Laporan Tugas Besar 1 IF2211

Strategi Algoritma

Pemanfaatan Algoritma Greedy dalam Aplikasi Permainan “Overdrive”

oleh

delTa : Kelompok 25

Hana Fathiyah (13520047)

Bayu Samudra (13520128)

Febryola Kurnia Putri (13520140)

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

Daftar Isi

Daftar Gambar

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 1.1. Ilustrasi permainan Overdrive

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

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

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

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

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

1. Peta permainan memiliki bentuk array 2 dimensi yang memiliki 4 jalur lurus. Setiap jalur dibentuk oleh block yang saling berurutan, panjang peta terdiri atas 1500 block. Terdapat 5 tipe block, yaitu Empty, Mud, Oil Spill, Flimsy Wall, dan Finish Line yang masing-masing karakteristik dan efek berbeda. Block dapat memuat powerups yang bisa diambil oleh mobil yang melewati block tersebut.
2. Beberapa powerups yang tersedia adalah:
    a. Oil item, dapat menumpahkan oli di bawah mobil anda berada.
    b. Boost, dapat mempercepat kecepatan mobil anda secara drastis.
    c. Lizard, berguna untuk menghindari lizard yang mengganggu jalan mobil anda.
    d. Tweet, dapat menjatuhkan truk di block spesifik yang anda inginkan.
    e. EMP, dapat menembakkan EMP ke depan jalur dari mobil anda dan membuat mobil musuh (jika sedang dalam 1 lane yang sama) akan terus berada di lane yang sama sampai akhir pertandingan. Kecepatan mobil musuh juga dikurangi 3.
3. Bot mobil akan memiliki kecepatan awal sebesar 5 dan akan maju sebanyak 5 block untuk setiap round. Game state akan memberikan jarak pandang hingga 20 block di depan dan 5 block di belakang bot sehingga setiap bot dapat mengetahui kondisi peta permainan pada jarak pandang tersebut.
4. Terdapat command yang memungkinkan bot mobil untuk mengubah jalur, mempercepat, memperlambat, serta menggunakan 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

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

Adapun peraturan yang lebih lengkap dari permainan Overdrive, dapat dilihat pada laman :

https://github.com/EntelectChallenge/2020-Overdrive/blob/develop/game-engine/game-rules.md

Bab II
Landasan Teori

2.1. Algoritma Greedy

2.1.1. Pengertian Algoritma Greedy

Dilansir dari diktat mata kuliah Strategi Algoritma yang disusun oleh Dr. Ir. Rinaldi Munir, M.T., algoritma greedy merupakan algoritma yang memecahkan suatu masalah dengan langkah per langkah. Hal-hal yang dilakukan pada setiap langkah tersebut adalah sebagai berikut.

1. Mengambil pilihan terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan, yaitu menggunakan prinsip “take what you can get now!”.
2. Berharap bahwa dengan memilih solusi optimum lokal pada setiap langkah akan berakhir dengan optimum global.

2.1.2. Skema Umum Algoritma Greedy

Algoritma greedy disusun oleh elemen-elemen berikut.

1. Himpunan kandidat

Himpunan kandidat berisi elemen-elemen yang membentuk solusi.

2. Himpunan solusi

Himpunan solusi berisi kandidat-kandidat yang terpilih sebagai solusi persoalan.

3. Fungsi seleksi (selection function)

Fungsi seleksi berfungsi untuk memilih kandidat yang paling memungkinkan mencapai solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.

4. Fungsi kelayakan (feasible)

Fungsi kelayakan berfungsi untuk memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala (constraints) yang ada. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan kandidat yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi.

5. Fungsi objektif

Fungsi objektif merupakan fungsi yang memaksimumkan atau meminimumkan nilai solusi.

2.2. Cara Kerja Program

2.2.1. Permainan Entellect 2020 Overdrive

Permainan overdrive mempertandingkan dua bot mobil di dalam suatu ajang balapan. Masing-masing bot saling bertanding untuk mencapai garis finish. Detail permainan ini terdapat pada situs https://github.com/EntelectChallenge/2020-Overdrive. Di dalam tugas besar ini, kami melanjutkan starter-bots yang terdapat dalam situs https://github.com/EntelectChallenge/2020-Overdrive/releases/tag/2020.3.4.

2.2.2. Struktur Direktori Java pada Permainan *Overdrive*

Pada permainan ini, terdapat file bot.json, folder src yang berisi folder main yang di dalamnya terdapat direktori program. Di dalam direktori tersebut terdapat file bot.java, main.java, folder entities, dan folder enums. Di dalam folder entities terdapat file Car.java, GameState.java, Lane.java, dan Position.java. Direktori enums berisi powerups.java dan direction.java. state.java, terrain.java. Selain itu terdapat direktori command yang di dalamnya terdapat file AccelerateCommand.java, BoostComand.java, ChangeLaneCommand.java, Command.java, DecelerateCommand.java, DoNothingCommand.java, serta OilCommand.java.

2.2.3. Mengimplementasikan Algoritma Greedy pada Bot

Pada mulanya, kita perlu untuk mengunduh versi terbaru dari starter pack tersebut, yakni pada situs https://github.com/EntelectChallenge/2020-Overdrive/releases/tag/2020.3.4. Dalam membuat bot, diperlukan Java (minimal Java 8), IntelliJ IDEA sebagai code editor, serta NodeJS.

Strategi greedy yang diimplementasikan dikaitkan dengan fungsi objektif permainan overdrive. Parameter yang menjadi penentu kemenangan pemain terdiri atas beberapa hal. Parameter utama yang menjadi penentu adalah waktu. Pemain yang berhasil mencapai garis finish lebih awal adalah pemenangnya. Jika ternyata kedua pemain mencapai

garis finish dalam waktu yang bersamaan, penentu selanjutnya adalah kecepatan pemain. Jika kecepatan masih sama, penentu selanjutnya adalah skor.

Algoritma greedy dalam bot dapat diimplementasikan secara modular di setiap kelas tertentu. Implementasi digunakan dengan bahasa Java dan berparadigma object-oriented, sehingga dibentuk kelas-kelas tertentu dalam pelaksanaannya.

2.2.4. Menjalankan Game Engine

Dalam menjalankan program ini, terdapat file “run.bat”. Pemain dengan sistem operasi berbasis windows dapat melakukan double-click dalam menjalankan permainannya, sedangkan pemain dengan sistem operasi Linux/Mac dapat menjalankan command “make run”. File “game-runner-condig.json” dan “bot.json” yang terdapat di dalam direktori “starter-bots” dapat diedit untuk mengatur informasi terkait bot yang dibuat.

Pada dasarnya, game-engine dijalankan dengan mode command-line interface (CLI). Untuk mempermudah visualisasi, dapat digunakan visualizer yang terdapat pada link https://github.com/Affuta/overdrive-round-runner.

Selain itu, game ini dapat dengan menggunakan bantuan docker untuk mempermudah proses kompilasi, sehingga tidak perlu meng-install dependencies lainnya, serta dapat dijalankan di semua platform tanpa menimbang kompabilitas.

Bab III
Aplikasi Strategi Greedy

3.1. Alternatif Solusi Permasalahan

Untuk mengoptimasi hasil pertandingan, terdapat beberapa metode yang dapat digunakan. Metode tersebut adalah sebagai berikut.

3.1.1. Greedy berdasarkan Point

Greedy berdasarkan point dalam proses pengambilan keputusan mengoptimalkan nilai poin yang dapat diperoleh dengan menjalankan suatu perintah. Prosesnya, setiap langkah yang akan diambil perlu dipertimangkan manakah dari setiap perintah yang dapat menghasilkan point maksimum sesuai dengan keadaan mobil saat ini.

3.1.1.1. Mapping Elemen Greedy

Pada solusi ini, terdapat beberapa elemen greedy di dalamnya yaitu sebagai berikut:

- Himpunan Kandidat: Semua perintah yang tersedia
- Himpunan Solusi: Sebuah perintah yang terpilih
- Fungsi solusi: Memeriksa apakah point yang dihasilkan setelah sebuah perintah dijalankan memberikan point tertinggi
- Fungsi seleksi: Memilih perintah yang dapat menghasilkan poin seperti dari pengambilan powerups.
- Fungsi kelayakan: Apakah perintah yang dipilih dapat dilakukan, seperti ketersediaan powerups atau posisi lane.
- Fungsi Objektif: Memaksimumkan hasil point dari sebuah perintah.

3.1.1.2. Analisis Kompleksitas

Dalam algoritma ini, terdapat 11 perintah yang harus diperiksa. Masing-masing perintah harus menghitung jumlah point yang dihasilkan dari suatu perintah. Pemeriksaan ini perlu melakukan scanning sebanyak n block berdasarkan kecepatan pemain. Oleh karena itu, kompleksitasnya adalah sebagai berikut

$$T(n) = 11*n = O(n)$$

3.1.1.3. Analisis Efektivitas

Algoritma ini akan memberikan nilai efektif bila kondisi berikut terpenuhi:

- Obstacle yang ada pada lane player lebih sedikit dibandingkan dengan jumlah powerups yang tersedia pada lane tersebut sehingga player dapat menambahkan poin dibandingkan dengan poin sebelumnya.
- Player memiliki kecepatan yang tinggi sehingga memperbesar kemungkinan untuk mendapatkan lebih banyak poin dibandingkan dengan kecepatan yang rendah.

3.1.2. Greedy berdasarkan Kecepatan

Pada solusi ini, greedy dilakukan berdasarkan analisis kecepatan pemain. Pada dasarnya, pemain yang menang adalah pemain yang berhasil sampai di finish line pertama kali. Namun, jika kedua pemain mencapai garis finish secara bersamaan, parameter kedua yang menentukan siapa pemenang permainan adalah kecepatan pemain, yaitu pemain yang memiliki kecepatan lebih cepat adalah pemenangnya jika keduanya sampai di waktu yang bersamaan. Maka dari itu, kami melakukan greedy berdasarkan kecepatan pemain. Elemen-elemen greedy yang terdapat pada alternatif ini adalah sebagai berikut.

3.1.2.1. Mapping Elemen Greedy

Pada alternatif ini, terdapat elemen-elemen greedy sebagai berikut:

- Himpunan Kandidat: Perintah untuk melakukan Accelerate, Boost, pindah lane, atau menggunakan Lizard.
- Himpunan Solusi : Perintah yang terpilih
- Fungsi seleksi: Pilihlah manakah perintah yang dapat memberikan kecepatan tertinggi setelah melakukan suatu perintah
- Fungsi kelayakan: Perintah yang dilakukan merupakan perintah yang valid, yaitu jika menggunaak powerups haruslah tersedia atau jika pindah lane, lane haruslah tersedia.
- Fungsi Objektif: Memaksimalkan kecepatan tertinggi

3.1.2.2. Analisis Kompleksitas

Pada alternatif ini, terdapat 4 perintah yang mungkin dijalankan. Selain itu, untuk perhitungan point diperlukan melakukan transversal dari posisi mobil pemain sebanyak n blok, disesuaikan dengan perintah yang dipilih. Oleh karena itu kompleksitanya adalah sebagai berikut:

$$T(n) = 4n = O(n)$$

3.1.2.3. Analisis Efektivitas

Solusi dari algoritma ini efektif bila ada poin ini terpenuhi:

- Tidak ada obstacle pada line yang sama sehingga player tidak perlu mengubah lane dan dapat meningkatkan kecepatan
- Tersedia powerups boost meningkatkan kecepatan dengan instan dan line yang saat ini tidak terdapat obstacle
- Tersedia powerups lizard sehingga player dapat menghindari obstacle didepannya tanpa menambah damage ataupun kecepatan

3.1.3. Greedy berdasarkan Jumlah Obstacle

Jumlah obstacle dianalisis pada setiap lane. Alternatif solusi yang digunakan terdiri atas tiga hal, yaitu menggunakan lizard, berpindah lane, dan menurunkan kecepatan. Namun, terdapat suatu kasus di mana setiap lane memiliki obstacle. Untuk kasus yang seperti ini, analisis dilakukan dengan menghitung jumlah obstacle beserta akumulasi damage yang mungkin dihasilkan akibat adanya collision dengan obstacle tersebut.

3.1.3.1. Mapping Elemen Greedy

Pada alternatif ini, terdapat elemen-elemen greedy sebagai berikut:

- Himpunan Kandidat: Semua perintah yang tersedia
- Himpunan Solusi : Sebuah perintah yang terpilih
- Fungsi seleksi: Memeriksa apakah jumlah obstacle di suatu lane tersebut lebih kecil dibandingkan lane lainnya atau sebaliknya serta memeriksa jumlah damage yang mungkin didapatkan.
- Fungsi kelayakan: Apakah perintah tersebut dapat dilakukan, apakah hal tersebut lebih mending dibandingkan opsi lainnya.
- Fungsi Objektif: Meminimumkan damage yang mungkin saja akan didapatkan

3.1.3.2. Analisis Kompleksitas

$$T(n) = 4n = O(n)$$

3.1.3.3. Analisis Efektivitas

Solusi dari algoritma ini efektif apabila:

- Perhitungan tingkat *damage* sesuai dengan prediksi
- Tidak terdapat *cyber truck* karena sejatinya *cyber truck* adalah *obstacle* yang *unpredictable.*

3.1.4. *Greedy* berdasarkan *Power Ups*

Power Ups memberi banyak manfaat bagi pemain. Dengan mengambil *power ups* yang tersedia pada *lane,* pemain bisa menggunakannya untuk menyerang lawan ataupun untuk mengamankan dirinya dari *obstacle.*

3.1.4.1. Mapping Elemen Greedy

Algoritma ini memiliki elemen greedy sebagai berikut:

- Himpunan Kandidat: Perintah Pindah lane perintah Do Nothing, perintah menjalankan powerups
- Himpunan Solusi : Perintah yang terpilih
- Fungsi seleksi: Memilih perintah yang dapat mengambil poweups paling maksimum
- Fungsi kelayakan: Perintah yang dipilih merupakan perintah yang valid, yaitu jika menggunakan powerups, maka harus ada atau lane yang akan dipindahi terseda
- Fungsi Objektif: Memaksimalkan pendapatan powerups

3.1.4.2. Analisis Kompleksitas

Dalam algoritma ini, setiap perintah harus dievaluasi apakah memberikan jumlah powerups yang maksimal. DIkarenakan jumlah powerups adalah 5, kompleksitas algoritma untuk algoritma ini adalah sebagai berikut:

$$T(n) = 7n = O(n)$$

3.1.4.3. Analisis Efektivitas

Algoritma ini akan efektif bila terdapat kondisi ini terpenuhi:

- Tidak terdapat obstacle setelah powerups sehingga tidak merusak kendaraan dan mengurangi kecepatan.
- Powerups boost tersedia lebih banyak sehingga player dapat memaksimalkan kecepatan dalam permainan.

3.1.5. *Greedy* berdasarkan *Attack*

Attack dilakukan untuk memperlemah kecepatan lawan. Dengan memberikan *attack* pada lawan, seperti menumpahkan *oil,* menembakkan EMP, dan menaruh *cybertruck* dapat menyebabkan lawan terkena *collision* dan meningkatkan akumulasi *damage* miliknya.

Damage lawan yang tinggi dapat mengurangi kecepatan miliknya yang membuat kita berpotensi tiba di *finish line* lebih cepat.

3.1.5.1. Mapping Elemen Greedy

Algoritma ini memiliki elemen greedy sebagai berikut:

1. Himpunan Kandidat: perintah *use* oli, *use* emp, dan *use* tweet
2. Himpunan Solusi : perintah yang dipilih
3. Fungsi seleksi: memilih perintah powerups mana yang ingin dilemparkan ke lawan
4. Fungsi kelayakan: perintah yang dipilih adalah perintah yang valid, yaitu memperhatikan posisi lawan dan ketersediaan powerups yang ingin dilemparkan
5. Fungsi Objektif: maksimalkan banyak powerups yang dilemparkan kepada lawan

3.1.5.2. Analisis Kompleksitas

Dalam algoritma ini, setiap perintah perlu diperiksa jumlah kemungkinan powerups yang dikeluarkan kepada lawan. Oleh karena itu, kompleksitasnya adalah sebagai berikut:

$$T(n) = 11n = O(n)$$

3.1.5.3. Analisis Efektivitas

Algoritma ini akan memberikan hasil efektif bila:

- Cybertruck yang dilemparkan ke depan lawan sebanyak mungkin karena akan memberikan damage sebesar 2 di tiap pelemparannya
- Mengeluarkan semua powerups yang bisa dilemparkan ke lawan sebanyak mungkin, oli, mud, dan cybertruck yang dapat menghambat pergerakan lawan

3.1.6. *Greedy* berdasarkan *Fix*

Seperti yang telah kita ketahui, *FixingCar* dapat digunakan untuk mengurangi *damage* yang dimiliki oleh kendaraan kita. Pengoptimalan ini dilakukan dengan menjalankan proses *FixingCar* setiap pemain terkena *collision* yang mengakibatkan *damage.*

3.1.6.1. Mapping Elemen Greedy

Algoritma ini memiliki elemen greedy sebagai berikut:

- Himpunan Kandidat: Perintah Fix, Boost, Accelerate, dan Pindah Lane
- Himpunan Solusi : Perintah yang terpilih
- Fungsi seleksi: Memilih perintah antara melakukan fix atau perintah lain

- Fungsi kelayakan: Perintah yang dipilih merupakan perintah yang valid, yaitu jika menggunakan powerups, maka harus ada atau lane yang akan dipindahi tersedia
- Fungsi Objektif: Maksimasi langkah output

3.1.6.2. Analisis Kompleksitas

Dalam algoritma ini, setiap perintah perlu dihitung jumlah fix yang bisa dilakukan. Oleh karena itu, kompleksitasnya adalah sebagai berikut:

$$T(n) = 4n = O(n)$$

3.1.6.3. Analisis Efektivitas

Algoritma ini akan memberikan hasil efektif bila:

- Fix dilakukan sesering mungkin bila terdapat damage sehingga dapat menyebabkan kecepatan selalu optimum.
- Obstacle jarang terjadi sehingga player dapat selalu mempercepat kendaraan tanpa perlu sering memperbaiki

3.1.7. Greedy berdasarkan Jumlah Damage

Greedy berdasarkan jumlah *damage* ini memengaruhi aksi yang akan dilakukan pemain. *Bot* di-*setting* dengan sistem perbaikan langsung apabila pemain terkena *collision* yang mengakibatkan meningkatnya *damage.* Berdasarkan eksperimen yang telah dilakukan, perbaikan langsung lebih optimal dibandingkan menunggu *damage* bertambah semakin besar.

3.1.7.1. Mapping Algoritma Greedy

Algoritma ini memiliki elemen greedy sebagai berikut:

- Himpunan Kandidat: Semua Perintah yang tersedia
- Himpunan Solusi : Perintah yang terpilih
- Fungsi seleksi: Memilih perintah yang memberikan damage terkecil
- Fungsi kelayakan: Perintah yang dipilih merupakan perintah yang valid, yaitu jika menggunakan powerups, maka harus ada atau lane yang akan dipindahi tersedia
- Fungsi Objektif: Meminimalkan jumlah damage

3.1.7.2. Analisis Kompleksitas

Setiap perintah yang dipilih, perlu diperiksa jumlah damage yang terjadi. Oleh karena itu, kompleksitas dari algoritma ini adalah sebagai berikut

$$T(n) = 11n = O(n)$$

3.1.7.3. Analisis Efektivitas

Algoritma ini akan memberikan hasil yang efektif bila terdapat hal ini terpenuhi:

- Terdapat sebuah lane yang dapat diakses tidak memiliki damage sehingga tidak perlu mendapatkan damage yang berarti
- Obstacle yang tersedia haruslah berturut-turut sehingga perbaikan tidak perlu dilakukan setiap kali terkena obstacle.

3.2. Solusi yang diimplementasikan

Pada implementasi program yang dilakukan merupakan kumulatif dari beberapa solusi greedy yang kami telah terangkan pada bagian sebelumnya. Kami menerapkan urutan prioritas greedy yaitu sebagai berikut

1. Memprioritaskan menghindari obstacle
2. Pengambilan Powerups
3. Melakukan Fixing Car
4. Melakukan perlawanan
5. Mempercepat Kecepatan

Pada tahap menghindari obstacle, kami memeriksa apakah lane yang kami tempati aman untuk dilewati dengan keadaan permainan yang diberikan. Sebuah lane aman bila tidak terdapat obstacle sama sekali. Jika terdapat ostacle pada lane saat ini dan player memiliki powerups lizard, powerups lizard perlu digunakan. Hal ini untuk mencegah pengurangan kecepatan ataupun penambahan damage. Jika tidak tersedia, dipilih sebuah jalur yang dapat diakses serta memiliki damage paling minimum. Dengan hal itu, damage dari kendaraan dapat ditekan. Bila lane yang memiliki damage minimum adalah lane posisi player saat ini, pertimbangkan prioritas greedy selanjutnya.

Pengambilan Powerups dilakukan dengan prioritas pengambilan powerups sebagai berikut:

1. Boost
2. Lizard
3. Tweet

4. EMP
5. Oil Power

Bila sebuah lane memiliki powerups pada suatu prioritas dan tidak menghasilkan damage lebih dari 2, maka lane itu perlu dipilih. Bila lane yang dipilih adalah lane player saat ini atau tidak ada powerups yang dapat diambil, maka perlu dipertimbangkan urutan prioritas greedy selanjutnya.

Pada prioritas ketiga, apabila mobil player sudah mengalami kerusakan, maka perlu dilakukannya fixing car. Dengan melakukan fixing car secara rutin, kecepatan mobil akan selalu berada di kecepatan yang cukup tinggi sehingga kendaraan dapat terus melaju tanpa perlu harus kehilangan kecepatan akibat damage yang dihasilkan.

Pada Prioritas keempat, mobil melakukan melakukan perlawanan dengan menggunakan powerups. Urutan perlawanan berdasarkan prioritasnya adalah sebagai berikut:

1. EMP
2. Oil
3. Tweet

EMP dilakukan apabila tersedia boost EMP. Player beserta lawan haruslah terdapat di lane yang sama. Player berada di belakang dari lawan pada babak selanjutnya berdasarkan speed saat ini. Jaeak antara lawan dan player tidak lebih daripada 20 block.

Jika syarat EMP tidak memenuhi, perlu diperiksa apakah Oil tersedia. Bila tersedia dan lawan akan melewati posisi player saat ini pada putaran selanjutnya, letakan oli pada posisi tersebut. Jarak antara player dan musuh tentunya harus kurang dari 5 untuk mengoptimasi langkah yang dilakukan.

Jika syarat Oil tidak terpenuhi, perlu diperiksa tweet ability tersedia. Syarat agar perintah ini dapat dijalankan apabila musuh berada di jarak pandang player. Selain itu, musuh harus dapat diprediksikan tidak akan mengubah lane dengan cara memeriksa apakah ada obstacle di depan musuh. Bila tidak ada, letakan cyber truck pada posisi player berhenti.

Bila semua syarat penyerangan tidak bisa terpenuhi, perlu dipertimbangkan tahap selanjutnya yaitu meningkatkan kecepatan. Peningkatan kecepatan perlu juga memprediksi apakah akan menghasilkan damage lebih daripada atau sama 2 atau tidak. BIla damage yang dihasilkan

kurang dari dua dan tersedia boost, maka digunakan boost. Bila dengan menerapkan accelerate tidak menghasilkan damage lebih dari dua, maka gunakan accelerate. Bila semua langkah tersebut tidak bisa dilakukan, maka tidak lakukan apa-apa.

Bab IV
Implementasi dan Pengujian

4.1. Pseudocode

Note: Apabila seluruh bot.java dan file modularnya dijadikan pseudocode akan membutuhkan banyak sekali halaman sehingga membuat laporan menjadi tidak efektif maka kami memutuskan untuk menjadikan pseudocode hanya program strateginya. Kode lengkap bisa dilihat langsung di source code-nya dan sudah diberi komentar.

```
//Strategi Greedy
//Program dibuat modular dengan membaginya menjadi beberapa file terpisah
//Ada beberapa folder utama yang didalamnya berisi file yang membentuk fungsi-fungsi sesuai greedy yang telah ditetapkan
//Folder Analyzer yang berisi file-file utama
//1.AttackAbility
//2.SpeedAbility
//3.FixingCar
//4.BaseAnalyzer
//5.PowerupsCollector
//6.ObstacleAvoid

function run () → Command
//Bagian pertama akan dijelaskan pseudocode mengenai kode pada AttackAbility
class AttackAbility
Deklarasi
opponentLane = int
playerLane = int
opponentBlock = int
playerBlock = int
isOpponentFrontClear = boolean
```

```
Terrain = list of Terrain
WAL,MUD,OIL_SPEED = list of Terrain
PowerUps = list of powerup

Body
// Persiapkan semua kondisi
opponentLane = opponentCar.position.lane
playerLane = playerCar.position.lane
opponentBlock = opponentCar.position.block
playerBlock = playerCar.position.block

//Fungsi mengecek apakah player mempunyai EMP
function checkIsEmpExist()→ boolean
 for (PowerUps ps : gameState.player.powerups)do
  if (ps=PowerUps.EMP) then
   → true
  end if
  → false
 end for
//Fungsi mengecek apakah player mempunyai OIL
function checkIsOilExist()→ boolean
 for (PowerUps ps : gameState.player.powerups)do
  if (ps=PowerUps.OIL) then
   → true
  end if
  → false
 end for
//Fungsi mengecek apakah player mempunyai TWEET
function checkIsTweetExist()→ boolean
 for (PowerUps ps : gameState.player.powerups)do
  if (ps=PowerUps.TWEET) then
   → true
  end if
```

```
  → false
 end for

procedure analyze()
//Dicek apakah ada EMP
 if(checkIsEmpExist()) then
           if      ((opponentLane=playerLane)and(opponentBlock      >
playerBlock)and(opponentBlock - playerBlock <= 20)and(opponentBlock
+ opponentCar.speed > playerBlock + playerCar.speed)) then
   setSolution(EmpCommand())
  end if
 end if

//Dicek apakah ada OIL
 if(checkIsOilExist()) then
       if        ((opponentLane=playerLane)and(opponentBlock       <
playerBlock)and(opponentBlock - playerBlock <= 5)and(opponentBlock +
opponentCar.speed >= playerBlock) then
   setSolution(OilCommand())
  end if
 end if

//Dicek apakah ada TWEET
 if(checkIsTweetExist()) then
  isOpponentFrontClear = true
   if((playerBlock-opponentBlock)>5  and  (playerBlock>opponentBlock)
or  (opponentBlock>playerBlock)  and  (opponentBlock-playerBlock>20))
then
   isOpponentFrontClear = false
  else
   isOpponentFrontClear = true
   for(Lane l : gameState.lanes.get(opponentLane-1))do
    if(l.isOccupiedByCyberTruck) then
```

```
     isOpponentFrontClear = false
    end if
    for(Terrain t : destructiveTerrain)do
    if(t=l.terrain) then
     isOpponentFrontClear = false
    end if
   end for
  end for
  if(isOpponentFrontClear) then
   setSolution(TweetCommant(opponentLane,opponentBlock))
  end if

SpeedAbility
class SpeedAbility
Deklarasi
Body
//Fungsi mengecek apakah player mempunyai BOOST
function checkIsBoostExist()→ boolean
 for (PowerUps ps : gameState.player.powerups)do
  if (ps=PowerUps.Boost) then
   → true
  end if
  → false
 end for

procedure analyze()
//Dicek apakah ada BOOST
 if(checkIsBoostExist()) then
   setSolution(EmpCommand())
 end if
 setSolution(AccelerateCommand())

FixingCar
```

```
class FixingCar
Deklarasi
Body
procedure analyze()
//Dicek apakah ada damage besar daripada 0
 if(gameState.player.damage>0) then
   setSolution(FixCommand())
 end if

PowerUpsCollector
class PowerUpsCollector
Deklarasi
BOOST,LIZARD,TWEET,EMP,OIL_POWER = list of Terrain
l = list of Terrain
i,j = int
Body
//Persiapkan semua yang diperlukan
l = get.lanes()
i = 0
j = 0
procedure analyze()
 for(Terrain : powerups)do
  for (i<l.size) do
   for (j< min(playerCar.speed+1,l.get(i).size()) do
    if(l.get(i).get(j) == powerups) then
      setCommandResult(i)
    end if
   end for
  end for
 end for

BaseAnalyzer
class BaseAnalyzer
```

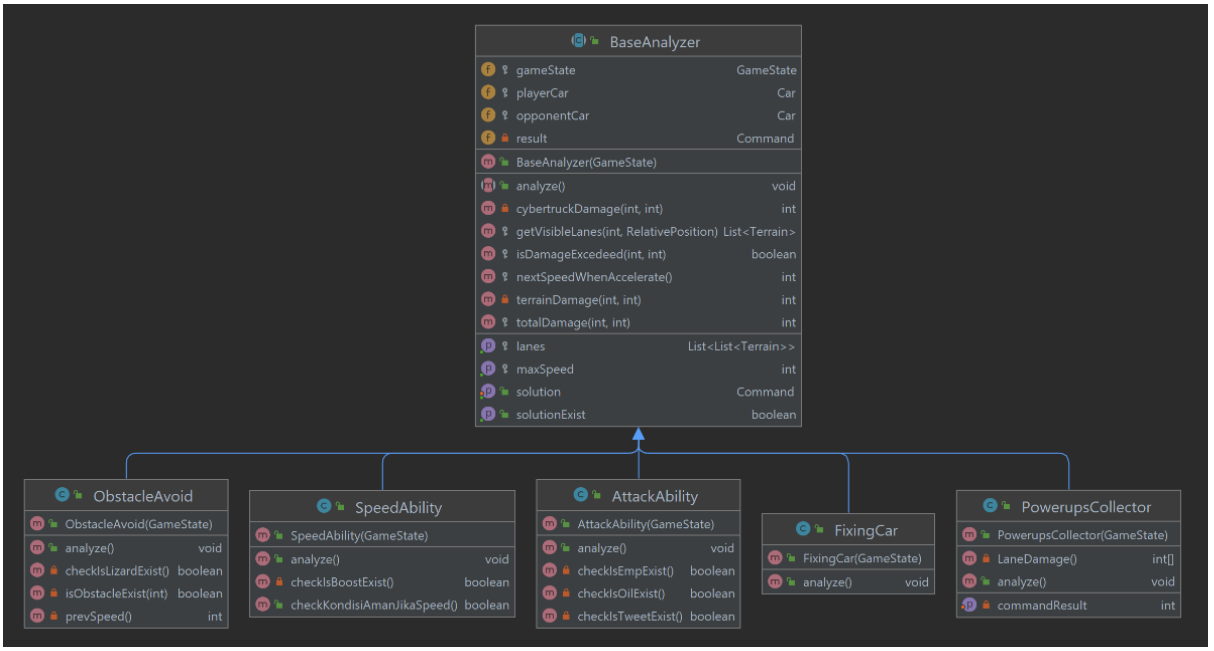
```
Deklarasi
gameState = GameState
playerCar = Car
opponentCar = Car
result = Command
solutionExist = false
Body
build constructor
 gameState = gameState
 playerCar = gameState.player
 opponentCar = gameState.opponent
 solutionExist = false

function isSolutionExist() → boolean
 → solutionExist

procedure setSolution(input solution: Command, output result : Command)
 result = solution
 solutionExist = true

function getSolution() → Command
 → result

function getVisibleLanes() → Lane
 → mengembalikan lane yang sesuai dan visible untuk ditempati oleh player dimana tidak ada halangan ataupun mobil lawan tepat di depan mobil player

ObstacleAvoid
class ObstacleAvoid
Deklarasi
result = int
```

```
playerSpeed = int
cnt = int
currentBlock = int
damage = int
Body
function prevSpeed()→ int
 playerSpeed = playerCar.speed
 if(playerSpeed>3) then
  result = 3
 if(playerSpeed>5) then
  result = 5
 if(playerSpeed>6) then
  result = 6
 if(playerSpeed>8) then
  result = 8
 if(playerSpeed>9) then
  result = 9

function checkIsLizardExist()→ boolean
 for (PowerUps ps : gameState.player.powerups)do
  if (ps=PowerUps.Lizard) then
   → true
  end if
  → false
 end for

function cybertruckDamage(int lane) → int
 cnt = 0
 currentBlock = playerCar.position.block
 for(Lane l : gameState.lanes.get(lane-1)) do
  if(l.position.block > playerCar.speed) then
   → cnt*2
       else if (l.position.block > currentBlock and
```

```
l.isOccupiedByCyberTruck) then
   cnt = cnt + 1
  end if
 end for
 → cnt * 2

function isObstacleExist(int lane) → boolean
 → totalDamege(lane) > 0

function totalDamage(int lane)→ int
 → cybertruckDamage(lane)+terrainDamage(lane)

function terrainDamage(int lane)→ int
 damage = 0
 currentBlock = playerCar.position.block
 for(Lane l : gameState.lanes.get(lane-1)) do
    if  (l.position.block>playerCar.position.block+  playerCar.speed)
then
   → damage
  end if
  else if (l.position.block>currentBlock)then
   if (l.terrain=MUD) then
    damage = damege + 1
   end if
   else if (l.terrain=WALL) then
    damage = damage + 2
   end if
  end if
 end for
 → damage

Bot.java
Deklarasi
```

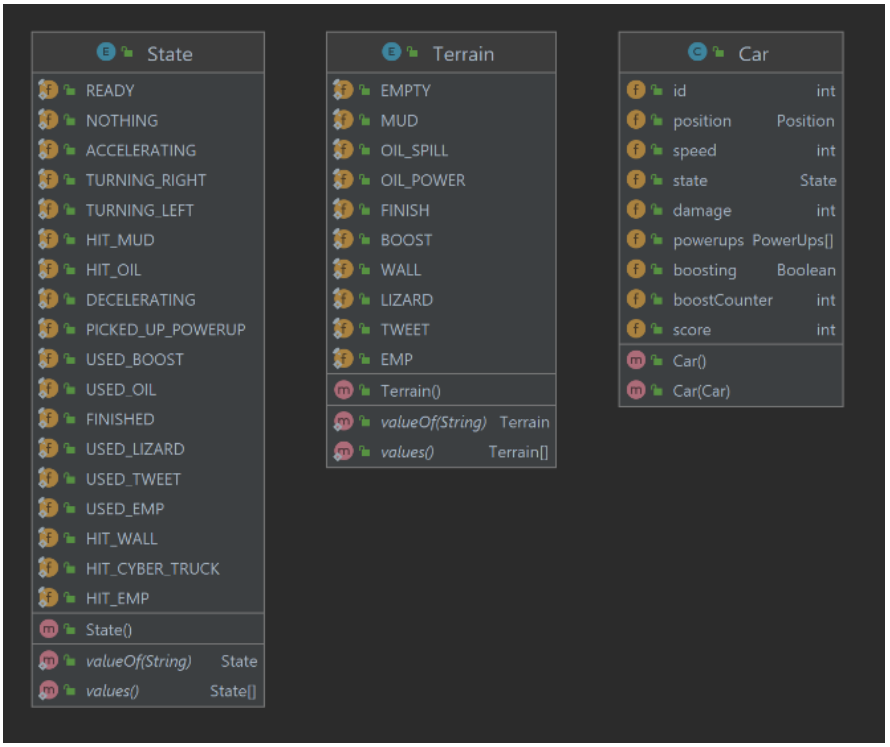
```
gameState = GameState
ctr = Controller
Body
build constructor
 gameState = gameState
 ctr = Controller()
 register()

procedure void register()
 ctr.addAnalyzer(ObstacleAvoid(gameState))
 ctr.addAnalyzer(SpeedAbility(gameState))
 ctr.addAnalyzer(AttackAbility(gameState))
 ctr.addAnalyzer(FixingCar(gameState))
 ctr.addAnalyzer(BaseAnalyzer(gameState))
 ctr.addAnalyzer(PowerUpsCollector(gameState))

function run() → Command
 ctr.nextAction()
```

4.2. Struktur Data

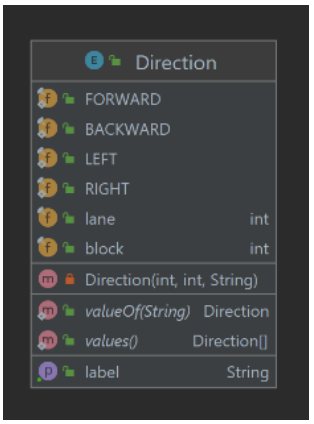


Gambar 4.2.1 Struktur Data Kelas Penggunaan Algoritma *Greedy* pada Bot

Pada gambar 4.2.1 di atas dijelaskan *struktur* data yang kami pakai dalam implementasi *bot* dalam permainan *overdrive. File BaseAnalyzer* berisi struktur data utama yang selanjutnya diimplementasikan pada kelas-kelas hasil dekomposisi persoalan, yaitu *AttackAbility, BaseAnalyzer, FixingCar, ObstacleAvoid, PowerupsCollector,* dan *SpeedAbility.* Keenam kelas hasil kelas tersebut merupakan implementasi *greedy* yang kami buat secara modular. Masing-masing kelas tersebut memiliki atribut tertentu beserta fungsi yang sama di setiap kelasnya, yaitu *analyze.* Di dalam fungsi *analyze* ini, algoritma *greedy* diterapkan.

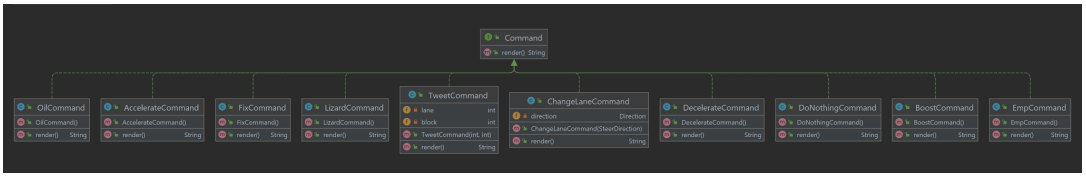


Gambar 4.2.2. Kelas Bawaan Permainan *Overdrive*



Gambar 4.2.3. Kelas Bawaan Permainan *Overdrive*

Gambar 4.2.2. dan 4.2.3. merupakan kelas yang berasal dari permainan *overdrive* itu sendiri. Kelas-kelas tersebut kemudian digunakan pada berbagai kelas lainnya dan memiliki keterkaitan satu sama lain.



Gambar 4.2.4. Kelas Command pada Permainan Overdrive

Secara *default,* permainan *overdrive* telah menyediakan *command* yang dapat diimplementasikan di dalam perjalanan menuju *finish line. Command-command* tersebut dapat dilihat secara detail pada gambar 4.2.4. *Command-command* tersebut adalah *OilCommand, AccelerateCommand, FixCommand, LizardCommand, TweerCommand, ChangeLaneCommand, DecelerateCommand, DoNothingCommand, BoostCommand, EmpCommand.* Di dalam masing-masing kelas tersebut terdapat konstruktor yang dijalankan ketika kelas tersebut dipanggil beserta fungsi *render* untuk menjalankannya.

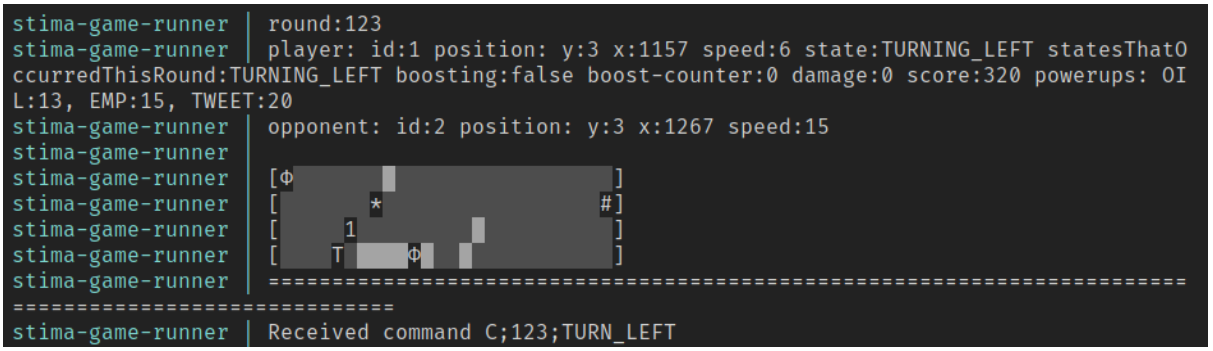
4.3. Analisis dan Pengujian

Pada bagian ini, kami akan paparkan pengujian yang kami lakukan pada program kami.

```
stima-game-runner | round:139
stima-game-runner | player: id:1 position: y:2 x:1258 speed:9 state:TURNING_RIGHT statesThat
OccurredThisRound:TURNING_RIGHT boosting:true boost-counter:3 damage:1 score:345 powerups: O
IL:13, BOOST:3, EMP:17, TWEET:23
stima-game-runner | opponent: id:2 position: y:4 x:1435 speed:9
stima-game-runner |
stima-game-runner | [                                                                ]
stima-game-runner | [      1                                                         ]
stima-game-runner | [                                                                ]
stima-game-runner | [    Φ         Φ                T                                ]
stima-game-runner | ================================================================================
==============================
stima-game-runner | Received command C;139;TURN_RIGHT
```

Gambar 4.3.1. Kasus 1

Pada kasus ini, player akan melakukan belok kanan. Hal ini dikarenakan pada kecepatan 9, player akan mengenai mud pada posisi block 1262. Oleh karena itu, perlu dipilih dua sebuah lane yang memiliki damage paling minimum. Karena damage paling minimum berada pada lane 3, maka dilakukan belok kanan. Oleh karena itu, algoritma *greedy* berhasil menghindari *obstacle*.



Gambar 4.3.2. Kasus 2

Pada kasus ini, player akan berbelok ke kiri. Hal ini dikarenakan terdapat powerups pada lane 2. Powerups pada lane 4 tidak dipilih dikarenakan adanya obstacle yang menyebabkan damage lebih dari 2. Obstacle pada lane 3 tidak termasuk dalam alasan berbeloknya player dikarenakan kecepatan player tidak mencukupi untuk meraih obstacle tersebut. Dengan begitu, langkah yang optimal untuk menghadapi permasalahan ini adalah berbelok ke kiri. Algoritma greedy memberikan hasil yang optimal untuk kasus ini.

Bab V
Kesimpulan dan Saran

5.1. Kesimpulan

Dari tugas besar IF2211 Strategi Algoritma semester 2 2021/2022 berjudul “Pemanfaatan Algoritma Greedy dalam Aplikasi Permainan '*Overdrive*', kami berhasil membuat sebuah *bot* dalam permainan "*Worms*" yang memanfaatkan algoritma greedy. Program yang dibuat tidak hanya menggunakan satu algoritma greedy saja melainkan beberapa algoritma greedy yang dikombinasikan untuk menghasilkan strategi yang terbaik. Program berhasil dijalankan tanpa bug dan sesuai dengan spesifikasi tugas besar 1 IF2211 yang diberikan.

5.2. Saran

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

1. Algoritma yang *dig*unakan pada Tugas Besar ini masih memiliki banyak kekurangan sehingga sangat memungkinkan untuk dilakukan efisiensi, misalnya dengan tidak menggunakan fungsi yang sama berulang-ulang. Oleh karena itu, dalam pengembangan program ini, masih bisa dilakukan efisiensi kinerja.
2. Penulisan *pseudocode* tampak kurang perlu dikarenakan program yang lumayan panjang dan membaca program lebih mudah daripada membaca *pseudocode* dengan asumsi program sudah *well commented.*
3. Memperjelas pemberian spesifikasi dan batasan-batasan setiap program pada *file* tugas besar untuk mencegah adanya multitafsir dan kesalahpahaman pada proses pembuatan program.

Lampiran

Link Video :

https://youtu.be/V_6a7zLlneg

Link Repository Github:

https://github.com//bayusamudra5502/Tubes-Stima-1

Daftar Pustaka

Slide kuliah IF2123 Aljabar Linear dan Geometri tahun ajaran 2021/2022.

2020-Overdrive/game-rules.md at develop · EntelectChallenge/2020-Overdrive (github.com)

diakses pada 7 Februari 2022

Affuta/overdrive-round-runner: Angular application which renders the contents of the match logs generated by the Entelect Challege Overdrive game runner (github.com)

diakses pada 7 Februari 2022