstacleRight) then
                if (countPowerUp < countPowerUpRight) then
                    → TURN_RIGHT

// LANE 2 ATAU 3
```

```
if (lanepos = 2 or lanepos = 3) then

    if (countObstacleLeft = 0) then
        if (countObstacleRight = 0) then
            if (countPowerUpLeft ≥ countPowerUpRight) then
                → TURN_LEFT
            else
                → TURN_RIGHT
        else
            → TURN_LEFT

    else
        if (countObstacleRight = 0) then
            → TURN_RIGHT
        else
            if (countObstacleLeft > countObstacleRight) then
                → TURN_RIGHT
            else
                → TURN_LEFT

// LANE 4
if (lanepos = 4) then
    if (tidak ada wall ataupun CT di lane kiri) then
        if (countObstacleLeft < countObstacle) then
            → TURN_LEFT

    // 2. ADA WALL DI KANAN
    if (tidak ada wall di kanan) then
        // 2.1
        if (countObstacle > countObstacleLeft) then
            → TURN_LEFT
        else if (countObstacle = countObstacleLeft) then
            if (countPowerUp < countPowerUpLeft) then
                → TURN_RIGHT

// Kena Mud → speed berkurang, skor berkurang 3, damage + 1
// Kena Oil → speed berkurang, skor berkurang 4, damage + 1
if (blocks depan mengandung mud ataupun spill oil) then
    // PRIORITAS LIZARD 1
    if (punya powerup lizard) then
        → LIZARD
// pindah2 lane. Kalau lane sebelah ada obstacles, mending nabrak mud
kiriaman ← true //dua variabel ini ngecek kiri kanan aman apa ga
kananaman ← true
if (lanepos = 1) then //kasus paling atas, hanya bisa turn right
    if (lanekanan ada obstacles) then
        kananaman ← false
    if (kananaman) then
        → TURN_RIGHT

if (lanepos = 4) then //kasus paling bawah, hanya bisa turn left
    if (lanekiri ada obstacles) then
        kiriaman ← false
    if (kiriaman) then
        → TURN_LEFT
if (lanepos = 2 or lanepos = 3) then
```

```
    if (block kanan ada obstacles) then  
        kananaman ← false  
    if (block kiri ada obstacles) then  
        kiriaman ← false  
    if (kiriaman and not kananaman) then  
        → TURN_LEFT  
    if (not kiriaman and kananaman) then  
        → TURN_RIGHT  
    if (kiriaman and kananaman) then  
        kiriadapower ← false  
        kananadapower ← false  
        if (blocks kanan ada powerup) then  
            kananadapower ← true  
        if (blocks kiri ada powerup) then  
            kiriadapower ← true  
        if (kiriadapower and kananadapower) then  
            → TURN_LEFT  
        if (not kiriadapower and not kananadapower) then  
            → TURN_RIGHT  
        if (kiriadapower and kananadapower) then  
            if (lanepos = 2) then  
                → TURN_RIGHT  
            if (lanepos = 3) then  
                → TURN_LEFT  
  
    // 5. MENGGUNAKAN POWERUP APABILA PUNYA  
    if (blocks depan mengandung mud ataupun spill oil) then  
        → TWEET  
    else if (punya powerup oil and opponent.lane = lanepos and myCar di depan  
    opponent) then  
        → OIL  
  
    // 6. PINDAH JALUR (PRIORITAS TENGAH)  
    if (speed mycar > 0) then  
        if (lanepos = 1 and countObstacleRight = countObstacle) then  
            if (countPowerUpRight > countPowerUp) then  
                → TURN_RIGHT  
        else if (lanepos = 2) then  
            if (countObstacle = 0 and countObstacleRight = 0) then  
                if (countPowerUpRight > countPowerUp) then  
                    → TURN_RIGHT  
            else  
                if (RightBen > SelfBen) then  
                    → TURN_RIGHT  
        else if (lanepos = 3) then  
            if (countObstacle = 0 and countObstacleLeft = 0) then  
                if (countPowerUpLeft > countPowerUp) then  
                    → TURN_LEFT  
            else  
                if (LeftBen > SelfBen) then  
                    → TURN_LEFT  
  
    // 7. MELAKUKAN AKSELERASI BILA AMAN  
    // 0 ke 3  
    if (myCar.speed = 0) then  
        → ACCELERATE
```

```
// 1 ke 3
if (myCar.speed = 1) then
    if (tidak ada obstacles dalam jangkauan 3 blocks depan) then
        → ACCELERATE

// 3 atau 5 ke 6
if (myCar.speed = 3 or myCar.speed = 5) then
    if (tidak ada obstacles dalam jangkauan 6 blocks depan) then
        → ACCELERATE

// 6 ke 8
if (myCar.speed = 6) then
    if (tidak ada obstacles dalam jangkauan 8 blocks depan) then
        → ACCELERATE

// 8 ke 9
if (myCar.speed = 8) then
    if (tidak ada obstacles dalam jangkauan 15 blocks depan) then
        → ACCELERATE

// 8. JIKA LEWAT SEMUA, DO NOTHING
→ NOTHING
```

## 4.2 Penjelasan Struktur Data

### 4.2.1 Car

Struktur data yang merepresentasikan objek utama dari permainan *Overdrive* ini, yakni mobil balap. Berisikan **id** bertipe integer, struktur data **Position**, kecepatan yang dinyatakan dalam **speed** dengan tipe integer, struktur data **State**, **damage** dengan tipe integer, larik berisi struktur data **Powerups**, status **boosting** dengan tipe boolean, dan **boostCounter** dengan tipe integer.

### 4.2.2 Position

Struktur data yang merepresentasikan posisi suatu objek dalam permainan *Overdrive*, direpresentasikan sebagai koordinat pada bidang 2 dimensi. Berisikan **lane** atau ekuivalen sumbu y dengan tipe integer dan **block** yang ekuivalen sumbu x dengan tipe integer juga.

### 4.2.3 State

Struktur data yang merepresentasikan keadaan bot mobil maupun perlombaan pada *round* yang bersangkutan, terdiri 16 atribut yakni **ACCELERATING** atau dalam keadaan percepatan, **READY** yang diartikan sebagai kesiapan untuk memulai lomba, **NOTHING** atau tidak dalam state lainnya, **TURNING\_RIGHT** yaitu sedang berpindah jalur ke arah kanan, **TURNING\_LEFT** yaitu sedang berpindah jalur ke arah kiri, **HIT\_MUD** atau kondisi sedang tertabrak *obstacle* Mud, **HIT\_OIL** atau kondisi sedang tertabrak *obstacle* Oil, **DECELERATING** atau kondisi perlambatan kecepatan, **PICKED\_UP\_POWERUP** atau kondisi sedang mengambil suatu *powerup*, **USED\_BOOST** atau kondisi penggunaan *powerup* jenis Boost, **USED\_OIL** atau kondisi penggunaan *powerup* jenis Oil Power, **USED\_LIZARD** atau kondisi penggunaan *powerup* jenis Lizard, **USED\_TWEET** atau kondisi penggunaan

*powerup* jenis Tweet, **HIT\_WALL** atau representasi keadaan ketika tertabrak *obstacle* Wall, **HIT\_CYBER\_TRUCK** atau representasi keadaan ketika tertabrak *obstacle* Cyber Truck, **FINISHED** atau kondisi permainan telah usai.

#### 4.2.4 Powerups

Struktur data yang merepresentasikan jenis-jenis *powerup* yang tersedia di permainan *Overdrive*, berisi *serialized name* yaitu **BOOST, OIL, TWEET, LIZARD, dan EMP**.

#### 4.2.5 Command

Struktur data yang memungkinkan bot untuk mengeksekusi aksi-aksi dalam perlombaan balapnya. Diimplementasikan dengan *string rendering*. Untuk jenis-jenis command sendiri dibagi menjadi fungsi percepatan dan perlambatan, fungsi *boost*, fungsi perpindahan jalur, fungsi *do nothing*, fungsi penggunaan *power-up oil*, fungsi penggunaan *tweet*, fungsi penggunaan *Lizard*, fungsi penggunaan EMP, dan fungsi perbaikan mobil.

#### 4.2.6 Terrain

Struktur data yang merepresentasikan jenis objek yang terdapat di arena balap mobil. Terdiri atas enumerasi variabel yang terdiri atas **EMPTY** sebagai block kosong, **MUD** sebagai block berisi lumpur, **OIL\_SPILL** sebagai block yang berisi tumpahan minyak, **OIL\_POWER** sebagai block yang berisi *power-up* berupa barel minyak, **FINISH** sebagai garis finish, **BOOST** sebagai block berisi *power-up* percepatan maksimal, **WALL** sebagai block yang berisi dinding, **LIZARD** sebagai block yang berisi *power-up* berjenis Lizard, **TWEET** block yang berisi *power-up* yang berjenis Tweet (untuk spawn Cyber Truck), dan **EMP** sebagai block yang berisi *power-up* EMP.

#### 4.2.7 Lane

Struktur data yang merupakan *wrapper* dari struktur-struktur data lain terkait dengan jalur balap, terdiri atas struktur data **Position**, struktur data **Terrain**, variabel bertipe integer **occupiedByPlayerId** yang merepresentasikan keterisian block oleh bot pemain, dan yang terakhir variabel bertipe boolean untuk merepresentasikan keberadaan Cyber Truck yaitu **cyberTruck**.

#### 4.2.8 Direction

Struktur data yang merepresentasikan arah gerak dari bot. Terdapat 4 arah gerak, yaitu *forward*, *backward*, *left*, dan *right*. Struktur data ini juga memiliki fungsi bernama *Direction* yang digunakan pada *command* **TURN\_LEFT** dan **TURN\_RIGHT**.

### 4.3 Analisis Desain Solusi Pada Setiap Pengujian

Pengujian 1	<p style="text-align: right;"><b>Winner : A – s4Rap</b></p> <p><b>s4Rap :</b>  score → 410  y → 2  x → 1491  damage → 0  <b>reference-bot :</b></p>
-------------	---

	score → -139 y → 4 x → 668 damage → 5
Pengujian 2	<p style="text-align: right;"><b>Winner : A – s4Rap</b></p> <b>s4Rap :</b> score → 416 y → 1 x → 1492 damage → 0 <b>reference-bot :</b> score → -45 y → 4 x → 592 damage → 5
Pengujian 3	<p style="text-align: right;"><b>Winner : A – s4Rap</b></p> <b>s4Rap :</b> score → 441 y → 3 x → 1497 damage → 1 <b>reference-bot :</b> score → -102 y → 4 x → 591 damage → 2
Pengujian 4	<p style="text-align: right;"><b>Winner : A – s4Rap</b></p> <b>s4Rap :</b> score → 412 y → 2 x → 1496 damage → 0 <b>reference-bot :</b> score → -125 y → 4 x → 623 damage → 5
Pengujian 5	<p style="text-align: right;"><b>Winner : A – s4Rap</b></p> <b>s4Rap :</b> score → 518 y → 4 x → 1491 damage → 0 <b>reference-bot :</b> score → -132 y → 4 x → 588 damage → 5

Berdasarkan hasil pengujian yang dilakukan dengan melawan *reference-bot* yang diberikan oleh Entelect, implementasi algoritma *greedy* dalam program bot ini dapat memenangkan 5 dari 5



permainan yang dicoba. Rata-rata perbedaan skornya adalah 548, sedangkan rata-rata perbedaan jaraknya adalah 881. Oleh karena itu, angka kemenangan dari bot kami adalah 100%.

Implementasi menghindari *obstacle* dapat dikatakan cukup optimal karena berhasil mengurangi angka *damage*. Dengan mengurangi angka *damage*, nilai maksimum kecepatan menjadi lebih tinggi dan tidak perlu dilakukan *fix* berulang-ulang. Selain itu, fokus selanjutnya terhadap *powerups* juga memperbesar peluang kemenangan karena ada *boost* yang dapat mempercepat laju bot, *lizard* untuk menghindari *damage*, dan EMP, *tweet*, juga *oil* yang dapat menyerang lawan.

Meskipun angka kemenangan terhadap bot adalah 100%, nilai ini belum tentu akurat karena pengujian yang hanya dilakukan sebanyak 5x dan lawan main yang merupakan *reference-bot* dengan implementasi kode dan strategi yang dapat dikatakan terlalu sedikit. Oleh karena itu, dapat disimpulkan juga bahwa kemenangan / kekalahan dari bot ini juga menyesuaikan dengan strategi lawan main. Apabila bot lawan melakukan implementasi yang mendalam dan lebih detail, besar kemungkinan bahwa bot ini akan kalah karena strategi algoritma *greedy* sendiri dibuat untuk fokus terhadap apa yang dihadapi sekarang, bukan menciptakan strategi yang dapat memprediksi gerak-gerik lawan dan langkah-langkah yang akan diambil selanjutnya.

## BAB V

### KESIMPULAN DAN SARAN

#### 5.1 Kesimpulan

Pada tugas besar mata kuliah IF2211 Strategi Algoritma yang ke-1 ini, telah berhasil diimplementasikan hasil pembelajaran algoritma *greedy* dalam bentuk permainan Overdrive yang merupakan *challenge* yang diberikan oleh Entelect pada tahun 2020.

Pendekatan menggunakan algoritma *greedy* sendiri diimplementasikan pada command run pada bot.java yang berada pada folder starter-bots. Setelah keseluruhan bot dibuat, bot kemudian digunakan dan diuji pada beberapa permainan. Dalam **5** permainan, bot berhasil memenangkan **5** permainan. Oleh karena itu, dapat disimpulkan bahwa persentase kemenangan adalah **100%**.

Dengan demikian, kelompok dapat menyimpulkan bahwa dengan mengerjakan Tugas Besar II IF2211 Strategi Algoritma Semester 2 Tahun 2021/2022 ini, dapat diketahui bahwa memenangkan suatu permainan Overdrive dapat dilakukan dengan cara mengimplementasikan algoritma *greedy*.

#### 5.2 Saran

Tugas besar IF2211 Strategi Algoritma Semester 2 Tahun 2021/2022 menjadi salah satu proses pembelajaran bagi kelompok dalam menerapkan ilmu-ilmu yang dipelajari pada perkuliahan ataupun dengan melakukan eksplorasi materi secara mandiri. Berikut ini adalah saran dari kelompok untuk pihak-pihak yang ingin melakukan atau mengerjakan hal serupa.

- Dibutuhkan *brainstorming* yang cukup dalam mengerjakan tugas ini. Dengan melakukan *brainstorming*, strategi yang digunakan akan menjadi lebih efisien.
- Lakukan analisis setiap ada pergantian masif terhadap algoritma. Hal ini dapat membantu para *programmer* untuk mencari letak kesalahan dan hal-hal lainnya yang sekiranya masih bisa diimplementasikan.
- Dalam mengerjakan suatu tugas secara berkelompok, penting untuk memiliki strategi serta distribusi tugas yang baik dan efisien. Cara penulisan kode dan kemampuan menulis komentar menjadi hal yang sangat penting dalam mengerjakan kode pemrograman secara berkelompok. Dengan adanya komentar pada kode, anggota lain pada kelompok dapat memahami cara kerja suatu kode dengan lebih cepat. Kemampuan tersebut juga didukung dengan adanya version control system (VCS) yang baik untuk digunakan oleh programmer dalam membuat sebuah kode pemrograman secara bersamaan. Kelompok sangat menyarankan penggunaan 'Github' untuk digunakan sebagai version control system (VCS) dalam pengerjaan tugas besar, maupun pada pembuatan program yang lainnya.

### **LINK PENTING**

Github Repository : [https://github.com/saulsayerz/Tubes1\\_S4RaP.git](https://github.com/saulsayerz/Tubes1_S4RaP.git)

Video : <https://youtu.be/-SNraBg572s>

### **DAFTAR PUSTAKA**

<https://www.techopedia.com/definition/16931/greedy-algorithm>

[https://en.wikipedia.org/wiki/Greedy\\_algorithm#:~:text=A%20greedy%20algorithm%20is%20used,algorithm%20for%20decision%20tree%20construction.](https://en.wikipedia.org/wiki/Greedy_algorithm#:~:text=A%20greedy%20algorithm%20is%20used,algorithm%20for%20decision%20tree%20construction.)

<https://www.geeksforgeeks.org/greedy-algorithms/>