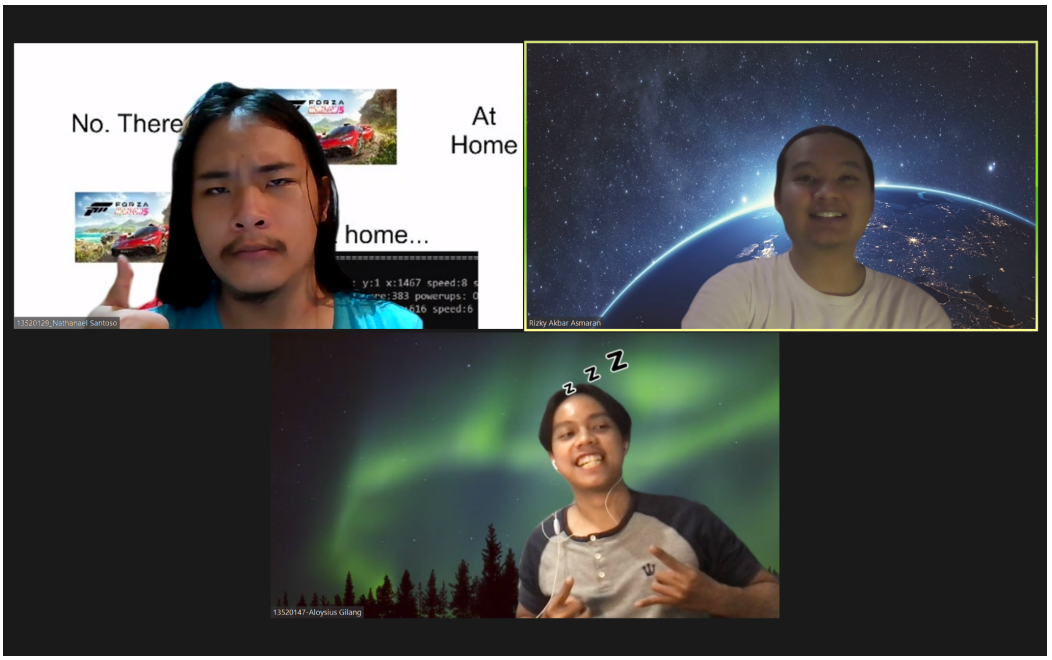
Laporan Tugas Besar 1 IF 2211

Strategi Algoritma

Penerapan Algoritma Greedy dalam bot permainan Entelect Challenge 2020 : Overdrive



Disusun Oleh:

Fast and Curious

Nathanael Santoso	**13520129**
Aloysius Gilang Pramudya	**13520147**
Rizky Akbar Asmaran	**13520111**

Institut Teknologi Bandung

Bandung

2022

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.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.

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 power ups. 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 IF2211.

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

Algoritma Greedy mempunyai beberapa elemen sebagai berikut;

1. Himpunan Kandidat, C : berisi kandidat yang akan dipilih pada setiap langkah.
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.
5. Fungsi Kelayakan (*feasible*) : memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi
6. Fungsi Obyektif : memaksimumkan atau meminimumkan.

Dengan menggunakan elemen diatas, 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 objektif.

Berikut merupakan skema umum dari algoritma greedy:

```
function greedy(C : himpunan_kandidat) -> himpunan_solusi
{mengembalikan solusi dari persoalan optimasi dengan
algoritma greedy}

Deklarasi
  x : kandidat
  S : himpunan_solusi

Algoritma:
  S <- {}
  while (not SOLUSI(S)) and (C != {}) do
    x <- SELEKSI(C)
    C <- C - {x}
    if LAYAK (S U {x}) then
      S <- S U {x}
    endif
  endwhile
  {SOLUSI(S) or C = {}}

  if SOLUSI(S) then
    return S
  else
    write('tidak ada solusi')
  endif
```

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

Akan tetapi, solusi optimum global pada algoritma greedy ini belum tentu merupakan solusi yang optimum (terbaik), bisa jadi merupakan solusi sub-optimum atau pseudo-optimum. Hal tersebut diakibatkan karena algoritma greedy tidak beroperasi secara menyeluruh terhadap semua kemungkinan solusi yang ada dan terdapat beberapa fungsi seleksi yang berbeda sehingga harus memilih fungsi yang tepat jika kita ingin algoritma menghasilkan solusi yang optimal.

2.2 Cara Kerja Program Secara Umum

agar dapat menjalankan permainan, kita hanya perlu mengeksekusi perintah file run.bat yang tersedia pada folder starter-pack. akan tetapi, sebelum kita melakukan hal tersebut, kita perlu menyesuaikan file game-config.json dan game-runner-config.json apabila kita ingin mengubah konfigurasi permainan yang dijalankan. Dengan file game-config.json. kita bisa mengatur panjang lintasan balapan dan jumlah lane yang ingin digunakan. Akan tetapi, di game-runner-config.json kita dapat menentukan bot yang ingin kita tandingkan. hal itu kita dapat lakukan dengan menuliskan path folder dimana bot terletak.

pada Folder bot sekurang-kurangnya berisi sebuah file bernama bot.json. bot.json ini bertugas untuk memberi tahu kepada game-engine beberapa informasi penting seperti nama dari file bot yang kita punya, bahasa pemrograman yang dipakai, lokasi dari file bot, dan nickname yang kita gunakan di dalam permainan. khusus untuk program kali ini kita menggunakan bahasa pemrograman JAVA sehingga file bot yang diterima oleh game-engine adalah sebuah file jar hasil dari kompilasi kode java. Setelah mengatur beberapa konfigurasi dan menentukan player yang akan bertanding. Kita tinggal menggunakan Command "*make run"* pada command prompt, lalu permainan akan berjalan.

BAB III
APLIKASI STRATEGI GREEDY

1. Proses Mapping

Elemen-elemen Algoritma Greedy by *speed state difference* dan *damage point* pada permasalahan optimasi bot game Overdrive ini adalah sebagai berikut :

A. Himpunan Kandidat (C)

a. NOTHING

Perintah ini memungkinkan mobil tidak melakukan apa-apa di ronde tersebut. Kecepatan mobil tidak akan berubah dan mobil akan tetap berada di jalur yang sama.

b. ACCELERATE

Perintah ini meningkatkan kecepatan mobil ke speed state selanjutnya. Mobil akan tetap berada di jalur yang sama.

c. DECELERATE

Perintah ini mengurangi kecepatan mobil ke speed state sebelumnya. Mobil akan tetap berada di jalur yang sama.

d. TURN_LEFT

Perintah ini mengubah jalur mobil player ke jalur sebelah kirinya. Pada blok pertama mobil akan bergerak ke kiri dan bergerak kedepan sepanjang sisa ronde.

e. TURN_RIGHT

Perintah ini mengubah jalur mobil player ke jalur sebelah kanannya. Pada blok pertama mobil akan bergerak ke kanan dan bergerak ke depan sepanjang sisa ronde.

f. USE_BOOST

Perintah ini digunakan untuk memakai power up boost yang didapatkan mobil. Mobil akan melaju dengan kecepatan MAXIMUM_SPEED selama 5 ronde.

g. USE_OIL

Perintah ini digunakan untuk memakai power up oil yang didapatkan mobil. Mobil menumpahkan oli tepat di bawahnya. Musuh yang terkena tumpahan oli akan melambat ke speed state sebelumnya.

h. USE_TWEET

Perintah ini digunakan untuk memakai power up tweet yang didapatkan mobil. Player menempatkan cyber truck di posisi tertentu. Musuh yang menabrak cyber truck akan berhenti selama satu ronde dan kecepatannya berubah ke speed state pertama.

i. USE_LIZARD

Perintah ini digunakan untuk memakai power up lizard yang didapatkan mobil. Mobil akan meloncat pada ronde tersebut. Semua power ups dan obstacle akan diabaikan.

j. USE_EMP

Perintah ini digunakan untuk memakai power up EMP yang didapatkan mobil. Mobil menembakan EMP ke depan dan jalur sebelahnya. Musuh yang terkena EMP akan berhenti selama ronde tersebut dan kecepatanya berubah ke 3.

k. FIX

Perintah ini digunakan untuk memperbaiki mobil. Perintah ini mengurangi damage point mobil sebesar 2. Mobil akan berhenti sepanjang ronde.

B. Himpunan Solusi (S)

Himpunan Solusi berisi kandidat perintah yang dipilih pada setiap ronde.

C. Fungsi Solusi

Memeriksa apakah perintah yang dipilih pada setiap ronde merupakan perintah yang valid. Jika tidak maka akan mendefault ke akselerasi.

D. Fungsi Seleksi

Goal dari permainan ini adalah agar mobil mencapai garis finish secepat mungkin dan jika mobil mencapai garis finish bersamaan dengan musuh maka

mobil harus memiliki kecepatan atau nilai yang lebih besar. Maka fungsi seleksinya adalah memilih perintah dengan keuntungan akselerasi terbesar (speed gain). Hal ini dilakukan dengan cara mengambil jalur yang akan ditempuh berdasarkan perintah yang ingin diperkirakan dan menghitung nilai speed state akhir jika mobil melewati jalur tersebut. Kemudian fungsi akan menghitung perbedaan speed state akhir dengan yang awal dan juga perbedaan damage akhir dengan yang awal. Oleh karena itu, perlu diperhatikan juga bahwa setiap perintah tidak memiliki bobot nilai yang konstan pada setiap ronde. Penilaian terhadap setiap perintah dipengaruhi oleh kondisi mobil dan jalur yang dilalui.

Selain itu, keberadaan power ups pada jalur akan memberikan bobot nilai pada tiap perintah yang ditambahkan pada akhir perhitungan kecepatan. Bobot ini ditentukan secara heuristik dengan memperkirakan keuntungan yang diperoleh jika mengambil power ups tersebut dan memakainya. Pembobotan terdapat pada tabel di bawah

Terrain	Bobot Speed State Gain	Bobot Damage
OIL_POWER	+1	-
TWEET	+1	-2
BOOST	+2	-
EMP	+1	-2
LIZARD	+1	-2

Tabel 1. Pembobotan Power Up pada Terrain

Ada pula pembobotan yang diberikan pada command tertentu secara heuristik untuk menghindari bot memilih pilihan yang merugikan jika keadaan akan membawa kerugian, namun fungsi seleksi standar tidak bisa mengetahui kerugian tersebut. Pembobotan terdapat pada tabel di bawah.

Command	Bobot Speed State Gain	Bobot Damage
FIX	-5, jika damage = 0 +5, jika damage >= 2	-
USE_BOOST	-5, jika Speed State = 5	-
USE_OIL	+1, jika mempunyai Power Up OIL dan musuh sedang di tidak jauh di belakang. -5, selain yang di atas.	-1, jika mempunyai Power Up OIL dan musuh sedang di tidak jauh di belakang. 1, selain yang di atas.
USE_TWEET	+SpeedStateMusuh - 1, Jika mempunyai Power Up TWEET. -5, selain yang di atas.	-2, Jika mempunyai Power Up TWEET. 2, selain yang di atas.
USE_EMP	+SpeedStateMusuh - 1, Jika mempunyai Power Up EMP dan musuh akan kena EMP jika ditembakkan -5, selain yang di atas.	-2, Jika mempunyai Power Up EMP dan musuh akan kena EMP jika ditembakkan 2, selain yang di atas.
USE_LIZARD	+2, Jika mempunyai Power LIZARD dan Speed State = 5. -5, Jika tidak mempunyai Power Up LIZARD.	-

Tabel 2. Pembobotan Command

Contoh kasus 1

Mobil bergerak dengan kecepatan 6 (speed state 2), damage point 0, dengan peta jalur pada ronde tersebut adalah sebagai berikut :

			Mud			
Mobil		Wall				

Maka pembobotan setiap perintah adalah sebagai berikut :

- ACCELERATE : Speed state difference -1, Damage taken 2

  Pertama kecepatan mobil akan naik ke speed state selanjutnya (+1) namun karena di jalur tersebut terdapat wall maka kecepatan mobil akan berubah ke speed state 1 (-2). Mobil juga akan mendapat damage point sebesar 2.
- TURN LEFT: Speed state difference -1, Damage taken 1

  Mobil berbelok ke kiri berhasil menghindari wall namun tetap melewati mud, maka kecepatan mobil turun ke state sebelumnya (-1) dan mendapat damage point sebesar 1.
- TURN RIGHT: Speed state difference 0, Damage taken 0
- FIX : Speed state difference -2, Damage taken 0

  Namun karena damage point mobil 0 perintah ini tidak berpengaruh dan player akan rugi satu ronde.
- USE BOOST : speed state difference -1, Damage taken 2

  Mobil akan memakai boost yang akan menaikan speed menjadi BOOST SPEED yaitu (15), namun karena di jalur tersebut terdapat wall maka kecepatan mobil akan langsung berubah ke speed state 1 (-4). Mobil juga akan mendapat damage point sebesar 2.
- USE OIL : speed state difference -6, Damage Taken 2
- USE TWEET : speed state difference -6, Damage Taken 2
- USE LIZARD : speed state difference -6, Damage Taken 2
- USE EMP : speed state difference -6, Damage Taken 2

Jika poin speed state difference di beberapa perintah sama, maka bandingkan poin damage taken. Berdasarkan pembobotan diatas maka perintah dengan poin terbaik adalah TURN RIGHT

Contoh Kasus 2

Mobil bergerak dengan kecepatan 8 (speed state 3), damage point 0, dengan peta jalur pada ronde tersebut adalah sebagai berikut :

Mobil		Truck				
			Oil			

Maka pembobotan setiap perintah adalah sebagai berikut :

- ACCELERATE : Speed state difference -3, Damage taken 2
  Pertama kecepatan mobil akan naik ke speed state selanjutnya (+1) menjadi Maximum Speed (9) namun karena di jalur tersebut terdapat cybertruck maka kecepatan mobil akan berubah ke speed state 1 (-3). Mobil juga akan mendapat damage point sebesar 2.
- TURN LEFT: Speed state difference 0, Damage taken 0
  Mobil berbelok ke kiri berhasil menghindari cybertruck dan tidak mempengaruhi speed state dan damage karena tidak terdapat *obstacle.*
- TURN RIGHT: Speed state difference -1, Damage taken 1
  Mobil berbelok ke kanan berhasil menghindari cybertruck namun tetap melewati oil, maka kecepatan mobil turun ke state sebelumnya (-1) dan mendapat damage point sebesar 1.
- FIX : Speed state difference 0, Damage taken 0
  Namun karena damage point mobil 0 perintah ini tidak berpengaruh dan player akan rugi satu ronde.
- USE BOOST : speed state difference -4, Damage taken 2

Mobil akan memakai boost yang akan menaikan speed menjadi BOOST SPEED yaitu (15), namun karena di jalur tersebut terdapat Truck maka kecepatan mobil akan langsung berubah ke speed state 1 (-4). Mobil juga akan mendapat damage point sebesar 2.

- USE OIL : speed state difference -6, Damage Taken 2
- USE TWEET : speed state difference -6, Damage Taken 2
- USE LIZARD : speed state difference -6, Damage Taken 2
- USE EMP : speed state difference -6, Damage Taken 2

Jika poin speed loss atau speed gain beberapa perintah sama, maka bandingkan poin damage taken. Berdasarkan pembobotan diatas maka perintah dengan poin terbaik adalah TURN LEFT

Contoh Kasus 3

Mobil bergerak dengan kecepatan 6 (speed state 2), damage point 0, dengan peta jalur pada ronde tersebut adalah sebagai berikut :

		Truck				
Mobil						
		Mud				

Maka pembobotan setiap perintah adalah sebagai berikut :

- ACCELERATE : Speed state difference +1, Damage taken 0
  Kecepatan mobil akan naik ke speed state selanjutnya (+1) menjadi speed state 3 (8) dan tanpa mengalami damage cost karena tidak terdapat *obstacle* sama sekali.
- TURN LEFT: Speed state difference -1, Damage taken 2
  Mobil berbelok ke kiri namun mobil akan melewati cybertruck, maka kecepataan mobil akan turun ke state 1 (3) dan mendapatkan damage point sebesar 2.
- TURN RIGHT: Speed state difference -1, Damage taken 1

Mobil berbelok ke kanan namun mobil akan melewati mud, maka kecepatan mobil turun ke state sebelumnya (-1) dan mendapat damage point sebesar 1.

- FIX : Speed state difference 0, Damage taken 0
  Namun karena damage point mobil 0 perintah ini tidak berpengaruh dan player akan rugi satu ronde.
- USE BOOST : Speed state difference +3, Damage taken 0
  Mobil akan memakai boost yang akan menaikan speed menjadi BOOST SPEED yaitu (15) dan juga tidak terdapat obstacle sama sekali di jalur tersebut sehingga mobil akan terus bergerak dengan kecepatan maksimal dan tidak akan terkena damage point.
- USE OIL : Speed state difference -5, Damage taken 0
- USE TWEET : Speed state difference -5, Damage taken 0
- USE LIZARD : Speed state difference -5, Damage taken 0
- USE EMP : Speed state difference -5, Damage taken 0

Jika poin speed loss atau speed gain beberapa perintah sama, maka bandingkan poin damage taken. Berdasarkan pembobotan diatas maka perintah dengan poin terbaik adalah USE BOOST

Contoh Kasus 4

Mobil bergerak dengan kecepatan 6 (speed state 2), damage point 2, dengan peta jalur pada ronde tersebut adalah sebagai berikut :

Mobil		Mud				
			Oil			

Maka pembobotan setiap perintah adalah sebagai berikut :

- ACCELERATE : Speed state difference -1, Damage taken 1
  Kecepatan mobil akan naik ke speed state selanjutnya (+1) menjadi speed state 3 (8) namun karena di jalur tersebut terdapat Mud maka

kecepatan mobil akan berubah ke speed state 1 (-1). Mobil juga akan mendapat damage point sebesar 1.

- TURN LEFT: Speed state difference 0, Damage taken 0
  Mobil berbelok ke kiri berhasil menghindari cybertruck dan tidak mempengaruhi speed state dan damage karena tidak terdapat *obstacle.*
- TURN RIGHT: Speed state difference -1, Damage taken 1
  Mobil berbelok ke kanan namun mobil akan melewati Oil, maka kecepatan mobil turun ke state sebelumnya (-1) dan mendapat damage point sebesar 1.
- FIX : Speed state difference 0, Damage taken -2
  Karena dalam kondisi mobil memiliki damage sebesar 2, maka akan dilakukan FIX yang tidak akan memengaruhi speed state dan mengurangi speed state sebesar 2 .
- USE BOOST : Speed state difference -1, damage taken 1
  Mobil akan memakai boost yang akan menaikan speed menjadi BOOST SPEED yaitu (15), namun karena di jalur tersebut terdapat mud maka kecepatan mobil akan langsung berubah ke speed state 2 (-1). Mobil juga akan mendapat damage point sebesar 1.
- USE OIL : Speed loss -6, Damage taken 1
- USE TWEET : Speed loss -6, Damage taken 1
- USE LIZARD : Speed loss -6, Damage taken 1
- USE EMP : Speed loss -6, Damage taken 1

Jika poin speed loss atau speed gain beberapa perintah sama, maka bandingkan poin damage taken. Berdasarkan pembobotan diatas maka perintah dengan poin terbaik adalah FIX

Contoh Kasus 5

Mobil bergerak dengan kecepatan 6 (speed state 2), damage point 2, dengan peta jalur pada ronde tersebut adalah sebagai berikut :

		Mud	Mud			
Lawan		Mobil				
			Oil			

Maka pembobotan setiap perintah adalah sebagai berikut :

- ACCELERATE : Speed loss 1, Damage taken 1

  Kecepatan mobil akan naik ke speed state selanjutnya (+1) menjadi speed state 3 (8) namun karena di jalur tersebut terdapat Mud maka kecepatan mobil akan berubah ke speed state 1 (-1). Mobil juga akan mendapat damage point sebesar 1.
- TURN LEFT: Speed loss 0, Damage taken 0

  Mobil berbelok ke kiri berhasil menghindari cybertruck dan tidak mempengaruhi speed state dan damage karena tidak terdapat *obstacle.*
- TURN RIGHT: Speed loss 1, Damage taken 1

  Mobil berbelok ke kanan namun mobil akan melewati Oil, maka kecepatan mobil turun ke state sebelumnya (-1) dan mendapat damage point sebesar 1.
- FIX : Speed loss 0, Damage taken -2

  Karena dalam kondisi mobil memiliki damage sebesar 2, maka akan dilakukan FIX yang tidak akan memengaruhi speed state dan mengurangi speed state sebesar 2 .
- USE BOOST : speed loss 1, damage taken 1

  Mobil akan memakai boost yang akan menaikan speed menjadi BOOST SPEED yaitu (15), namun karena di jalur tersebut terdapat mud maka kecepatan mobil akan langsung berubah ke speed state 2 (-1). Mobil juga akan mendapat damage point sebesar 1.
- USE OIL : Speed loss 0, Damage taken 0
- USE TWEET : Speed loss 0, Damage taken 0
- USE LIZARD : Speed loss 0, Damage taken 0
- USE EMP : Speed loss 0, Damage taken 0

Jika poin speed loss atau speed gain beberapa perintah sama, maka bandingkan poin damage taken. Berdasarkan pembobotan diatas maka perintah dengan poin terbaik adalah FIX

E. Fungsi Kelayakan

Memeriksa apakah perintah yang dipilih pada setiap ronde merupakan perintah yang tidak menghasilkan perintah DO_NOTHING. Contohnya memeriksa apakah power up tersedia jika hendak digunakan, memeriksa apakah mobil di jalur paling kiri/kanan jika hendak berbelok, dan memeriksa apakah mobil berada pada speed state maksimum jika hendak accelerate. Jika akan menghasilkan DO_NOTHING, maka diberikan pembobotan agar tidak melakukan aksi tersebut. Pembobotan yang diberikan sudah dicantumkan pada tabel 2 bagian fungsi seleksi.

F. Fungsi Objektif

Memastikan perintah yang digunakan memperoleh nilai akselerasi yang paling besar untuk mencapai garis finish.

2. Eksplorasi Alternatif Solusi

Berikut adalah beberapa alternatif solusi greedy yang dapat digunakan dalam permainan Overdrive :

A. Greedy by block gain

Greedy by block gain adalah strategi memenangkan balapan berdasarkan besar perpindahan mobil. Pada setiap ronde, bot akan memilih perintah yang menghasilkan perpindahan terbesar.

B. Greedy by power ups

Greedy by power ups adalah strategi memenangkan balapan berdasarkan penggunaan power ups. Pada setiap ronde, bot akan memilih jalur yang memiliki power ups terbanyak dan mengabaikan obstacle. begitu memiliki

power up bot akan memakainya guna memperlambat dan menghentikan bot musuh.

C. Greedy by damage point

Greedy by avoiding obstacle adalah strategi memenangkan balapan berdasarkan obstacle. Pada setiap ronde, bot akan memilih jalur yang tidak memiliki / paling sedikit *obstacle* untuk menghindari *damage point.* Bot juga akan melakukan FIX setiap kali mendapat *damage point*.

3. Analisis Efisiensi

Berikut adalah analisis efisiensi untuk solusi yang dipilih dan alternatif solusi yang dipaparkan :

A. Secara umum, strategi *greedy by speed state difference* dan *damage point* kurang efisien untuk diimplementasikan dibanding alternatif yang lain karena bot harus menghitung total poin setiap *obstacle* dan *power up* pada jalur di depan, kanan, dan kiri bot. Bot juga harus memperhatikan keadaan bot musuh pada pembobotan.

B. Strategi *greedy by block gain* lebih efisien dari pada *greedy by speed state difference* dan *damage point* karena bot hanya perlu menghitung nilai setiap obstacle pada jalur di depan, kanan, dan kiri bot dan tidak perlu memerhatikan power ups dan keadaan bot musuh.

C. Strategi *greedy by power ups* juga lebih efisien dari pada *greedy by speed state difference* dan *damage point* karena bot hanya perlu menghitung nilai setiap power ups pada jalur di depan, kanan, dan kiri bot dan tidak perlu memerhatikan obstacle dan keadaan bot musuh.

D. Strategi *greedy by damage point* lebih efisien dari pada *greedy by speed state difference* dan *damage point* karena bot hanya perlu menghitung nilai setiap obstacle pada jalur di depan, kanan, dan kiri bot dan tidak perlu memerhatikan keadaan bot musuh.

4. Analisis Efektivitas

Berikut adalah analisis efektivitas untuk solusi yang dipilih dan alternatif solusi yang dipaparkan :

A. Strategi *greedy by speed state difference* dan *damage point* sangat efektif untuk diimplementasikan dibanding alternatif yang lain karena bot tidak hanya berfokus pada mencari perintah dengan akselerasi terbaik tapi juga berusaha untuk memperlambat dan menghentikan musuh dan mengusahakan perbaikan jiga dibutuhkan. Obstacle dan power ups juga menjadi variabel dalam pembobotan sehingga menghasilkan solusi yang terbaik.

B. Strategi *greedy by block gain* cukup efektif untuk diimplementasikan karena bot hanya berfokus pada mencari perintah dengan akselerasi terbaik. Namun bot bisa saja mengabaikan power ups yang memiliki efek besar hanya karena *block gain* pada jalur tersebut lebih kecil.

C. Strategi *greedy by power ups* kurang efektif untuk diimplementasikan karena bot tidak terlalu memperhatikan akselerasi dan mengabaikan *obstacle* yang ada.

D. Strategi *greedy by damage point* juga kurang efektif untuk diimplementasikan karena bot tidak terlalu memperhatikan akselerasi dan mengabaikan *power ups* yang ada.

5. Strategi Greedy yang dipilih

Langkah-langkah Strategi Greedy yang diimplementasikan pada bot adalah sebagai berikut :

1) Untuk setiap perintah yang tersedia dalam game, lakukan pembobotan selisih *speed state* dan *damage taken* berdasarkan keadaan jalur mobil. Pembobotan dilakukan dengan menghitung total efek yang dihasilkan perintah, *power ups*, dan *obstacle* pada jalur di depan, kiri, dan kanan

mobil. Pembobotan pemakaian power ups selain *speed boost* dilakukan dengan menghitung nilai *speed state difference* dan *damage taken* mobil musuh jika terkena efek *power ups*.

2) Pilih perintah yang mempunyai nilai *speed state difference* maksimum. Jika terdapat nilai *speed state difference* yang sama maka pilih perintah dengan nilai *damage taken* paling minimum.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

1. Implementasi Algoritma Greedy

```
Program Bot
KAMUS

use command, entities, enums

Constant SSMin  : integer = 0
Constant SS1    : integer = 3
Constant SS2    : integer = 6
Constant SS3    : integer = 8
Constant SSMax  : integer = 9
Constant SSB    : integer = 15
Constant SSI    : integer = 5

function getSpeedStateFromSpeed(Speed: integer) -> integer
{- mengembalikan speedstate (0-5) berdasarkan kecepatan -}

function getSpeedFromSpeedState(SpeedState: integer) -> integer
{- mengembalikan speed (0-15) berdasarkan state kecepatan -}

function getMaxSpeedStateFromDamage(damage: integer) -> integer
{- mengembalikan speedstate (0-5) maksimum berdasarkan damage -}

function ACCELERATE() -> Command
{- mengembalikan Objek Command Accelerate -}
function DECELERATE() -> Command
{- mengembalikan Objek Command Decelerate -}
function LIZARD() -> Command
{- mengembalikan Objek Command Use_Lizard -}
function OIL() -> Command
{- mengembalikan Objek Command Use_Oil -}
function BOOST() -> Command
{- mengembalikan Objek Command Use_Boost -}
function EMP() -> Command
{- mengembalikan Objek Command Use_EMP -}
function FIX() -> Command
{- mengembalikan Objek Command Fix -}
function TWEET() -> Command
{- mengembalikan Objek Command Tweet dengan parameter default -}
function TWEET(lane, block: integer) -> Command
{- mengembalikan Objek Command Tweet dengan parameter lane, block -}
function TURN_RIGHT() -> Command
{- mengembalikan Objek Command Turn_Right -}
function TURN_LEFT() -> Command
{- mengembalikan Objek Command Turn_Left -}

function run(gameState: GameState) -> Command
Kamus Lokal
```

```
    idx, i: integer
    myCar, opponent: Car
    front, left, right: Array of Terrain
ALGORITMA
    myCar <- gameState.player
    opponent <- gameState.opponent

    front <- getBlocksInLane(myCar.position.lane, myCar.position.block + 1,
gameState)

    if (myCar.position.lane != 1) then:
        left <- getBlocksInLane(myCar.position.lane - 1,
myCar.position.block, gameState)
    else:
        left <- null
    endif

    if (myCar.position.lane != 4) then:
        right <- getBlocksInLane(myCar.position.lane + 1,
myCar.position.block, gameState)
    else:
        right <- null
    endif

    PREDACCEL <- PredictState(ACCELERATE, myCar,opponent, front);
    PREDDECEL <- PredictState(DECELERATE, myCar,opponent, front);
    PREDLIZARD <- PredictState(LIZARD, myCar,opponent, front);
    PREDOIL <- PredictState(OIL, myCar,opponent, front);
    PREDBOOST <- PredictState(BOOST, myCar,opponent, front);
    PREDEMP <- PredictState(EMP, myCar,opponent, front);
    PREDFIX <- PredictState(FIX, myCar,opponent, front);
    PREDTWEET <- PredictState(TWEET, myCar,opponent, front);
    PREDLEFT <- PredictState(TURN_LEFT, myCar,opponent, left);
    PREDRIGHT <- PredictState(TURN_RIGHT, myCar,opponent, right);
    PREDCOMMANDS <- [PREDACCEL, PREDDECEL, PREDLIZARD, PREDOIL, PREDBOOST,
PREDEMP, PREDFIX, PREDTWEET, PREDLEFT, PREDRIGHT]

    BESTPRED <- null
    i traversal [0..9]
        PRED <- PREDCOMMANDS[i]
        if (BESTPRED = null) then:
            BESTPRED <- PRED
        else if (PRED != null) then:
            if (BESTPRED.get(0) < PRED.get(0)) then:
                BESTPRED <- PRED
            else if (BESTPRED.get(0).equals(PRED.get(0)) and
BESTPRED.get(1) > PRED.get(1)) then:
                BESTPRED <- PRED
            endif
        endif

    i traversal [0..9]
        if (BESTPRED = PREDCOMMANDS[i]) then:
            idx <- i
        endif

    depend on (idx)
```

```
        idx = 0: -> ACCELERATE()
        idx = 1: -> DECELERATE()
        idx = 2: -> LIZARD()
        idx = 3: -> OIL()
        idx = 4: -> BOOST()
        idx = 5: -> EMP()
        idx = 6: -> FIX()
        idx = 7: -> TWEET(opponent.position.lane, opponent.position.block +
opponent.speed + 1)
        idx = 8: -> TURN_LEFT()
        idx = 9: -> TURN_RIGHT()
    else: -> ACCELERATE()
    endif

function hasPowerUp(ToCheck: PowerUps, PList: Array of PowerUps) -> boolean
KAMUS LOKAL
    i: integer
ALGORITMA
    i traversal [0..PList.Neff - 1]
        if (PList[i] = ToCheck) then:
            -> true
        endif
    -> false

function getBlocksInLane(lane, block: integer, gameState: GameState) ->
Array Of Terrain
KAMUS LOKAL
    map: Array of Array of Lane
    blocks: Array of Terrain
    startBlock, i: integer
    laneList: Array of Lane
ALGORITMA
    map <- gameState.lanes
    startBlock <- map[0][0].position.block
    laneList <- map[lane - 1]
    i traversal [max(block - startBlock, 0)..block - startBlock + 19]
        if (laneList[i] != null and laneList[i].terrain != Terrain.FINISH)
then:
            if (laneList[i].isOccupiedByCyberTruck = true) then:
                blocks[i] <- Terrain.WALL
            else:
                blocks[i] <- laneList[i].terrain
            endif
        endif
    -> blocks

function isEnemyDirectlyBehind(myCar, oppCar: Car) -> boolean
KAMUS LOKAL
    posDiff: integer
ALGORITMA
    posDiff <- myCar.position.block - oppCar.position.block
    -> (myCar.position.lane = oppCar.position.lane and posDiff > 0 and
posDiff <= oppCar.speed)

function isEnemyEMPable(myCar, oppCar: Car) -> boolean
KAMUS LOKAL
ALGORITMA
```

```
    -> (abs(myCar.position.lane - oppCar.position.lane) <= 1 and
oppCar.position.block > myCar.position.block)

function getNewStateFromCommand(command: Command, SSDMG: Array of integer,
myCar, opponent: Car) -> Array of integer
KAMUS LOKAL
    MaxSS: integer
ALGORITMA
    MaxSS <- getMaxSpeedStateFromDamage(SSDMG[1])

    if (command = ACCELERATE()) then:
        if (SSDMG[0] != 5) then:
            -> [min(min(SSDMG[0] + 1, 4), MaxSS), SSDMG[1], 0, 0]
        else:
            -> [min(5, MaxSS), SSDMG[1], 0, 0]
        endif
    else if (command = DECELERATE()) then:
        -> [max(SSDMG[0] - 1, 0), SSDMG[1], 0, 0]
    else if (command = FIX()) then:
        depend on (SSDMG[1]):
            SSDMG[1] = 0:
                -> [SSDMG[0], 0, -5, 0]
            SSDMG[1] = 1:
                -> [SSDMG[0], 0, 0, 0]
            SSDMG[1] = 2:
                -> [SSDMG[0], 0, 5, 0]
        else:
            -> [SSDMG[0] , SSDMG[1] - 2, 5, 0]
    else if (command = BOOST()) then:
        if (hasPowerUp(PowerUps.BOOST, myCar.powerups) and SSDMG[0] < 5)
then:
            -> [min(5, MaxSS), SSDMG[1], MaxSS - 5, 0]
        else:
            -> [SSDMG[0], SSDMG[1], -5, 0]
        endif
    else if (command = OIL()) then:
        if (hasPowerUp(PowerUps.OIL, myCar.powerups) and
isEnemyDirectlyBehind(myCar, opponent)) then:
            -> [SSDMG[0] , SSDMG[1], 1, -1]
        else:
            -> [SSDMG[0] , SSDMG[1], -5, 1]
        endif
    else if (command = TWEET()) then:
        if (hasPowerUp(PowerUps.TWEET, myCar.powerups)) then:
            -> [SSDMG[0] , SSDMG[1],
(getSpeedStateFromSpeed(opponent.speed) - 1), -2]
        else:
            -> [SSDMG[0] , SSDMG[1], -5, 2]
        endif
    else if (command = EMP()) then:
        if (hasPowerUp(PowerUps.TWEET, myCar.powerups) and
isEnemyEMPable(myCar, opponent)) then:
            -> [SSDMG[0] , SSDMG[1],
(getSpeedStateFromSpeed(opponent.speed) - 1), -2]
        else:
            -> [SSDMG[0] , SSDMG[1], -5, 2]
        endif
```

```
    else if (command = LIZARD()) then:
        if (hasPowerUp(PowerUps.LIZARD, myCar.powerups) and myCar.speed =
SSB) then:
            -> [SSDMG[0] , SSDMG[1], 2, 0]
        else if (!hasPowerUp(PowerUps.LIZARD, myCar.powerups)) then:
            -> [SSDMG[0] , SSDMG[1], -5, 0]
        endif
    endif
    -> [SSDMG[0], SSDMG[1], 0, 0]

function getNewStateFromTerrain(terrain: Terrain, SSDMG: Array of integer)
-> Array of integer
KAMUS LOKAL
    ss, dmg, bobotacc, bobotdmg: integer
ALGORITMA
    ss <- SSDMG[0]
    dmg <- SSDMG[1]
    bobotacc <- SSDMG[2]
    bobotdmg <- SSDMG[3]
    if (terrain = Terrain.WALL) then:
        ss <- min(ss, 1)
        dmg <- min(dmg + 2, 5)
    else if (terrain = Terrain.MUD or terrain = Terrain.OIL_SPILL) then:
        ss <- max(0, ss - 1)
        dmg <- min(dmg + 1, 5)
    else if (terrain = Terrain.OIL_POWER) then:
        bobotacc <- bobotacc + 1
    else if (terrain = Terrain.TWEET) then:
        bobotacc <- bobotacc + 1
        bobotdmg <- bobotdmg - 2
    else if (terrain = Terrain.BOOST) then:
        bobotacc <- bobotacc + 2
    else if (terrain = Terrain.EMP) then:
        bobotacc <- bobotacc + 1
        bobotdmg <- bobotdmg - 2
    else if (terrain = Terrain.LIZARD) then:
        bobotacc <- bobotacc + 1
        bobotdmg <- bobotdmg - 2
    endif
    -> [ss, dmg, bobotacc, bobotdmg]

function PredictState(command: Command, myCar, opponent: Car, Lanes: Array
of Terrain) -> Array of integer
KAMUS LOKAL
    State: Array of Integer
    LaneCopy: Array of Terrain
    i: integer
ALGORITMA
    if (Lanes = null) then:
        -> null
    endif

    State <- [getSpeedStateFromSpeed(myCar.speed), myCar.damage, 0, 0]
    State <- getNewStateFromCommand(command, State, myCar, opponent)
    LaneCopy <- Lanes.subList(0, min(getSpeedFromSpeedState(State[0]),
Lanes.size()))
```

```
    if (command = LIZARD()) then:
        if (LaneCopy.size() != 0) then:
            State <- getNewStateFromTerrain(LaneCopy[LaneCopy.size() - 1],
State)
        endif
    else if (!command = FIX()) then:
        i traversal [0..LaneCopy.Neff - 1]
            State <- getNewStateFromTerrain(LaneCopy[i], State)

    -> [State[0] - getSpeedStateFromSpeed(myCar.speed) + State[2], State[1]
- myCar.damage + State[3])

function fromJson(state: SEQFILE) -> GameState
{- Returns gamestate from Json -}

gameState: GameState
state: SEQFILE

ALGORITMA UMUM
gamestate <- fromJson(state)
output(run(gamestate))
```

2. **Penjelasan Struktur Data**

   Adapun struktur data yang kelompok kamu gunakan terbagi menjadi 3, yaitu command, entities, enum yang disertakan dengan bot.java dan main.java

   1. Command

      struktur data yang berfungsi untuk menyimpan kelas kelas yang berhubungan dengan prosedur game-enginenya.

      a. AccelerateCommand : Command untuk meningkatkan speed state sebesar satu tingkat state (sampai MAXIMUM_SPEED) dan mobil tidak berpindah jalur.
      b. BoostCommand : Command untuk menggunakan power up *Boost* yang meningkatkan speed state menjadi maksimal.
      c. ChangeLaneCommand : Command yang digunakan untuk membelokkan mobil dalam permainan, kiri dan kanan yang mengakibatkan pengurangan speed sebesar -1.
      d. DecelerateCommand : Command yang digunakan untuk mengurangi speed state sebesar satu tingkat (sampai MINIMUM_SPEED).
      e. DoNothingCommand : Command yang digunakan untuk tidak melakukan apa apa sehingga tidak berpengaruh terhadap speed state dan tetap berada di jalur awal.
      f. EmpCommand : Command yang digunakan untuk memakai power up EMP yang dapat memengaruhi mobil lawan.
      g. FixCommand : Command yang digunakan untuk memperbaiki mobil dari damage yang telah diterima, sehingga damage dari mobil akan berkurang sebesar dua.

h. LizardCommand : Command yang digunakan untuk meletakkan power up lizard.
i. OilCommand : Command yang digunakan untuk menumpakan oil dibelakang mobil pemain.
j. TweetCommand : Command yang digunakan untuk meletakkan *Cybertruck*.

2. Entites
   Struktur data yang berguna sebagai objek yang terdapat pada gamestate
   a. Car : Objek *Car*.
   b. GameState: Objek yang menyimpan state tertentu mulai dari currentRound, maxRounds, player, opponent dan worldMap.
   c. Lane : Objek yang dilewati mobil pemain sebagai jalur balapan.
   d. Position : Objek posisi yang berupa koordinat.

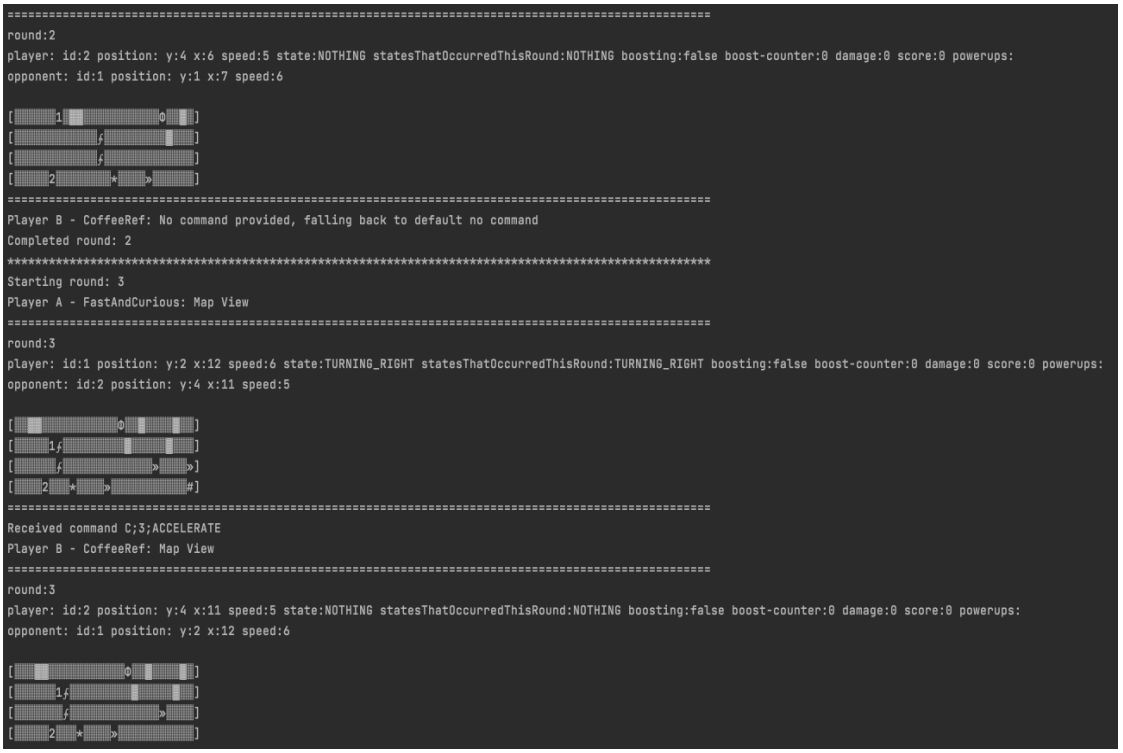
3. Enum
   Struktur data yang berguna sebagai alat iterasi yang dibuat untuk mencocokan setiap kemungkinan yang ada
   a. Direction : Berisi Arahan mobil untuk bergerak
   b. PowerUps : Berisi Power up yang dapat digunakan.
   c. State : Berisi perintah yang bisa diberikan kepada mobil.
   d. Terrain : Berisi deskripsi dari obstacle pada lane.

4. bot.java
   Struktur data yang berisi algoritma bot yang dibuat oleh kelompok kami

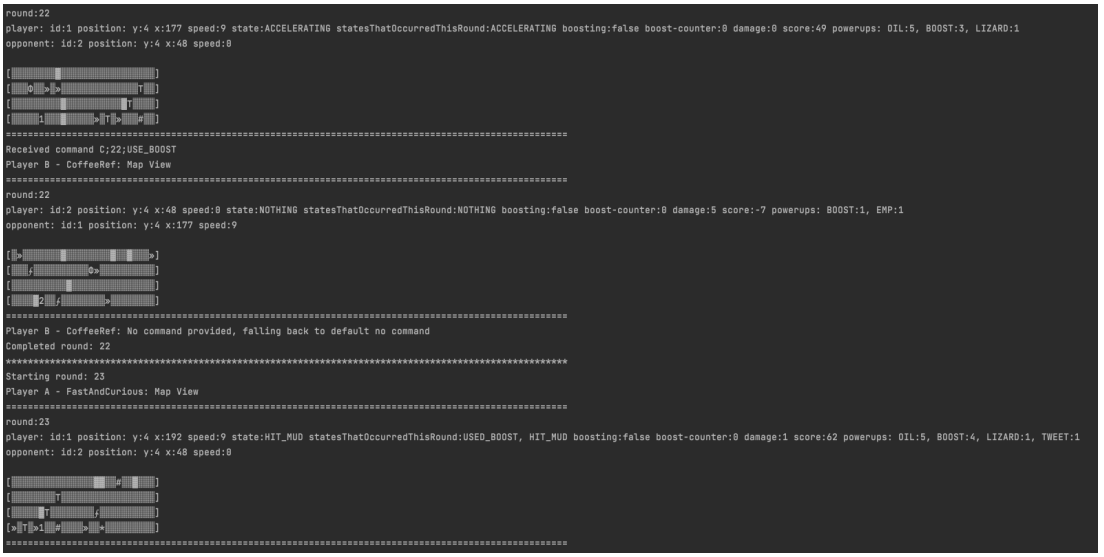
5. main.java.
   Struktur data yang berisi mekanisme pembacaan state dan perintah eksekusi state.

3. Analisis Desain Solusi Algoritma Greedy

Berikut hasil pengujian strategi yang diimplementasikan beserta analisisnya. Pengujian dilakukan dengan melawan bot referensi yang sudah tersedia.



Pada ronde ke-2, bot kita sedang berada jalur paling kiri dengan kecepatan 6. Pada 6 block di depan mobil terdapat dua *obstacle* mud sedangkan 6 block pada jalur kanannya terdapat *power up Lizard.* Berdasarkan strategi greedy yang sudah dibuat maka bot akan mengeluarkan perintah TURN_RIGHT. Pada ronde ini strategi greedy yang dipilih kebetulan menghasilkan nilai optimal.



Pada ronde ke-22, bot kita sedang berada pada jalur paling kanan dengan kecepatan 9. Pada 6 blok di depan dan kiri terdapat *obstacle mud* yang bisa membatalkan efek *power up speed boost*. Namun bot kita tetap menggunakan *power up speed boost* meskipun terdapat *obstacle mud* di depan. Kasus ini merupakan salah satu contoh solusi tidak optimal yang dihasilkan oleh strategi *greedy* yang dibuat. Hal ini terjadi karena bot tidak memperhatikan pembatalan *power up speed boost* jika terkena *obstacle* pada proses pembobotan. Maka dapat disimpulkan bahwa solusi yang dihasilkan oleh strategi greedy yang dibuat oleh tim kami tidak menghasilkan solusi yang optimal.

BAB V
KESIMPULAN, SARAN, DAN REFLEKSI

1. Kesimpulan

Dari tugas besar IF2211 Strategi Algoritma Semester 2 2021/2022 berjudul "Pemanfaatan Algoritma Greedy dalam Aplikasi Permainan *Overdrive"*, kami berhasil membuat sebuah bot dalam permainan *overdrive* yang memanfaatkan algoritma *Greedy*. Strategi *Greedy* yang kami buat tidak menghasilkan solusi yang optimal karena pembobotan yang tidak sempurna pada setiap perintah yang tersedia.

2. Saran

Saran-saran yang dapat kami berikan untuk tugas besar IF2211 Strategi Algoritma Semester 2 2021/2022 adalah:

a. Pembobotan untuk setiap command dan terrain dilakukan dengan lebih matematis yaitu dengan cara mengetes bot, kemudian mengubah angka untuk mendapatkan hasil yang lebih baik dan mengetes lagi, hingga mencapai hasil yang terbaik.
b. Algoritma menambahkan pembobotan untuk posisi musuh (contoh: Jika musuh jauh di belakang, tidak perlu dihalangi lagi dengan power ups dan fokus untuk maju saja)
c. Menulis *source code* dengan lebih efisien agar waktu perhitungan lebih pendek dan bisa memainkan permainan dengan lebih cepat.

3. Refleksi

Materi yang telah didapat dari mata kuliah Strategi Algoritma membuat pembuatan algoritma greedy pada permainan *Overdrive* ini menjadi lebih terbayang dan lebih mudah. Bahasa Java terbilang cukup mudah dan cukup efektif untuk membuat program ini.

Link Github : https://github.com/nart4hire/Tubes1_FastAndCurious.git

Link Video Demo : https://youtu.be/ri3_LE9rb7I

Daftar Pustaka

EntelectChallenge. (n.d.). *EntelectChallenge/2020-overdrive: Main repository for entelect challenge 2020*. GitHub. Retrieved February 18, 2022, from https://github.com/EntelectChallenge/2020-Overdrive.git

Munir, R. (n.d.). *Bahan Kuliah IF2211 strategi algoritma algoritma greedy*. Retrieved February 18, 2022, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf