

LAPORAN TUGAS BESAR I

IF2211 STRATEGI ALGORITMA

Laporan dibuat untuk memenuhi salah satu tugas mata kuliah
IF2211 Strategi Algoritma



Disusun oleh:

Kelompok Bot Jaya 99

13520001 Fayza Nadia

13520014 Muhammad Helmi Hibatullah

13520100 Averrous Saloom

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
SEMESTER 2 TAHUN 2021/2022

DAFTAR ISI

DAFTAR ISI	2
BAB I	3
BAB II	4
Dasar Teori Algoritma Greedy	4
Cara Kerja Program	5
BAB III	6
Elemen Algoritma Greedy	6
Eksplorasi Alternatif Solusi Greedy	9
Analisis Efisiensi	10
Analisis Efektivitas	11
Pemilihan Strategi Greedy	12
BAB IV	14
Implementasi pada Program Bot dalam Game Engine	14
Struktur Data	19
Analisis pada Pengujian	22
BAB V	26
Kesimpulan	26
Saran	26
TAUTAN REPOSITORY GITHUB	27
DAFTAR PUSTAKA	27

BAB I

DESKRIPSI TUGAS

Overdrive adalah sebuah game yang mempertandingkan dua 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. Pada tugas besar pertama Strategi Algoritma ini, digunakan sebuah *game engine* yang mengimplementasikan permainan Overdrive. Tugas mahasiswa adalah mengimplementasikan bot mobil dalam permainan Overdrive dengan menggunakan strategi *greedy* untuk memenangkan permainan.

Strategi *greedy* yang diimplementasikan tiap kelompok harus dikaitkan dengan fungsi objektif dari permainan itu sendiri, yaitu memenangkan permainan dengan cara mencapai garis *finish* lebih awal atau mencapai garis *finish* bersamaan tetapi dengan kecepatan lebih besar atau memiliki skor terbesar jika kedua komponen tersebut masih bernilaiimbang. Salah satu contoh pendekatan *greedy* yang bisa digunakan (pendekatan tak terbatas pada contoh ini saja) adalah menggunakan *power ups* begitu ada untuk mengganggu mobil musuh. Strategi *greedy* harus dibuat sebaik mungkin, karena setiap bot dari masing-masing kelompok akan diadu satu sama lain dalam suatu kompetisi Tubes 1 (TBD). Strategi *greedy* harus dijelaskan dan ditulis secara eksplisit pada laporan, karena akan diperiksa pada saat demo apakah strategi yang dituliskan sesuai dengan yang diimplementasikan. Tiap kelompok dapat menggunakan kreativitas mereka dalam menyusun strategi *greedy* untuk memenangkan permainan. Implementasi pemain harus dapat dijalankan pada *game engine* yang telah disebutkan pada spesifikasi tugas besar, serta dapat dikompetisikan dengan pemain dari kelompok lain.

BAB II

LANDASAN TEORI

Dasar Teori Algoritma *Greedy*

Secara informal, algoritma *greedy* adalah algoritma yang rakus. Kelas algoritma ini disebut rakus karena selalu mengambil jalan yang terbaik di setiap langkah dengan harapan bahwa himpunan dari langkah-langkah terbaik tersebut dapat menghasilkan solusi paling optimal.

Algoritma *greedy* adalah algoritma yang memecahkan persoalan secara langkah per langkah sedemikian sehingga pada tiap langkah mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan dan berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Algoritma ini memiliki enam komponen penting, yakni:

1. Himpunan kandidat, kandidat yang dapat dipilih pada setiap langkah.
2. Himpunan solusi, kandidat yang sudah dipilih
3. Fungsi solusi, mengevaluasi apakah kandidat yang dipilih memberikan solusi
4. Fungsi seleksi, memilih kandidat dengan strategi *greedy* tertentu
5. Fungsi kelayakan, mengevaluasi apakah kandidat yang dipilih sudah layak untuk menjadi solusi
6. Fungsi objektif, mengevaluasi apakah solusi sudah minimum atau maksimum

Secara formal, dapat dikatakan bahwa:

Algoritma greedy melibatkan pencarian sebuah himpunan bagian, S , dari himpunan kandidat, C ; yang dalam hal ini, S harus memenuhi beberapa kriteria yang ditentukan, yaitu S menyatakan suatu solusi dan S dioptimisasi oleh fungsi obyektif. (Munir, 2021)

Algoritma *greedy* tidak akan selalu menghasilkan solusi yang optimal. Sesuai definisinya, algoritma ini hanya mengambil langkah yang terbaik pada saat itu dan tidak mengambil solusi untuk jangka panjang atau solusi optimum lokal. Oleh karena itu, tidak jarang pula ditemukan

algoritma *greedy* menghasilkan solusi yang mendekati solusi optimal atau disebut solusi sub optimum.

Cara Kerja Program

Overdrive adalah sebuah kompetisi algoritma yang menggunakan balapan mobil sebagai masalah utama. Kompetisi ini sudah menyediakan berbagai hal dasar seperti *game engine*, konfigurasi standar, serta console sederhana untuk analisis. Untuk menjalankan program, kita perlu mendefinisikan `bot.json` dalam folder bot sesuai dengan nama bot, identitas pembuat, bahasa bot, lokasi, serta nama file bentuk final bot yang sudah bisa dirun (biasanya di-*build*). Lalu setup pada `game-runner-config.json`, dua buah nama serta folder bot untuk dipertandingkan. Setelah setup selesai, jalankan file `run.bat` untuk menjalankan pertandingan serta memunculkan console.

Secara standar, implementasi algoritma ditulis pada file `bot.*`, ekstensi tergantung bahasa yang dipilih. Overdrive juga sudah menyediakan template dasar untuk implementasi pada tiap bahasa sehingga para peserta tidak perlu mengerjakan banyak *overhead* agar bisa menjalankan kode.

Untuk memulai sebuah pertandingan, siapkan dua buah file build dari bot. Lalu taruh kedua bot di dua folder berbeda. Untuk memberikan identifikasi dan setup pada setiap bot buat file '`bot.json`' pada tiap folder sesuai pada aturan. Selanjutnya, konfigurasikan file '`game-runner-config.json`' yang berada di folder utama untuk menunjuk player A ke satu bot dan player B ke bot lain. Setelah semuanya sudah disetup, jalankan file '`run.bat`'.

BAB III

APLIKASI STRATEGI *GREEDY*

Elemen Algoritma *Greedy*

a. Pemetaan Umum ke dalam Algoritma *Greedy*

Berdasarkan enam komponen yang telah diidentifikasi, didapatkan elemen-elemen algoritma *greedy* pada permainan Overdrive adalah sebagai berikut:

Elemen Algoritma <i>Greedy</i>	Pemetaan pada Permainan Overdrive
Himpunan Kandidat (C)	Seluruh perintah yang bisa dijalankan pada permainan Overdrive, yakni tidak melakukan apa-apa, mempercepat, memperlambat, belok kiri, belok kanan, menggunakan <i>boost</i> , menggunakan <i>oil</i> , menggunakan <i>tweet</i> , menggunakan <i>lizard</i> .
Himpunan Solusi (S)	Perintah terbaik pada tiap ronde permainan.
Fungsi Solusi	Memeriksa apakah perintah yang dipilih adalah perintah yang terdefinisi atau tidak di dalam permainan.
Fungsi Seleksi (<i>Selection Function</i>)	Melakukan pemilihan perintah sesuai prioritas strategi yang dimiliki.
Fungsi Kelayakan (<i>Feasibility</i>)	Memeriksa apakah dampak dari suatu perintah terbaik yang dipilih dapat dipenuhi oleh aturan permainan.
Fungsi Objektif	Memenangkan permainan Overdrive, yakni, dengan prioritas sesuai urutan: memenangkan pertandingan, memiliki kecepatan tertinggi saat <i>finish</i> , dan memiliki

	skor tertinggi.
--	-----------------

b. Pemetaan Penghindaran *Obstacle* ke dalam Algoritma *Greedy*

Halangan membuat bot menjadi rusak sehingga membatasi kecepatan yang bisa diraih. Dengan kemampuan menghindari halangan yang baik, bot dapat memaksimalkan kecepatan dan memenangkan pertandingan. Selain itu, halangan mengurangi skor bot.

Elemen Algoritma <i>Greedy</i>	Pemetaan pada Permainan <i>Overdrive</i>
Himpunan Kandidat (C)	Perintah-perintah yang berkaitan dengan penghindaran dalam permainan <i>Overdrive</i> , yaitu belok kiri, belok kanan, memperlambat, mempercepat, dan menggunakan <i>lizard</i> .
Himpunan Solusi (S)	Perintah terbaik yang dipilih sesuai dengan kondisi jalan di depan dan di sekitar mobil pada setiap ronde permainan selama belum menyentuh garis <i>finish</i> .
Fungsi Solusi	Memeriksa apakah perintah yang dipilih adalah perintah yang terdefinisi atau tidak di dalam permainan.
Fungsi Seleksi (<i>Selection Function</i>)	Pemilihan perintah yang sesuai dengan prioritas strategi yang dihitung menggunakan penilaian jalur. Perhitungan skor jalur didasarkan pada keberadaan rintangan dan <i>power up</i> . Skor jalur yang lebih tinggi menjadi dasar pengambilan keputusan.
Fungsi Kelayakan (<i>Feasibility</i>)	Memeriksa bahwa semua kondisi yang diperlukan untuk mengesekusi perintah yang dipilih sudah terpenuhi atau belum. Validasi dilakukan terhadap koordinat, kecepatan, <i>obstacle</i> yang ada, serta <i>power</i>

	<i>up</i> yang dimiliki saat ronde permainan.
Fungsi Objektif	Meminimalkan kerusakan yang diterima, pengurangan kecepatan yang terjadi.

c. Pemetaan Penggunaan *Power Up* ke dalam Algoritma *Greedy*

Power up merupakan salah satu strategi yang dapat digunakan oleh pemain untuk mempercepat bot pemain ataupun memperlambat bot lawan. Penggunaan *power up* yang tidak tepat dapat merugikan bot pemain sehingga harus melihat solusi seoptimal mungkin di tengah keterbatasan bot pemain.

Elemen Algoritma <i>Greedy</i>	Pemetaan pada Permainan <i>Overdrive</i>
Himpunan Kandidat (C)	Perintah-perintah yang berkaitan dengan menggunakan <i>power up</i> .
Himpunan Solusi (S)	Perintah terbaik untuk dalam menggunakan <i>power up</i> .
Fungsi Solusi	Memeriksa apakah perintah yang dipilih adalah perintah yang terdefinisi atau tidak di dalam permainan.
Fungsi Seleksi (<i>Selection Function</i>)	Memeriksa kondisi bot serta bot lawan apakah optimal untuk menggunakan <i>power ups</i> . Memeriksa kerusakan yang dialami bot yang dapat menghambat penggunaan <i>power ups</i> .
Fungsi Kelayakan (<i>Feasibility</i>)	Memeriksa apakah <i>power ups</i> yang dipilih dimiliki sehingga perintah penggunaan <i>power up</i> tidak merugikan pemain.
Fungsi Objektif	Memaksimalkan dampak dari <i>power up</i> . Dampak kerusakan dan perlambatan bagi bot musuh serta

	dampak percepatan dan penghindaran halangan bagi bot.
--	---

Eksplorasi Alternatif Solusi *Greedy*

Dalam menyelesaikan permainan Overdrive, terdapat berbagai pendekatan yang dapat dipilih oleh pemain. Pendekatan yang dipilih akan sangat berpengaruh terhadap hasil dari permainan Overdrive sehingga diperlukan strategi yang tepat guna untuk mencapai tujuan masing-masing pemain dalam pertandingan.

a. Pendekatan *Speed* sebagai Prioritas Utama

Pendekatan dengan elemen *speed* sebagai prioritas utama memiliki arti memaksimalkan *speed* bot pemain pada setiap ronde. Untuk memaksimalkan *speed*, pemain dapat memprioritaskan untuk melakukan percepatan pada setiap kondisi yang memungkinkan. Kondisi tersebut di antaranya adalah dengan mempertimbangkan posisi bot lawan serta letak *obstacle*, terutama *truck* dan *wall*, karena dapat mengurangi *speed* pemain secara drastis. Hal ini dikarenakan tabrakan baik dengan bot lawan maupun *obstacle* dapat mengurangi *speed*.

Selain mengutamakan percepatan, pendekatan ini juga akan memprioritaskan penggunaan *power up boost* jika pemain memilikinya. Pendekatan ini juga membutuhkan pemain untuk melakukan *command fix* jika *damage* yang bot pemain miliki lebih kecil dari batas tertentu.

b. Pendekatan *Score* sebagai Prioritas Utama

Pendekatan *score* berarti memaksimalkan skor yang dimiliki oleh bot. Hal ini dilakukan dengan menghindari *obstacle* yang dapat mengurangi skor (*mud* dan *oil spill*), mengambil *power up*, menggunakan *power up*, serta menghindari *invalid command*.

Pada pendekatan ini, dapat dilakukan perhitungan terhadap skor jalur. Perhitungan skor ini akan berguna bagi bot pemain dalam menentukan jalur dengan *power up* terbanyak

dan *obstacle* tersedikit. Jalur kosong akan dianggap memiliki skor *default* nol. Dalam beberapa *blocks* ke depan, jika ditemukan adanya *power up*, maka skor jalur akan bertambah dan sebaliknya jika ditemukan adanya *obstacle* pada jalur tersebut. Banyaknya *power up* yang didapatkan akan memiliki pengaruh yang besar terhadap *score*. Hal ini dikarenakan aturan permainan Overdrive yang memberikan tambahan *score* jika pemain menggunakan *power up*.

c. Pendekatan *Boost* sebagai Prioritas Utama

Pendekatan dengan *power up boost* sebagai prioritas utama memiliki arti melakukan pengambilan *power up boost* dan menggunakannya semaksimal mungkin. Pengambilan *power up boost* dapat dilakukan dengan melakukan pemeriksaan jauh terhadap *block* di depan dan sekitar pemain sehingga pemain dapat maju bersama dengan melakukan belok ke arah *power up boost* berada.

Selain itu, pendekatan ini juga dilakukan dengan menjaga *damage* bot pemain tetap pada angka 0. Hal ini dikarenakan *damage* yang diterima pemain memiliki dampak terhadap *speed* maksimal yang pemain bisa capai. Untuk menjaga *damage* seminimal mungkin, pemain dapat melakukan *command fix* agar dapat menggunakan *power up boost* seoptimal mungkin.

Analisis Efisiensi

Program kebanyakan hanya melibatkan logika *if then else* sehingga pada setiap ronde jumlah komputasi yang dilakukan cenderung bernilai konstan, tidak bertumbuh *respective* terhadap apa pun. Maka dari itu kompleksitas algoritma pada setiap round adalah $O(K)$ dengan K adalah jumlah pengecekan blok serta logika *if then else*. Dan untuk seluruh kompetisi adalah $O(R * K)$ dengan R sebagai jumlah ronde yang dilakukan, dimana nilainya juga cenderung konstan.

Analisis Efektivitas

Dengan melakukan uji coba pada berbagai *match*, kami menemukan berbagai temuan yang membantu kami merancang strategi *greedy* terbaik dalam permainan Overdrive.

Objektif dari permainan Overdrive ini adalah mencapai *block finish* lebih cepat dari bot lawan. Jika bot pemain dan bot lawan melewati *block finish* pada waktu bersamaan, akan dilakukan pemeriksaan terhadap *speed* keduanya. Pemain dengan *speed* yang lebih tinggi akan diprioritaskan untuk menjadi pemenang permainan Overdrive. Namun, jika kedua pemain memiliki *speed* yang sama, maka permainan akan memprioritaskan bot dengan skor terbesar antara keduanya. Objektivitas tersebut kemudian menjadi dasar dalam menentukan tingkat efektivitas masing-masing pendekatan.

Pada percobaan awal, dilakukan pendekatan dengan mengutamakan *speed*, yakni melakukan percepatan jika tidak ada *obstacle* yang menghambat jalannya bot pemain. Jika terdapat *obstacle*, maka mobil akan melakukan *turn* ke jalur di sebelahnya. Kedua hal ini merupakan hal utama dan terpenting yang menjadi dasar permainan Overdrive ini. Pada saat ini, bot pemain sudah dapat mengalahkan *reference* bot yang tersedia, namun bot pemain belum bergerak dikarenakan logika yang masih sangat sederhana. *Power up* selain *boost* yang didapatkan juga menjadi terbuang-buang jika hanya memprioritaskan *speed* tanpa mementingkan objektif lain.

Sedangkan, pendekatan dengan *score* sebagai prioritas utama merupakan cara efektif untuk mendapatkan *score* sebanyak mungkin. Namun, penting untuk diingat bahwa objektivitas utama permainan Overdrive terletak pada pemain yang mencapai *block finish* terlebih dahulu. Sehingga, pendekatan ini tidak dapat diambil sepenuhnya karena ketidakefektifannya dalam mencapai *finish* secepat mungkin. Tetapi, perhitungan skor pada jalur yang dilewati merupakan hal yang sangat efisien untuk membantu pemain menentukan jalur yang mana yang terbaik untuk ditempuh jika dihadapi oleh *obstacle* pada lebih dari satu jalur.

Terakhir, pendekatan dengan *power up boost* sebagai prioritas utama di mana bot pemain akan berusaha mendapatkan dan menggunakan *boost* sesering mungkin. *Power up boost* merupakan satu dari beberapa *command* yang memiliki dampak sangat besar. Namun, jika pemain terus

melakukan *command fix* agar dapat menggunakan *power up boost* pada kondisi optimalnya, pemain dapat terus tertinggal sehingga pendekatan ini tidak dapat dikatakan efektif sepenuhnya. Hal ini dikarenakan *command fix* yang mengharuskan pemain untuk diam selama satu ronde. *Power up boost* yang dapat digagalkan jika musuh menggunakan *power up* juga menjadikan pendekatan kurang efektif karena dapat membuang *power up boost*.

Pemilihan Strategi Greedy

Setelah melakukan analisis lebih lanjut terhadap masing-masing pendekatan, kami memilih untuk menggabungkan seluruh pendekatan di atas. Dengan mengurutkan pendekatan-pendekatan tersebut sesuai heuristik dan analisis langkah per langkah, kami mendapati urutan yang paling optimal adalah: memastikan kecepatan tetap tinggi dengan memperbaiki kerusakan dan mempercepat, menghindari halangan di depan baik dengan berbelok atau menggunakan *lizard*, menggunakan *boost*, menyerang pemain lawan dengan *EMP* dan *tweet*, dan memperbaiki bot untuk menggunakan *boost*.

Selagi kecepatan belum menyampai maksimal, bot yang kami buat akan berusaha untuk mempercepat sambil mempertahankan kecepatan yang sudah dibangun. Untuk mempertahankan kecepatan, bot harus pandai menghindari *obstacle*. Selain itu, menghindari *obstacle* juga dapat mencegah pengurangan poin. Dari kedua hal ini, kami memutuskan bahwa algoritma penghindaran adalah hal yang sangat penting. Algoritma penghindaran yang kami buat dilakukan dengan menilai ketiga lane (dua jika sedang ada di samping) mulai dari depan mobil hingga pada kecepatan saat itu. Penilaiannya adalah sebagai berikut.

1. *Mud* mengurangi skor jalur sebanyak dua. *Mud* dapat menurunkan tingkat kecepatan sampai satu tingkat dan memberi kerusakan sebanyak satu.
2. *Oil spill* mengurangi skor jalur sebanyak satu. Walaupun *oil spill* mengurangi poin lebih banyak daripada *mud* dan dapat mengurangi kecepatan seperti *mud*, tetapi *oil spill* lebih jarang muncul dan tidak memberi kerusakan sama sekali.
3. *Wall* dan *cyber truck* mengurangi skor jalur sebanyak lima. *Wall* dan *cyber truck* sama-sama menurunkan kecepatan menjadi tiga. Maka dari itu kedua rintangan ini sangat berbahaya.

4. *Boost* akan menambah skor jalur sebanyak dua. Karena kami menggabungkan ketiga pendekatan yang kami temukan, kami juga mengatur harga yang cukup tinggi untuk *power up boost* ini.
5. *Tweet* akan menambah skor jalur sebanyak satu. *Tweet* merupakan salah satu *power up* yang cukup kuat namun *power up* ini sulit diprediksi keakuratan penempatannya karena kita tidak tahu pasti pergerakan lawan dan hanya bisa menebak-nebak.
6. *EMP* akan menambah skor jalur sebanyak dua jika kita ada di belakang lawan dan nol jika di depan lawan. Hal ini dilakukan karena kami hanya memprioritaskan *EMP* ketika kami tertinggal jauh di belakang lawan.

Dengan akumulasi skor di tiap jalur, akan dipilih jalur dengan poin terbesar, yaitu *obstacle* yang sesedikit mungkin dan *power ups* sebanyak mungkin.

Selain mengutamakan *speed*, kami membuat beberapa kasus khusus di antaranya adalah kasus di mana bot sudah memiliki kecepatan maksimum dan tidak memiliki *damage*. Pada saat ini, kami akan menggunakan *power up* yang kami miliki, dalam hal ini *oil*. Tentunya, *oil* yang digunakan kecil kemungkinannya untuk tepat sasaran, terutama jika jarak antara kedua bot sangatlah jauh. Namun, penggunaan *oil* selagi masih memiliki *power up oil* menjadi pilihan yang cukup menguntungkan dibandingkan melakukan *acceleration* ataupun *boosting* ketika *speed* yang dimiliki tidak dapat meningkat lagi. Selain masih berkemungkinan untuk mencelakai bot lawan, penggunaan *power up oil* jugalah menambahkan skor.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

Implementasi pada Program *Bot* dalam *Game Engine*

Berikut implementasi beberapa fungsi penting dalam bentuk *pseudocode*:

```
function run() -> Command
    // Fix first if too much damage
    if damage >= 2 then FIX

    // Accelerate if speed below 4 and there's no obstacle ahead
    if speed <= 3 and no obstacle in front until next speed state then ACCELERATE

    // Dodge logic
    if there is obstacle in front until current speed state or collide with
    opponent or (there is obstacle in front until boost speed state and speed =
    15) then
        if speed = 15 then
            if lizard is fit in boost state then LIZARD
            else turnObstacle with boost state // dodge
        else
            if lizard is fit in current speed then LIZARD
            else turnObstacle with current speed // dodge

    // Improvement logic
    if speed < 8 then
        if boost is fit then BOOST
        else if no obstacle in front until next speed state then ACCELERATE

    // Attack logic
    // Case when player's car is in front of the opponent
    if in front of opponent then
        if boost is fit then BOOST
```

```

    else if has oil and opponent is in two blocks behind and opponent is on our
sidelines then OIL
    else if has tweet then
        // opponent is more likely to prioritize ACCELERATE if speed <= 3
        if opponent speed <= 3 then TWEET at 4 blocks in front of the opponent
        else TWEET at a block right in front of the opponent
// Case when player's car is behind
if opponent in front then
    if emp is fit then EMP
    else if boost is fit then BOOST
    else if has tweet then
        // opponent is more likely to prioritize ACCELERATE if speed <= 3
        if opponent speed <= 3 then TWEET at 4 blocks in front of the opponent
        else TWEET at a block right in front of the opponent

// Waste oil to gain more points
if speed >= 9 and has oil then OIL

// Fix to perfect condition, preparation to use boost if
// player's position is behind the opponent
if has boost and damage >= and speed = 9 and opponent is in front then FIX

// Default state if nothing happened
if there is obstacle in front until next speed state then turnObstacle //dodge
else ACCELERATE

```

```

functions turnObstacle(Car: myCar, Array_of_block: currentBlocks, lane1Blocks,
lane2Blocks, lane3Blocks, lane4Blocks, accelerateBlocks) -> Command
    scoreAccelerate ← calculate accelerateBlocks score // call scoreLane
    scoreFront ← calculate currentBlocks score // call scoreLane
    score1 ← calculate lane1Blocks score // call scoreLane
    score2 ← calculate lane2Blocks score // call scoreLane
    score3 ← calculate lane3Blocks score // call scoreLane
    score4 ← calculate lane4Blocks score // call scoreLane

```

```

// scoreAccelerate will evaluate (current speed + accelerate) blocks points in
front of the car
// scoreFront will evaluate (current speed) blocks points in front of the car
// score1 will evaluate (current speed) blocks points in lane 1
// score2 will evaluate (current speed) blocks points in lane 2
// score3 will evaluate (current speed) blocks points in lane 3
// score4 will evaluate (current speed) blocks points in lane 4

if lane = 1 then
    if there is obstacle in lane 2 then
        if scoreFront < score2 then TURN_RIGHT
        else
            if scoreAccelerate >= scoreFront and accelerateBlocks do
not contain wall then ACCELERATE
            if speed >= 9 and has oil then OIL
            else NOTHING
        else TURN_RIGHT

if lane = 4 then
    if there is obstacle in lane 3 then
        if scoreFront < score3 then TURN_LEFT
        else
            if scoreAccelerate >= scoreFront and accelerateBlocks do
not contain wall then ACCELERATE
            if speed >= 9 and has oil then OIL
            else NOTHING
        else TURN_LEFT

if lane = 2 then
    if there is obstacle in lane 1 and 3 then
        if score1 higher than scoreFront and score3 then TURN_LEFT
        else if score3 higher than scoreFront and score1 then TURN_RIGHT
        else if scoreFront higher than score1 and score3 then
            if scoreAccelerate >= scoreFront and accelerateBlocks do
not contain wall then ACCELERATE
            if speed >= 9 and has oil then OIL
            else NOTHING

```



```

    else if there is no obstacle in lane 1 and 3 then
        if score1 >= score3 and score1 > scoreFront then TURN_LEFT
        else if score3 > score1 and score3 > scoreFront then TURN_RIGHT
    else if there is obstacle in lane 1 but not 3 then TURN_RIGHT
    else if there is obstacle in lane 3 but not 1 then TURN_LEFT
if lane = 3 then
    if there is obstacle in lane 2 and 4 then
        if score2 higher than scoreFront and score4 then TURN_LEFT
        else if score4 higher than scoreFront and score2 then TURN_RIGHT
        else if scoreFront higher than score2 and score4 then
            if scoreAccelerate >= scoreFront and accelerateBlocks do
                not contain wall then ACCELERATE
            if speed >= 9 and has oil then OIL
            else NOTHING
    else if there is no obstacle in lane 2 and 4 then
        if score2 >= score4 and score2 > scoreFront then TURN_LEFT
        else if score4 > score2 and score4 > scoreFront then TURN_RIGHT
    else if there is obstacle in lane 2 but not 4 then TURN_RIGHT
    else if there is obstacle in lane 4 but not 2 then TURN_LEFT
else NOTHING

```

```

functions isEMPNice(Car: myCar, Car: opponent, Array_of_block: blocks) -> boolean
    if opponentSpeed >= 5 and side by side and has EMP and distance between not
    too small then
        → true
    else
        → false

```

```

functions isBoostNice(Car: myCar, Array_of_block: blocks) -> boolean
    if has Boost and no obstacle ahead and has boost and myCar not damaged and
    not boosting then
        → true
    else
        → false

```

```

functions isLizardNice(Car: myCar, Array_of_block: blocks) -> boolean
    if has Lizard and near finish block then
        → true
    elif has Lizard and scoreLane < -2 and no obstacle on arrival block then
        → true
    else
        → false

```

```

functions isObstacle(Car: myCar, Array_of_block: blocks) -> Command
    if blocks contains mud or wall or oil_spill then
        → true
    else
        → false
    // any isObstacle function work like this the different only on the range of
    // blocks checked

```

```

functions scoreLane(Car: myCar, Array_of_block: blocks, int: mode) -> int
    score ← 0
    iterate blocks[0..blocks.size()]
        if blocks contains mud then
            score ← score - 2
        if blocks contains oilSpill then
            score ← score - 1
        if blocks contains wall then
            score ← score - 5
        if blocks contains boost then
            score ← score + 2
        if blocks contains tweet then
            score ← score + 1
        if blocks contains emp then
            if opponent is in front then
                score ← score + 2
    → score

```

```
// blocks scoring is based on priority in which:
// 1. avoiding wall
// 2. avoiding other obstacles
// 3. gaining boost
// 4. gaining other power ups
```

Struktur Data

Struktur berorientasi objek banyak digunakan pada program ini.

a. Modul Command

Nama	Deskripsi
AccelerateCommand	Kelas yang menghasilkan objek perintah yang dapat menaikkan <i>state</i> kecepatan bot.
BoosCommand	Kelas yang menghasilkan objek perintah yang dapat menaikkan <i>state</i> kecepatan bot menjadi 15 sebanyak 5 ronde, yakni di atas kecepatan yang bisa dicapai oleh perintah <i>accelerate</i> .
ChangeLaneCommand	Kelas yang menghasilkan objek perintah yang dapat menggeser bot ke <i>lane</i> kanan atau kiri.
DecelerateCommand	Kelas yang menghasilkan objek perintah yang dapat menurunkan <i>state</i> kecepatan bot.
DoNothingCommand	Kelas yang menghasilkan objek perintah yang secara praktikal tidak melakukan perubahan apa-apa pada bot.
EmpCommand	Kelas yang menghasilkan objek perintah yang dapat menembakkan <i>electromagnetic pulse</i> di <i>lane</i> bot dan

	lane yang beradjajensi dengannya ke arah depan tanpa batas.
FixCommand	Kelas yang menghasilkan objek perintah yang memperbaiki bot dengan mengurangi <i>state damage</i> sebanyak 2.
LizardCommand	Kelas yang menghasilkan objek perintah yang menyebabkan bot terbang dan tidak terdampak rintangan di depannya selama satu ronde.
OilCommand	Kelas yang menghasilkan objek perintah yang dapat menaruh <i>oil spill</i> pada lajur.
TweetCommand	Kelas yang menghasilkan objek perintah yang dapat menaruh <i>cyber truck</i> karya Elon Musk dengan parameter posisi <i>block</i> dan lajur.

b. Modul Entities

Nama	Deskripsi
Car	Representasi mobil yang diperbalapkan. Terdapat atribut <i>id</i> , <i>position</i> , <i>speed</i> , <i>damage</i> , <i>state</i> dari mobil, <i>powerups</i> , <i>boosting</i> , serta <i>boostCounter</i> .
GameState	Menyimpan <i>state</i> dari ronde game yang sedang berlangsung. Berisi <i>currentRound</i> , <i>maxRounds</i> , <i>player</i> , <i>opponent</i> , serta <i>lanes</i> .
Lane	Menyimpan atribut <i>position</i> , <i>terrain</i> , serta <i>occupiedByPlayerId</i> .
Position	Menyimpan atribut <i>lane</i> , dan <i>block</i> .

c. Modul Enums

Modul Enums mengenumerasi perintah, arah, *state*, serta *power ups*. Enumerasi diperlukan untuk pencatatan daripada Direction, PowerUps, State, dan Terrain.

d. Kelas Bot

Kelas ini adalah kelas yang berisi logika-logika utama yang digunakan bot dalam permainan Overdrive. *Method* pada kelas ini turut menggunakan variabel serta objek yang terdefinisi pada kelas lain.

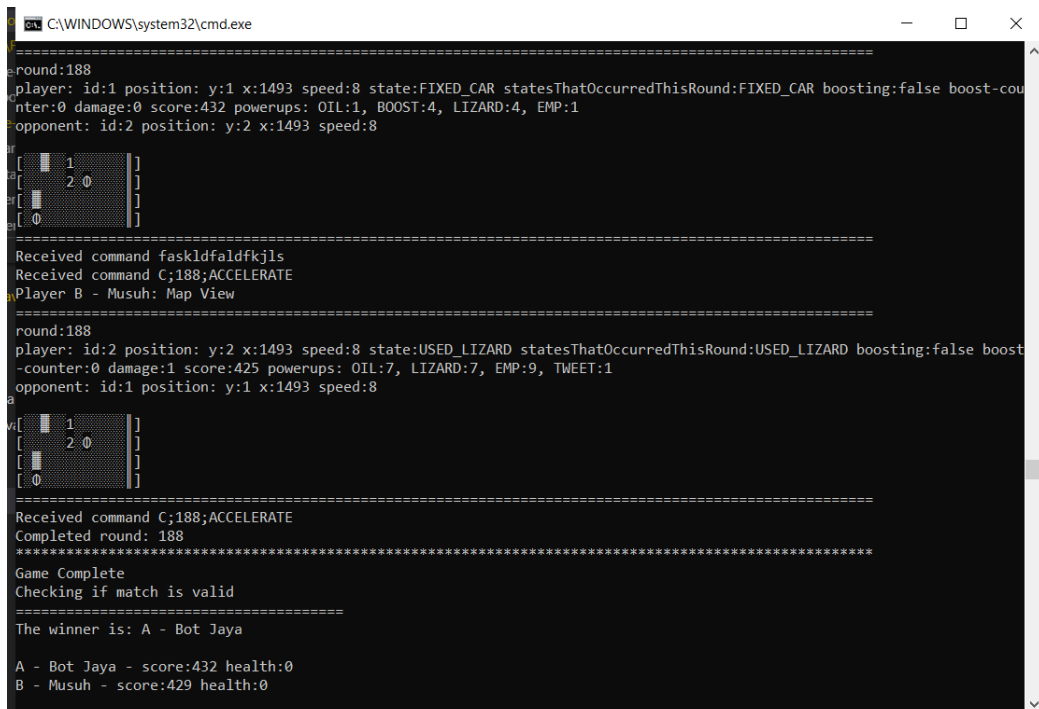
e. Kelas Main

Kelas ini adalah kelas yang akan mengkonstruksi objek bot dan menjalankannya bergantung pada *game state* yang ada. *Command* dikirimkan kepada *game engine* oleh kelas ini dengan melakukan *print* pada *console* sesuai format *command*.

Analisis pada Pengujian

Pengujian dilakukan dengan mempertandingkan bot kami dengan bot kelompok lain. Dengan mengatur konfigurasi sesuai aturan dan menjalankan 'run.bat', pertandingan berjalan dan kami gugup berharap hasil kemenangan.

a. Pengujian I



```
C:\WINDOWS\system32\cmd.exe
=====
round:188
player: id:1 position: y:1 x:1493 speed:8 state:FIXED_CAR statesThatOccurredThisRound:FIXED_CAR boosting:false boost-counter:0 damage:0 score:432 powerups: OIL:1, BOOST:4, LIZARD:4, EMP:1
opponent: id:2 position: y:2 x:1493 speed:8

[ 1 ]
[ 2 0 ]
[  ]
[  ]

=====
Received command faskldfalkjls
Received command C:188;ACCELERATE
Player B - Musuh: Map View
=====
round:188
player: id:2 position: y:2 x:1493 speed:8 state:USED_LIZARD statesThatOccurredThisRound:USED_LIZARD boosting:false boost-counter:0 damage:1 score:425 powerups: OIL:7, LIZARD:7, EMP:9, TWEET:1
opponent: id:1 position: y:1 x:1493 speed:8

[ 1 ]
[ 2 0 ]
[  ]
[  ]

=====
Received command C:188;ACCELERATE
Completed round: 188
=====
Game Complete
Checking if match is valid
=====
The winner is: A - Bot Jaya

A - Bot Jaya - score:432 health:0
B - Musuh - score:429 health:0
```

Di pengujian pertama ini, kami melawan bot kelompok lain yang cukup kuat. Kami menang tipis dengan skor lebih banyak 10. Dari sini, dapat dilihat bahwa untuk bisa menang, tidak cukup dengan mempercepat diri, tetapi juga mencari skor-skor disekitar. Terdapat banyak sekali perebutan posisi pertama, strategi yang paling ampuh untuk merebut posisi pertama bagi bot yang tertinggal adalah dengan menembakkan *EMP*. Selain memberhentikan bot lawan, penggunaan *EMP* juga dapat menambahkan skor yang dapat menjadi salah satu evaluasi penentuan pemenang seperti pada kasus ini di mana kedua bot sampai *block finish* secara bersamaan dengan besar *speed* yang juga sama.

b. Pengujian II

```
C:\WINDOWS\system32\cmd.exe
Starting round: 179
Player A - Bot Jaya: Map View
=====
round:179
player: id:1 position: y:3 x:1497 speed:6 state:HIT_MUD statesThatOccurredThisRound:NOTHING, HIT_MUD bo
osting:false boost-counter:0 damage:1 score:297 powerups: BOOST:4, LIZARD:1, EMP:16, TWEET:2
opponent: id:2 position: y:1 x:1432 speed:9

[
  [
    [
      1
    ]
  ]
]
=====
Received command C;179;ACCELERATE
Player B - Musuh: Map View
=====
round:179
player: id:2 position: y:1 x:1432 speed:9 state:NOTHING statesThatOccurredThisRound:NOTHING boosting:fa
lse boost-counter:0 damage:0 score:416 powerups: OIL:3, LIZARD:2, EMP:1, TWEET:7
opponent: id:1 position: y:3 x:1497 speed:6

[
  [
    [
      2
    ]
  ]
]
=====
Received command C;179;NOTHING
Completed round: 179
*****
Game Complete
Checking if match is valid
=====
The winner is: A - Bot Jaya

A - Bot Jaya - score:297 health:0
B - Musuh - score:416 health:0
=====
```

Pada pengujian kedua ini, kami memenangkan pertandingan dengan selisih jarak kurang lebih 70 *blocks* di depan lawan. Namun, yang perlu menjadi fokus perhatian adalah selisih skor yang terbilang jauh. Hal ini dapat dilihat dari sisa *power ups* yang tersisa, baik yang bot kami miliki maupun yang bot lawan miliki. Di akhir ronde, bot lawan memiliki *power ups* yang lebih sedikit dibandingkan dengan *power ups* yang kami miliki. Posisi bot kami yang seringkali berada di depan lawan menjadi salah satu alasan dari tidak terpakainya *power ups* yang kami miliki. Sesuai dengan prioritas yang kami tetapkan, *speed* menjadi prioritas utama. Oleh karena itu, ketika berada di depan lawan,

Sehingga, dapat dikatakan bahwa *speed* tidak hanya menjadi titik fokus utamanya, terlebih ketika bot yang kami miliki bersaing sengit dengan bot lawan, karena adanya kemungkinan untuk sampai *block finish* secara bersamaan. Namun, pada kasus ini, jika jarak dengan bot lawan terbilang jauh dan sulit untuk disaingi, maka bukan suatu masalah jika skor tidaklah menjadi prioritasnya.

c. Pengujian III

```
C:\WINDOWS\system32\cmd.exe
```

```
Player A - Bot Jaya: Map View  
=====  
round:176  
player: id:1 position: y:2 x:1468 speed:6 state:ACCELERATING statesThatOccurredThisRound:ACCELERATING boosting:false boost-counter:0 damage:0 score:354 powerups: OIL:3, BOOST:2, EMP:4, TWEET:2  
opponent: id:2 position: y:2 x:1496 speed:15  
  
[ * ]  
[ # [ B ] ]  
[ | ]  
[ ]  
=====  
Received command C;176;TURN_LEFT  
Player B - Musuh: Map View  
=====  
round:176  
player: id:2 position: y:2 x:1496 speed:15 state:TURNING_RIGHT statesThatOccurredThisRound:TURNING_RIGHT boosting:true boost-counter:4 damage:0 score:373 powerups: LIZARD:1, TWEET:1, EMP:3, OIL:1, BOOST:1  
opponent: id:1 position: y:2 x:1468 speed:6  
  
[ | ]  
[ 2 ] ]  
[ * ] ]  
[ T * ] ]  
=====  
Received command C;176;USE_BOOST  
Completed round: 176  
*****  
Game Complete  
Checking if match is valid  
=====  
The winner is: B - Musuh  
  
A - Bot Jaya - score:354 health:0  
B - Musuh - score:377 health:0  
  
=====
```

Pada pengujian kasus ketiga, kami mendapati kami kalah melawan bot lawan. Setelah dilakukan pemeriksaan pada setiap ronde yang dilewati, kami mendapat bot kami jauh unggul pada awal hingga pertengahan pertandingan. Namun, unggul pada awal

pertandingan tidak selalu berarti hal yang baik. Hal ini dikarenakan, bot lawan cenderung akan menggunakan *power up* untuk mencelakai kami, dalam hal ini penggunaan *power up EMP*. *Power up EMP* yang dapat menghentikan gerak mobil selama satu ronde serta mengakibatkan *speed* berkurang drastis menjadi 3 sangatlah merugikan kami. Seringkali ditemukan kasus di mana bot kami baru saja menjalankan *power up boost* namun tiba-tiba harus berhenti dikarenakan adanya penggunaan *EMP* dari lawan. Pertengahan hingga akhir pertandingan, bot kami berhasil disusul oleh lawan dibuktikan dengan seringkalinya kedua mobil berpapasan hingga akhirnya bot kami benar-benar disusul pada enam ronde terakhir. Strategi untuk menghindarkan diri dari *obstacle* menjadi peranan yang sangat penting pada kasus ini. Namun, *power up EMP* menjadi salah satu hal yang susah untuk dihindari. Posisi bot kami yang berada di depan lawan menjadikan kami tidak dapat membalas *EMP* yang diberikan dan hanya bisa fokus pada hal lain seperti *speed* atau skor.

BAB V

KESIMPULAN DAN SARAN

Kesimpulan

Dengan menggunakan Algoritma *Greedy*, strategi algoritma yang telah kami buat untuk mencoba memenangkan permainan Overdrive ini adalah pendekatan campuran yang terdiri dari pendekatan prioritas kecepatan, pendekatan prioritas poin, dan pendekatan prioritas *boost*. Setelah dilakukan uji coba serta studi kasus, dapat disimpulkan bahwa algoritma kami dapat berjuang untuk memenangkan permainan sebagaimana yang diharapkan.

Saran

Setelah menyelesaikan pengerjaan tugas besar ini, beberapa saran yang dapat kami berikan adalah lebih mendalami Algoritma *Greedy* sampai benar-benar paham dan juga lebih membaca peraturan permainan dengan lebih seksama. Pemahaman tentang Algoritma *Greedy* yang mendalam akan mempercepat pengerjaan tugas besar pertama strategi algoritma ini. Begitu juga dengan membaca peraturan permainan lebih teliti, tidak akan ada hal yang tertinggal di tengah-tengah perancangan algoritma hingga harus membuat ulang algoritmanya dari awal.

TAUTAN REPOSITORY GITHUB

Berikut tautan untuk mengakses *source code* yang kami simpan pada github:

https://github.com/mhelimih/Tubes1_BotJaya99

DAFTAR PUSTAKA

Entelect Forum. Entelect Overdrive 2020 Challenge. Diakses pada 12 Februari 2022 pukul 18.21 WIB melalui <https://forum.entelect.co.za/>

Munir, R. (2020). Algoritma Greedy Bagian 1[PDF]. Institut Teknologi Bandung. Diakses pada 16 Februari 2022 pukul 11.21 WIB melalui [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf)

Munir, R. (2020). Algoritma Greedy Bagian 2[PDF]. Institut Teknologi Bandung. Diakses pada 16 Februari 2022 pukul 11.33 WIB melalui [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/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf)

Munir, R. (2020). Algoritma Greedy Bagian 3[PDF]. Institut Teknologi Bandung. Diakses pada 16 Februari 2022 pukul 11.47 WIB melalui [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-Bag3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag3.pdf)

asyique