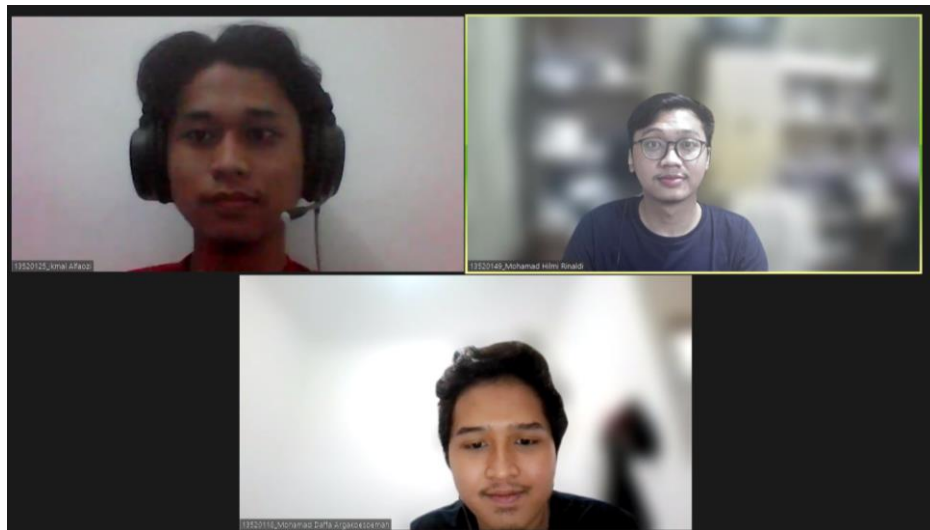


TUGAS BESAR 1 IF 2211
STRATEGI ALGORITMA

oleh

Mohamad Daffa Argakoeosemah	13520118
Ikmal Alfaozi	13520125
Mohamad Hilmi Rinaldi	13520149



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

DAFTAR ISI

DAFTAR ISI.....	2
BAB I DESKRIPSI TUGAS	3
BAB II LANDASAN TEORI.....	6
2.1. Dasar Teori	6
2.2. Cara Kerja Program.....	6
BAB III APLIKASI STRATEGI GREEDY.....	8
3.1. Proses Mapping	8
3.2. Ekplorasi Alternatif Solusi	8
3.3. Analisis Efisiensi.....	9
3.4. Analisis Efektivitas.....	9
3.5. Strategi Greedy yang Dipilih.....	10
BAB IV IMPLEMENTASI DAN PENGUJIAN	12
4.1. Implementasi Algoritma Greedy	12
4.2. Struktur Data.....	13
4.3. Analisis Solusi	15
BAB V KESIMPULAN DAN SARAN.....	18
5.1. Kesimpulan	18
5.2. Saran.....	18
DAFTAR LINK.....	19
DAFTAR PUSTAKA.....	20

BAB I

DESKRIPSI TUGAS

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



Gambar 1. Ilustrasi permainan *Overdrive*

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

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

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

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

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

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

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

Adapun peraturan yang lebih lengkap dari permainan *Overdrive*, dapat dilihat pada laman :
<https://github.com/EntelectChallenge/2020-Overdrive/blob/develop/game-engine/game-rules.md>

BAB II

LANDASAN TEORI

2.1. Dasar Teori

Algoritma Greedy merupakan metode yang paling populer dan sederhana untuk menyelesaikan berbagai permasalahan optimasi atau pencarian solusi optimal. Terdapat dua permasalahan optimasi yaitu maksimisasi dan minimasi. Algoritma ini merupakan algoritma yang memecahkan persoalan secara langkah per langkah sedemikian sehingga pada setiap langkah:

1. Mengambil pilihan terbaik yang dapat diperoleh pada saat itu tanpa memedulikan konsekuensi ke depannya
2. Memilih optimum lokal dengan berharap bahwa akan berakhir dengan optimum global

Terdapat beberapa elemen dalam Algoritma Greedy, yaitu:

1. Himpunan kandidat: berisi kandidat yang akan dipilih pada setiap langkah
2. Himpunan solusi: berisi kandidat yang sudah dipilih
3. Fungsi solusi: menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi
4. Fungsi seleksi: memilih kandidat berdasarkan strategi *greedy* tertentu. Strategi *greedy* ini bersifat heuristik
5. Fungsi kelayakan: memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi
6. Fungsi obyektif: memaksimumkan atau meminimumkan

2.2. Cara Kerja Program

Di dalam aplikasi permainan “Overdrive”, terdapat dua pemain (bot) yang dapat digunakan untuk menjalankan permainan. Dalam permainannya, terdapat beberapa komponen yang digunakan agar bot ini dapat berjalan, yaitu:

1. *Game engine*
Game engine bertugas agar bot yang dibuat dapat mengikuti aturan yang sudah ditentukan dan akan memproses perintah dari bot ke dalam game jika perintah yang diberikan valid.
2. *Game runner*
Game runner bertugas untuk menjalankan pertandingan antar bot dan akan menerima perintah dari bot yang nantinya akan diteruskan ke *game engine* untuk dieksekusi.
3. *Reference bot*
Bot referensi yang sudah disediakan dan dapat digunakan untuk menguji bot yang akan dibuat.
4. *Starter bot*
Bot awal dengan strategi dasar yang dapat digunakan dan dikembangkan sesuai dengan strategi yang diinginkan.

Dalam pengembangan bot yang akan dibuat, kita dapat menambahkan strategi greedy kita ke dalam *starter* bot yang terdapat di dalam file “starter-pack\starter-bots\java\src\main\java\za\co\entelect\challenge\Bot.java”. Di sana kita dapat mengimplementasikan strategi greedy dengan memanfaatkan kelas-kelas lain, strategi akan dituliskan di dalam method run yang terdapat pada kelas bot.

Setelah menambahkan strategi ke dalam starter bot, untuk menguji botnya kita memerlukan beberapa aplikasi seperti Java (minimal Java 8), IntelliJ IDEA, dan NodeJS. Lalu, kita memerlukan file *executable* java yang didapatkan melalui build dengan IntelliJ IDEA. Setelah itu, perlu penyesuaian pada game-runner-config.json untuk mengubah lokasi data player-a menjadi “./starter-bots/java”. Lalu, untuk menjalankan *game engine*-nya, kita dapat menjalankan file run.bat sehingga muncul state setiap round dan pemenang dari permainan melalui cmd.

BAB III

APLIKASI STRATEGI GREEDY

3.1. Proses Mapping

Berikut adalah elemen-elemen algoritma *greedy* dalam permasalahan *Overdrive*:

1. Himpunan kandidat : Semua *command* yang tersedia (*NOTHING*, *ACCELERATE*, *DECELERATE*, *TURN_LEFT*, *TURN_RIGHT*, *USE_BOOST*, *USE_OIL*, *USE_LIZARD*, *USE_TWEET* <lane> <block>, *USE_EMP*, *FIX*).
2. Himpunan solusi : Semua *command* yang valid. Misalnya *command TURN_LEFT* hanya bisa digunakan ketika mobil tidak berada pada jalur paling kiri.
3. Fungsi solusi: Memeriksa apakah *command* yang dipilih dapat digunakan.
4. Fungsi seleksi: Memilih *command* yang dapat memperbesar jarak mobil lawan dengan mobil kita (jika mobil kita dalam posisi memimpin) atau memperkecil jarak mobil lawan dengan mobil kita (jika mobil kita dalam posisi tertinggal).
5. Fungsi kelayakan: Memeriksa apakah *command* yang dipilih valid.
6. Fungsi Objektif: Meminimumkan waktu *finish*, memaksimumkan kecepatan mobil, dan memaksimumkan *score*.

3.2. Ekplorasi Alternatif Solusi

Berikut adalah beberapa alternatif solusi yang kita temukan

1. Greedy Berdasarkan *Obstacle* yang Bisa Dihindari.

Pada setiap ronde, mobil bergerak menghindari *obstacle* dengan berbelok atau menggunakan *powerups* yang dimilikinya. Strategi ini berusaha meminimumkan waktu *finish*, memaksimumkan kecepatan mobil, dan memaksimumkan *score* dengan meminimumkan *obstacle* yang mengenainya.

2. Greedy Berdasarkan Kecepatan

Strategi ini berusaha memaksimumkan kecepatan mobil dengan menggunakan *command ACCELERATE* atau *USE_BOOST*. Pada setiap ronde, mobil akan dipercepat jika tidak ada *obstacle* yang menghalanginya.

3. Greedy Berdasarkan Pemakaian Powerups.

Strategi ini berusaha memaksimumkan *score* dengan menggunakan *powerups* yang dimilikinya sebanyak mungkin. Pada setiap ronde, mobil akan berusaha menggunakan *powerups* yang dimilikinya dengan mempertimbangkan kondisi pada ronde tersebut. Berikut adalah pemakaian *powerups* berdasarkan kondisi yang mungkin dialami mobil:

- a. Mobil akan menggunakan *Boost* jika tidak ada *obstacle* yang menghalanginya.
- b. Mobil akan menggunakan *Oil Item* jika pemain lawan berada di belakangnya dan pada jalur yang sama.
- c. Mobil akan menggunakan *Lizard* untuk menghindari *obstacle* yang ada di depannya.

- d. Mobil akan menggunakan *Tweet* jika mobil lawan tidak berada di depan kita karena dapat menyebabkan mobil kita menabrak *Cybertruck*.
- e. Mobil akan menggunakan *EMP* jika mobil lawan berada di depannya.

4. Greedy Berdasarkan Pengambilan Powerups

Strategi ini berusaha memaksimalkan *score* dengan mengambil powerups sebanyak mungkin. Pada setiap ronde, mobil akan berusaha mengambil powerups yang tersedia di jalur lintasan terdekat.

3.3. Analisis Efisiensi

1. Greedy Berdasarkan *Obstacle* yang Bisa Dihindari.

Strategi ini akan mengecek terlebih dahulu *obstacle* yang berada di 20 block ke depan. Lalu jika terdapat *obstacle* pada 20 block terdekat, maka cara menghindarinya akan disesuaikan dengan ketersediaan *powerups* lizard dan kecepatan mobil. Jika kecepatan mobil melebihi dengan jarak *obstacle*, maka mobil akan memutuskan menghindari *obstacle* ke lane yang memiliki *obstacle* terjauh. Sehingga kompleksitas waktu algoritmanya yaitu : $T(n) = 20 * 3 * 20 * 20 = O(1)$

2. Greedy Berdasarkan Kecepatan

Strategi ini mengutamakan penambahan kecepatan dan pemakaian *powerups* boost sesuai dengan jarak *obstacle*. Pada awalnya strategi ini akan mengecek *obstacle* pada 20 block ke depan dan membandingkan kecepatan mobil dengan jarak *obstacle*, jika jarak *obstacle* lebih kecil dari kecepatan mobil, maka bot akan menambah kecepatan. Sehingga kompleksitas waktu algoritmanya yaitu : $T(n) = 20 * 1 = O(1)$

3. Greedy Berdasarkan Pemakaian Powerups.

Strategi ini berfokus pada pemakaian *powerups* yang bersifat menyerang pada pemain lawan. Pemakaiannya akan menyesuaikan dengan posisi lawan sehingga strategi ini awalnya akan mengecek lane musuh dan jika lane musuh tidak sama dengan lane kita maka akan menggunakan *powerups* *tweet* sedang jika lane musuh sama akan menggunakan *powerups* *emp* atau *oil*. Sehingga kompleksitas waktu algoritmanya yaitu : $T(n) = 2 = O(1)$

4. Greedy Berdasarkan Pengambilan Powerups

Strategi ini mengutamakan pada pengambilan *powerups* terlebih dahulu agar *score* yang didapat bisa maksimum. Pada awalnya strategi ini akan mengecek pada 20 block terdekat dan menghitung berapa banyak *powerups* yang tersedia lalu akan dibandingkan dengan keberadaan *powerups* yang terdapat pada lane kiri dan kanan pemain. Sehingga kompleksitas waktu algoritmanya yaitu : $T(n) = 20 * 20 * 20 * 2 = O(1)$

3.4. Analisis Efektivitas

1. Greedy Berdasarkan *Obstacle* yang Bisa Dihindari.

Strategi ini cukup efektif menghindari *obstacle* yang ada selama permainan berlangsung. Pada strategi ini, *powerups* yang berkaitan dengan menangkal *obstacle* secara efektif terpakai. Hal tersebut karena pemakaian *powerups* akan diutamakan terlebih dahulu dibandingkan dengan berpindah jalur untuk menghindari *obstacle*. Strategi ini juga secara efektif menghindari pengurangan skor karena terkena *obstacle*. Namun, strategi ini tidak efektif untuk memperlambat mobil lawan dan menambah skor karena hanya mementingkan *obstacle* yang ada untuk menghindarinya. Selain itu, strategi ini tidak efektif jika sering ditemukan *obstacle* yang jaraknya berdekatan dari posisi mobil saat itu karena perpindahan jalur akan mengurangi jangkauan perpindahan yang dilakukan

2. Greedy Berdasarkan Kecepatan

Strategi ini hanya akan efektif jika di sepanjang jalur yang dilalui mobil jarang terdapat *obstacle*. Sebaliknya, strategi ini tidak akan efektif jika jalur yang dilalui terdapat banyak *obstacle* dengan jarak yang cukup berdekatan. Hal ini karena dalam strategi ini, mobil hanya akan mempercepat kecepatannya atau menggunakan *boost powerup* jika tidak ada *obstacle* yang menghalanginya sesuai pertambahan kecepatannya. Selain itu, strategi ini tidak efektif untuk memperlambat mobil lawan dan menambah skor karena hanya mementingkan peningkatan kecepatan mobil.

3. Greedy Berdasarkan Pemakaian Powerups.

Strategi ini kurang efektif untuk meningkatkan skor karena *powerups* digunakan untuk menyerang mobil lain, tetapi risiko terkena *obstacle* tetap akan cukup besar. Hal ini disebabkan dalam strategi ini, penghindaran *obstacle* hanya akan dilakukan menggunakan *lizard powerup*. Jika mobil tidak mempunyai *lizard powerup*, mobil akan terkena *obstacle* sehingga skor serta kecepatan mobil berkurang dan tingkat kerusakan mobil bertambah.

4. Greedy Berdasarkan Pengambilan Powerups

Strategi ini akan efektif jika diimplementasikan bersamaan dengan strategi nomor tiga. Selain itu, untuk menambah keefektifan, strategi ini juga bisa digunakan Bersama dengan strategi ke-3. Namun, sama seperti strategi yang ke-3, strategi ini masih sangat berisiko untuk mengurangi skor, kecepatan mobil, dan menambah tingkat kerusakan. Hal ini karena pengambilan *powerups* tidak memperhatikan *obstacle* yang ada di jalur tersebut.

3.5.Strategi Greedy yang Dipilih

Berdasarkan analisis efisiensi dan efektivitas yang sudah dipaparkan sebelumnya, kami memilih strategi *greedy* sebagai berikut:

1. Greedy berdasarkan *obstacle* yang bisa dihindari
2. Greedy berdasarkan kecepatan
3. Greedy berdasarkan pemakaian *powerups*

Kami memilih strategi *greedy* di atas karena tiga strategi tersebut jika diimplementasikan bersamaan akan menutupi kekurangan antarstrategi lainnya. Misalnya, strategi *greedy* berdasarkan pemakaian *powerups* berisiko untuk mengurangi skor karena dapat terkena *obstacle* dan strategi *greedy* berdasarkan *obstacle* menghindari *obstacle* tersebut. Selain itu, strategi *greedy* berdasarkan kecepatan akan menutupi kekurangan strategi *greedy* pertama yang bisa mengurangi jangkauan perpindahan mobil.

Dengan begitu, kami mengkombinasikan antara ketiga strategi tersebut yang akan diurutkan berdasarkan prioritas. Prioritas pertama yang kami ambil yaitu strategi *greedy* berdasarkan kecepatan karena aspek penilaian kemenangan dari permainan ini siapa yang lebih dahulu sampai di garis finish. Bot akan memprioritaskan menambah kecepatan hingga max speed beriringan dengan pengecekan *obstacle* sejauh dengan kecepatan sekarang. Jika *obstacle* terdekat melebihi dari speed yang dapat ditingkatkan, maka mobil akan menambah kecepatannya dan jika jarak *obstacle*-nya kurang dari kecepatan saat ini bot akan memakai strategi untuk menghindari *obstacle*.

Dalam menghindari *obstacle*, kami akan menyesuaikan dengan *powerups* yang tersedia jika terdapat *powerups* lizard dan tempat jatuh bot tidak ada *obstacle* maka bot akan menggunakan *powerups* tersebut. Dan sebaliknya jika tidak mempunyai *powerups* tersebut maka bot akan berbelok sesuai dengan lane dan ke arah *obstacle* yang memiliki jarak terjauh. Jika bot terkena *obstacle*, maka bot kami akan menyesuaikan dengan damage yang terkena untuk menentukan apakah perlu untuk diperbaiki terlebih dahulu agar kecepatan di round selanjutnya dapat ditingkatkan secara maksimal. Lalu, untuk penggunaan *powerups* yang bersifat menyerang bot lawan kita akan mengecek posisi lawan dan jika sudah sesuai dengan kriterianya maka langsung akan memakai *powerups* yang tersedia.

BAB IV IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi Algoritma Greedy

Pseudocode dari algoritma *greedy* yang diimplementasikan adalah sebagai berikut:

```
function run() -> Command
{ mengembalikan perintah per round dengan algoritma greedy }

Algoritma:
    if (DAMAGE >= 3) then { Mobil akan diperbaiki terlebih dahulu jika damaganya lebih besar dari 3 }
    |   -> FIX
    endif

    { pakai boost jika tidak ada obstacle yg menghalangi di depannya }
    if (HAS_BOOST and (NEAREST_OBSTACLE_CURRENT_LANE > 15 or NEAREST_OBSTACLE_CURRENT_LANE == 0)) then
    { NEAREST_OBSTACLE = 0 berarti tidak ada obstacle terdekat }
    |   -> BOOST
    endif

    { implementasi percabangan di bawah menggunakan switch pada source code java }

    if (CURRENT_SPEED = 0) then { jika tidak ada speed, naikan kecepatan }
    |   -> ACCELERATE
    { sebelum speed dinaikkan ke state selanjutnya, akan dicek obstacle terdekat pada lane saat itu agar tidak mengenai obstacle }
    else if (CURRENT_SPEED = 3 and NEAREST_OBSTACLE_CURRENT_LANE > 5 || NEAREST_OBSTACLE_CURRENT_LANE = 0) then
    |   -> ACCELERATE
    else if (CURRENT_SPEED = 5 and NEAREST_OBSTACLE_CURRENT_LANE > 6 || NEAREST_OBSTACLE_CURRENT_LANE = 0) then
    |   -> ACCELERATE
    else if (CURRENT_SPEED = 6 and NEAREST_OBSTACLE_CURRENT_LANE > 8 || NEAREST_OBSTACLE_CURRENT_LANE = 0) then
    |   -> ACCELERATE
    else if (CURRENT_SPEED = 8 and NEAREST_OBSTACLE_CURRENT_LANE > 9 || NEAREST_OBSTACLE_CURRENT_LANE = 0) then
    |   -> ACCELERATE
    endif

    { belok hanya untuk menghindari obstacle, selebihnya mengikuti current lane }

    { menghindari obstacle }
    if (NEAREST_OBSTACLE_CURRENT_LANE != 0) then
    |   if (HAS_LIZARD and not LANDING_IN_OBSTACLE) then
    |   |   { cek agar saat landing setelah menggunakan lizard tdk mengenai obstacle }
    |   |   -> LIZARD
    |   endif
    |   if (CURRENT_SPEED >= NEAREST_OBSTACLE_CURRENT_LANE) then
    |   |   { jika akan menabrak obstacle }
    |   |   if (MY_POSITION_LANE = 4) then
    |   |   |   if (NEAREST_OBSTACLE_CURRENT_LANE >= NEAREST_OBSTACLE_LEFT_LANE) && NEAREST_OBSTACLE_LEFT_LANE != 0)) then
    |   |   |   |   { jika obstacle di lane saat ini lebih jauh daripada obstacle di lane kirinya }
    |   |   |   |   -> ACCELERATE
    |   |   |   endif
    |   |   |   -> TURN_LEFT
    |   |   else if (MY_POSITION_LANE = 1) then
    |   |   |   if (NEAREST_OBSTACLE_CURRENT_LANE >= NEAREST_OBSTACLE_RIGHT_LANE) && NEAREST_OBSTACLE_RIGHT_LANE != 0)) then
    |   |   |   |   { jika obstacle di lane saat ini lebih jauh daripada obstacle di lane kanannya }
    |   |   |   |   -> ACCELERATE
    |   |   |   endif
    |   |   |   -> TURN_RIGHT
    |   |   else { jika di 2 lane tengah (lane 2 dan 3) }
    |   |   |   { menghindari dgn belok ke lane yang memiliki obstacle terjauh }
    |   |   |   if (NEAREST_OBSTACLE_RIGHT_LANE = 0) then
    |   |   |   |   -> TURN_RIGHT
    |   |   |   else if (NEAREST_OBSTACLE_LEFT_LANE = 0) then
    |   |   |   |   -> TURN_LEFT
    |   |   |   else if (NEAREST_OBSTACLE_LEFT_LANE <= NEAREST_OBSTACLE_RIGHT_LANE) then { obstacle di lane kanan lebih jauh }
    |   |   |   |   -> TURN_RIGHT
```

```

        else { obstacle di lane kiri lebih jauh }
            -> TURN_LEFT
        endif
    endif
endif
endif

{ setelah coba menghindari obstacle, pakai powerups yg ada }

{ ageresif (penggunaan powerups untuk menyarang lawan) }

if (HAS_TWEET and MY_POSITION_LANE != OPPONENT_POSITION_LANE) then
    { cek agar cybertruck tdk di-spawn di lane saat ini }
    -> TWEET
endif

if (MY_POSITION_LANE = OPPONENT_POSITION_LANE) then
    if (HAS_OIL and MY_POSITION_BLOCK > OPPONENT_POSITION_BLOCK) then
        { jika posisi mobil lebih jauh dari lawan }
        -> OIL
    endif
    if (HAS_EMP and MY_POSITION_BLOCK < OPPONENT_POSITION_BLOCK) then
        { jika posisi lawan lebih jauh }
        -> EMP
    endif
endif

{ default ACCELERATE }
-> ACCELERATE

```

4.2.Struktur Data

Struktur data yang kami gunakan dalam bot Overdrive secara garis besar terbagi menjadi tiga, yaitu command, entities, dan enum.

1. Modul Command

Modul ini berisi perintah-perintah yang berfungsi untuk menjalankan bot mobil. Perintah-perintah tersebut diimplementasikan dalam masing-masing kelas objek.

- a. *Class Command* merupakan kelas abstrak yang akan diimplementasikan di kelas turunannya.
- b. *Class AccelerateCommand*, perintah ini digunakan untuk mempercepat bot mobil ke level kecepatan selanjutnya.
- c. *Class BoostCommand*, perintah ini digunakan untuk mempercepat kecepatan mobil sampai level kecepatan *boost*.
- d. *Class ChangeLaneCommand*, perintah ini digunakan untuk mengubah jalur bot dengan berbelok ke kiri atau ke kanan dari jalur yang sedang dilalui.
- e. *Class DecelerateCommand*, perintah ini digunakan untuk memperlambat kecepatan mobil ke level kecepatan dibawahnya.
- f. *Class DoNothingCommand*, perintah ini digunakan agar bot mobil tidak melakukan aktivitas apa-apa, tetapi tetap melaju sesuai keadaannya.
- g. *Class EmpCommand*, perintah ini digunakan untuk menyerang bot lawan dengan menembakkan gelombang *emp* ke arah depan.
- h. *Class FixCommand*, perintah ini digunakan untuk memperbaiki kerusakan yang diterima mobil akibat terkena *obstacle*.
- i. *Class LizardCommand*, perintah ini digunakan untuk menghindari *obstacle* yang ada di depannya dengan melompati *obstacle* tersebut.
- j. *Class OilCommand*, perintah ini digunakan untuk menyerang bot lawan dengan meletakkan oli ditempat bot kita berada.

- k. *Class TweetCommand*, perintah ini digunakan untuk menyerang bot lawan dengan meletakkan CyberTruck di posisi tertentu.

2. Modul Entities

Modul ini berisi objek-objek yang berada di gamestate.

- a. *Class Car*, kelas ini merupakan objek mobil yang berisi informasi tentang *id*, *position*, *speed*, *state*, *damage*, *powerups*, *boosting*, dan *boostCounter*.
- b. *Class GameState*, kelas ini berisi informasi tentang *currentRound*, *maxRounds*, *player*, *opponent*, dan *worldMap*.
- c. *Class Lane*, kelas ini berisi informasi tentang *position*, *terrain*, dan *occupiedByPlayerId*.
- d. *Class Position*, kelas ini berisi informasi tentang letak bot mobil kita, yaitu *lane* dan *block*.

3. Modul Enums

Modul ini berfungsi sebagai alat iterasi yang digunakan untuk memeriksa kemungkinan yang ada.

- a. *Enum Direction*, berisi data arah, yaitu *FORWARD*, *BACKWARD*, *LEFT*, dan *RIGHT*.
- b. *Enum PowerUps*, berisi data powerups, yaitu *BOOST*, *OIL*, *TWEET*, *Lizard*, dan *EMP*.
- c. *Enum State*, berisi data keadaan bot, yaitu *ACCELERATING*, *READY*, *NOTHING*, *TURNING_RIGHT*, *TURNING_LEFT*, *HIT_MUD*, *HIT_MUD*, *DECELERATING*, *PICKED_UP_POWERUP*, *USED_BOOST*, *USED_OIL*, *FINISHED*.
- d. *Enum Terrain*, berisi *EMPTY*, *MUD*, *OIL_SPILL*, *OIL_POWER*, *FINISH*, *BOOST*, *WALL*, *LIZARD*, *TWEET*, *EMP*.

4. Bot.java

File Bot.java berisi algoritma bot yang dibuat oleh pemain.

5. Main.java

File Main.java berisi mekanisme pembacaan state dan perintah eksekusi state.

4.3. Analisis Solusi

Berdasarkan strategi *greedy* yang kami pilih, berikut merupakan hasil pengujian strategi yang telah diimplementasikan, yaitu :

1. Pengujian 1

Round Details Max Rounds: 600 Current Round: 10

A - Belum Kepikiran (selected) ✓

Position	Speed	Lane	Distance	Boosts	Boosting	Powerups
1	9	1	78	0	No	EMP

Bot Command
 Command: ACCELERATE
 Execution time: 4ms
 Exception: null

Pada tiap rondanya, strategi dari bot kami akan memprioritaskan untuk menambahkan kecepatan seiring dengan pengecekan obstacle ke blocks yang ada di depan. Jika jarak dari *obstacle* melebihi dari kecepatan terkini, maka bot akan menambah kecepatan hingga max speed.

2. Pengujian 2

[82, 1]	[83, 1]	[84, 1]	[85, 1]	[86, 1]	[87, 1] <PLAYER>	[88, 1]	[89, 1]	[90, 1]	[91, 1] <LIZARD>	[92, 1] <SLUD>	[93, 1]	[94, 1]	[95, 1]	[96, 1]	[97, 1]	[98, 1]	[99, 1]	[100, 1]
[82, 2] <SLUD>	[83, 2] <SLUD>	[84, 2] <SLUD>	[85, 2]	[86, 2]	[87, 2]	[88, 2]	[89, 2]	[90, 2]	[91, 2]	[92, 2]	[93, 2]	[94, 2] <SLUD>	[95, 2]	[96, 2]	[97, 2]	[98, 2]	[99, 2] <SLUD>	[100, 2]
[82, 3]	[83, 3]	[84, 3]	[85, 3]	[86, 3]	[87, 3]	[88, 3]	[89, 3]	[90, 3]	[91, 3]	[92, 3] <LIZARD>	[93, 3]	[94, 3]	[95, 3]	[96, 3] <SLUD>	[97, 3] <SLUD>	[98, 3]	[99, 3]	[100, 3]
[82, 4]	[83, 4]	[84, 4]	[85, 4]	[86, 4]	[87, 4] <TWEST>	[88, 4]	[89, 4]	[90, 4]	[91, 4]	[92, 4]	[93, 4]	[94, 4]	[95, 4]	[96, 4]	[97, 4]	[98, 4]	[99, 4]	[100, 4]

< First < Prev 11 Next > Last >








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

Round Details

Max Rounds: 600

Current Round: 11

A - Belum Kepikiran
(selected) v

						
Position 1 <small>[x: 87, y: 1]</small>	Speed 9	Lane 1	Distance 87	Boosts 0	Boosting No	Powerups EMP

Bot Command
 Command: TURN_RIGHT
 Execution time: 5ms
 Exception: null

Ketika pengecekan jarak obstacle kurang dari kecepatan terkini, maka bot akan memilih antara berbelok atau menabrak *obstacle* yang ada di depan. Penentuan hal ini berdasarkan pada ketersediaan *powerups lizard*, lane yang kosong, dan jarak *obstacle*-nya. Pada kondisi ini bot akan memilih lane dengan obstacle yang terjauh jika tidak memiliki *powerups lizard*.

3. Pengujian 3

[201, 1]	[202, 1]	[203, 1]	[204, 1]	[205, 1]	[206, 1] <PLAYER>	[207, 1]	[208, 1]	[209, 1]	[210, 1]	[211, 1]	[212, 1] <SLUD>	[213, 1] <SLUD>	[214, 1]	[215, 1]	[216, 1]	[217, 1] <SLUD>	[218, 1]	[219, 1]
[201, 2]	[202, 2]	[203, 2]	[204, 2]	[205, 2]	[206, 2]	[207, 2]	[208, 2]	[209, 2]	[210, 2]	[211, 2]	[212, 2]	[213, 2]	[214, 2] <SLUD>	[215, 2] <SLUD>	[216, 2]	[217, 2]	[218, 2]	[219, 2]
[201, 3]	[202, 3]	[203, 3]	[204, 3]	[205, 3]	[206, 3]	[207, 3]	[208, 3]	[209, 3]	[210, 3]	[211, 3]	[212, 3]	[213, 3]	[214, 3]	[215, 3]	[216, 3]	[217, 3]	[218, 3]	[219, 3]
[201, 4]	[202, 4]	[203, 4]	[204, 4]	[205, 4]	[206, 4]	[207, 4]	[208, 4] <SLUD>	[209, 4] <SLUD>	[210, 4]	[211, 4]	[212, 4]	[213, 4]	[214, 4] <SLUD>	[215, 4]	[216, 4]	[217, 4]	[218, 4]	[219, 4]

< First < Prev 29 Next > Last >








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

Round Details

Max Rounds: 600

Current Round: 29

A - Belum Kepikiran
(selected) v

						
Position 1 <small>[x: 206, y: 1]</small>	Speed 15	Lane 1	Distance 206	Boosts 3	Boosting YES	Powerups EMPLIZARD

Bot Command
 Command: USE_LIZARD

Ketika pengecekan jarak *obstacle* kurang dari kecepatan terkini dan memiliki *powerups lizard*, maka bot akan menggunakan *powerups* tersebut dan hal ini akan menguntungkan saat permainan karena jika lurus ataupun belok pada keadaan ini akan tetap menabrak *obstacle*.

4. Pengujian 4

Position	Speed	Lane	Distance	Boosts	Boosting	Powerups
1	9	1	161	0	No	EMP, TWEET, BOOST, LIZARD

Bot Command
Command: FIX

Ketika bot mendapatkan *powerups boost*, bot kami akan memprioritaskan penggunaan *powerups* ini. Sebelum penggunaannya, bot kami akan mengecek terlebih dahulu apakah terdapat *damage* pada mobil. Jika terdapat *damage* maka bot kami akan memperbaiki terlebih dahulu untuk menggunakan *boost* di round selanjutnya agar maksimum *speed* yang didapatkan akan maksimal.

BAB V

KESIMPULAN DAN SARAN

5.1.Kesimpulan

Pada tugas besar 1 IF 2211 yang berjudul “Pemanfaatan Algoritma Greedy dalam Aplikasi Permainan Overdrive”, kami berhasil mengimplementasikan algoritma greedy yang telah dibuat ke dalam bot yang tersedia. Algoritma greedy yang kami pilih merupakan gabungan dari beberapa strategi agar dapat menutupi kekurangan antar strategi yang satu dengan yang lainnya.

5.2.Saran

Strategi yang kami pilih dalam pembuatan bot ini tentunya masih memiliki banyak kekurangan yang bisa dikembangkan lagi, sebaiknya sebelum menentukan strategi yang akan dipilih lebih banyak mengeksplorasi alternatif-alternatif strategi yang lainnya. Selain itu, dalam pengimplementasiannya perlu rajin untuk mendebug tiap rondanya agar mengetahui implementasi yang dibuat sudah sesuai atau belum.

DAFTAR LINK

Link Github :

<https://github.com/ikmalalfaozi/Tubes-Stima-1>

Link Video :

<https://youtu.be/kGU9vRHYSb0>

DAFTAR PUSTAKA

Munir, Rinaldi. 2021. “Algoritma *Greedy* (Bagian 1)”
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf)