TUGAS BESAR I

IF2211 STRATEGI ALGORITMA

Pemanfaatan Algoritma Greedy dalam Aplikasi Permainan “Overdrive”



Kelompok 12 - DAMRI

13520002 Muhammad Fikri Ranjabi

13520017 Diky Restu Maulana

13520035 Damianus Clairvoyance Diva P.

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

Daftar Isi

Bab 1
Deskripsi Tugas

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

Pada tugas besar pertama Strategi Algoritma ini, digunakan sebuah *game engine* yang mengimplementasikan permainan Overdrive. *Game engine* dapat diperoleh pada laman berikut.
https://github.com/EntelectChallenge/2020-Overdrive

Buatlah implementasi bot mobil dalam permainan Overdrive dengan menggunakan strategi greedy untuk memenangkan permainan. Untuk mengimplementasikan bot tersebut, mahasiswa disarankan melanjutkan program yang terdapat pada starter-bots di dalam starter-pack pada laman berikut ini.
https://github.com/EntelectChallenge/2020-Overdrive/releases/tag/2020.3.4

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

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

c. DECELERATE
d. TURN_LEFT
e. TURN_RIGHT
f. USE_BOOST
g. USE_OIL
h. USE_LIZARD
i. USE_TWEET
j. USE_EMP
k. FIX 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 berikut.
   https://github.com/EntelectChallenge/2020-Overdrive/blob/develop/game-engine/game-rules.md

Strategi greedy yang diimplementasikan tiap kelompok harus dikaitkan dengan fungsi objektif dari permainan itu sendiri, yaitu memenangkan permainan dengan cara mencapai garis finish lebih awal atau mencapai garis finish bersamaan tetapi dengan kecepatan lebih besar atau memiliki skor terbesar jika kedua komponen tersebut masih bernilai imbang. Salah satu contoh pendekatan greedy yang bisa digunakan (pendekatan tak terbatas pada contoh ini saja) adalah menggunakan powerups begitu ada untuk mengganggu mobil musuh.

Strategi greedy harus dijelaskan dan ditulis secara eksplisit pada laporan, karena akan diperiksa pada saat demo apakah strategi yang dituliskan sesuai dengan yang diimplementasikan. Tiap kelompok dapat menggunakan kreativitas mereka dalam menyusun strategi greedy untuk memenangkan permainan. Implementasi pemain harus dapat dijalankan pada game engine yang telah disebutkan pada spesifikasi tugas besar, serta dapat dikompetisikan dengan pemain dari kelompok lain.

Bab 2
Landasan Teori

A. Dasar Teori Algoritma Greedy

Algoritma *greedy* merupakan metode sederhana yang digunakan untuk memecahkan persoalan optimasi. Dua persoalan optimasi yang dapat diselesaikan dengan algoritma ini adalah *maximization* dan *minimization*.

Secara definisi, algoritma *greedy* adalah algoritma yang memecahkan persoalan secara langkah per langkah (*step by step*). Dengan ini, pada setiap langkahnya akan diambil pilihan terbaik pada saat itu tanpa memperhatikan konsekuensi ke depan dan berharap bahwa setiap langkah yang dipilih menghasilkan optimum lokal dan berakhir dengan optimum global.

Berbagai macam persoalan dapat diselesaikan dengan algoritma *greedy* seperti persoalan penukaran uang, persoalan memilik aktivitas, minimasi waktu dalam sistem, persoalan *knapsack*, pohon merentang minimum, dan lain-lain. Persamaan pendekatan yang digunakan pada seluruh persoalan tersebut yaitu dengan mengambil langkah terbaik yang ada tanpa mempertimbangkan langkah-langkah lain dengan asumsi bahwa langkah yang diambil merupakan solusi terbaik pada saat itu.

B. Cara Kerja Program dan Bot

Program permainan Overdrive adalah permainan balapan antara dua mobil dalam satu jalur yang sama dan berkompetisi untuk mencapai garis finish. Setiap mobil akan dikendalikan oleh bot yang ada. Pada dasarnya, bot dibedakan menjadi *starter-bots* (bot miliki pengguna) dan *reference-bot* (bot AI/lawan).

Bot akan berjalan sepanjang permainan dan komunikasi bot dengan permainan dilakukan melalui interaksi *input/output*. Dalam setiap ronde, bot akan membaca situasi saat itu yang mengandung kondisi peta dan properti dari ronde tersebut. Setelah itu bot akan menjalankan langkah untuk ronde selanjutnya berdasarkan logika yang sudah ada dan *output* langkah yang diambil akan tercatat di permainan untuk selanjutnya ditampilkan.

Pengguna tidak perlu memasukkan perintah pada setiap ronde secara manual karena bot akan otomatis menjalankan command berdasarkan logika yang sudah disiapkan pengguna sejak awal permainan dalam file Bot.Java pada direktori starter-bots.

Program ini secara *default* dapat berjalan melalui Command Line Interface (lihat Bab 4 Implementasi dan Pengujian), dengan keterangan output tersedia pada dokumentasi Overdrive. Akan tetapi, untuk memperoleh pengalaman bermain yang lebih memanjakan mata, sudah tersedia sejumlah visualizer web-based yang dapat digunakan. Pengguna hanya tinggal mengunggah file zip match-logs.

Apabila diinginkan, untuk mempermudah pengembangan strategi, pemain juga dapat mengatur seed map yang sama, yakni dengan menambahkan line “seed: <integer>” pada game-runner-config.json”. Terdapat pula pengaturan max-runtime dan timeout yang mengakibatkan bot otomatis menjalankan command DO NOTHING jika melebihi batas timeout tersebut.

Satu hal kecil lagi, terdapat informasi pengaturan yang tidak tertulis pada dokumentasi Overdrive, yakni menambrak wall mengurangi 5 skor, menabrak Cybertruck 7 skor, dan menerima EMP lawan 7 skor.

C. Implementasi Algoritma Greedy pada Bot

Algoritma Greedy dibuat dalam Bahasa Java untuk diimplementasikan pada bot. File yang menyimpan algoritma ini dapat ditemukan pada direktori berikut.

starter-pack/starter-bot/java/src/main/java/za/co/entelect/challenge/Bot.java

Penulisan kode dilakukan dengan Intellij, sebuah IDE untuk Bahasa Java. Di dalamnya terdapat Maven, sebuah perangkat lunak berupa alat otomatisasi build untuk proyek Java. Maven juga dapat digunakan untuk membangun proyek yang ditulis dalam C#, Ruby, Scala, dan Bahasa lainnya. Kode yang sudah selesai dituliskan selanjutnya di-build ke dalam file .jar. Jika proses build berhasil, file tersebut akan tersimpan di:

starter-pack/starter-bot/java/target/java-starter-bot.jar
starter-pack/starter-bot/java/target/java-starter-bot-with-dependencies.jar

D. Menjalankan Game Engine

Bot yang kami buat dan bot lawan berada pada direktori sebagai berikut.

starter-pack/starter-bots/java
starter-pack/reference-bots/java

Atur isi file game-runner-config.json sesuai direktori penyimpanan kedua bot. Jika sudah sesuai, selanjutnya kita dapat menjalankan permainan (atau mengadu bot kita) dengan bot referensi, klik run.bat (untuk pengguna OS Windows). Selanjutnya akan terbuka jendela command prompt yang menampilkan status pertandingan untuk kedua Bot. Untuk pengguna Linux atau MacOS, jalankan make run di Terminal dalam direktori yang sesuai, dan akan langsung tampil status pertandingan untuk kedua Bot. Pertandingan akan selesai ketika salah satu bot berhasil mencapai garis finish. Pada bagian akhir, akan ditampilkan pemenangnya beserta poin kedua bot.

Hasil pertandingan akan disimpan dalam bentuk file yang dapat ditemukan pada starter-pack/match-logs. Sedikit informasi tambahan, menjalankan game engine Overdrive cukup memakan *resource* CPU dan *storage* data, terutama karena match-logs yang cukup besar tiap pertandingannya (hingga 100 MB), sehingga perlu persiapan sistem yang memadai.

Di dalam direktori tersebut terdapat file GlobalState.json yang menyimpan state kedua bot dalam setiap round. Lalu, terdapat folder yang menyimpan command kedua bot pada setiap round. .

Bab 3
Aplikasi Strategi *Greedy*

3.1 Mapping Persoalan *Overdrive* Menjadi Elemen-Elemen Algoritma Greedy

1. Himpunan Kandidat, *C*: Himpunan command yang dapat digunakan di setiap round.
   C = {NOTHING, ACCELERATE, DECELERATE, TURN_LEFT, TURN_RIGHT, BOOST, OIL, LIZARD, TWEET <lane> <block>, EMP, FIX}
2. Himpunan Solusi, *S*: Himpunan command yang valid untuk digunakan di suatu round.
3. Fungsi Solusi: Command valid yang dipilih berdasarkan strategi yang ada.
4. Fungsi Seleksi: Memilih command berdasarkan kondisi mobil kita dan mobil lawan serta kondisi jalur yang sedang dilewati. Selalu pilih jalur yang memiliki obstacle minimal dan powerups maksimal.
5. Fungsi Kelayakan: Memvalidasi suatu command untuk digunakan di suatu round. Misalnya, ketika ingin menggunakan sebuah powerup, terlebih dahulu dilakukan pengecekan di list powerups yang dimiliki oleh mobil kita.
6. Fungsi Objektif: Memastikan command yang dipilih membawa mobil kepada garis finish dengan urutan prioritas waktu tercepat, kecepatan terbesar, dan score terbesar sesuai dengan peraturan pada game engine.

3.2 Alternatif Solusi Greedy

3.2.1 Alternatif Solusi Pertama

Menganggap bahwa menghalangi lawan merupakan kondisi optimal lokal sehingga command seperti OIL, EMP, dan TWEET harus menjadi command prioritas.

3.2.2 Alternatif Solusi Kedua

Kebalikan dari strategi pertama, yakni menganggap bahwa memfokuskan bot untuk maju dengan kecepatan tertinggi dengan menghindari segala gangguan, serta tidak menghalangi lawan merupakan kondisi optimum lokal sehingga command seperti ACCELERATE, BOOST, LIZARD, TURN_LEFT, dan TURN_RIGHT harus menjadi command prioritas.

3.2.3 Alternatif Solusi Ketiga

Setiap ronde akan selalu dilakukan pengecekan terlebih dahulu pada damage car. Kemudian dilakukan penghitungan rintangan yang ada di depan blok saat ini. Keputusan berikutnya yang diambil akan memperhatikan jumlah rintangan, bot akan memprioritaskan jalur yang tidak memiliki rintangan terlebih dahulu. Kemudian setelah itu akan dipertimbangan jumlah jalur dengan rintangan yang paling sedikit dibandingkan dengan jalur lainnya. Boost akan digunakan ketika tidak ada rintangan di jalur saat ini. Kemudian accelerate akan selalu dilakukan jika kecepatan kurang dari atau sama dengan 3.

Kemudian membagi permasalahan ke dalam dua situasi yaitu ketika berada di depan mobil lawan dan ketika berada di belakang mobil lawan. Ketika berada di depan mobil lawan, akan digunakan tweet pada jalur lawan dengan koordinat block_lawan+16. Setelah itu akan digunakan oil jika jalur saat ini sama dengan jalur lawan. Apabila posisi berada di belakang lawan, maka gunakan emp dan tweet jika tersedia.

3.3 Analisis Efisiensi

3.3.1 Alternatif Solusi Pertama

Dilihat dari efisiensi, alternatif ini kurang optimal karena setiap menggunakan powerups, bot harus mengorbankan perpindahan ke depan, sehingga bisa jadi malah memperlama bot mendekati garis finish.

3.3.2 Alternatif Solusi Kedua

Alternatif ini juga bukanlah strategi optimal karena bot lawan dapat dengan mudah mengurangi kecepatan kita melalui penggunaan Powerups mereka. Secara efisiensi, alternatif ini juga kurang baik karena bot harus banyak berbelok, yang mengurangi perpindahan bot ke arah garis finish.

3.3.3 Alternatif Solusi Ketiga

Strategi meprioritaskan command FIX menjaga agar mobil dapat berjalan dengan kecepatan maksimal. Ronde yang dikorbankan untuk melakukan FIX sangat sepadan untuk memperoleh kecepatan di ronde-ronde berikutnya. Sebelumnya strategi untuk menghindari rintangan hanya berdasarkan apakah ada objek rintangan yang berada di depan mobil pada jalur yang sama tanpa menghitung jumlah rintangan. Setelah itu mobil akan belok ke jalur lain untuk menghindari rintangan tersebut. Tetapi hal ini tidak efisien karena bisa saja jalur baru sesudah belok memiliki rintangan yang lebih banyak dibandingkan dengan jalur sebelumnya dan hal ini akan mengurangi kecepatan secara drastis.

Dari percobaan yang dilakukan didapatkan kesimpulan bahwa mendapatkan keunggulan dari mobil lawan di early round sangat berdampak nantinya di ronde-ronde berikutnya sehingga penggunaan EMP ketika berada di belakang mobil lawan sangat efisien untuk mendahului mobil lawan. TWEET akan diletakkan pada jalur yang sama dengan jalur lawan dan berada pada block_lawan+16. Hal ini dilakukan agar TWEET tetap dapat mengenai lawan walaupun lawan berada di kecepatan maksimal (memakai BOOST).

Penggunaan BOOST harus memperhatikan rintangan yang ada karena rintangan seperti wall, oil spill, dan mud dapat mengurangi efek BOOST. Sebelumnya telah dicoba solusi greedy dengan selalu menggunakan BOOST apabila tersedia, namun hal ini tidak begitu efisien karena jika terdapat rintangan maka efek BOOST akan berkurang dan penggunaannya menjadi tidak maksimal. Strategi penggunaan LIZARD ini sangat efisien karena dari percobaan yang sudah dilakukan seringkali terdapat tiga jalur pada block yang sama yang memiliki rintangan, hal ini sangat berdampak buruk karena rintangan tidak dapat dihindari dengan berbelok.

3.4 Analisis Efektivitas

3.3.1 Alternatif Solusi Pertama

Alternatif ini ternyata bukanlah strategi optimal karena berdasarkan konfigurasi `game-config.js`,

```
<redacted>
    “MUD_GENERATION_PERCENTAGE”: 5,
    “BOOST_GENERATION_PERCENTAGE”: 1,
    “OIL_GENERATION_PERCENTAGE”: 1,
    “WALL_GENERATION_PERCENTAGE”: 1.25,
    “TWEET_GENERATION_PERCENTAGE”: 1,
    “LIZARD_GENERATION_PERCENTAGE”: 1,
```

```
    “EMP_GENERATION_PERCENTAGE”: 1
<redacted>
```

Dari kode tersebut, dapat disimpulkan bahwa karena persentase munculnya powerups OIL, EMP, dan TWEET lebih kecil dibandingkan dengan munculnya obstacle.

3.4.2 Alternatif Solusi Kedua

Strategi ini sebenarnya cukup efektif untuk meraih tujuan utama, yaitu mencapai garis finish paling cepat. Bot juga mengurangi penalti atas tabrakan dengan obstacle. Namun, terkadang kita dihadapkan pada kondisi di mana jalur yang sedang mobil kita tempati, jalur kiri, dan jalur kanannya terdapat obstacle. Di samping itu, mobil sedang tidak memiliki powerup LIZARD. Mau tidak mau, mobil akan terkena obstacle diikuti pengurangan score dan penambahan damage. Peraturan game menyebutkan bahwa semakin tinggi damage, maka kecepatan maksimal mobil akan semakin kecil. Oleh karena itu, command ACCELERATE akan menjadi percuma karena mobil kita tidak dapat melaju lebih cepat. Kita tetap harus menjalankan command FIX yang akan membuat mobil kita berdiam di block yang sama untuk satu round. Penggunaan LIZARD yang berlebihan juga merugikan karena akan banyak powerups yang terlompati sehingga tidak masuk ke dalam list powerups yang dimiliki oleh mobil.

Selain itu, berfokus kepada kecepatan akan memberikan kesempatan bagi lawan untuk melakukan serangan terhadap mobil kita dengan EMP dan TWEET. Akibatnya, mobil kita akan terus mengalami pengurangan kecepatan dan penambahan damage. EMP termasuk powerup yang sangat efektif digunakan di late game. Sejauh apapun kita memimpin, jika lawan memiliki banyak EMP, maka akan dengan mudah menyusul karena mobil kita hanya bisa berdiam di block yang sama selama satu round. TWEET merupakan alternatif jika tidak memiliki EMP. Cyber Truck yang dihasilkan oleh command ini tidak dapat dideteksi oleh mobil lawan. Jika lawan memiliki algoritma yang baik dan berhasil menaruh Cyber Truck di tempat yang tepat, maka mobil kita hanya bisa pasrah sambil menabrak truk tersebut.

3.4.3 Alternatif Solusi Ketiga

Saat berada di belakang lawan terutama ketika early round, penggunaan EMP terhitung lebih efektif daripada TWEET karena pada saat early round, power ups yang dimiliki lawan juga masih sedikit. Dengan ini EMP dapat mengurangi kecepatan mobil lawan dan memungkinkan untuk menyusul. Dibandingkan strategi tanpa melakukan pengecekan damage pada mobil, strategi fix memberikan hasil yang lebih baik, terlihat dari akumulasi kecepatan dari mobil menjadi lebih besar karena damage menjadi berkurang setelah dilakukan FIX.

Dengan kemampuan untuk menghitung banyak rintangan yang ada, maka pemilihan jalur menjadi lebih efektif dengan mengambil jalur yang memiliki rintangan paling sedikit. Kemudian ketika berada di depan mobil lawan, TWEET dapat dipastikan akan mengenai mobil lawan apabila mobil lawan tidak berganti jalur selama 15 blok ke depan sejak TWEET diletakkan. BOOST menjadi lebih efektif penggunaannya karena lebih baik untuk tidak menggunakan BOOST apabila masih terdapat rintangan di jalur saat ini. BOOST dapat disimpan untuk digunakan pada kesempatan yang lain agar efeknya dapat menjadi maksimal. Penggunaan LIZARD sangat efektif terutama jika ketiga lane di depan memiliki rintangan karena dengan itu rintangan dapat dilewati dan tidak perlu mengurangi kecepatan.

Bab 4
Implementasi dan Pengujian

4.1 Implementasi Algoritma Greedy pada Program Bot

Berikut adalah implementasi dari algoritma greedy yang disusun dari beberapa fungsi dengan fungsi run sebagai algoritma utama untuk mengeksekusi command ketika permainan dimulai.

4.1.1 Fungsi Run

```
function run() -> Command
{ Mengembalikan command yang akan menentukan langkah yang akan diambil bot pada suatu round
berdasarkan strategi yang telah dibuat }

Kamus Lokal:
blocks : List of object
my_block : int
my_lane : int
my_speed : int
opp_block : int
opp_lane : int
numObstacleRight : int
numObstacleLeft : int
numObstacleForward : int
numObstacleForwardIfBoost : int
safetyWay : int
numPowerUpsRight : int
numPowerUpsLeft : int
numPowerUpsForward : int
maxPowerUps : int

Algoritma:
{ Melakukan perbaikan pada mobil }
if (myCar.damage >= 1) then
   -> FIX

{ Strategi untuk menghindari rintangan }
if (blocks.contains(Terrain.MUD) or blocks.contains(Terrain.WALL) or
blocks.contains(Terrain.OIL_SPILL)) then
   if (numObstacleLeft > 0 and numObstacleForward > 0 and numObstacleRight > 0 and
hasPowerUp(PowerUps.LIZARD, myCar.powerups)) then
      -> LIZARD
   if (numObstacleForward = safetyWay) then
      if (numObstacleLeft = safetyWay and numPowerUpsLeft = maxPowerUps)
         -> TURN_LEFT
      if (numObstacleRight = safetyWay and numPowerUpsRight = maxPowerUps)
         -> TURN_RIGHT
      -> ACCELERATE
   else
      if (numObstacleLeft < numObstacleRight) then
         -> TURN_LEFT
      else if (numObstacleRight < numObstacleLeft) then
         -> TURN_RIGHT
```

```
        else
            if (numPowerUpsLeft > numPowerUpsRight) then
                -> TURN_LEFT
            else if (numPowerUpsRight > numPowerUpsLeft) then
                -> TURN_RIGHT
            else
                if (my_lane = 3 or my_lane = 4) then
                    -> TURN_LEFT
                else
                    -> TURN_RIGHT

{ Penggunaan Powers Ups BOOST }
if (hasPowerUp(PowerUps.BOOST, myCar.powerups) and numObstacleForwardIfBoost = 0) then
    -> BOOST

{ Mempertahankan kecepatan mobil }
if (my_speed <= 3) then
    -> ACCELERATE

{ Apabila posisi berada di depan mobil lawan, gunakan TWEET dan OIL }
if (my_block > opp_block) then
    if (hasPowerUp(PowerUps.TWEET, myCar.powerups)) then
        -> new TweetCommand(opp_lane, opp_block+16)
    if ((hasPowerUp(PowerUps.OIL, myCar.powerups) and (my_lane = opp_lane)) or (my_speed =
maxSpeed)) then
        -> OIL

{ Apabila posisi berada di belakang atau sejajar mobil lawan, gunakan EMP dan TWEET }
else
    if (hasPowerUp(PowerUps.EMP, myCar.powerups) and (abs(my_lane-opp_lane) <= 1)) then
        -> EMP
    if (hasPowerUp(PowerUps.TWEET, myCar.powerups)) then
        -> new TweetCommand(opp_lane, opp_block+16)

{ Jika tidak ada BOOST dan tidak ada obstacle, maka ACCELERATE }
if (not(myCar.boosting and my_speed != maxSpeed and numObstacleForward = 0)) then
    -> ACCELERATE

{ Tidak melakukan apa-apa jika seluruh kondisi di atas tidak memenuhi }
-> DO  NOTHING
```

4.1.2 Fungsi hasPowerUp

```
function hasPowerUp (PowerUps powerUpToCheck, PowerUps[] available) -> Boolean
{ Mengembalikan nilai True jika PowerUps tersedia }
Kamus Lokal:
Algoritma:
for (PowerUps powerUp: available):
    if (powerUp.equals(powerUpToCheck)) then
        -> true
-> false
```

4.1.3 Fungsi getBlocksInFront

```
Function getBlocksInFront (lane : int, block : int) -> int
{ Mengembalikan objek-objek yang ada pada jalur saat ini }

Kamus Lokal:
map : List of lane
blocks : List of object
startBlock : int
lanelist <- List of lane

Algoritma:
map <- gameState.lanes
blocks <- new ArrayList<>()
int startBlock <- map.get(0)[0].position.block

laneList <- map.get(lane - 1)
for (int i = max(block - startBlock, 0) i <= block - startBlock + myCar.speed i++)
   if (laneList[i] = null or laneList[i].terrain = Terrain.FINISH) then
      break
   blocks.add(laneList[i].terrain)
-> blocks
```

4.1.4 Fungsi numberObstacle

```
Function numberObstacle (lane : int, block : int) -> int
{ Mengembalikan jumlah rintangan yang ada pada block }

Kamus Lokal:
blocks : List of object
total : int

Algoritma:
if (lane < 1 or lane > 4) then
   -> 99
else
   total <- 0
   List<Object> blocks <- getBlocksInFront(lane, block)
   for (Object o : blocks)
      if (o = Terrain.OIL_SPILL or o = Terrain.MUD) then
         total += 1
      if (o = Terrain.WALL) then
         total += 2
-> total
```

4.1.5 Fungsi numberObstacleIfBoost

```
Function numberObstacleIfBoost (lane : int, block : int) -> int
{ Mengembalikan jumlah rintangan yang ada pada block saat menggunakan BOOST }

Kamus Lokal:
map : List of lane
blocks : List of object
```

```
startBlock : int
total : int
laneList : List of lane

Algoritma:
if (lane < 1 or lane > 4) then
   -> 99
else
   map <- gameState.lanes
   blocks <- new ArrayList<>()
   startBlock <- map.get(0)[0].position.block
   total <- 0
```

4.1.6 Fungsi numberObstaacleIfAccelerate (lane : int, block : int)

```
Function numberObstacleIfAccelerate(int lane, int block)
{ Mengembalikan jumlah rintangan yang ada pada block saat ACCELERATE }

Kamus Lokal:
map : List of lane
startBlock : int
speedState : List of int
index : int
total : int
laneList : List of lane

Algoritma :
if (lane < 1 or lane > 4) then
   -> 99
else
   map <- gameState.lanes
   startBlock <- ((Lane[])map.get(0))[0].position.block
   speedState <- new int[]0, 3, 6, 8, 9
   index <- 5
   total <- 0

   laneList <- map.get(lane - 1)
   for(int k = 0 k < speedState.length ++k) do
      if (this.myCar.speed = speedState[k]) then
         index <- k
         break

   if (myCar.speed = 5) then
      for (int i = max(block - startBlock, 0) i <= block - startBlock + 6 i++) do
         if (laneList[i] = null or laneList[i].terrain = Terrain.FINISH) then
            break
         if (laneList[i].terrain = Terrain.MUD) then
            total++
         if (laneList[i].terrain = Terrain.WALL) then
            total += 2
         if (laneList[i].terrain = Terrain.OIL_SPILL) then
            total++
```

```
    else if (index = 4) then
      for (int i = max(block - startBlock, 0) i <= block - startBlock + 9 i++) do
        if (laneList[i] = null or laneList[i].terrain = Terrain.FINISH) then
          break
        if (laneList[i].terrain = Terrain.MUD) then
          total++
        if (laneList[i].terrain = Terrain.WALL) then
          total += 2
        if (laneList[i].terrain = Terrain.OIL_SPILL) then
          total++
    else
      for (int i = max(block - startBlock, 0) i <= block - startBlock + 15 i++) do
        if (laneList[i] = null or laneList[i].terrain = Terrain.FINISH)
          break
        if (laneList[i].terrain = Terrain.MUD) then
          total++
        if (laneList[i].terrain = Terrain.WALL) then
          total += 2
        if (laneList[i].terrain = Terrain.OIL_SPILL) then
          total++
    -> total
```

4.1.7 Fungsi numberPowerUps

```
Function numberPowerUps(lane : int, block : int)
{ Mengembalikan jumlah power ups yang tersedia pada jalur }
Kamus Lokal:
total : int
blocks : List of object

Algoritma:
if (lane < 1 or lane > 4) then
   -> -99
   else
   total <- 0
   blocks <- getBlocksInFront(lane, block)
   for (Object o : blocks) do
      if (o = Terrain.BOOST or o = Terrain.EMP) then
         total += 5
      if (o = Terrain.TWEET or o = Terrain.LIZARD) then
         total += 3
      if (o = Terrain.OIL_POWER) then
         total += 1
   -> total
```

4.1 Struktur Data yang Digunakan

Program bawaan bot Overdrive mengimplementasikan paradigma pemrograman berorientasi objek, sehingga struktur data tersusun dalam sejumlah class. Program bot mengandung tiga folder yang berisi struktur data, yakni command untuk implementasi program command, entities untuk implementasi entitas di dalam game

yang terus berubah seiring waktu, dan enums atau enumerations untuk implementasi elemen-elemen yang bersifat konstan.

Dalam folder command, setiap command didefinisikan dalam class masing-masing. Isi folder command beserta penjelasannya adalah sebagai berikut. Seluruh command menggunakan struktur data String.

Nama File	**Penjelasan**
AccelerateCommand.java	Menjalankan perintah ACCELERATE
BoostCommand.java	Menjalankan perintah BOOST
ChangeLaneCommand.java	Menjalankan perintah TURN_LEFT dan TURN_RIGHT
Command.java	Base class untuk command lainnya. Pada dasarnya semua command akan melakukan render.
DecelerateCommand.java	Menjalankan perintah DECELERATE
DoNothingCommand.java	Menjalankan perintah DO_NOTHING
EmpCommand.java	Menjalankan perintah EMP
FixCommand.java	Menjalankan perintah FIX
LizardCommand.java	Menjalankan perintah LIZARD
OilCommand.java	Menjalankan perintah USE_OIL
TweetCommand.java	Menjalankan perintah USE_TWEET <lane><block>

Dalam folder entities, setiap entitas didefinisikan dalam class masing-masing. Isi folder entities beserta penjelasannya adalah sebagai berikut. ENUMS JUGA YE?

Nama File	**Penjelasan**
Car.java	Berisi struktur kelas Car. Kelas ini digunakan untuk menyimpan informasi umum tentang player car dengan atribut sebagai berikut. • id, bertipe integer (bilangan bulat). • position, bertipe Position. • speed, beripe integer (bilangan bulat). • state, bertipe State. • damage, bertipe integer (bilangan bulat). • powerups, bertipe list of Powerups. • boosting, bertipe boolean. • boost counter, bertipe integer (bilangan bulat).
GameState.java	Berisi struktur kelas GameState. Kelas ini digunakan untuk menyimpan status game saat runtime dengan atribut sebagai berikut. • currentRound, bertipe integer. • maxRounds, bertipe integer. • player, bertipe Car. • opponent, bertipe Car. • Lanes, bertipe list of list of Lane.

Lane.java	Berisi struktur kelas Lane. Kelas ini menyimpan informasi pada setiap jalur dengan atribut sebagai berikut. • position, bertipe Position. • terrain, bertipe Terrain. • occupiedByPlayerId, bertipe integer (bilangan bulat).
Position.java	Berisi struktur kelas Position. Kelas ini digunakan untuk menyimpan posisisi car pada player saat ronde tertentu. Nilai yang disimpan adalah LANE dalam variabel Y dan BLOCK dalam variabel X. Keduanya bertipe integer (bilangan bulat).

Struktur data buatan dalam folder ini ialah State (diimplementasikan dalam State.java), Powerups (diimplementasikan dalam PowerUps.java), Lane (diimplementasikan dalam Lane.java), Position (diimplementasikan dalam Position.java), dan Terrain (diimplementasikan dalam Terrain.java).

Dalam folder enums, setiap operasi enumerasi didefinisikan dalam class masing-masing. Isi folder enums beserta penjelasannya adalah sebagai berikut.

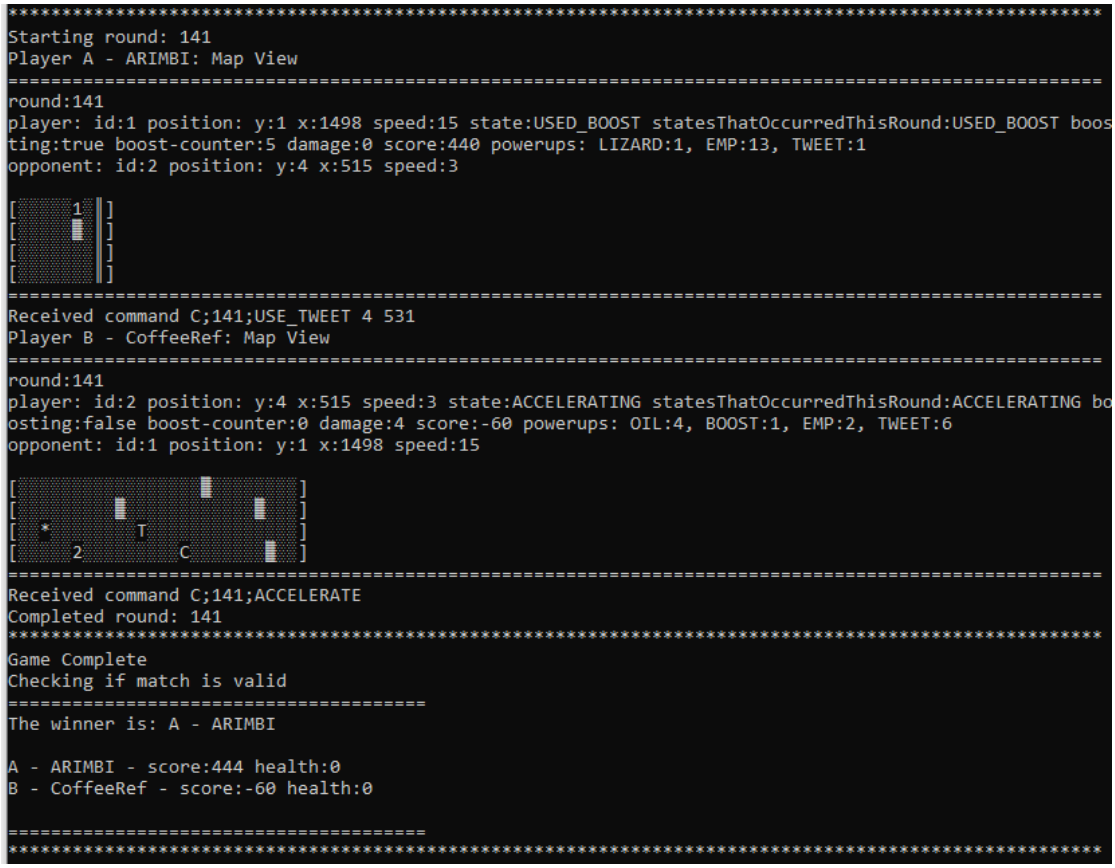
Nama File	Penjelasan
Direction.java	Berisi struktur kelas Direction. Kelas ini digunakan untuk menyimpan arah yang dimiliki oleh sebuah Car dengan atribut sebagai berikut. • lane, bertipe integer. • block, bertipe integer. • label, bertipe string.
PowerUps.java	Berisi struktur kelas PowerUps. Kelas ini digunakan untuk menyimpan powerups yang dapat dimiliki oleh sebuah Car. Jenis powerups tersebut adalah sebagai berikut. • BOOST • OIL • TWEET • LIZARD • EMP
State.java	Berisi struktur kelas State. Kelas ini digunakan untuk menyimpan status sebuah Car setelah melakukan command dalam suatu round. Jenis state tersebut adalah sebagai berikut. • ACCELERATING • READY • NOTHING • TURNING_RIGHT • TURNING_LEFT

	• HIT_MUD • HIT_OIL • DECELERATING • PICKED_UP_POWERUP • USED_BOOST • USED_OIL • USED_LIZARD • USED_TWEET • HIT_WALL • HIT_CYBER_TRUCK • FINISHED
Terrain.java	Berisi struktur kelas Terrain. Kelas ini digunakan untuk menyimpan informasi jenis block yang terdapat pada sebuah lane. Block dapat menyimpan obstacle maupun powerups. Jenis block adalah sebagai berikut. • EMPTY • MUD • OIL_SPILL • OIL_POWER • FINISH • BOOST • WALL • LIZARD • TWEET • EMP

4.2 Analisis Desain Solusi Algoritma Greedy

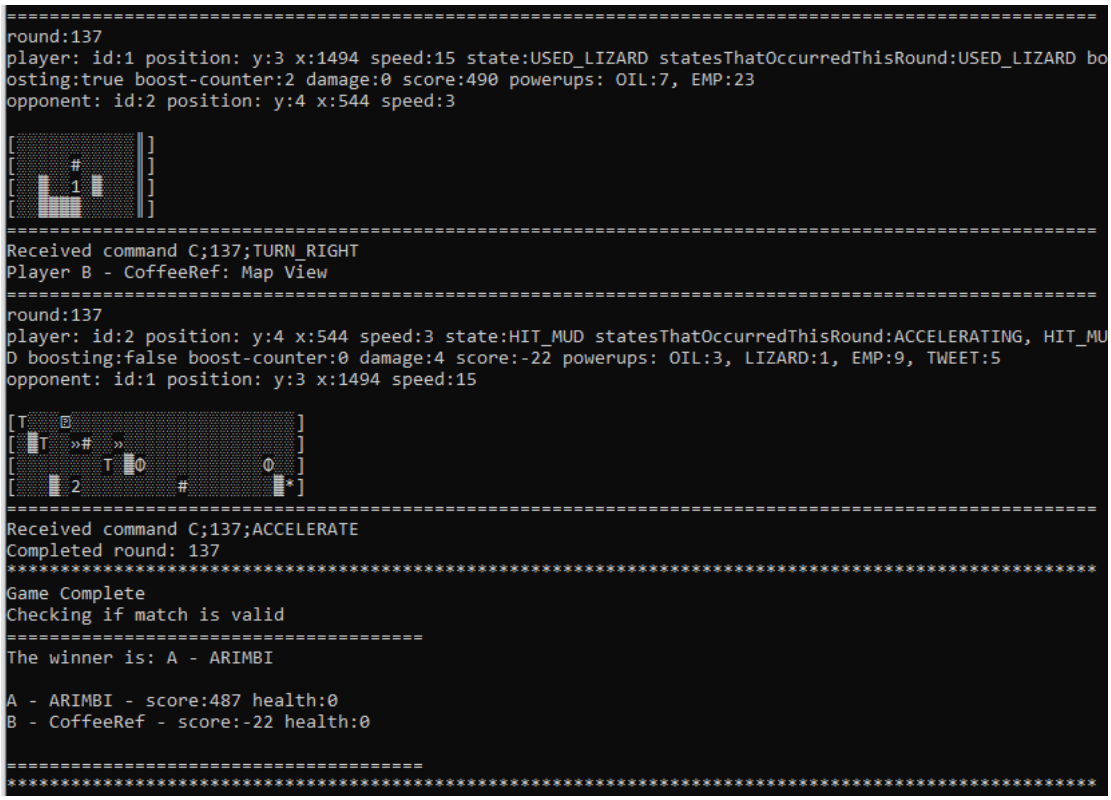
Pengujian akan dilakukan dengan mengambil 3 sampel percobaan dengan melawan Reference Bot yang disediakan oleh EntelectChallange. Tujuan utama yang ingin dicapai dari pengujian ini adalah dengan mendapatkan keunggulan ronde dan jumlah skor sebanyak-sebanyaknya melawan Reference Bot.

A. Percobaan Pertama



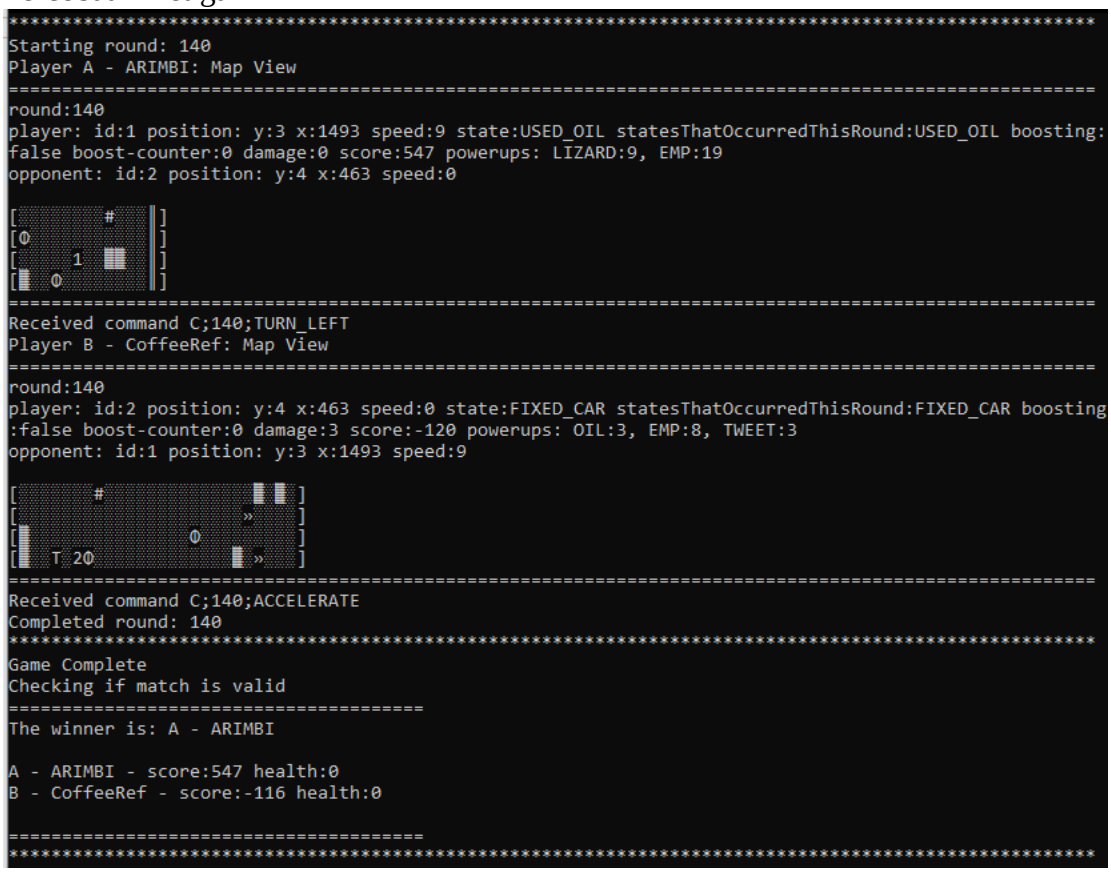
Pada pengujian pertama, dapat dilihat bahwa bot tim yang kami buat mengungguli reference bot sebanyak 983 ronde dan perbedaan skor sebanyak 504. Sisa power up yang tersedia adalah 13 EMP, 1 TWEET, dan 1 LIZARD. Hal ini dapat terjadi karena bot mendominasi kemeanangan ronde sejak awal permainan hingga akhir permainan sehingga power up EMP tidak perlu digunakan untuk memperlambat laju lawan. Pada ronde 27 bot menggunakan TWEET di jalur lawan pada saat itu dan mengenai bot lawan. Hal ini menunjukkan bahwa penggunaan TWEET efektif selama lawan tidak berpindah jalur sejak penempatan TRUCK akibat power up tersebut.

B. Percobaan Kedua



Pada percobaan ini bot kami memiliki keunggulan sebanyak 950 ronde dan perbedaan skor sebanyak 509. Kemudian sisa power up yang tersedia adalah OIL sebanyak 7 buah dan EMP sebanyak 23 buah. Alasan mengapa masih banyak EMP yang tersedia sama dengan percobaan sebelumnya yang mana bot kami mendominasi kemenangan ronde dari bot lawan sehingga EMP tidak perlu digunakan. Kemudian sisa OIL sebanyak 7 item terjadi karena kemungkinan ketika power up tersebut tersedia dan jalur bot sama dengan jalur lawan adalah kecil sehingga OIL tidak digunakan walaupun berada di depan mobil lawan. Power up TWEET juga sempat digunakan sama seperti percobaan sebelumnya dan tepat mengenai bot lawan. Kemudian terdapat penggunaan power up LIZARD ketika terdapat rintangan dua buah MUD dan satu buah WALL yang berada pada jalur yang sama berurutan, hal ini menunjukkan bahwa strategi penggunaan LIZARD menjadi sangat efisien untuk menghindari rintangan yang bertumpuk pada jalur yang sama. Dengan ini bot tidak mendapat pengurangan kecepatan.

C. Percobaan Ketiga



Pada percobaan ketiga, bot kami mengungguli bot lawan sebanyak 1030 ronde dengan perbedaan skor 663. Kemudian power up yang tersisa adalah LIZARD sebanyak 9 buah dan EMP sebanyak 19 buah. Dengan alasan yang sama seperti percobaan sebelumnya, kelompok kami tidak banyak menggunakan EMP dalam permainan. Kemudian banyak LIZARD yang tersisa menunjukkan bahwa rintangan yang ada pada permainan tidak terlalu rumit sehingga power up ini tidak perlu digunakan. Perlu diketahui bahwa syarat penggunaan LIZARD adalah banyaknya rintangan di depan, kiri, dan kanan dari jalur bot saat ini lebih dari 0. Kondisi ini jarang terjadi dalam percobaan kali ini.

Bab 5
Kesimpulan dan Saran

5.1 Kesimpulan

Dari analisis desain algoritma greedy pada bab 4.3 didapat bahwa bot kami memenangkan tiga percobaan yang dilakukan melawan bot lawan dengan rata-rata kemenangan bot melawan bot lawan adalah 718 ronde dan keunggulan skor sebanyak 558. Power up yang banyak tersisa dari ketiga percobaan tersebut adalah EMP, hal ini menunjukkan bahwa bot kami mendominasi kemenangan ronde dalam permainan. Beberapa strategi greedy seperti menggunakan LIZARD ketika terdapat banyak rintangan yang ada di jalur bot, penempatan power up TWEET, pemilihan jalur dengan rintangan yang sedikit juga telah terealisasi. Hal ini menunjukkan bahwa strategi algoritma greedy yang kami buat sudah efektif penggunaannya.

5.2 Saran

Secara umum tugas sudah diselesaikan dengan baik. Namun, ada beberapa hal yang masih dapat dilakukan untuk mendapatkan hasil yang lebih maksimal seperti perbaikan pada modularitas dan penulisan komentar yang lebih deskriptif pada kode program dan formatting pada laporan. Strategi algoritma greedy sudah dibuat semaksimal mungkin. Akan tetapi, masih ada beberapa kasus yang dapat dievaluasi untuk memberikan hasil yang lebih baik. Penulisan laporan juga dapat dilakukan sedikit demi sedikit agar dapat selesai sebelum waktu pengumpulan tugas. Atas perhatiannya kami ucapkan terima kasih.

Daftar Pustaka

Munir, R. (2022). Algoritma Greedy Bagian 1. Institut Teknologi Bandung. Diakses pada tanggal 3 Februari 2022 melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf.

Munir, R. (2022). Algoritma Greedy Bagian 2. Institut Teknologi Bandung. Diakses pada tanggal 3 Februari 2022 melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf.

Munir, R. (2022). Algoritma Greedy Bagian 3. Institut Teknologi Bandung. Diakses pada tanggal 3 Februari 2022 melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf

Lampiran

Link Repository Github:

https://github.com/ranjabi/Greedy-Algorithm-EntelectChallenge2020

Link Video:

https://youtu.be/c1awd1vtBg