LAPORAN TUGAS BESAR I

PEMANFAATAN ALGORITMA GREEDY
DALAM APLIKASI PERMAINAN “OVERDRIVE”

Laporan dibuat untuk memenuhi salah satu tugas mata kuliah

IF2211 Strategi Algoritma

Disusun oleh :

Kelompok Overcooked

Kristo Abdi	**13520058**
Yakobus Iryanto P	**13520104**
Hilda Carissa Widelia	**13520164**

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

DAFTAR ISI

Bab 1 : Deskripsi Tugas

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

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 power-up yang bisa diambil oleh mobil yang melewati block tersebut.

2. Beberapa power-up 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 power-up. 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 2 : Landasan Teori

2.1 Dasar Teori Algoritma Greedy

Algoritma Greedy adalah algoritma yang bersifat heuristik dan urutan logisnya disusun berdasarkan langkah-langkah penyelesaian masalah yang disusun secara sistematis. Algoritma ini adalah metode paling populer dan sederhana untuk memecahkan persoalan optimasi. Algoritma ini berfungsi dengan menentukan solusi secara *step by step* dengan mencari optimum lokalnya. Dari optimum lokal - optimum lokal yang dipilih, diharapkan kemudian menjadi optimum global yang merupakan solusi terefektif dari permasalahan yang ada.

Algoritma Greedy dapat digunakan untuk menghasilkan solusi hampiran (*approximation*). Hal ini karena tidak mungkin untuk mengevaluasi masing-masing solusi dari setiap langkahnya sehingga mungkin hasil yang didapatkan bukanlah hasil yang paling optimal. Meskipun solusi paling optimalnya tidak dapat ditemukan, solusi dengan menggunakan algoritma Greedy dapat dianggap sebagai hampiran solusi optimal.

Berikut merupakan elemen-elemen yang menggambarkan sebuah Algoritma Greedy.

1. Himpunan Kandidat (C) Himpunan yang berisi kandidat yang akan dipilih pada setiap langkah, misalnya simpul sisi di dalam graf, job, task, koin, benda, karakter.
2. Himpunan Solusi (S) Himpunan yang berisi kandidat yang sudah dipilih.
3. Fungsi Solusi Fungsi yang digunakan untuk menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi paling optimal.
4. Fungsi Seleksi (Selection Function) Fungsi yang digunakan untuk memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik.
5. Fungsi kelayakan (Feasibility) Memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak).
6. Fungsi Objektif Fungsi yang digunakan untuk memaksimumkan atau meminimumkan solusi yang kami miliki.

2.2 Cara Kerja Program

Pada permainan Overdrive, game engine dijalankan dengan run.bat. File tersebut akan menjalankan file jar game-runner-with-dependencies yang telah di compile. Game engine menggunakan file game-config.json dan game-runner-config.json untuk menyimpan konfigurasi permainan dan state - state yang ada. Pada direktori `match-logs` yang telah diisi dengan hasil run program, kita juga bisa melihat ada file `GlobalState.json` yang menyimpan perubahan dan kondisi permainan untuk setiap ronde permainan ataupun file csv per player yang menyimpan perubahan, *command* dan state player semua ronde permainan. Pengguna tidak perlu untuk mengatur konfigurasi permainan tersebut dan dapat menjalankan permainan yang ada. Bot dasar telah tersedia melalui konfigurasi tersebut dan pengguna dapat

melakukan perubahan pada bagian tertentu dalam permainan dengan memodifikasi file-file .json ini seperti misalnya, untuk mengubah nama bot yang kita inginkan. Perubahan ini dapat kita aplikasikan dengan mengubah atribut `nickName` di dalam file bot.json yang ada pada direktori Java maupun bahasa lainnya.

Ketika pengguna ingin mengubah cara kerja bot dengan bahasa pemrograman pilihan kita maka pengguna perlu mengubah direktori file bahasa pemrograman bot pada file game-runner-config.json pada player-a sehingga cara kerja bot dapat dengan bebas diubah dalam file starter-bots sesuai bahasa pemrograman pilihan. Sebagai tambahan, game engine ini juga dilengkapi dengan sebuah visualizer yang bisa diunduh secara terpisah untuk menampilkan program dengan grafis pada localhost.

Pada permainan Overdrive, setiap ronde akan dimulai dengan game engine menerima command dari kedua player dan akan menjalankan objek yang berlomba mencapai garis finish terlebih dahulu. Pada awalnya, cara kerja Bot hanya akan mengimplementasikan logika biasa dan atribut tertentu seperti getBlocksInFront pada Java. Agar implementasi bot lebih baik maka pengguna perlu untuk menambahkan atau memodifikasi atribut - atribut yang sekiranya diperlukan. Dalam bahasa Java, implementasi Bot dilakukan berdasarkan paradigma pemrograman berorientasi objek sehingga pengguna perlu mengimplementasikan method - method dalam kelas Bot.

Dalam penggunaannya, build program dapat menggunakan aplikasi Intellij IDEA. Intellij IDEA telah menyediakan Maven Toolbox dan kita dapat memanfaatkan fitur `Install` di dalam `Lifecycle` untuk menghasilkan file .jar. File ini akan dibuat dalam direktori file target pada direktori bot yang menggunakan bahasa pemrograman Java untuk digunakan dalam proses menjalankan permainan. Pengguna dapat menjalankan permainan dengan menjalankan program `run.bat`. File batch ini akan menjalankan game engine secara otomatis di dalam Command Prompt atau Terminal (jika tanpa visualizer). Permainan ini akan ditampilkan dalam format text-based.

Bab 3 : Aplikasi Strategi *Greedy*

3.1 Pemetaan Algoritma Greedy pada Proses Persoalan Overdrive

a. Pemetaan Umum Permasalahan Permainan

Berikut pemetaan permasalahan permainan *Overdrive* secara umum menjadi elemen-elemen penyusun Algoritma Greedy

Tabel 3.1.1 Pemetaan umum permasalahan permainan

Elemen Algoritma Greedy	Pemetaan pada permainan *Overdrive*
Himpunan Kandidat (C)	Perintah-perintah atau command yang dapat dijalankan di dalam permainan. Command yang dimaksud dapat berupa perintah untuk menambahkan *speed*, mengurangi *speed*, berpindah *lane*, menggunakan *power up*, *fix* bot (mengurangi damage yang dimiliki bot) atau tidak melakukan apa apa.
Himpunan Solusi (S)	Perintah terbaik yang dipilih sesuai dengan kondisi atau *state* yang ada di setiap *round*-nya.
Fungsi Solusi	Memeriksa apakah perintah atau *command* yang dipilih merupakan *command* yang terdefinisi di dalam permainan. Jika *command* tidak terdefinisi maka tentu tidak dapat dijalankan.
Fungsi Seleksi (*Selection Function*)	Pemilihan perintah atau *command* yang sesuai dengan prioritas strategi yang digunakan di dalam *bot* yang dibangun untuk implementasi permainan *overdrive* ini. Penentuan prioritas dalam bahasa pemrograman yang digunakan dapat digunakan dengan membuat fungsi penyeleksian atau kondisi terurut sesuai dengan strategi yang sudah disusun.
Fungsi Kelayakan (*Feasibility*)	Memeriksa apakah perintah atau *command* yang digunakan memenuhi kondisi untuk dijalankan, validasi yang dimaksud adalah misalnya memastikan memiliki *power-up* yang dimaksud sebelum digunakan , memastikan bot tidak berada di lane 1 sebelum

	memberikan perintah belok kiri atau memastikan bot tidak berada di lane 4 sebelum memberikan perintah belok kanan.
Fungsi Objektif	Memenangkan permainan *overdrive* dengan menjadi yang tercepat untuk sampai di garis akhir (sejauh 1500 block) atau dengan mengumpulkan poin sebanyak-banyaknya yaitu dengan menggunakan power-up atau mengambil power-up, karena jika kedua bot tiba di saat yang sama, maka pemenangnya adalah bot dengan skor tertinggi.

b. Pemetaan Pergerakan ke dalam Algoritma Greedy

Pergerakan bisa dibilang aspek terpenting dari permainan ini, karena dengan pergerakan yang baik maka dapat menjadi yang tercepat untuk mencapai garis *finish* dan dengan pergerakan yang baik juga, bisa mendapatkan power up-power up yang kemudian dapat menambahkan skor sehingga jika kebetulan kedua player tiba di waktu yang sama di garis *finish*, player masih dapat menang berdasarkan skornya yang lebih banyak.

Tabel 3.1.2 Pemetaan pergerakan ke dalam Algoritma Greedy

Elemen Algoritma Greedy	Pemetaan pada permainan *Overdrive*
Himpunan Kandidat (C)	Perintah-perintah atau command yang dapat dijalankan di dalam permainan yang mempengaruhi pergerakan dari car seperti Accelerate, Decelerate, Turn Left, Turn Right, Use Boost, dan Use Lizard.
Himpunan Solusi (S)	Perintah terbaik yang dipilih sesuai dengan kondisi atau *state* dari lane dan musuh untuk setiap round selama permainan berlangsung.
Fungsi Solusi	Memeriksa apakah perintah atau *command* yang dipilih merupakan *command* yang terdefinisi di dalam permainan. Jika *command* tidak terdefinisi maka tentu tidak dapat dijalankan.
Fungsi Seleksi (*Selection Function*)	Pemilihan perintah atau *command* yang sesuai dengan prioritas strategi yang digunakan di dalam *bot* yang dibangun untuk implementasi permainan *overdrive* ini. Penentuan prioritas

	dalam bahasa pemrograman yang digunakan dapat digunakan dengan membuat fungsi penyeleksian atau kondisi terurut sesuai dengan strategi yang sudah disusun.
Fungsi Kelayakan (*Feasibility*)	Memeriksa apakah perintah atau *command* yang digunakan memenuhi kondisi untuk dijalankan, validasi yang dimaksud adalah misalnya memastikan memiliki *power-up* yang dimaksud sebelum digunakan, memastikan bot tidak berada di lane 1 sebelum memberikan perintah belok kiri atau memastikan bot tidak berada di lane 4 sebelum memberikan perintah belok kanan, kemudian memastikan bahwa bot tidak memiliki kecepatan maksimal sebelum melakukan accelerate, dan hal lain yang berhubungan dengan pergerakan mobil.
Fungsi Objektif	Memaksimalkan score dan pengambilan power-up yang bisa didapatkan *car* ketika melakukan pergerakan. Selain itu, *car* juga diminimalisir untuk tidak melakukan apa-apa serta menghindari *obstacle* yang ada jika bergerak

c. Pemetaan penggunaan *power-up* dalam algoritma Greedy

Tabel 3.1.3 Pemetaan penggunaan *power-up* dalam Algoritma Greedy

Elemen Algoritma Greedy	Pemetaan pada permainan *Overdrive*
Himpunan Kandidat (C)	Perintah-perintah atau command yang dapat dijalankan di dalam permainan yang menggunakan *power-up,* dimulai dengan USE_(nama power-up)
Himpunan Solusi (S)	Perintah terbaik yang dipilih sesuai dengan kondisi atau *state* dari lane dan musuh untuk setiap round selama permainan berlangsung.
Fungsi Solusi	Memeriksa apakah perintah atau *command* yang dipilih merupakan *command* yang terdefinisi di dalam permainan. Jika *command* tidak terdefinisi maka tentu tidak dapat dijalankan.

Fungsi Seleksi (*Selection Function*)	Pemilihan perintah atau *command* yang sesuai dengan prioritas strategi yang digunakan di dalam *bot* yang dibangun untuk implementasi permainan *overdrive* ini. Penentuan prioritas dalam bahasa pemrograman yang digunakan dapat digunakan dengan membuat fungsi penyeleksian atau kondisi terurut sesuai dengan strategi yang sudah disusun.
Fungsi Kelayakan (*Feasibility*)	Memeriksa apakah perintah atau *command* yang digunakan memenuhi kondisi untuk dijalankan, validasi yang dimaksud adalah misalnya memastikan memiliki *power-up* yang dimaksud sebelum digunakan, serta memastikan penggunaannya tepat sehingga dapat mengganggu musuh atau membantu player mencapai tujuan
Fungsi Objektif	Memaksimalkan *score* dan penggunaan serta pengambilan *power-up* agar kemungkinan untuk menangnya lebih besar.

3.2 Eksplorasi Alternatif Solusi Greedy

Permainan *overdrive* ini bisa dibilang cukup kompleks apalagi dengan adanya *power-up* dan *damage* yang juga berpengaruh pada jalannya bot. Oleh karena itu ada beberapa alternatif solusi yang dapat diimplementasikan untuk mencapai tujuan akhir dari game ini yaitu memenangkan permainan.

Hal utama dalam permainan ini adalah terdapat 2 kategori utama tipe block yang memberi efek berbeda pada player. Kategori pertama ada block yang akan memberi damage pada player yang bisa disebut juga sebagai obstacle, seperti Mud, Oil Spill, dan Wall. Ketiga obstacle ini akan memberikan damage dengan jumlah berbeda pada player yang kemudian akan mempengaruhi kecepatan maksimum dari player. Ketiga block ini dapat muncul secara acak pada map. Oleh karena itu dibutuhkan strategi untuk menghadapi block-block ini. Idelanya, saat menemukan obstacle tentu saja untuk berbelok ke kanan atau ke kiri, namun karena obstacle yang muncul acak, maka bisa saja ternyata terdapat obstacle lain di sebelah kanan dan kiri jalur kita. Maka untuk menghadapi obstacle seperti ini, bisa disusun strategi seperti membiarkan player menabrak obstacle yang memberi damage lebih kecil saat sudah terpojok (tidak memiliki pilihan lain), atau bisa juga dengan membuat player menggunakan *power-up* yang dimiliki. Pemain bisa menyusun strategi dari awal permainan untuk misalnya mencari mana jalur yang memiliki obstacle lebih sedikit lalu kemudian berpindah ke jalur itu. Bisa juga dengan mengurangi kecepatan sehingga tidak menabrak obstacle yang ditemukan.

Selain block obstacle, terdapat juga block *power-up*. Pengambilan dan penggunaan *power-up* akan menambah skor player. Oleh karena itu, pergerakan player untuk mengambil *power-up* juga akan berpengaruh dan harus ada strategi agar tetap mendapat *power-up* namun tidak menjadi kalah dari musuh. Salah satu strategi yang bisa digunakan adalah dengan mencari *power-up* terdekat lalu kemudian berpindah ke jalur yang memiliki block *power-up*. Namun jika berpindah jalur, bisa saja ternyata jalur baru memiliki obstacle lebih banyak. Oleh karena itu, player bisa juga mengecek terlebih dahulu, dan baru berpindah jalur apabila *power-up* yang ada sesuai dengan keadaan saat itu. Misalnya saat berada di belakang musuh, dan ada *block power-up* EMP barulah diambil dengan cara berpindah jalur.

Power-up yang tersedia tidak hanya untuk menambah skor tetapi juga berguna untuk membantu kita dalam menghadapi obstacle-obstacle. Terdapat juga *power-up* yang dapat memperlambat musuh dan memberi obstacle pada jalan musuh sehingga harus berpindah jalur. Alternatif solusi untuk menggunakan *power-up* juga ada banyak. Misalnya untuk *power-up* lizard, karena fungsinya untuk melompat sepanjang round untuk menghindari obstacle maka dapat digunakan saat terdapat obstacle di depan dan di kanan serta kiri. Namun, bisa juga player memilih untuk langsung menggunakan *power-up* ini saat baru mendapatkan *power-up*n-nya agar tidak harus menghindari obstacle-obstacle di *lane*. Selain lizard, ada juga *power-up* lain seperti tweet yang akan menyimpan *cyber truck* di *lane* sesuai yang kita mau dan akan menambah damage musuh saat mereka menabrak. Ada juga boost yang berguna untuk menambah kecepatan player pada round tersebut. Dengan banyaknya *power-up* yang tersedia, dibutuhkan juga strategi yang baik untuk menggunakannya agar tidak terbuang sia-sia.

Dalam permainan ini, strategi yang disusun dapat digunakan dalam kondisi tertentu, sesuai dengan keadaan map dan player juga musuh. Tentu saja, setiap strategi memiliki kelebihan dan kekurangannya masing-masing. Oleh karena itu, dibutuhkan penyusunan strategi yang tepat, yang bisa saling melengkapi sehingga pada akhirnya player dapat memenangkan permainan ini dengan maksimal.

3.3 Analisis Efisiensi dan Efektivitas Kumpulan Solusi

Untuk menemukan solusi berupa strategi yang paling baik untuk menghadapi musuh dalam game ini, dibutuhkan analisis efisiensi dan efektivitas sehingga saat dijalankan, dalam waktu yang sama, player dapat mendapatkan lebih banyak skor atau bisa berjalan dengan lebih cepat hingga sampai di garis *finish*. Oleh karena itu, berikut merupakan analisis efisiensi dan efektivitas dari kumpulan solusi.

3.3.1 Analisis Efisiensi

Untuk strategi pergerakan, kita harus mengecek jalur di kanan, kiri, dan jalur kita saat ini apakah ada obstacle yang dapat memberi damage pada player. Oleh karena itu, kita harus mengiterasi sebanyak jumlah block yang bisa dilihat atau 20 block. Hal ini

berarti kompleksitas waktu yang dibutuhkan untuk mengecek apakah ada obstacle atau tidak sebesar O(n).

Selanjutnya, untuk mencari block dengan *power-up* yang paling dekat dengan *player* dibutuhkan waktu yang lebih lama. Hal ini karena pertama, kita harus mengecek terlebih dahulu apakah di jalur kanan, kiri, dan jalur kita saat ini terdapat block *power-up*. Kemudian setelah ditemukan *power-up*, kita kembali mengiterasi jalur untuk mencari dimana *power-up* yang indeks blocknya berada paling dekat dengan indeks player sekarang. Maka, kompleksitas waktu untuk mencari *power-up* ini adalah sebesar $O(n^2)$.

Untuk strategi menggunakan *power-up*, karena kebanyakan hanya berupa cek kondisi, maka kompleksitas waktu penggunaannya sebesar O(1), misalnya hanya mengecek posisi player dan musuh, atau mengecek posisi musuh saja, atau menambahkan salah satu status dari player.

3.3.2 Analisis Efektivitas

Dalam penyusunan strategi, terdapat beberapa komponen yang kemudian lebih baik untuk diprioritaskan. Namun, dengan memprioritaskan satu hal, tentu saja ada hal lain yang harus dikorbankan. Oleh karena itu, analisis efektivitas ini dilakukan untuk menemukan apa yang harus diprioritaskan.

Pertama, jika kita memprioritaskan untuk terus bergerak maju. Salah satunya adalah dengan tidak mencoba untuk mengambil *power-up* atau menggunakan namun jika kebetulan musuh kita menggunakan *power-up* seperti EMP pada player, maka kecepatan player akan berkurang, dan kemudian dapat dikejar oleh musuh. Strategi memprioritaskan pergerakan ini akan efektif apabila kita mendapat map yang tidak banyak obstacle atau ketika musuh berada jauh di belakang kita.

Kemudian jika kita memprioritaskan untuk mengurus damage pada bot kita. Menabrak obstacle-obstacle yang ada akan membuat bot kita terkena damage yang kemudian akan mempengaruhi kecepatan maksimal bot. Dengan memprioritaskan damage, berarti kita selalu mencoba untuk fix bot yang berarti menghabiskan satu round untuk memperbaiki bot sehingga kemudian tidak jalan sama sekali, namun kecepatan maksimal bot bisa bertambah. Selain itu, bisa juga dengan selalu mencoba untuk menghindari obstacle yang muncul. Strategi ini tidak akan efektif apabila kita kebetulan menghadapi obstacle yang mungkin berhimpitan dengan jalur kita saat ini.

Selanjutnya untuk prioritas *power-up* yang berefek pada player. Penggunaan dan pengambilan *power-up* akan menambah skor kita. Memang jika kebetulan kita tiba di saat yang sama dengan musuh maka skor lebih tinggi akan menang. Sehingga penting juga untuk kita menambah skor. Akan tetapi, jika menggunakan strategi yang terfokus untuk mengambil dan menggunakan *power-up* bisa jadi malah bot kita tertinggal.

Strategi ini akan efektif apabila map sedang tidak banyak obstacle, dan jalur yang sedang kita lalui memiliki banyak block *power-up* karena dengan begitu player tidak harus berpindah-pindah jalur untuk mengambil *power-up*. Namun tentu saja jika ternyata map memiliki banyak obstacle dan *power-up* yang muncul berada di jalur yang berbeda, strategi ini malah akan mengurangi kecepatan dan membatasi pergerakan player.

Terakhir adalah prioritas menyabotase musuh. Pemakaian prioritas ini contohnya seperti menggunakan *power-up* oil, tweet, dan emp atau bisa juga dengan mencoba untuk memblok musuh dengan berada di jalur yang sama dengan musuh. Penggunaan strategi ini akan efektif bila kita memiliki *power-up* yang sesuai, dan musuh berada di depan kita sehingga kita harus mengejar musuh. Akan tetapi, jika kita sudah tertinggal, dan player memprioritaskan untuk mendapatkan *power-up* yang menyabotase musuh, malah akan merugikan player karena tetap tertinggal.

3.4 Strategi yang Dipilih beserta Alasan

Pada akhirnya, karena sistem kerja kami yang tiap anggota kelompok mencoba membuat algoritma bot masing - masing, kami mencoba menggabungkan algoritma yang optimal saat digunakan dan mengganti - ganti urutan prioritas yang bisa dikombinasikan. Hasil yang paling optimal kami dapatkan setelah berbagai pengujian dan kami memutuskan untuk menggunakan prioritas command sebagai berikut : strategi perbaikan mobil dengan memaksimalkan *fix command*, strategi penghindaran penghalang atau *obstacle*, strategi agresif dengan memaksimalkan *power-up commands* seperti *boost, emp, lizard, oil,* dan *tweet*.

Algoritma yang kami pilih untuk diimplementasikan adalah algoritma yang memastikan mobil dapat menempuh jarak yang lebih jauh daripada mobil musuh. Kami memprioritaskan perbaikan mobil terlebih dahulu, dengan tujuan agar kecepatan mobil tidak menjadi terbatas. Selanjutnya, kami memastikan mobil dapat berpindah jalur setiap kali muncul penghalang atau *obstacle*. Algoritma greedy juga kami desain agar tidak "*stuck*", dalam kata lain, bot melakukan *NOTHING*, agar mobil bisa terus melaju setiap ronde.

Selain itu, algoritma *greedy* kami akan memprioritaskan pencarian *power-ups*, karena selain menambah skor, penggunaan *power-up* tentu akan membantu bot untuk memenangkan pertandingan secara lebih mudah, dengan mengganggu bot lawan. Algoritma greedy kami juga memprioritaskan power-up aggresive dengan damage paling tinggi misalnya saat posisi bot player di belakang musuh, menggunakan EMP jika memiliki powerup nya dan menyerang musuh lalu TWEET adalah opsi lain ketika powerup EMP tidak memiliki. Algoritma aggresive ini membuat jarak bot player dan bot musuh semakin jauh jika memimpin dan semakin dekat jika tertinggal, agar dapat memenangkan permainan.

Selain itu, algoritma greedy kami desain untuk membuat bot selalu bergerak dan menghindari munculnya respon do nothing atau invalid command. Urutan prioritas yang kami tentukan berasal dari proses pengujian bot kami melawan reference bot ataupun bot kelompok

lain. Hasilnya, prirotias yang disusun menghasilkan winrate tertinggi dibanding prioritas algoritma lain yang kami kembangkan. Berdasarkan pengujian, bot kami hanya kalah ketika mendapatkan map yang kurang menguntungkan dan merupakan edge case dalam kasus ini.

Bab 4 : Implementasi dan Pengujian

4.1 Implementasi Algoritma Greedy (Pseudocode)

```
function run() -> Command
{ Fungsi utama yang akan menjalankan algoritma greedy bot }

    playerPosBlock <- myCar.position.block
    currentDamage <- myCar.damage

     { Jika mobil terlalu rusak untuk melaju dengan kecepatan penuh, maka lakukan
reparasi }
    if (currentDamage >= 2) then
        -> FIX

     { Jika status mobil terhenti tiba - tiba dan tidak melakukan apapun, lakukan
percepatan }
    if (myCar.state = State.NOTHING) then
        -> ACCELERATE

    { Jika mobil musuh menghalangi mobil kita, maka cari jalur lain }
    if (isEnemyBlocking(myCar, opponentCar)) then
        enemyBlocks <- tryToOvertake(myCar)
        -> enemyBlocks

    { Logika untuk menghindari rintangan }
    if ((checkLaneClearance(myCar, 1)) and (myCar.state != State.HIT_EMP)) then
        ordersLane <- findClearLane(myCar)
        -> ordersLane

    { Jika mobil kita terkena EMP, coba untuk blokir jalur musuh }
    if (myCar.state = State.HIT_EMP) then
        ordersBlock = tryToBlock(myCar, opponentCar)
        -> ordersBlock
```

```
    { Penggunaan TWEET }
    if (hasPowerUp(power-up.TWEET, myCar.power-up)) then
        tweetUsage <- useTweet(myCar, opponentCar)
        -> tweetUsage

    { Jika posisi player sedang berada di belakang musuh }
    if (isPlayerinFront(myCar, opponentCar) = false) then
        { Penggunaan EMP untuk memperlambat musuh }
        if (hasPowerUp(power-up.EMP, myCar.power-up)) then
            -> EMP

        { Penggunaan BOOST untuk mengejar musuh }
        if (hasPowerUp(power-up.BOOST, myCar.power-up)) then
            -> BOOST

        { Mencari power-up tambahan }
           if ((hasAllpower-up(myCar.power-up) = false) and (isEnemyFar(myCar,
opponentCar) = false) and (playerPosBlock > 5) and (playerPosBlock < 1300)) then
            ordersPower <- findpower-up(myCar)
            -> ordersPower

        -> ACCELERATE

    { Jika posisi player sedang berada di depan musuh }
    else if (isPlayerinFront(myCar, opponentCar)) then
        { Penggunaan OIL untuk memperlambat musuh }
        if (hasPowerUp(power-up.OIL, myCar.power-up)) then
            oilUsage <- useOil(myCar, opponentCar)
            -> oilUsage

        { Penggunaan BOOST untuk mengejar musuh }
        if (hasPowerUp(power-up.BOOST, myCar.power-up)) then
            -> BOOST
```

```
        -> ACCELERATE

    -> ACCELERATE

{ Fungsi untuk mengecek ketersediaan power-up di mobil kita }
function hasPowerUp(powerUpToCheck, available) -> boolean
    for (power-up powerUp: available)
        if (powerUp.equals(powerUpToCheck)) then
            -> true
    -> false

{ Fungsi untuk mengecek apakah player memiliki setiap jenis power-up }
function hasAllpower-up(available) -> boolean
    boost, emp, lizard, oil, tweet = 0
    for (power-up powerUp: available) then
        if (powerUp.equals(power-up.BOOST)) then
            boost <- boost + 1
        if (powerUp.equals(power-up.EMP)) then
            emp <- emp + 1
        if (powerUp.equals(power-up.LIZARD)) then
            lizard <- lizard + 1
        if (powerUp.equals(power-up.OIL)) then
            oil <- oil + 1
        if (powerUp.equals(power-up.TWEET)) then
            tweet <- tweet + 1

     if ((boost > 0) and (emp > 0) and (lizard > 0) and (oil > 0) and (tweet >
0)) then
        -> true
    else
        -> false

{ Fungsi untuk mengecek apakah musuh jauh dari player atau tidak }
```

```
function isEnemyFar(myCar, opponentCar) -> boolean
    myPosBlock <- myCar.position.block
    opponentPosBlock <- opponentCar.position.block

    if (isPlayerinFront(myCar, opponentCar) = false) then
        if (opponentPosBlock - myPosBlock > 20) then
            -> true
    else if (isPlayerinFront(myCar, opponentCar)) then
        if (myPosBlock - opponentPosBlock > 20) then
            -> true
    -> false

{ Fungsi untuk mengecek apabila musuh menghalangi posisi mobil kita }
function isEnemyBlocking(myCar, opponentCar) -> boolean
    myPosLane <- myCar.position.lane
    myPosBlock <- myCar.position.block
    opponentPosLane <- opponentCar.position.lane
    opponentPosBlock <- opponentCar.position.block

    if (myPosLane = opponentPosLane) then
        if (opponentPosBlock - myPosBlock = 1) then
            -> true
    -> false

{ Fungsi untuk mengecek apabila posisi player berada di depan musuh }
function isPlayerinFront(myCar, opponentCar) -> boolean
    myPosBlock <- myCar.position.block
    opponentPosBlock <- opponentCar.position.block
    diff <- myPosBlock - opponentPosBlock

    if (diff > 0) then
        -> true
    else
```

```
        -> false


{ Fungsi untuk mengkalkulasi penempatan powerUp TWEET }
function useTweet(myCar, opponentCar) -> Command
    opponentPosLane <- opponentCar.position.lane
    opponentPosBlock <- opponentCar.position.block
    opponentSpeed <- opponentCar.speed
    -> new TweetCommand(opponentPosLane, opponentPosBlock + opponentSpeed + 1)


{ Fungsi untuk menggunakan powerUp OIL, apabila kita berada di lane yang sama
dengan musuh }
function useOil(myCar, opponentCar) -> Command
    myPosLane <- myCar.position.lane
    opponentPosLane <- opponentCar.position.lane


    if (myPosLane = opponentPosLane) then
        -> OIL
    -> ACCELERATE


function tryToOvertake(myCar)-> Command
        myPosLane <- myCar.position.lane;
        { Jika di lane 1, satu - satunya cara menghindari adalah turn_right }
        if (myPosLane = 1) then
            if (!checkLaneClearance(myCar, 2)) then
                -> TURN_RIGHT;
       { Jika di lane 1, satu - satunya cara menghindari adalah turn_left }
        else if (myPosLane == 4) then
            if (!checkLaneClearance(myCar, 0)) then
                -> TURN_LEFT;
       { Jika di lane 2 atau 3, mencari cara optimumn untuk menghindar }
        else
            { Mengecek lane kiri, jika kosong turn left }
            if (!checkLaneClearance(myCar, 0)) then
                -> TURN_LEFT;
```

```
            { Mengecek lane kanan, jika kosong turn right}
            else if (!checkLaneClearance(myCar, 2)) then
                -> TURN_RIGHT;
        -> ACCELERATE;

{ Fungsi untuk mencoba memblokir jalan musuh, membiarkan musuh menabrak kita }
function tryToBlock(myCar, opponentCar) -> Command
    myPosLane <- myCar.position.lane
    opponentPosLane <- opponentCar.position.lane

    { Jika musuh berada di kiri kita atau di kanan kita persis}
    if ((myPosLane = 1) or (myPosLane = 4)) then
        if (opponentPosLane = myPosLane) then
            -> ACCELERATE
        else if ((myPosLane = 1) and (opponentPosLane = 2)) then
            if (checkLaneClearance(myCar, 2) = false) then
                -> TURN_RIGHT
        else if ((myPosLane = 4) and (opponentPosLane = 3)) then
            if (checkLaneClearance(myCar, 0) = false) then
                -> TURN_LEFT

    else if (myPosLane - opponentPosLane = 1) then
        if (checkLaneClearance(myCar, 2) = false) then
            -> TURN_LEFT

    else if (myPosLane - opponentPosLane = -1) then
        if (checkLaneClearance(myCar, 0) = false) then
            -> TURN_RIGHT
    -> ACCELERATE

{ Fungsi untuk memindai list of object untuk mencari keberadaan cybertruck }
function checkCyberTruck(blocks) -> boolean
    for (i = 0; i < blocks.size(); i++)
```

```
        if (blocks.get(i) = "true") then
            -> true
    -> false

{ Fungsi untuk mendapatkan List of Object berisi keberadaan cybertruck }
function getCTinLaneBased(lane, block, whichLane) -> List of Object
    map = gameState.lanes
    blocks = new ArrayList<>()
    startBlock = map.get(0)[0].position.block

    { whichLane = 0 -> left lane, whichLane = 1 -> current lane, whichLane = 2
-> right lane }
    laneList = {}
    if (whichLane = 0) { then
        laneList = map.get(lane - 2)
    else if (whichLane = 1) then
        laneList = map.get(lane - 1)
    else if (whichLane = 2) then
        laneList = map.get(lane)

    for (i = max(block - startBlock, 0); i <= block - startBlock + Bot.maxSpeed;
i++)
        if (laneList[i] == null || laneList[i].terrain == Terrain.FINISH) then
            break
        var = laneList[i].isOccupiedByCyberTruck
        if (var) then
            blocks.add("true")
        else
            blocks.add("false")

    -> blocks

{ Fungsi untuk mendapatkan List of Object berisi elemen terrain di jalur yang
diminta }
function getInfoinLaneBased(lane, block, whichLane) -> List of Object
```

```
    map = gameState.lanes;
    blocks = new ArrayList<>();
    startBlock = map.get(0)[0].position.block;


     { whichLane = 0 -> left lane, whichLane = 1 -> current lane, whichLane = 2
-> right lane }
    laneList = {}
    if (whichLane = 0) { then
        laneList = map.get(lane - 2)
    else if (whichLane = 1) then
        laneList = map.get(lane - 1)
    else if (whichLane = 2) then
        laneList = map.get(lane)


    for (int i = max(block - startBlock, 0); i <= 20; i++)
        if (laneList[i] == null || laneList[i].terrain == Terrain.FINISH) then
            break
        blocks.add(laneList[i].terrain)


    -> blocks
```

4.2 Struktur Data Permainan Overdrive

Pada permainan “Overdrive”, struktur data yang digunakan adalah berbasis objek atau class. Dalam proses pembuatannya, kelompok kami menambahkan beberapa method dalam class Bot untuk menjalankan algoritma greedy. Algoritma greedy yang dikembangkan membagi player dalam bagian yaitu Command, Entities, Enums, Bot, dan Main.

a. Command

Kelas Command berguna untuk menyimpan aksi - aksi yang dapat dilakukan Bot yang akhirnya akan di return di main untuk menjalankan player di permainan.

- Accelerate Command

  Command ini akan menambah speed bot secara perlahan dan melangkah ke depan bergerak sejumlah beberapa block sesuai speed yang dimiliki.

- Boost Command

Command ini akan menggunakan powerup Boost dan membuat speed bot menjadi boost_speed yaitu bernilai 15 dan bertahan selama 5 ronde. Boost akan terhentikan secara otomatis ketika bot mengalami perlambatan karena melewati mud ataupun oil. Jika player menggunakan command ini ketika tidak memiliki powerup Boost maka bot akan diam di tempat.

- Change Lane Command

  Command ini akan menggerakkan bot ke kanan dengan TURN_RIGHT atau TURN_LEFT ke kiri.

- Decelerate Command

  Command ini akan memperlambat speed bot dan tetap berjalan di lane yang sama.

- Do Nothing Command

  Command ini tidak akan menggerakkan bot. Speed dan posisi bot akan berada di tempat yang sama.

- Emp Command

  Command ini akan menembakkan EMP/roket dari depan bot menuju ke lane yang sama, kiri dan kanan posisi player saat menembakkan EMP/roket. EMP akan menghentikan gerak bot lawan dan menurunkan speed bot lawan menjadi 3.

- Fix Command

  Command ini akan menghilangkan 2 point damage dari bot dan akan berhenti sementara di posisi tersebut.

- Lizard Command

  Command ini akan menggunakan powerup Lizard dan membuat bot melompat melewati obstacle yang ada sesuai dengan speed yang dimiliki bot. Ketika melompat pada block yang sudah ada bot lawan maka otomatis bot player akan dimundurkan satu block sebelum tujuan block mula - mula. Jika player menggunakan command ini ketika tidak memiliki powerup Lizard maka bot akan diam di tempat.

- Oil Command

  Command ini akan memberikan obstacle oil tepat di bawah mobil bot player. Jika bot lawan melewati oil yang ditaruh bot player, speed bot lawan akan berkurang. Jika player menggunakan command ini ketika tidak memiliki powerup Oil maka bot akan diam di tempat.

- Tweet Command

Command ini menerima dua integer yaitu posisi lane dan block dimana bot player ingin menaruh obstacle cybertruck pada round selanjutnya. Jika bot player mencoba menaruh cybertruck di lokasi dimana cybertruck musuh sudah terletak maka powerup akan di refund dan cybertruck tetap berada di tempat semula. Jika kedua bot berusaha menaruh cybertruck di lokasi yang sama maka kedua powerup akan di refund pada masing - masing bot dan cybertruck tetap berada di tempat semula. Jika player menggunakan command ini ketika tidak memiliki powerup Tweet maka bot akan diam di tempat.

b. Entities

Kelas-kelas pada bagian ini berguna untuk menyimpan data state dari player.

- Car

  Dalam kelas ini, disimpan beberapa data yang berhubungan dengan mobil dari player. Seperti *id, position, speed, damage, state, power-up, boosting* dan *boostcounter*. Speed berisi kecepatan car sekarang dengan maksimal di 15. Damage berisi banyaknya damage yang diterima car dengan minimal 0 dan maksimal 5, semakin bertambah damage maka kecepatan maksimal yang dimiliki car akan berkurang, state berisi status terakhir mobil setelah dilakukan perintah pada round sebelumnya seperti setelah menabrak *object surface*, atau menggunakan *power-up*. power-up berisi power-up yang dimiliki player, boosting menunjukkan apakah player menggunakan boost atau tidak, boostercounter menunjukkan sisa round hingga boost habis.

- GameState

  Berisi state permainan saat ini. Seperti currentRound, maxRounds, player, opponent, dan worldMap. CurrentRound menunjukkan sekarang round ke berapa, maxRounds menunjukkan maksimum Round yang ada dalam match ini. Player menunjukkan detail player, dan opponent menunjukkan detail musuh yaitu id, posisi dan kecepatan. Terakhir WorldMap berisi array dari objek yang mendeskripsikan block-block yang terlihat di map.

- Lane

  Lane berisi position, surfaceobject, occupiedbyplayerid, isoccupiedbycybertruck. Position berisi tempat block tertentu di mapnya. SurfaceObject berisi apa yang ada di blok ini. Identifier blocknya berupa integer dari 0-9, contohnya ada block yang berisi mud, berisi oil spill, atau block finish atau beberapa block berisi power-up lain.

- Position

Berisi x dan y dimana y melambangkan lane (1-4) dan x melambangkan block tempat player berada saat ini.

c. Enums

Pada bagian ini menyimpan tipe - tipe block yang ada seperti kondisi block, powerup, arah dan state pada map permainan.

- Direction

  Enumerasi direction memiliki 4 jenis arah yang memiliki atribut lane, block, dan label. Terdapat direction “FORWARD”, “BACKWARD’, “LEFT”, dan “RIGHT”.

- power-up

  Enumerasi power-up adalah jenis - jenis power-up yang terdapat pada map yaitu boost, emp, lizard, tweet, dan oil.

- State

  Enumerasi state berisi jenis - jenis keadaan bot pada suatu ronde yaitu misalnya, ACCELERATING, READY, NOTHING, TURNING_RIGHT, TURNING_LEFT, HIT_MUD, HIT_OIL, HIT_EMP, DECELERATING, PICKED_UP_POWERUP, USED_BOOST, USED_OIL, USED_LIZARD, USED_TWEET, HIT_WALL, HIT_CYBER_TRUCK, FINISHED.

- Terrain

  Enumerasi terrain adalah hal - hal apa saja yang bisa terdapat pada suatu block di map yaitu termasuk powerup ability yang sudah didefinisikan di power-up dan obstacle yang ada seperti WALL, MUD, dan OIL_SPILL ataupun kosong.

d. Kelas Bot

Kelas ini berisi pengembangan implementasi Algoritma Greedy yang kami buat untuk memenangkan permainan *overdrive* ini. Implementasi algoritma ini dilakukan dengan memanfaatkan kelas-kelas yang ada sebelumnya,

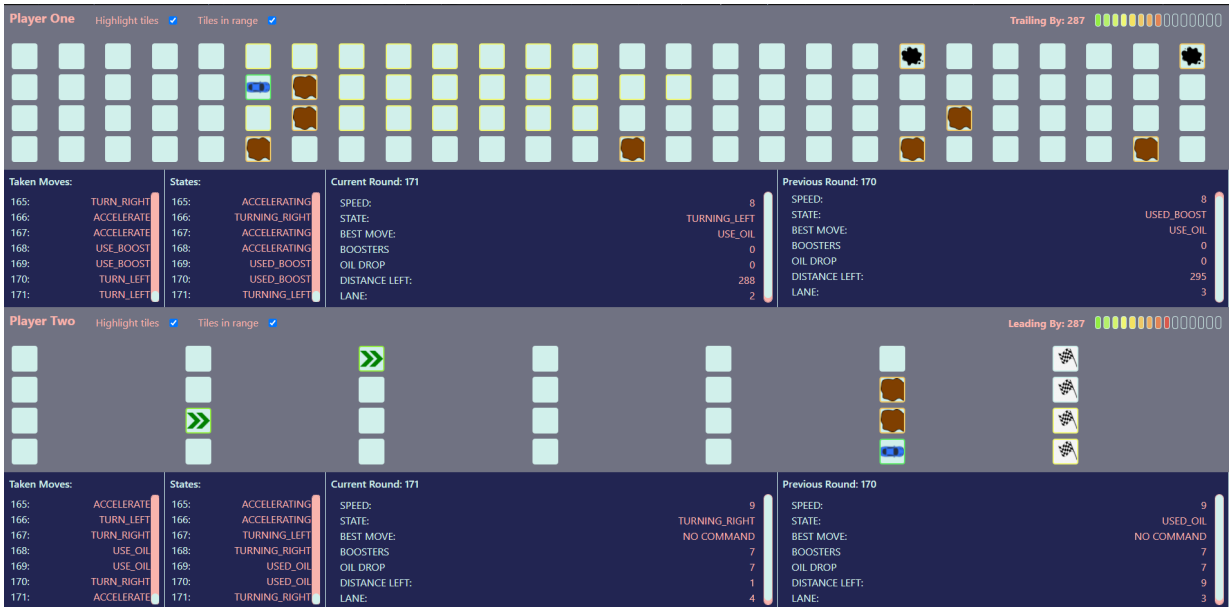
e. Kelas Main

Kelas ini berfungsi untuk memanggil kelas Bot dan menerima command yang dijalankan dengan method run lalu menjalankan atau memulai program dengan menjalankan seluruh algoritma dan command-command yang sudah dibentuk untuk memulai permainan.

4.3 Analisis Desain Solusi Algoritma Greedy

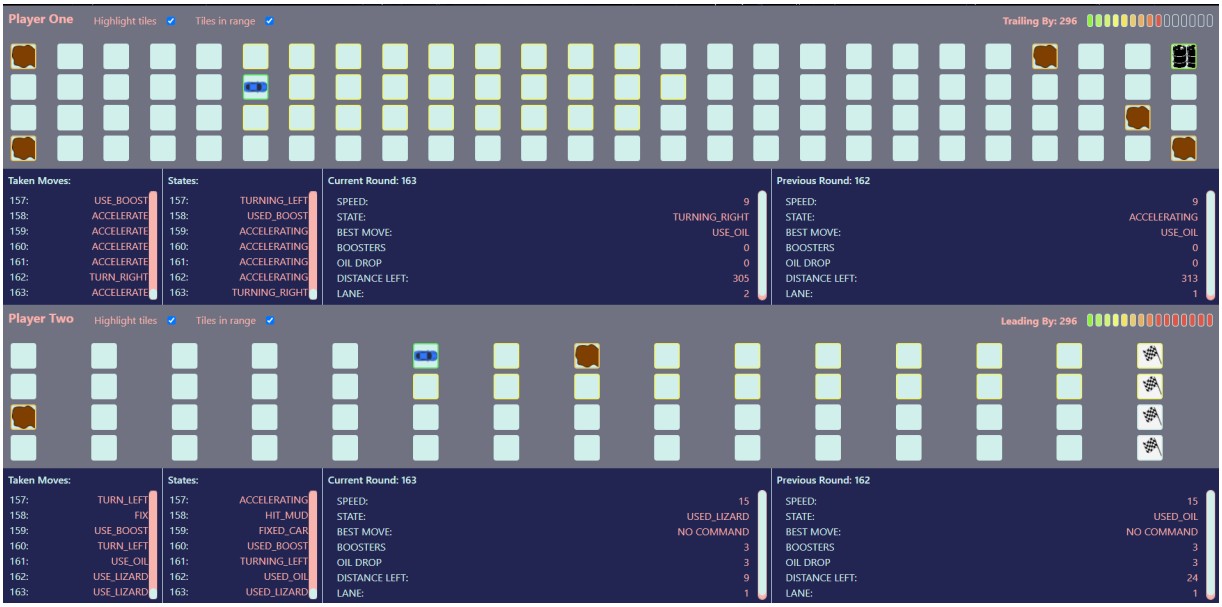
Percobaan dilakukan dengan beberapa data uji dengan menjalankan `run.bat` pada folder starter-pack (src jika sudah diganti) yang berisi bot yang telah kami kembangkan. Dilakukan tiga kali pengujian terhadap bot yang telah kami rancang.

a. Pengujian 1



Pada awal round, Bot kami kalah cepat dibanding dengan Bot 1. Akan tetapi, memasuki round ke-5, bot kami dapat mengejar dan di round selanjutnya malah mendahului Bot 1. Pada round-round selanjutnya, bot kami juga memperlebar jarak yang ada. Hal ini mungkin terjadi karena setelah dilihat dari match log, bot musuh banyak menggunakan belok kanan atau kiri yang mengakibatkan player tidak maju dengan penuh (misalnya speed 6, tapi karena berbelok jadi 5) sehingga jarak dengan bot kami semakin jauh. Pada round akhir, terlihat bahwa bot kami memenangkan pertandingan dengan perbedaan yang cukup jauh yaitu sekitar 200 block. Hal ini disebabkan karena bot yang kami buat sudah memiliki algoritma yang cukup baik, dan bot musuh yang kami gunakan untuk perbandingan masih merupakan bot awal yang kami buat sehingga algoritma yang dipakai oleh bot musuh juga masih belum begitu efisien dan efektif.

b. Pengujian 2



Pada pengujian kedua, *bot* yang kami rancang memenangkan pertandingan melawan *bot* musuh. Dapat terlihat dari data *leading by* yang tertera di gambar, *bot* kami menang sejauh 296 *block* dibandingkan dengan musuh. Hal ini disebabkan oleh algoritma greedy yang kami gunakan membuat *bot* akan selalu mencoba untuk menghindari *obstacle* seperti *wall* atau *mud*. Alhasil, *bot* kami tidak mengalami banyak pengurangan kecepatan dan *damage* yang diterima pun minimal, sehingga tidak memerlukan *fix command* yang terlalu sering.

c. Pengujian 3

Pada pengujian ketiga, *bot* yang kami rancang memenangkan pertandingan dengan perbedaan yang lebih tipis dibandingkan kedua pengujian sebelumnya. Hal ini dikarenakan *bot* musuh yang kami gunakan adalah *bot* yang sudah menggunakan algoritma greedy, tetapi belum diimplementasikan secara penuh (dalam kata lain, *bot* musuh merupakan iterasi *bot* utama kami sebelumnya). Oleh karena itu, *bot* musuh mampu menghindari *obstacle* secara efisien, sehingga jarak antar mobil dekat. Akan tetapi, *bot* musuh tidak memiliki algoritma greedy untuk mencari *power-up*, sehingga jumlah *power-up* yang bisa digunakan oleh musuh tergolong sedikit. Sedangkan untuk *bot* kami, algoritma greedy untuk mencari *power-up* sudah cukup efisien sehingga jumlah *power-up* yang bisa digunakan lebih banyak.

Bab 5 :Kesimpulan dan Saran

5.1 Kesimpulan

Implementasi permainan Overdrive berhasil kami lakukan menggunakan algoritma greedy yang bisa mencapai tujuan objektif dari permainan ini yaitu mencapai garis finish terlebih dahulu atau memiliki score yang lebih banyak dari musuh ketika finish bersamaan. Selain itu, kami juga menggunakan beberapa pendekatan untuk memenangkan permainan yaitu mencari power-up terdekat, navigasi selalu mencari lane tanpa obstacle, dan menggunakan *power-ups* terbaik tiap ronde.

Pada tugas besar 1 Strategi Algoritma kali ini pun, kami juga menyadari bahwa algoritma greedy yang kelompok kami buat tidak selalu membuahkan hasil yang paling optimum. Hal ini dapat terlihat ketika kami kalah dalam pengujian bot kami melawan bot teman kelompok lain. Selain karena perubahan gameState dan map yang acak pada setiap kali pengujian, beberapa kondisi map yang sangat jarang ditemukan dan tidak kami masukkan ke dalam algoritma greedy navigasi lane kosong (findClearLane) kami, sesuai dengan nature algoritma greedy dimana kurang meninjau semua kemungkinan yang ada secara keseluruhan.

5.2 Saran

Untuk pengembangan selanjutnya, bot dapat dikembangkan untuk memiliki urutan prioritas power-up dan penghindaran obstacle yang optimal dengan dites pada berbagai map yang ada.

Lampiran

Link Github Repository : https://github.com/hcarissa/overcooked-bot

Daftar Pustaka

Entelect Forum. Entelect Overdrive 2021 Challenge. Diakses pada 5 Februari 2022 pukul 18.21 WIB melalui https://forum.entelect.co.za/.

Munir, R. (2022). Algoritma Greedy Bagian 1[PDF]. Institut Teknologi Bandung. Diakses pada 5 Februari 2022 pukul 11.21 WIB melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

Munir, R. (2022). Algoritma Greedy Bagian 2[PDF]. Institut Teknologi Bandung. Diakses pada 8 Februari 2022 pukul 11.33 WIB melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag2.pdf

Munir, R. (2022). Algoritma Greedy Bagian 3[PDF]. Institut Teknologi Bandung. Diakses pada 11 Februari 2022 pukul 11.47 WIB melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag3.pdf

Aziz, F., Lukman, M., Rivaldo, R. (2021). Laporan Hasil Tugas Besar I Pemanfaatan Algoritma Greedy dalam Aplikasi Permainan “Worms”, Institut Teknologi Bandung. Diakses pada 16 Februari 2022 pukul 11.53 WIB melalu https://github.com/RichardRivaldo/Worms/blob/main/doc/send%20hlp%20pls.pdf i