# LAPORAN TUGAS BESAR

# IF2211 STRATEGI ALGORITMA

# PEMANFAATAN ALGORITMA GREEDY DALAM APLIKASI PERMAINAN “OVERDRIVE”

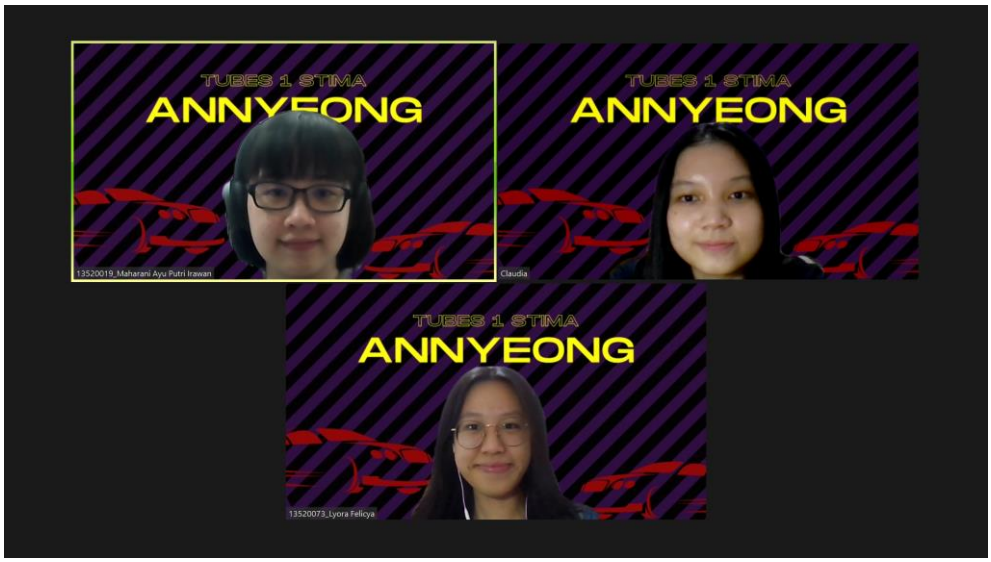

Disusun oleh:

Kelompok 22 - Annyeong

Maharani Ayu Putri Irawan 13520019

Lyora Felicya 13520073

Claudia 13520076


## SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

## INSTITUT TEKNOLOGI BANDUNG

## 2022

# 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 1.1 Ilustrasi Permainan *Overdrive***

Pada tugas besar pertama Strategi Algoritma ini, kami diminta untuk mengimplementasikan bot mobil dalam permainan *Overdrive*. Bot yang dibuat harus memanfaatkan strategi *greedy* untuk memenangkan permainan. Implementasi bot dapat dilakukan dengan melanjutkan program yang terdapat pada starter-bots di dalam starter-pack pada laman https://github.com/EntelectChallenge/2020-Overdrive/releases/tag/2020.3.4. Pengerjaan tugas besar ini mengikuti paduan singkat sebagai berikut.

1. Unduh *latest release* starter pack.zip Entelect Challenge 2020.
2. Untuk menjalankan permainan, dibutuhkan beberapa *requirement* dasar sebagai berikut.
   a. Java (minimal Java 8)
   b. IntelIiJ IDEA.
   c. NodeJS
3. Buka file “run.bat” untuk menjalankan permainan
4. Secara *default*, permainan akan dilakukan diantara *reference bot* dan *starter bot* yang disediakan. Untuk mengubah hal tersebut, silahkan edit file “game-runner-config.json”. Anda juga dapat mengubah file “bot.json” dalam direktori “starter-bots” untuk mengatur informasi terkait bot anda.

5. Modifikasilah bot yang disediakan di starter-bots. Bot harus menggunakan bahasa Java dan di-build menggunakan IntelliJ sebelum menjalankan permainan kembali. Dilarang menggunakan kode program yang sudah ada untuk pemainnya atau kode program lain yang diunduh dari Internet. Mahasiswa harus membuat program sendiri, tetapi belajar dari program yang sudah ada tidak dilarang.
6. Hasil permainan dapat dilihat dengan menggunakan visualizer berikut https://github.com/Affuta/overdrive-round-runner

Strategi *greedy* yang diimplementasikan harus dikaitkan dengan fungsi objektif dari permainan itu sendiri, yaitu untuk 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. Hasil akhir yang diharapkan adalah terbentuknya suatu bot untuk permainan Overdrive ini yang memanfaatkan strategi greedy dalam proses pemrogramannya untuk memenangkan permainan.

# BAB 2 – LANDASAN TEORI

## Dasar Teori *Greedy* secara Umum

Algoritma greedy merupakan metode yang populer dan sederhana untuk memecahkan persoalan optimasi. Dalam hal ini, hanya terdapat dua macam persoalan optimasi, yaitu maksimasi dan minimasi. Algoritma ini memecahkan persoalan secara langkah per langkah (step by step) sedemikian sehingga, pada setiap langkah dilakukan pengambilan pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensinya ke depan (prinsip "take what you can get now!") dan "berharap" bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global. Optimum lokal ini dapat ditentukan dari banyak faktor tergantung persoalan, seperti contohnya selalu mengambil pilihan dengan nilai variabel tertentu yang terbesar. Elemen-elemen algoritma greedy:

1. Himpunan kandidat, C : berisi kandidat yang akan dipilih pada setiap Langkah
2. Himpunan solusi, S : berisi kandidat yang sudah dipilih
3. Fungsi solusi: menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi
4. Fungsi seleksi (*selection function*): memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik.
5. Fungsi kelayakan (*feasible*): memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak)
6. Fungsi objektif : memaksimumkan atau meminimumkan

Berikut merupakan skema umum algoritma greedy :

```
function greedy(C : himpunan_kandidat) → himpunan_solusi
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy }
Deklarasi
   x : kandidat
   S : himpunan_solusi

Algoritma:
   S ← {}      { inisialisasi S dengan kosong }
   while (not SOLUSI(S)) and (C ≠ {} ) do
         x ← SELEKSI(C)     { pilih sebuah kandidat dari C}
         C ← C – {x}        { buang x dari C karena sudah dipilih }
         if LAYAK(S ∪ {x}) then      { x memenuhi kelayakan untuk dimasukkan ke dalam himpunan solusi }
            S ← S ∪ {x}      { masukkan x ke dalam himpunan solusi }
         endif
   endwhile
   {SOLUSI(S) or C = {} }

   if SOLUSI(S) then     { solusi sudah lengkap }
      return S
   else
      write('tidak ada solusi')
   endif
```

## Cara Kerja Program Secara Umum

Untuk menjalankan permainan dibutuhkan sebuah game engine. Game engine yang digunakan untuk menjalankan bot tidak dirancang dari awal melainkan sudah disediakan dan dapat diunduh melalui https://github.com/EntelectChallenge/2020-Overdrive. Game engine ini akan digunakan menyimpan state-state dalam permainan dan menjalankan permainan. Menjalankan program setelah mengunduh starter-kit dapat dilakukan dengan membuka file “run.bat” pada sistem operasi Windows atau membuka root dari starter-pack dan menjalankannya dengan make run pada terminal untuk sistem operasi Linux.

Program ini terdiri atas 4 hal utama yaitu:

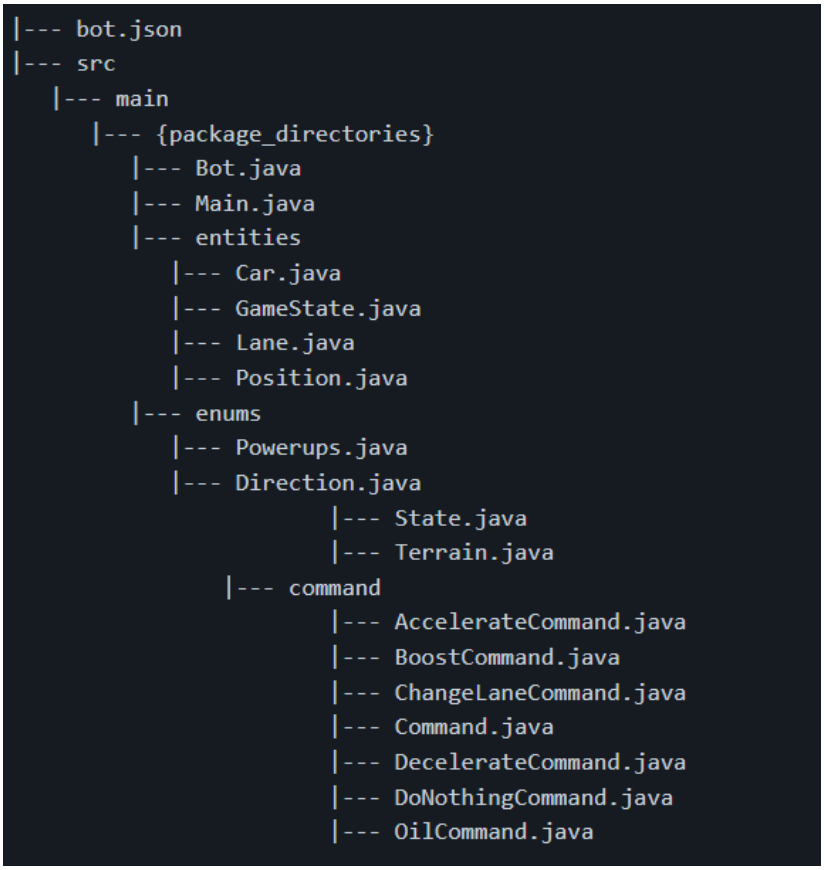
- ***game-engine*** yang berfungsi untuk menerapkan aturan pada bot ke game-state
- ***game-runner*** yang berfungsi untuk menjalankan pertandingan antarbot dan memanggil perintah yang sesuai dengan perintah yang dikeluarkan bot agar dieksekusi oleh game-engine
- ***reference-bot*** yang digunakan untuk bertanding dengan starter-bot
- ***starter-bot*** yang merupakan bot yang harus diimplementasikan dengan algoritma Greedy

Awalnya, permainan akan berlangsung antara reference-bot dan starter-bot yang sudah disediakan. Namun, reference-bot dan starter-bot dapat diganti sehingga permainan bisa berlangsung dengan menggunakan bot yang sudah dikembangkan sendiri. Cara untuk mengganti bot yang digunakan dalam pertandingan adalah dengan memodifikasi file game-runner-config.json yaitu pada player-a dan player-b diganti dengan direktori menuju file

bot.json yang diinginkan. Selain itu kita juga bisa mengganti identitas seperti nickname bot dan lain-lain pada file bot.json.

Bot dapat melakukan pertandingan dengan bantuan dari game runner sebagai perantara bagi bot dan game engine. Bot melakukan komunikasi dengan stdin dan stdout. Angka babak pertandingan akan dibaca sebagai stdinput lalu file state.json yang menyimpan peta dan informasi mengenai pertandingan akan dibaca, setelah itu bot akan memilih command yang akan dieksekusi sesuai dengan strategi yang sudah ditetapkan. Command yang dikeluarkan oleh bot sebagai stdout akan dibaca dan dieksekusi oleh game runner.

Algoritma Greedy akan diimplementasikan pada starter-bot menggunakan bahasa Java. Awalnya starter-bot hanya berisi logika sederhana namun nantinya akan diubah dan dikembangkan sesuai dengan strategi Greedy yang sudah dipilih. Struktur starter-bot dapat dilihat pada gambar di bawah.




**Gambar 2.1 Struktur Folder bot berbahasa Java**

Implementasi strategi *Greedy* dilakukan pada file Bot.java, program yang berisi implementasi strategi *Greedy* dimasukkan pada method run(), selain itu juga ada method-method tambahan yang digunakan untuk membantu implementasi program.

# BAB 3 – APLIKASI STRATEGI GREEDY

## Proses *Mapping* Persoalan *Overdrive* menjadi Elemen Algoritma *Greedy*

Dalam mengimplementasikan strategi greedy pada bot yang terdapat dalam starter-bot, kami telah mendesain strategi greedy by command yang merupakan kombinasi dari beberapa faktor pertimbangan untuk memenangkan permainan Overdrive, yaitu untuk mencapai objektif utamanya dalam mencapai garis finish. Adapun uraian dari mapping persoalan overdrive menjadi elemen algoritma Greedy adalah sebagai berikut :

1. **Himpunan kandidat**

   Command-command yang digunakan dalam permainan yaitu :
   {ACCELERATE, DECELERATE, TURN_LEFT, TURN_RIGHT, USE_BOOST, USE_OIL, USE_TWEET, USE_LIZARD, DO_NOTHING, USE_EMP}
2. **Himpunan solusi:**

   Command yang terpilih pada setiap round yang bertujuan untuk memenangkan permainan Overdrive
3. **Fungsi solusi:**

   Menentukan command yang paling menguntungkan
4. **Fungsi seleksi:**

   Memilih command yang paling menguntungkan dengan urutan prioritas apakah command tersebut akan memberikan kecepatan terbaik, jika command yang dipertimbangkan memberikan kecepatan yang sama, maka akan dibandingkan lagi dengan urutan prioritas command yang memberikan lokasi x terjauh, mengambil power up terbanyak, memberikan damage terkecil, dan memberikan score terbesar
5. **Fungsi kelayakan:**

   Memeriksa apakah tersedia power up untuk mengaktifkan command power up, memeriksa apakah bot masih bisa belok ke kiri atau ke kanan,
6. **Fungsi objektif:**

   Memenangkan permainan dengan mencapai garis finish terlebih dahulu

## Eksplorasi dan Analisis Efisiensi serta Efektivitas Alternatif Solusi *Greedy* yang mungkin dipilih dan Strategi *Greedy* yang digunakan

Sebelum akhirnya menemukan strategi greedy yang menurut kami paling efektif, kelompok kami melakukan banyak eksplorasi mengenai kemungkinan alternatif solusi greedy yang dapat diimplementasikan pada starter bot yang disediakan. Alternatif tersebut terdiri dari ***greedy by***

***score, greedy by position, greedy by speed,*** **dan** ***greedy by powerups***. Berikut penjelasan mengenai hasil eksplorasi kami untuk setiap alternatif solusi greedy, termasuk analisis efisiensi dan efektivitasnya :

***1. Greedy by Speed***

Strategi greedy by speed ini merupakan strategi yang pertama kali kami pikirkan. Hal ini karena tujuan utama dari permainan overdrive ini adalah untuk mencapai garis finish terlebih dahulu. Apabila kedua mobil mencapai garis finish disaat yang bersamaan, mobil yang memiliki kecepatan lebih besar yang akan memenangkan permainan. Untuk itu, strategi greedy by speed ini memiliki objektif untuk memaksimalkan kecepatan mobil dan mencapai garis finish. Mobil memiliki initial speed sebesar 5 dan dapat ditingkatkan menjadi speed state berikutnya hingga mencapai speed maksimal yaitu 9. Mobil juga dapat memiliki kecepatan sebesar 15 selama 5 round saat menggunakan powerup boost. Namun, speed mobil dapat berkurang saat melewati block yang berisi mud, oil spill, wall, atau rintangan lainnya. Pada implementasinya, kami mengutamakan command Accelerate pada setiap round dan menggunakan powerup Boost apabila memungkinkan, serta sebisa mungkin menghindari block-block yang berisi mud, oil spill, dan wall. Namun akhirnya kami memutuskan untuk tidak menggunakan strategi ini karena kami menganggap bahwa strategi ini akan menjadi kurang efisien karena tidak memanfaatkan powerup lainnya yang disediakan. Selain itu, peta permainan terdiri dari 4 lane sepanjang 1500 block, sepanjang map pasti akan terdapat banyak sekali block rintangan yang posisinya tidak dapat diprediksi, akan terdapat kondisi-kondisi saat command accelerate atau boost tidak dapat digunakan, hingga akhirnya mobil akan fokus menghindari block-block tersebut dan justru menjadi tidak efisien. Perkiraan kompleksitas untuk algoritma ini O(n).

***2. Greedy by Score***

Selain mencapai garis finish, faktor lainnya yang menjadi penentu pemenang dari permainan Overdrive ini adalah score akhir yang didapatkan oleh masing-masing pemain. Score digunakan sebagai penentu kemenangan apabila kedua mobil mencapai garis finish disaat yang bersamaan dan memiliki kecepatan seimbang. Oleh karena itu, kelompok kami merancang strategi greedy by score sebagai salah satu alternatif solusi. Strategi ini berfokus pada memaksimalkan score yang diperoleh dan meminimalisir pengurangan score yang didapatkan. Kita memperoleh skor 4 setiap kali mengambil sebuah powerup, dan mendapatkan 4 lagi setiap kali menggunakan powerup. Pengurangan skor dilakukan

saat mobil menabrak wall, melewati oil spill atau mud, serta melakukan command yang invalid seperti menggunakan powerup saat tidak memilikinya, ataupun command turn left dan turn right saat sudah berada di lane paling ujung. Dengan demikian, implementasi greedy by score ini akan dilakukan dengan mengambil setiap powerups yang ada dan selalu menggunakannya setiap memenuhi kondisi. Strategi ini sebenarnya kurang efektif karena memperoleh skor yang besar tidak 100% menjamin kemenangan, seperti yang telah disebutkan sebelumnya, skor bukanlah prioritas penentu kemenangan. Skor akan menjadi penentu kemenangan hanya pada saat kedua mobil mencapai garis finish bersamaan dan memiliki besar kecepatan yang sama. Oleh karena itu maka kelompok kami memutuskan untuk tidak menggunakan strategi greedy by score, untuk analisis kompleksitasnya kami perkiraan O(n).

***3. Greedy by Powerups***

Pada permainan Overdrive ini, terdapat beberapa powerups yang dapat digunakan untuk memberi rintangan pada lawan dan memperlambat kecepatan mobil lawan. Strategi ini akan berfokus pada memaksimalkan penggunaan powerup untuk mengganggu mobil pemain lawan dan memperlambat kecepatan mobilnya. Implementasi strategi greedy by powerups ini berfokus pada 3 powerups, yaitu oil, tweet, dan emp. Ketiga powerup tersebut memiliki efek yang dapat memperlambat mobil lawan dan menghentikan pergerakan mereka. Powerup oil digunakan dengan meletakkan oil spill diblock mobil kita berada, dan menurunkan kecepatan mobil lawan ke state sebelumnya apabila dilewati, Tweet digunakan dengan meletakkan tweet di block yang kita inginkan, jika lawan menabrak cybertruck yang diletakkan, maka mereka akan berhenti pada round tersebut dan kecepatannya menjadi 3. Powerupditembakkan ke jalur depan, kiri, dan kanan mobil kita. EMP akan menghentikan mobil lawan pada round itu dan menurunkan kecepatan mobil lawan menjadi 3. Perlu diperhatikan bahwa strategi penggunaan powerup ini mengharuskan kita untuk mencari lokasi yang paling strategis, yaitu lokasi yang kemungkinan besar akan dilewati lawan. Hal ini menjadikan strategi greedy by powerups menjadi tidak efektif karena sulit untuk memprediksi pergerakan lawan pada round tersebut, mengingat command dari kedua pemain akan dijalankan bersamaan. Maka dari itu kelompok kami tidak menggunakan strategi greedy by powerups untuk diimplementasikan pada bot mobil kelompok kami. Kompleksitas untuk algoritma ini kami perkirakan O(n).

4. ***Greedy by Damage***

Selama permainan berlangsung, mobil kita dapat menerima damage apabila tidak berhati-hati dalam memperhitungkan pergerakan. Akumulasi damage yang diterima akan membatasi kecepatan maksimal mobil. Damage maksimal yang dapat diterima adalah 5, pada damage sebesar ini mobil akan memiliki kecepatan 0 dan harus segera di perbaiki agar dapat kembali berjalan. Ketika menggunakan command fix, mobil tidak akan maju pada round tersebut. Oleh karena itu semakin banyak damage yang diterima akan semakin sering digunakan command fix yang kemudian akan menyebabkan mobil kita ketinggalan dari mobil lawan. Implementasi dari strategi ini akan berfokus pada meminimalisir damage yang diterima oleh mobil. Maka, fokus utamanya adalah untuk menghindari block yang berisi oil spill, mud, ataupun wall. Namun strategi greedy by damage ini tidak efisien karena menurut kami, untuk menghindari block-block tersebut juga dapat diterapkan saat menggunakan strategi greedy lainnya yang lebih baik, tidak hanya mengutamakan damage seminimal mungkin. Perkiraan kompleksitas untuk algoritma ini O(n).

5. ***Greedy by Command***

Strategi greedy terakhir yang didapatkan dari hasil eksplorasi kami adalah greedy by command. Pada strategi ini, kami mengutamakan command yang dapat memberi keuntungan terbesar pada round tersebut. Faktor-faktor yang menjadi perbandingan terdiri dari 5, yaitu speed after, damage, location, powerups, dan score. Implementasi dari algoritma ini adalah dengan memeriksa terlebih dahulu command apa saja yang valid dan dapat digunakan pada saat round tersebut, kemudian membandingkan faktor-faktor tadi dan memilih command yang terbaik untuk digunakan. Misalnya pada suatu round, sudah ada himpunan command yang valid untuk digunakan. Untuk memilih command terbaik yang akan dipakai maka akan dibandingkan terlebih dahulu command yang memberikan kecepatan yang lebih besar. Setelah mendapatkan command yang memberikan kecepatan yang lebih besar, maka command tersebut akan digunakan. Jika setelah dilakukan perbandingan ada beberapa command yang akan menghasilkan kecepatan yang sama maka akan dibandingkan lagi command mana yang akan memberikan lokasi x terjauh, jika masih sama maka akan dilanjutkan perbandingan dengan prioritas command yang akan mengambil power up lebih banyak, damage lebih kecil, dan score terbesar. Menurut kami strategi ini adalah strategi yang paling efektif karena tidak hanya mementingkan satu aspek saja.

Setelah melakukan analisis terhadap seluruh alternatif solusi greedy yang kami temukan, akhirnya kami memilih strategi *greedy by command* untuk diimplementasikan pada bot kami. Menurut kami, strategi ini merupakan strategi yang terbaik karena merupakan gabungan dari faktor-faktor perhitungan yang terdapat pada strategi greedy lainnya sehingga tidak hanya terpaku pada satu sisi saja. Terdapat lima parameter yang menjadi pertimbangan bot dalam melakukan perhitungan sebelum memberikan command. Melalui strategi *greedy by command* ini, bot yang telah diimplementasikan dapat memaksimalkan speed yang menjadi prioritas utama kami dan sekaligus mempertimbangkan faktor lainnya agar mendapatkan dampak negatif yang seminimal mungkin dan mendapat keuntungan yang maksimal dengan memanfaatkan command yang tersedia pada round tertentu. Analisis kompleksitasnya kami perkirakan O(n).

# BAB 4 – IMPLEMENTASI DAN PENGUJIAN

## Implementasi algoritma Greedy pada program

```
// Mengambil array of blocks di depan bot mobil
blocks <- getBlocksInFront(myCar.position.lane,
myCar.position.block)

// Inisiasi untuk melakukan komparasi command terbaik
// Diawali dengan command DO_NOTHING
temp_best <- DO_NOTHING

// Cek apakah command ACCELERATE valid
if (command ACCELERATE valid) then
      // Jika valid, bandingkan dengan temp_best (command
DO_NOTHING)
      // Jika command ACCELERATE lebih menguntungkan daripada
command DO_NOTHING
      if (advantage ACCELERATE > advantage temp_best ) then
            // temp_best diupdate untuk dibandingkan dengan command
selanjutnya
            temp_best <- ACCELERATE

// Cek apakah command DECELERATE valid
if (command DECELERATE valid) then
      // Jika valid, bandingkan dengan temp_best
      if (advantage DECELERATE > advantage temp_best ) then
            // temp_best diupdate untuk dibandingkan dengan command
selanjutnya
            temp_best <- DECELERATE

// Cek apakah command TURN_LEFT valid
if (command TURN_LEFT valid) then
      // Jika valid, bandingkan dengan temp_best
      if (advantage TURN_LEFT > advantage temp_best ) then
            // temp_best diupdate untuk dibandingkan dengan command
selanjutnya
            temp_best <- TURN_LEFT

// Cek apakah command TURN_RIGHT valid
if (command TURN_RIGHT valid) then
      // Jika valid, bandingkan dengan temp_best
      if (advantage TURN_RIGHT > advantage temp_best ) then
            // temp_best diupdate untuk dibandingkan dengan command
selanjutnya
            temp_best <- TURN_RIGHT

// Cek apakah command USE_BOOST valid
```

```
if (command USE_BOOST valid) then
      // Jika valid, bandingkan dengan temp_best
      if (advantage USE_BOOST > advantage temp_best ) then
            // temp_best diupdate untuk dibandingkan dengan command
selanjutnya
            temp_best <- USE_BOOST

// Cek apakah command USE_OIL valid
if (command USE_OIL valid) then
      // Jika valid, bandingkan dengan temp_best
      if (advantage USE_OIL > advantage temp_best ) then
            // temp_best diupdate untuk dibandingkan dengan command
selanjutnya
            temp_best <- USE_OIL

// Cek apakah command USE_LIZARD valid
if (command USE_LIZARD valid) then
      // Jika valid, bandingkan dengan temp_best
      if (advantage USE_LIZARD > advantage temp_best ) then
            // temp_best diupdate untuk dibandingkan dengan command
selanjutnya
            temp_best <- USE_LIZARD

// Cek apakah command USE_EMP  valid
if (command USE_EMP valid) then
      // Jika valid, bandingkan dengan temp_best
      if (advantage USE_EMP > advantage temp_best ) then
            // temp_best diupdate untuk dibandingkan dengan command
selanjutnya
            temp_best <- USE_EMP

// Cek apakah command FIX valid
if (command FIX valid) then
      // Jika valid, bandingkan dengan temp_best
      if (advantage FIX > advantage temp_best ) then
            // temp_best diupdate untuk dibandingkan dengan command
selanjutnya
            temp_best <- FIX

// Cek apakah command USE_TWEET valid
if (command USE_TWEET valid) then
      // Jika valid, bandingkan dengan temp_best
      if (advantage USE_TWEET > advantage temp_best ) then
            // temp_best diupdate untuk dibandingkan dengan command
selanjutnya
            temp_best <- USE_TWEET

// return best command
```

```
-> temp_best
```

## Penjelasan Struktur Data pada Program

Secara garis besar, struktur data yang kami gunakan pada program *Overdrive* ini terbagi menjadi 3 bagian, yaitu *command*, *entities*, *enums* yang disertakan dengan file bot.java dan main.java. Berikut merupakan penjelasan singkat dari setiap struktur data yang ada.

| Struktur Data dan Kelas | Penjelasan singkat |
|---|---|
| Bot.java | Terdapat metode utama yaitu run() dimana strategi *greedy* diimplementasikan untuk menjalankan bot sesuai perintah. |
| Main.java | Program utama dari permainan untuk menghubungkan kelas-kelas yang ada dengan *game engine.* |
| ***Package : Command*** | |
| AccelerateCommand.java | Deklarasi kelas untuk menjalankan *command Accelerate.* |
| BoostCommand.java | Deklarasi kelas untuk menjalankan *command* USE_BOOST, yaitu perintah untuk menggunakan *powerup boost.* |
| ChangeLaneCommand.java | Deklarasi kelas beserta konstruktor untuk menjalakan *command* TURN_LEFT atau TURN_RIGHT. |
| Command.java | Deklarasi kelas utama *command* untuk melakukan *rendering* terhadap perintah yang akan dieksekusi. |
| DecelerateCommand.java | Deklarasi kelas untuk menjalankan *command Decelerate.* |
| DoNothing.java | Deklarasi kelas beserta konstruktor untuk menjalankan *command Do Nothing*. |
| EmpCommand.java | Deklarasi kelas untuk menjalankan *command* USE_EMP, yaitu perintah untuk menggunakan *powerup Emp*. |
| FixCommand.java | Deklarasi kelas beserta konstruktor untuk menjalankan *command* FIX. |
| LizardCommand.java | Deklarasi kelas untuk menjalankan *command* USE_LIZARD, yaitu perintah untuk menggunakan *powerup Lizard*. |
| OilCommand.java | Deklarasi kelas untuk menjalankan *command* USE_OIL, yaitu perintah untuk menggunakan *powerup oil.* |

| | |
|---|---|
| TweetCommand.java | Deklarasi kelas beserta konstruktor untuk menjalankan *command* USE_TWEET, yaitu perintah untuk menggunakan *powerup Tweet*. |
| ***Package : Entities*** | |
| Car.java | Deklarasi kelas dari *Car* yang memiliki atribut *id, position, speed, damage, state. powerups, boosting*, dan *boostCounter*. |
| GameState.java | Deklarasi kelas dari *GameState* yang memiliki atribut *currentRound, maxRounds, player, opponent,* dan *worldMap*. |
| Lane.java | Deklarasi kelas dari *Lane* yang memiliki atribut *position, surfaceObject*, dan *occupiedByPlayerId*. |
| Position.java | Deklarasi kelas dari position yang memiliki atribut y sebagai *lane* dan x sebagai *blocks*. |
| ***Package : Enums*** | |
| Direction.java | Deklarasi dari semua jenis *Direction* yang ada pada permainan yaitu *Forward, Backward, Left,* dan *Right.* |
| PowerUps.java | Deklarasi dari semua jenis *powerups* yang ada yaitu *boost, oil, tweet, lizard*, dan *emp*. |
| State.java | Deklarasi dari semua *state* yang ada dalam permainan yaitu *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*, dan *finished*. |
| Terrain.java | Deklarasi dari semua jenis Terrain yang ada yaitu *empty*, *mud*, *oil_spill*, *oil_power*, *finish*, *boost*, *wall*, *lizard*, *tweet*, dan *emp*. |

## Analisis Desain Solusi dari Algoritma Greedy yang diimplementasikan pada setiap Pengujian

Setelah beberapa kali melakukan pengujian terhadap program dan juga revisi untuk mengoptimasi bot agar dapat berjalan dengan lebih baik, berikut kami uraikan analisis dari desain solusi algoritma Greedy yang telah diimplementasikan pada setiap kasus pengujian. Kami menemukan bahwa program kami berhasil berjalan sesuai dengan yang kami harapkan, atau dengan kata lain program kami berhasil untuk menerapkan command berdasarkan strategi algoritma greedy by command. Hasil yang didapatkan sudah cukup optimal untuk sebagian besar kasus. Bot mobil yang kami rancang mampu bekerja sesuai dengan prioritas command

yang telah kami jelaskan sebelumnya. Perhitungan – perhitungan yang dilakukan bot sebelum memberi command untuk setiap round sudah cukup valid untuk dilakukan perbandingan sebelum akhirnya dipilih satu command yang paling baik untuk dilakukan pada round tersebut. Bot mobil kami telah berhasil mengutamakan untuk memilih command dengan speed maksimal dan dapat membuat bot mobil mencapai posisi akhir x yang terjauh. Bot juga telah mampu mengambil keputusan untuk mengambil jalur yang memiliki powerups untuk digunakan nantinya. Kemudian, bot kami juga mampu menggunakan powerups yang telah diperoleh untuk mengganggu bot mobil lawan. Bot kami juga mampu memperhitungkan damage yang akan diterima jika melakukan command tertentu dan memilih menggunakan command yang dapat memberikan damage minimum pada bot. Terakhir, bot kami juga dapat memperoleh skor akhir yang cukup baik. Kami juga telah melakukan uji coba dengan mengadu bot kami dengan bot-bot lainnya dan mendapatkan hasil yang cukup baik.

Namun perlu diperhatikan bahwa pada beberapa kasus, kami menemukan bahwa strategi greedy yang kami implementasikan masih tidak begitu optimal. Misalnya, bot kami kurang berhasil dalam mendapatkan nilai yang optimal dalam memanfaatkan powerup. Hal ini dapat dilihat dari penggunaan command Tweet dengan pilihan posisi Cybertruck yang kurang tepat sehingga lawan tidak terpengaruh. Menurut kami hal ini merupakan tantangan karena harus memprediksi pergerakan lawan untuk round berikutnya, mengingat penggunaan command Tweet berlakuk untuk round selanjutnya. Pada kasus tertentu, kami juga menemukan bahwa algoritma yang diterapkan masih menghasilkan command yang kurang optimal, misalnya memilih untuk beripindah jalur hanya untuk mengambil powerups, padahal masih terdapat block mud atau rintangan lainnya. Saat di uji coba dengan menandingkan bot kami dengan bot lain, kadang-kadang bot kami masih sering terkena dampak dari powerups yang diberikan lawan, sehingga menyebabkan kecepatan bot menjadi berkurang dan tidak optimal. Hal ini terjadi karena bot masih kurang optimal dalam menghindari jebakan yang diberikan lawan. Jadi, strategi greedy by command ini masih belum dapat menjamin 100% bahwa solusi yang ditemukan adalah solusi terbaik yang ada.

# Bab 5 – KESIMPULAN DAN SARAN

## Kesimpulan

Algoritma Greedy bisa digunakan sebagai salah satu strategi untuk menjalankan bot dalam sebuah permainan, walaupun algoritma Greedy hanya mempertimbangkan pengambilan keputusan terbaik pada optimum lokal namun algoritma ini menurut kami sudah baik karena dari sejumlah pertandingan yang dilakukan dengan reference-bot, bot kami memenangkan semuanya. Selain itu, alternatif solusi Greedy yang bisa dipilih juga cukup banyak sehingga banyak pilihan yang bisa digunakan untuk mencapai solusi yang lebih optimal.

## Saran

Algoritma Greedy yang sudah kami terapkan pada starter-bot sudah baik namun mungkin bisa disempurnakan lagi dengan menambahkan alternatif kasus yang ada dan bisa mengoptimasi penggunaan TWEET agar bisa lebih akurat dalam memprediksi posisi lawan. Ada beberapa spesifikasi yang agak rancu pada Github Entellect Challenge 2020 sehingga kami sempat kebingungan sehingga spesifikasi mungkin bisa diperjelas di spek tugas besar.

**LINK REPOSITORY**

https://github.com/clauculus/stima-overdrive

**LINK VIDEO**

**Video Tugas Besar**

# DAFTAR PUSTAKA

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)-Bag2.pdf

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

GitHub - EntelectChallenge/2020-Overdrive: Main repository for Entelect Challenge 2020