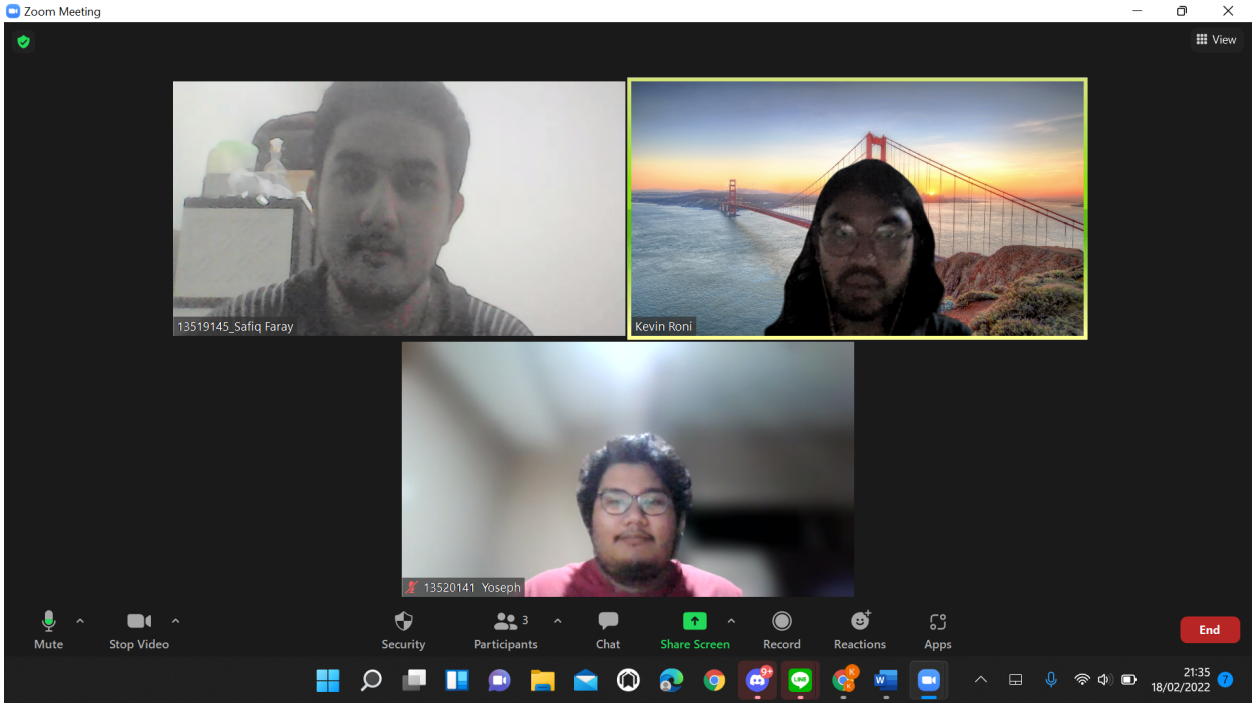
IF2211 STRATEGI ALGORITMA
TUGAS BESAR 1 SEMESTER II TAHUN 2021/2022

PEMANFAATAN ALGORITMA GREEDY DALAM APLIKASI PERMAINAN "OVERDRIVE"



13519145 Safiq Faray

13520114 Kevin Roni

13520141 Yoseph Alexander Siregar

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.

Gambar 1. Ilustrasi permainan Overdrive

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.

Tugas yang diberikan adalah untuk mengimplementasikan bot mobil dalam permainan Overdrive dengan menggunakan strategi greedy untuk memenangkan permainan.

Pada tugas besar kali ini, diminta untuk membuat sebuah bot untuk bermain permainan
Overdrive yang telah dijelaskan sebelumnya. Strategi greedy yang diimplementasikan 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.

Bab 2

Landasan Teori

2.1. Algoritma Greedy

Algoritma greedy adalah salah satu strategi untuk memecahkan masalah secara heuristik dan memilih pilihan yang paling optimal pada setiap tahap. Ini berarti algoritma greedy tidak melihat hasil jangka panjang dan hanya melihat hasil jangka pendeknya saja sehingga algoritma greedy tidak menjamin hasil yang paling optimal. Walaupun begitu, algoritma greedy dapat menghasilkan solusi yang cukup baik dalam waktu yang tidak terlalu lama.

Secara general terdapat beberapa komponen algoritma greedy, yaitu :

1. Set Kandidat : Elemen-elemen pembentuk solusi
2. Fungsi Seleksi : Fungsi untuk memilih kandidat terbaik untuk solusi
3. Fungsi Kelayakan : Fungsi pembatas yang memeriksa kelayakan setiap elemen pada himpunan kandidat.
4. Fungsi Objektif : Fungsi yang merupakan batasan optimum dari masalah yang akan diselesaikan.
5. Set Solusi : Kumpulan solusi dari setiap langkah dalam penyelesaian masalah

Secara umum, cara kerja algoritma greedy adalah sebagai berikut. Fungsi seleksi akan menyeleksi semua elemen pada himpunan kandidat. Elemen yang terpilih dari fungsi seleksi tersebut akan diuji kelayakannya sebagai solusi dengan menggunakan fungsi kelayakan dan akan dikeluarkan dari himpunan kandidat. Jika solusi tersebut layak, maka solusi akan dimasukkan ke dalam himpunan solusi. Jika tidak, maka akan mencari kandidat lain.

2.2. Game Engine

Game engine didapat dari Entelect Challenge. Game engine terdiri dari game-engine dan game-runner yang bertanggung jawab dalam pelaksanaan game beserta aturan-aturannya, serta pemanggilan command berdasarkan algoritma bot. Bot dapat diimplementasikan pada folder starter-bots.

Pada bot java, bot dipecah menjadi beberapa objek, yaitu objek commands, objek entities (Car, GameState, Lane, Position), serta objek bot utama pada Bot.java. Pengimplementasian algoritma greedy dapat dilakukan dengan menambahkan algoritma pada fungsi run pada objek Bot dengan return type Command.

Pada setiap game, informasi mengenai segala entitas yang berada di suatu game akan disimpan pada file GlobalState.json. File ini berisi informasi mengenai peta permainan dan informasi mobil tiap pemain. Pada tiap pemain juga direkam state individualnya pada JsonMap.json dan juga jejak command yang dieksekusi pada PlayerCommand.txt.

Untuk menjalankan permainan, bot harus di-*build* menggunakan jdk pada IntelliJ. Pada folder java, terdapat file bot.json yang menyimpan informasi dasar tentang pemain yang bisa

dimodifikasi. Sebelum dijalankan dipastikan untuk merubah jenis bot yang akan dijalankan pada file game-runner-config.json pada folder utama. Setelah .java file di-*build*, jalankan run.bat.

Bab 3

Aplikasi Strategi Greedy

3.1. Pemetaan Masalah

Permainan ini pada dasarnya adalah balapan mobil diantara dua pemain di sebuah trek yang disediakan. Mobil ini memiliki kecepatan tertentu dan memiliki batas *damage* yang diterima. Selain itu, mobil juga dapat menerima berbagai macam *power up*. Trek ini berbentuk array 2 dimensi yang memiliki 4 jalur lurus dan memiliki 5 tipe blok, yaitu *Empty, Mud* yang dapat mengurangi kecepatan mobil pemain jika terkena*, Oil Spill* yang dapat mengurangi kecepatan pemain jika terkena*, Flimsy Wall* yang dapat mengurangi kecepatan pemain sekaligus memberikan *damage,* dan *Finish Line* yang merupakan blok terakhir yang harus dilalui agar permainan dapat berakhir.

Setiap mobil dapat melakukan sebuah aksi dan mendapatkan skor darinya dan akan mengurangi poin jika pemberian aksi tidak valid. Aksi-aksi yang dapat dieksekusi oleh pemain terhadap mobil yaitu :

a. Nothing : Tidak melakukan apa-apa dalam suatu ronde.
b. Accelerate : Menambah kecepatan mobil
c. Decelerate : Mengurangi kecepatan mobil
d. Turn Left : Merubah lajur mobil ke kiri
e. Turn Right : Merubah lajur mobil ke kanan
f. Use Boost : Menggunakan *power up* yang menambah kecepatan selama 5 ronde.
g. Use Oil : Menggunakan *power up* yang akan meletakkan *"oil block"* di posisi mobil pemain.
h. Use Tweet : Menggunakan *power up* yang akan mengeluarkan *cyber truck* pada suatu lajur dan suatu blok, tergantung dari masukan pemain.
i. Use Lizard : Menggunakan *power up* yang membuat mobil melompat untuk menghindari kadal yang berjalan di trek.
j. Use EMP : Menggunakan *power up* yang akan menembakkan *emp* dan yang terkena akan berhenti di trek sampe ronde berakhir dan mengurangi kecepatan mobil hingga 3.
k. Fix : Memperbaiki mobil dengan mengurangi damage yang diterima sebanyak dua.

Pemenang dari permainan ini akan ditentukan dengan dua cara, yaitu jika kedua mobil telah melalui *finish line*, maka penentu pemenangnya adalah mobil yang memiliki kecepatan tertinggi. Jika kedua mobil kecepatannya sama, maka akan ditentukan berdasarkan skor tertinggi.

Berdasarkan deskripsi diatas, dapat dipetakan permasalahan permainan *Overdrive* menjadi elemen-elemen algoritma greedy.

3.1.1. Himpunan Kandidat

Himpunan command yang dapat dipilih (sesuai dengan penjelasan di atas)

3.1.2. Himpunan Solusi

Berisikan command yang dipilih

3.1.3. Fungsi Solusi

Memeriksa apakah command yang dipilih menghasilkan value terbesar

3.1.4. Fungsi Seleksi

Memilih command berdasarkan value command yang tertinggi pada keberjalanan round

3.1.5. Fungsi Kelayakan

Memeriksa apakah command yang dipilih merupakan command yang valid

3.1.6. Fungsi Objektif

Memeriksa di antara command yang tersedia, command mana yang bernilai lebih tinggi

3.2. Alternatif Solusi serta Analisis Efisiensi & Efektivitas

3.2.1. Greedy by Speed

Greedy by speed adalah mengurutkan command berdasarkan kecepatan yang mungkin untuk round berikutnya. Alternatif ini akan menganalisis situasi bot pada saat tersebut, lalu menganalisis command yang dapat memberikan kecepatan paling maksimal bagi bot untuk round berikutnya. Idenya adalah dengan mengoptimasi kecepatan tiap ronde, akan mencapai garis finish dengan waktu secepat mungkin.

Analisis Efisiensi dari greedy by speed adalah sebagai berikut:

Algoritma ini akan mencoba memaksimalkan kecepatan bot pada tiap ronde, harapannya dengan menjadi paling cepat dapat menjadi yang pertama di garis finish. Algoritma akan mengutamakan lane yang tidak memiliki rintangan sama sekali agar kecepatan tidak berkurang. Apabila pada semua lane yang memungkinkan ada rintangan, algoritma akan menganalisis lane mana yang memiliki beban paling ringan sehingga tidak begitu berdampak pada round berikutnya. Apabila jalan yang paling lapang adalah lurus, bot akan mencoba menggunakan *power up* apabila tersedia. Karena pengecekan yang dilakukan adalah sebanyak n buah command, maka kompleksitas waktu adalah O(n)

Analisis Efektivitas dari greedy by speed adalah sebagai berikut:

Algoritma ini akan sangat efektif dalam memilih perjalanan yang paling menghindari masalah. Akan tetapi, algoritma ini akan lemah pada map yang memiliki *power up* sangat banyak dan tersebar. Algoritma ini cenderung mengabaikan *power up* yang tersebar sehingga mengurangi beberapa kemungkinan meraih poin. Selain itu, mobil cenderung akan mengabaikan *power up* yang sedang tersedia, kecuali *boost*. Algoritma ini akan sangat efektif jika mobil tidak memiliki *power up*. Algoritma ini memberikan keadaan menguntungkan apabila sedang dalam keadaan memimpin karena tidak perlu memikirkan penggunaan *power up* untuk memperlambat

musuh. Pada keadaan tidak memimpin, algoritma tidak memprioritaskan usaha untuk menggagalkan lawan.

3.2.2. Greedy by Power Ups

Greedy by Power Ups adalah strategi greedy yang mengusahakan penggunaan *power ups* yang tersedia. Strategi ini merupakan salah satu strategi paling dasar yang dapat dilakukan oleh bot karena penggunaannya yang mudah. Algoritma ini memiliki ide semakin banyak *power up* yang digunakan, akan semakin mudah jalan kita untuk mencapai garis finish. *Power up* adalah salah satu komponen yang memberikan bot kemampuan lebih dari bot musuh sehingga penggunaannya memiliki 2 kemungkinan, mempermudah bot untuk mencapai garis finish (contohnya pada *power up lizard* serta *boost*) atau mempersulit musuh untuk mencapai garis finish (contohnya pada *power up oil, tweet, EMP*).

Analisis efisiensi greedy by power up adalah sebagai berikut:

Algoritma ini akan berfokus pada pemakaian *power up*. Apabila *power up* tersedia, maka *power up* tersebut akan digunakan. Penggunaan strategi ini akan menghabiskan waktu untuk mengecek ketersediaan m *power up* sebanyak m kali maka kompleksitasnya adalah O(m).

Analisis efektivitas greedy by power up adalah sebagai berikut:

Fokus dari strategi ini adalah memakai *power up* jika ada. Karena itu, penggunaan *power up* haruslah sebaik mungkin. Contohnya untuk menggunakan *tweet* serta *emp* haruslah memprediksi pergerakan lawan sebaik mungkin agar kena. Algoritma ini akan sangat efektif jika bot memiliki *power up* yang banyak. Algoritma ini akan menjadi tidak efektif apabila tidak beruntung dalam mendapatkan lane sehingga lane yang dipilih memiliki banyak hambatan. Karena fokus dari strategi ini adalah *power up*, maka pemilihan lane tidak dipikirkan sehingga sangat mungkin bagi mobil untuk mendapatkan damage hasil bertemu dengan rintangan yang ada.

3.2.3. Greedy by Score

Sesuai dengan namanya, strategi ini akan memprioritaskan pengumpulan poin sebanyak-banyaknya ketika round berjalan dan menghindari pengurangan poin. Idenya adalah dengan mengumpulkan poin sebanyak-banyaknya pada round yang akan datang, pemain dapat memenangkan pertandingan. Penggunaan command akan diurutkan berdasarkan command yang menghasilkan score paling tinggi, seperti penggunaan *power up* akan menambahkan poin dan menabrak akan mengurangi poin. Command yang menghasilkan score tertinggi akan dipilih menjadi command bot.

Analisis efisiensi greedy by score:

Strategi ini akan menilai command tertentu akan mendapatkan score berapa. Strategi akan menilai command sebanyak n memiliki nilai berapa dan melakukan algoritma *sorting* untuk

mengurutkan command terurut berdasarkan score, lalu memilih command yang valid untuk dipilih. Algoritma ini memiliki kompleksitas O(n^2).

Analisis efektivitas greedy by score:

Algoritma ini akan sangat efektif jika kondisi akhir adalah seri karena sudah pasti strategi ini akan memiliki score yang sangat banyak. Akan tetapi, karena strategi ini adalah *score-oriented*, maka bot cenderung tidak memperhatikan kondisi kalah atau menang, serta kondisi mobil sendiri. Hal ini akan menyebabkan bot cenderung melakukan command yang menambahkan score tetapi tidak melakukan command yang merupakan *winning condition* (seperti menghindari jalan atau menggunakan boost untuk menambah kecepatan)

3.3. Strategi Greedy yang Dipilih

Strategi yang kami pilih adalah greedy by speed. Hal ini disebabkan oleh hal-hal berikut:

-Greedy by speed adalah yang paling efektif karena command yang diperlukan merupakan command optimal lokal yang merupakan winning condition (bergerak semakin dekat ke garis finish)

-Greedy by power up memiliki kecenderungan untuk mengabaikan *obstacle* yang ada di depan dan greedy by score sangat *score-oriented* yang menurut kelompok kami bukan merupakan winning condition utama

-Greedy by speed memiliki ide yang sangat sesuai dengan ide utama strategi greedy, yaitu dengan memilih optimal lokal akan dihasilkan optimal global

Strategi yang kami pilih, memiliki skema umum sebagai berikut:

-Fix adalah command paling penting, jika mobil rusak, kecepatan akan tidak maksimal. Fix jika *damage* >= 2.

-Cari lane yang tidak memiliki obstacle, jika ternyata lurus gunakan power up

-Jika semua lane ada obstacle, gunakan lizard jika ada

-Jika tidak ada lizard, maka carilah lane memungkinkan yang paling tidak menimbulkan *damage* paling besar

-Tempuhlah lane tersebut, jika ternyata lurus, gunakan power up

Bab 4

Implementasi dan Pengujian

4.1. Penerapan Algoritma Dalam Pseudocode

Program Bot

KAMUS

```
random : Random
gameState : gameState
opponent : Car ← gameState.opponent
myCar : Car ← gameState.player
maxSpeed : integer
ACCELERATE : Command ← AccelerateCommand()
ACCELERATE : Command ← AccelerateCommand()
BOOST : Command ← BoostCommand()
TURN_LEFT : Command ← ChangeLaneCommand(-1)
TURN_RIGHT : Command ← ChangeLaneCommand(1)
DECELERATE : Command ← DecelerateCommand()
DO_NOTHING : Command ← DoNothingCommand()
EMP : Command ← EmpCommand()
FIX : Command ← FixCommand()
LIZARD : Command ← LizardCommand()
OIL : Command ← OilCommand()

function run() → Command
{dari algoritma yang diterapkan, fungsi akan mengeluarkan tipe data
Command, yang merupakan perintah bot}

function TWEET     (lane : integer, block : integer) → Command
{fungsi yang akan membuat player menggunakan power up tweet pada lane dan
block tertentu }

function getLaneDamage(lane : integer, block : integer) → integer
{fungsi akan mendapati damage yang mungkin di dapat  berdasarkan block
atau rintangan yang ada. Akan mengembalikan nilai integer}

function ForwardDamageComparison (lane : integer, block : integer, left :
boolean, right : boolean, leftPriority : boolean) → boolean
{membandingkan damage yang diterima dari lane depan, kanan, dan belakang.
Akan mengembalikan nilai integer yang berupa damage }

function getLaneDamageBoosted(lane : integer, block : integer) → integer
{ fungsi akan mendapati damage yang mungkin di dapat  berdasarkan block
atau rintangan yang ada dengan kondisi mobil sedang boosted, yang berarti jarak
yang ditempuh makin jauh. Akan mengembalikan nilai integer}
```

function isLaneSafe(lane : integer, block : integer) → boolean
{mengembalikan nilai boolean yang menandakan apakah sebuah lane aman dilewati atau tidak, yang berarti melakukan pengecekan apakah terdapat sebuah rintangan pada lane, yang ditandai dengan damage = 0 dari nilai getLaneDamage }

function isLaneSafeBoosted(lane : integer, block : integer) → boolean
{ mengembalikan nilai boolean yang menandakan apakah sebuah lane aman dilewati atau tidak, dengan asumsi mobil dalam keadaan boosted yang berarti jarak yang ditempuh makin jauh, yang berarti melakukan pengecekan apakah terdapat sebuah rintangan pada lane, yang ditandai dengan damage = 0 dari nilai getLaneDamageBoosted}

function isOpponentBehind() → boolean
{Melakukan pengecekan apakah musuh berada di belakang pemain atau tidak, yang dilihat dari posisi lane. Akan mengembalikan true jika benar berada di belakang, dan sebaliknya jika tidak }

function opponentLanePosition () → integer
{Akan mengembalikan posisi lane musuh dalam bentuk integer }

function opponentBlockPosition() → integer
{ Akan mengembalikan posisi block musuh dalam bentuk integer }

function getMyLane() → integer
{Akan mengembalikan posisi lane player }

function isTurnValid(direction : integer, currLane : integer) → boolean
{ Akan mengecek apakah jalur yang akan dituju aman atau tidak, dalam artian melanggar constraints atau tidak. }

function isPowerUpAvailable(tocheck : PowerUps) → boolean
{Mengecek apakah sebuah powerup tersedia atau tidak, dan akan mengembalikan tipe data boolean }

function usePowerUps () → Command
{ Akan mengembalikan Command berupa penggunaan power up berdasarkan power up yang dimiliki dan kelayakan penggunaan }

ALGORITMA

```
function run() → Command
KAMUS LOKAL
LaneNow : integer
BlockNow : integer
ALGORITMA
LaneNow ← myCar.position.lane
BlockNow = myCar.position.block
If (myCar.damage >= 4) then
      → FIX
If (myCar.speed = this.maxSpeed) then
```

```
            If (isLaneSafe(LaneNow, BlockNow)) then
                  usePowerUps()
            else if (isTurnValid(-1, LaneNow) and isLaneSafe(LaneNow-1,
BlockNow-1)) then
                  → TURN_LEFT
            else if (isTurnValid(1, LaneNow) and isLaneSafe(LaneNow+1,
BlockNow-1)) then
                  → TURN_RIGHT
            else
                  if (isPowerUpAvailable(PowerUps.LIZARD)) then
                        → LIZARD
                  else
                        if (!isTurnValid(-1, LaneNow) or !isTurnValid(1,
LaneNow)) then
                              if (ForwardDamageComparison(LaneNow, BlockNow,
!isTurnValid(-1, LaneNow), !isTurnValid(1, LaneNow), false)) then
                              usePowerUps()
                              else
                                    if (!isTurnValid(-1, LaneNow))then
                                          → TURN_RIGHT
                                    else
                                          → TURN_LEFT
                        else
                              if (ForwardDamageComparison(LaneNow, BlockNow,
true, true, false)) then
                                    usePowerUps();
                              else if (ForwardDamageComparison(LaneNow,
BlockNow, true, true, true)) then
                                    → TURN_LEFT
                              else
                                    → TURN_RIGHT
      else
            if (isPowerUpAvailable(PowerUps.BOOST) and
isLaneSafeBoosted(LaneNow, BlockNow)) then
                  → BOOST
            else
                  If (isLaneSafe(LaneNow, BlockNow)) then
                        usePowerUps()
                  else if (isTurnValid(-1, LaneNow) and isLaneSafe(LaneNow-1,
BlockNow-1)) then
                  → TURN_LEFT
            else if (isTurnValid(1, LaneNow) and isLaneSafe(LaneNow+1,
BlockNow-1)) then
                  → TURN_RIGHT
            else
                  if (isPowerUpAvailable(PowerUps.LIZARD)) then
                        → LIZARD
                  else
                        if (!isTurnValid(-1, LaneNow) or !isTurnValid(1,
```

```
LaneNow)) then
                              if (ForwardDamageComparison(LaneNow, BlockNow,
!isTurnValid(-1, LaneNow), !isTurnValid(1, LaneNow), false)) then
                              usePowerUps()
                              else
                                    if (!isTurnValid(-1, LaneNow))then
                                          → TURN_RIGHT
                                    else
                                          → TURN_LEFT
                        else
                              if (ForwardDamageComparison(LaneNow, BlockNow,
true, true, false)) then
                                    usePowerUps();
                              else if (ForwardDamageComparison(LaneNow,
BlockNow, true, true, true)) then
                                    → TURN_LEFT
                              else
                                    → TURN_RIGHT
```

```
function TWEET(lane : integer, block : integer) → integer
ALGORITMA
→ TweetCommand(lane, block)

function getLaneDamage(lane : integer, block : integer) → integer
KAMUS LOKAL
map : Array[] of Array[] of Lane
damage : integer
startBlock : integer
laneList : Array[] of Lane
ALGORITMA
map ← gameState.lanes
damage ← 0
startBlock ← map[0][0].position.block
laneList ← map[lane-1]
i traversal[max(block-startBlock,0)...(block-startBlock+myCar.speed)]
      If (laneList[i] = null or laneList[i].terrain = Terrain.FINISH)
then
            break
      else if (laneList[i].terrain = Terrain.OIL_SPILL) then
            damage ← damage+2
      else if (laneList[i].terrain = Terrain.MUD) then
            damage ←damage+2
      else if (laneList[i].terrain = Terrain.WALL) then
            damage ← damage+4
→ damage

function ForwardDamageComparison (lane : integer, block : integer, left :
boolean, right : boolean, leftPriority : boolean) → boolean
KAMUS LOKAL
```

```
      forwardDamage : integer
      rightOrLeftDamage : integer
      rightDamage : integer
      leftDamage : integer
      ALGORITMA
      forwardDamage ← getLaneDamage(lane,block)
      If (left) then
            rightOrleftDamage ← getLaneDamage(lane+1,block-1)
      else
            rightOrLeftDamage ← getLaneDamage(lane-1, block-1)
      if (left and right) then
            rightDamage ← getLaneDamage(lane+1, block-1)
            leftDamage ← getLaneDamage(lane-1,block-1)
            if (leftPriority) then
                  → leftDamage >= forwardDamage and leftDamage >= rightDamage
            else
                  → forwardDamage >= leftDamage and forwardDamage >=
rightDamage
      → forwardDamage >= rightOrLeftDamage

      function getLaneDamage(lane : integer, block : integer) → integer
      KAMUS LOKAL
      map : Array[] of Array[] of Lane
      damage : integer
      startBlock : integer
      laneList : Array[] of Lane
      ALGORITMA
      map ← gameState.lanes
      damage ← 0
      startBlock ← map[0][0].position.block
      laneList ← map[lane-1]
      i traversal[max(block-startBlock,0)...(block-startBlock+this.maxSpeed)]
            If (laneList[i] = null or laneList[i].terrain = Terrain.FINISH)
then
                  break
            else if (laneList[i].terrain = Terrain.OIL_SPILL) then
                  damage ← damage+2
            else if (laneList[i].terrain = Terrain.MUD) then
                  damage ←damage+2
            else if (laneList[i].terrain = Terrain.WALL) then
                  damage ← damage+4
      → damage

      function isLaneSafe(lane : integer, block : integer) → boolean
      ALGORITMA
      → getLaneDamage(lane, block) = 0

      function isLaneSafeBoosted(lane : integer, block : integer) → boolean
      ALGORITMA
```

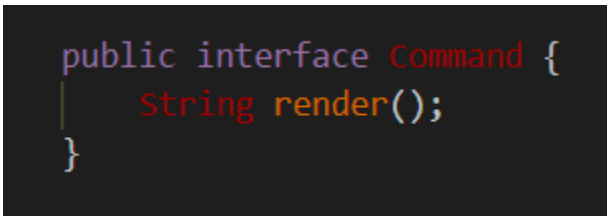
```
→ getLaneDamageBoosted(lane, block) = 0

function isOpponentBehind() → boolean
ALGORITMA
If (opponent.position.block < myCar.position.block) then
      → true
else
      → false

function opponentLanePosition() → integer
ALGORITMA
→ opponent.position.lane

function opponentBlockPosition() → integer
ALGORITMA
→ opponent.position.block

function getMyLane() → integer
ALGORITMA
→ myCar.position,lane

function isTurnValid(direction : integer, currLane : integer) → boolean
ALGORITMA
If (currLane = 1 and  direction = -1) then
      → false
else if (currLane = 4 and  direction = 1) then
      → false
else

      → true

function isPowerUpAvailable(tocheck : PowerUps) → boolean
ALGORITMA
i traversal [0…length(myCar.powerUps)]
      if (myCar.powerUps[i] = tocheck) then
            → true
→ false

function usePowerUps() → Command
ALGORITMA
      if (isPowerUpAvailable(PowerUps.EMP)  and  (getMyLane() =
opponentBlockPosition())) then
            →  EMP
      if (isPowerUpAvailable(PowerUps.TWEET)) then
            → TWEET(opponentLanePosition(), opponentBlockPosition())
      if (isPowerUpAvailable(PowerUps.OIL) and isOpponentBehind())
            → OIL
      else
            → ACCELERATE
```

4.2. Penjelasan Struktur Data

Struktur-struktur data yang digunakan dalam program implementasi bot dalam permainan Overdrive.

a. Commands

Struktur data commands terdapat pada directory ‘command’ yang merupakan implementasi dari class utama Command pada Command.Java :



Command.Java

Terdapat 10 class yang diimplementasikan menggunakan class Command di atas yaitu,

a). AccelerateCommand

b). BoostCommand

c). ChallengeLaneCommand

d). DecelerateCommand

e). DoNothingCommand

f). EmpCommand

g). FixCommand

h). LizardCommand

i). OilCommand

j). TweetCommand

b. Entities

Struktur data entities terdapat pada directory ‘entities’ yang merupakan berbagai macam class dengan atribut-atribut public yang menjadi informasi yang bisa diakses.

Entitas-entitas yang dimaksud yaitu,

a). Car, yang memiliki atribut id, position, speed, state, damage, powerups, boosting, boostcounter

b). GameState, yang memiliki atribut currentRound, maxRounds, player, opponent, dan lanes

c). Lane, yang memiliki atribut position, terrain, occupiedByPlayerId

d). Position, yang memiliki atribut lane dan block

c. Enums

Enums atau enumerated type adalah sekumpulan konstanta, yang kali ini telah disediakan dari game enginenya. Enums yang terdapat pada program ini yaitu,

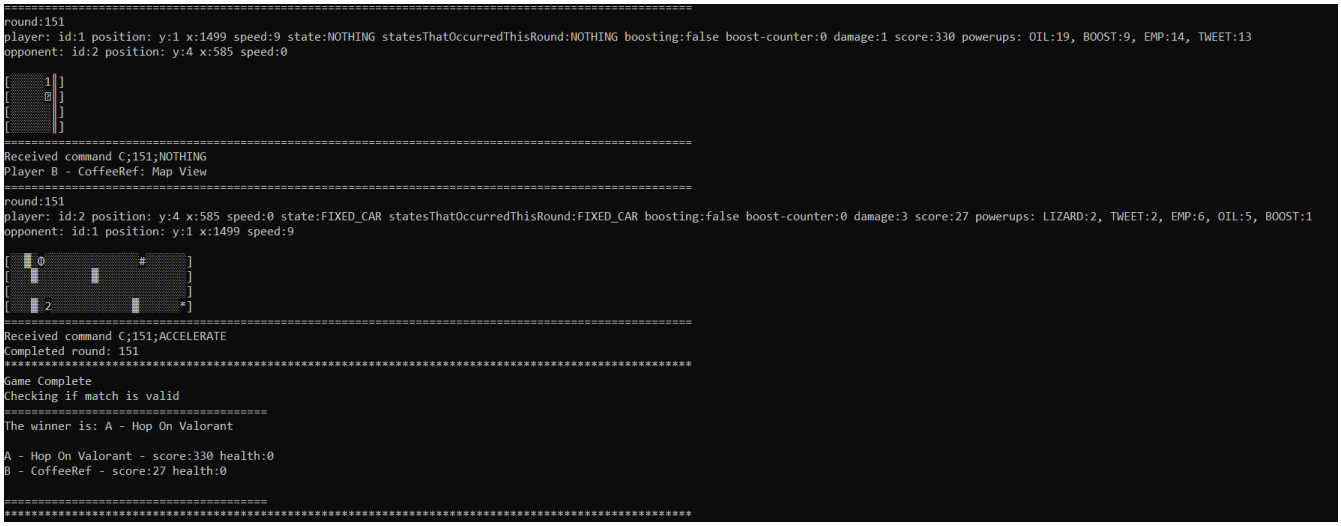
a). Direction, yang merupakan konstanta arah yang relatif jalan mobil pada permainan ini. Bergantung pada masukan x dan y, direction ini memiliki konstanta berupa FORWARD, BACKWARD, LEFT, dan RIGHT

b). PowerUps, yang mengandung berbagai macam power up dalam permainan ini. Power up tersebut adalah BOOST, OIL, TWEET, LIZARD, dan EMP

c). State, yang mengandung berbagai macam kondisi mobil pemain dalam bentuk konstanta. Kondisi-kondisi tersebut adalah ACCELERATING, READY, NOTHING, TURNING_RIGHT, TURNING_LEFT, HIT_MUD, HIT_OIL, DECELERATING, PICKED_UP_POWERUP, USED_BOOST, USED_OIL, USED_TWEET, USED_LIZARD, HIT_WALL, HIT_CYBER_TRUCK, dan FINISHED

d) Terrain, yang mengandung berbagai macam terrain yang berada pada trek permainan ini. Terrain tersebut adalah EMPTY, MUD, OIL_SPILL, OIL_POWER, FINISH, BOOST, WALL, LIZARD, TWEET, dan EMP

4.3. **Analisis**



Gambar di atas merupakan hasil menjalankan dari program bot yang sudah dibuat. Dapat dilihat pada round akhir bot kami berada pada posisi block 1491 dan reference bot pada block 562. Meskipun bot kami berhasil menang dengan lead dan score yang lumayan jauh, tetapi apabila diperhatikan pada bagian power up terjadi penumpukan :

Hal ini terjadi karena strategi algrotima greedy yang kami gunakan benar-benar terfokus pada cara untuk mendapatkan speed tercepat pada round itu dan hanya menggunakan power up apabila algoritma memutuskan untuk mengambil jalan lurus (tidak berpindah lane). Sehingga penggunaan power up menjadi terbengkalai dan tidak dimanfaatkan sebaik mungkin karena keterbatasan kondisi untuk pemakainnya dalam strategi greedy yang kami buat.

Hal yang berbeda terlihat pada reference bot, dimana bisa terlihat pada kode program reference bot yang ada pada starter-pack, menggunakan greedy by power ups yang akan langsung menggunakan power up yang ada secara langsung dengan prioritas tertentu. Akan tetapi, dapat dilihat bahwa strategi ini tidak efektif dan efisien karena power up yang memberikan keuntungan secara langsung kepada bot hanyalah power up BOOST.

Bab 5

Kesimpulan dan Saran

5.1. Kesimpulan

Berdasarkan pengujian, algoritma yang kami buat sudah dapat berfungsi dan berhasil mengalahkan reference bot yang merupakan bot default. Berdasarkan data uji, hasil kemenangan kami juga cukup jauh dengan reference bot sehingga dapat dinyatakan dengan meyakinkan. Akan tetapi, karena salah satu ide dari algoritma greedy adalah ambil pilihan optimal lokal tanpa memikirkan konsekuensi kedepannya, strategi yang kami susun memiliki kekurangan, yaitu kurang manajemen risiko untuk visi permainan kedepannya. Oleh karena itu kami menyimpulkan bahwa untuk menjadi juara dalam permainan ini tidak cukup dengan hanya mengandalkan algoritma greedy

5.2. Saran

Melihat kesimpulan yang kami dapatkan bahwa algoritma greedy saja tidak cukup untuk benar-benar merancang suatu program yang efektif dan efisien, sebaiknya eksplorasi algoritma diluar greedy diperbolehkan asal tidak menjadi algoritma utama.

Daftar Pustaka

1. https://github.com/EntelectChallenge/2020-Overdrive
2. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

3. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf
4. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf

Link Repository dan Video

1. Link Repository : https://github.com/yosalx/Stima01Tubes
2. Link Video : https://youtu.be/6D9RIg17pPI