Laporan Tugas Besar 1 - IF2211

Pemanfaatan Algoritma Greedy dalam Permainan “Overdrive”

Disusun Oleh: Kelompok Solo Balapan

Fikri Khoiron Fadhila	- 13520056
Malik Akbar Hashemi Rafsanjani	- 13520105
Hafidz Nur Rahman Ghozali	- 13520117

Semester II Tahun 2021/2022

Institut Teknologi Bandung

Bab 1

Deskripsi Tugas

Overdrive adalah sebuah *game* yang mempertandingkan 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

Tugas besar kali ini bertujuan untuk membuat sebuah bot terbaik untuk bermain permainan Overdrive dengan menggunakan algoritma *greedy*. Strategi *greedy* yang diimplementasikan harus dikaitkan dengan fungsi objektif dari permainan itu sendiri, yaitu memenangkan pertandingan dengan cara mengalahkan bot lawan dalam balapan melalui *command-command* yang disediakan dalam permainan. Implementasi pemain harus dapat dijalankan pada *game engine* yang disebutkan pada spesifikasi tugas besar, serta dapat dikompetisikan dengan pemain dari kelompok lain.

Bab 2

Landasan Teori

A. Algoritma Greedy

Algorimta Greedy adalah algoritma yang membentuk solusi langkah per langkah dengan mencari solusi terbaik lokal pada setiap langkahnya. Pada kebanyakan kasus, algoritma *greedy* tidak menghasilkan solusi yang paling optimal, namun menemukan solusi yang mendekati optimum dnegan waktu yang cukup cepat. Algoritma *Greedy* memiliki prinsip “take what you can get now” artinya adalah algoritma akan mengambil solusi yang optimal pada langkah tersebut tanpa mempertimbangkan konsekuensinya pada langkah berikutnya.

Elemen-elemen algoritma *greedy:*

1. Himpunan kandidat, C: berisi kandidat yang akan dipilih pada setiap langkah (misal : simpul sisi di dalam graf , job, task, koin , benda , karakter , dsb).
2. Himpunan solusi, S : berisi kandidat yang sudah dipilih.
3. Fungsi solusi: menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi.
4. Fungsi seleksi (selection function): memilih kandidat berdasarkan strategi greedy tertentu . Strategi greedy ini bersifat heuristik.
5. Fungsi kelayakan (feasible): memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak).
6. Fungsi obyektif : memaksimumkan atau meminimumkan

Dengan menggunakan elemen-elemen di atas, maka 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 dioptimasi oleh fungsi obyektif.

B. Permainan Overdrive

Dalam permainan *overdrive* terdapat 2 pemain yang akan melakukan pertandingan balap dengan *command-command* telah disediakan. *Command* yang tersedia berupa perlakuan pada bot mobil. Pemain yang mencapai garis *finish* terlebih dahulu adalah pemenangnya, atau apabila *finish* bersamaan, maka pemenangnya adalah pemain yang memiliki skor lebih tinggi.

Permainan ini dimainkan di peta dengan 4 jalur yang memiliki panjang 1500 blok. Peta tersebut diimplementasikan dalam bentuk matriks 2 dimensi. Ada beberapa jenis tipe sel permukaan pada jalur.

Berikut adalah tipe sel permukaan pada jalur yang ada dalam *game:*

- *Empty* - blok sel yang tidak memiliki efek kepada mobil.
- *Mud* - blok sel yang memiliki lumpur, akan berefek pada perlambatan kecepatan mobil.
- *Oil Spill* - blok sel yang mengandung minyak, blok ini akan memperlambat kecepatan mobil.
- *Flimsy wall* - blok sel ini akan mengandung mobil, apabila mobil menabraknya akan menyebabkan kerusakan pada mobil.
- *Finish Line* - blok garis *finish* yang menandakan mobil telah mencapai akhir dari peta.

Selain tipe sel permukaan, setiap blok juga dapat berisi *powerup*. *Powerup* diambil ketika mobil bergerak ke blok itu.

- *Oil item* - blok memiliki drum minyak di atasnya untuk dikumpulkan dan memungkinkan pemain untukmembuat tumpahan minyak.
- *Boost* - blok memiliki *powerup boost* apabila digunakan akan mempercepat mobil.
- *Lizard* - blok memiliki *powerup lizard* yang akan membuat mobil bisa melompat menghindari rintangan pada jalur.
- *Tweet* - blok memiliki *powerup tweet* Elon Musk yang akan memarkir *Cybertruck* di lokasi yang diinginkan.
- *EMP* - blok berisi *EMP* yang akan menembakkan gelombang *EMP* ke depan

Bot mobil akan bergerak di jalur yang dimulai dengan kecepatan 5. Hal ini berarti akan membuat mobil bergerak maju lima blok di setiap putaran. *Command* yang tersedia untuk pemain akan memungkinkan pemain untuk mengubah jalur, mempercepat, serta memperlambat, dan menggunakan *powerup.*

Powerup dapat dikumpulkan oleh mobil dengan melewati blok yang memiliki *powerup* didalamnya. Setelah *powerup* dikumpulkan, *command* yang sesuai dapat digunakan untuk mengaktifkan *powerup* tersebut. Jika pemain mencoba menggunakan *powerup* yang tidak dimiliki, maka bot mobil akan melakukan *command* DO_NOTHING dan pemain akan dikenakan skor penalti.

Berikut beberapa status yang diakibatkan *Command* yang ada dalam permainan:

- READY: menyatakan semua pemain memulai di awal balapan.
- NOTHING: status pemain setelah menjalankan perintah DO_NOTHING.
- TURNING_LEFT: status pemain masuk setelah berhasil menjalankan *command* TURN_LEFT
- TURNING_RIGHT: status pemain masuk setelah berhasil menjalankan *command* TURN_RIGHT
- ACCELERATING: status pemain masuk setelah menjalankan perintah ACCELERATING
- DECELERATING: status pemain masuk setelah menjalankan perintah DECELERATING
- PICKED_UP_POWERUP: status pemain masuk setelah mengambil power up
- USED_BOOST: status pemain masuk setelah berhasil menggunakan boost.
- USED_OIL: status pemain setelah berhasil menggunakan power up oil.
- USED_LIZARD: status pemain masuk setelah berhasil menggunakan power up lizard.
- USED_TWEET: status pemain masuk setelah berhasil menggunakan power up tweet.
- USED_EMP: status pemain masuk setelah berhasil menggunakan power up EMP
- HIT_MUD: status pemain setelah menabrak blok lumpur.
- HIT_OIL: status pemain setelah menabrak blok minyak.
- HIT_WALL: status pemain setelah menabrak dinding.
- HIT_CYBER_TRUCK: status pemain setelah menabrak Cybertruck.
- HIT_EMP: status pemain setelah terkena EMP.
- FINISH: status pemain ketika mencapai garis finish.

Pada setiap round, pemain dapat mengirimkan satu command untuk bot mobil mereka. Semua command pemain divalidasi sebelum dijalankan. Command yang tidak valid mengakibatkan mobil tidak melakukan apa-apa untuk putaran tersebut. Command yang tidak valid akan mengurangi skor. Terlalu banyak command yang tidak valid akan membatalkan balapan. Selain itu, command dari kedua pemain akan dijalankan secara bersamaan dalam satu round.

Penskalaan status kecepatan pada permainan ini dinyatakan sepeti di bawah ini

- MINIMUM_SPEED = 0
- SPEED_STATE_1 = 3
- INITIAL_SPEED = 5
- SPEED_STATE_2 = 6
- SPEED_STATE_3 = 8
- MAXIMUM_SPEED = 9
- BOOST_SPEED = 15

Sedangkan semua perintah pada permainan ini dinyatakan sebagai berikut

- NOTHING : mobil tidak akan melakukan perubahan apapun pada round tersebut.
- ACCELERATE : mobil akan meningkatkan kecepatan mobil ke status kecepatan berikutnya.
- DECELERATE : mobil akan menurunkan kecepatan mobil ke status kecepatan sebelumnya.
- TURN_LEFT : mobil akan mengubah jalur mobil ke jalur berikutnya di sebelah kiri.
- TURN_RIGHT : mobil akan mengubah jalur mobil ke jalur berikutnya di sebelah kanan.
- USE_BOOST : mobil akan menggunakan powerup boost, boost akan meningkatkan kecepatan mobil menjadi BOOST_SPEED selama 5 round. Semua perlambatan akan menggagalkan boost dan mengembalikan kecepatan ke MAXIMUM_SPEED.
- USE_OIL : mobil akan menempatkan blok oil tepat di bawah mobil pemain. Setiap mobil yang melewatinya (kecuali pemain) akan mengalami penurunan kecepatan ke kecepatan sebelumnya.
- USE_LIZARD : mobil akan melompat menghindari semua *obstacle* termasuk powerup.
- USE_TWEET : command ini akan memunculkan cybertruck di jalur dan blok yang diinginkan, dika pemain bertabrakan dengan cybertruc, maka kecepatan dikurangi menjadi 3 dan akan terjebak sampai round berakhir.
- USE_EMP : mobil akan menembakkan EMP ke depan, jalur kanan, dan jalur kiri dari mobil pemain. EMP akan menghentikan musuh di jalurnya selama round tersebut dan mengurangi kecepatannya menjadi 3
- FIX : command ini akan memperbaiki kerusakan mobil sebanyak 2 poin. Saat menggunakan command ini, mobil pemain akan tetap diam selama round tersebut.

Permainan ini memiliki beberapa *obstacle* yang akan mempengaruhi status dari mobil seperti berikut:

- MUD: jika pemain melewati MUD, kecepatan mobil akan berkurang ke level yang lebih rendah. Jika menggunakan boost, maka boost akan berhenti. Dengan status kecepatan yang baru mengacu pada skala seperti berikut:
  - SPEED_STATE_1 => SPEED_STATE_1
  - INITIAL_SPEED => SPEED_STATE_1
  - SPEED_STATE_2 => SPEED_STATE_1
  - SPEED_STATE_3 => SPEED_STATE_2
  - MAXIMUM_SPEED => SPEED_STATE_3
  - BOOST_SPEED => MAXIMUM_SPEED
- WALL: jika pemain bertabrakan dengan dinding, kecepatannya akan berkurang menjadi 3 terlepas dari kecepatan saat ini. Jika mobil menggunakan boost, maka boost akan berakhir.

Kerusakan akibat dari tabrakan akan mengurangi kecepatan maksimum yang dapat dicapai. Jika menerima terlalu banyak kerusakan maka mobil tidak dapat bergerak. Jumlah kerusakan yang didapat setiap bertabrakan dengan objek adalah sebagai berikut:

- Lumpur: 1
- Minyak: 1
- Dinding: 2
- Cybertruck: 2

Kerusakan tersebut akan mempengaruhi kecepatan maksimal dari mobil pemain dengan aturan sebagai berikut:

- Kerusakan 5 sama dengan kecepatan maksimal 0
- Kerusakan 4 sama dengan kecepatan maksimal 3
- Kerusakan 3 sama dengan kecepatan maksimal 6
- Kerusakan 2 sama dengan kecepatan maksimal 8
- Kerusakan 1 sama dengan kecepatan maksimal 9
- Kerusakan 0 sama dengan kecepatan maksimal 15

Akhir dari permainan ini adalah mobil mencapai ujung dari jalur yang ditentukan oleh blok yang berisi objek FINISH_LINE. Jika kedua bot mencapai garis finish bersamaan, maka bot yang melewati garis finish dengan kecepatan tertinggi akan menang. Apabila finish bersamaan dengan kecepatan yang sama, maka bot dengan skor tertinggi akan menang.

Skorsing pada permainan ini dimulai dengan skor 0. Beberapa tindakan yang terjadi dalam permainan akan mempengaruhi skor Pemain seperti di bawah ini:

- Menabrak MUD akan mengurangi skor sebesar 3
- Menabrak OIL akan mengurangi skor sebesar 4
- Mengambil powerup akan menambah skor sebesar 4
- Menggunakan powerup akan menambah skor sebesar 4
- Command yang tidak valid akan mengurangi skor pemain sebesar 5. Perintah yang tidak valid didefinisikan sebagai berikut:
    - Belok kiri saat berada di jalur 1
    - Belok kanan saat berada di jalur 4
    - Menggunakan power up yang tidak dimiliki
    - Mengeluarkan command tidak sesuai sintaks.
    - Bot gagal mengeluarkan perintah.

Bab 3

Pemanfaatan Algoritma Greedy

A. Elemen - Elemen Algoritma Greedy Pada Permainan Overdrive

Dalam permainan Overdrive, setiap pemain akan memiliki sebuah mobil yang berlomba dengan tujuan untuk mencapai garis finish terlebih dahulu. Namun, di dalam lintasan, terdapat beberapa rintangan maupun *powerups*. Ketika pemain melewati blok lintasan yang mengandung rintangan, maka pemain akan mendapatkan *damage* yang sesuai dengan rintangan yang dilewati. Ketika pemain melewati blok lintasan yang mengandung *powerups*, pemain akan mendapatkan *powerups* tersebut yang dapat digunakan sesuai dengan *powerups* tersebut pada ronde-ronde berikutnya.

Pada setiap ronde, pemain akan memberikan satu perintah kepada mobilnya. Semua perintah akan divalidasi oleh game. Untuk memenangkan permainan, maka pemain perlu untuk mencapai garis finish terlebih dahulu. Jika kedua pemain mencapai garis finish bersamaan, maka pemenang ialah pemain dengan kecepatan tercepat, jika tetap sama maka pemenang adalah yang memiliki score tertinggi. Di samping itu, terdapat beberapa batasan, seperti kecepatan maksimum ditentukan oleh *damage* yang diterima mobil, serta *damage* dapat dikurangi 2 unit dengan menggunakan command FIX, tetapi mobil harus berhenti selama 1 ronde. Dari sini dapat kita simpulkan bahwa elemen greedy permainan Overdrive sebagai berikut.

1. Himpunan Kandidat:

   Himpunan perintah yang dapat dilakukan, meliputi perintah: DO_NOTHING, ACCELERATE, DECELERATE, TURN_LEFT, TURN_RIGHT, USE_BOOST, USE_OIL, USE_TWEET {block} {lane}, USE_LIZARD, USE_EMP, FIX.

2. Himpunan Solusi:

   Himpunan perintah yang terpilih dan akan dilakukan.

3. Fungsi Solusi:

   Memeriksa perintah yang terpilih dapat dijalankan pada ronde tersebut sesuai dengan state game saat itu.

4. Fungsi Seleksi:

Memilih perintah terbaik berdasarkan *damage*, rintangan, *powerups* yang dimiliki, *powerups* yang dapat diambil, kecepatan, posisi diri sendiri, serta posisi lawan.

5. Fungsi Kelayakan:

   Memeriksa apakah perintah yang akan dipilih dapat dilakukan, seperti apakah memiliki *powerups* yang akan dipilih.

6. Fungsi Objektif:

   Mencapai garis finish secepat mungkin.

B. Solusi greedy yang dapat diimplementasikan

Terdapat banyak aspek yang dapat dipertimbangkan pada strategi greedy yang bisa dibuat. Pada bagian ini, akan dijelaskan strategi greedy berdasarkan aspek tertentu serta fungsi seleksinya.

1. Greedy by damage

   Pada strategi ini, dalam memilih perintah terbaik, akan dipertimbangkan rintangan apa saja yang ada di jalur depan baik ketika mempercepat atau dengan kecepatan yang sama, kiri, dan kanan. Dari *damage* yang dihasilkan oleh rintangan-rintangan tersebut, akan dipilih jalur yang memiliki damage paling kecil.

   Algoritma ini akan mengecek semua blok yang ada di jalur kiri, kanan, dan depan sesuai dengan kecepatan saat itu. Efisiensi dari algoritma ini adalah O(n) dengan n merupakan kecepatan saat itu karena perlu mengiterasi setiap blok sebanyak sekitar kecepatan saat itu.

   Efektivitas dari algoritma ini cukup baik karena *damage* yang diterima oleh pemain akan sangat berakibat pada performansi kecepatan karena semakin besar *damage* yang dimiliki pemain, akan semakin kecil kecepatan maksimum yang dapat dicapai. Di samping itu, untuk memperbaiki *damage*, perlu untuk menggunakan perintah FIX, yang mengharuskan pemain untuk berhenti satu ronde dan hanya mengurangi *damage* sebesar 2 unit.

2. Greedy by score

   Pada strategi ini, dalam memilih perintah terbaik, akan dipertimbangkan perintah dengan score terbanyak pada ronde tersebut. Score akan berkurang ketika

pemain melewati rintangan mud dan oil, serta score akan bertambah jika pemain mengambil powerups dan menggunakannya.

Algoritma ini akan mengecek semua blok yang ada di jalur kiri, kanan, dan depan sesuai dengan kecepatan saat itu. Efisiensi dari algoritma ini adalah O(n) dengan n merupakan kecepatan saat itu.

Efektivitas dari algoritma ini kurang baik karena pengaruh score tidaklah besar terhadap kemenangan. Score merupakan prioritas ketiga dalam penentuan kemenangan, setelah pemain mana yang paling cepat mencapai garis finish dan kecepatan tertinggi pada ronde terakhir ketika sama sama cepat mencapai garis finish.

3. Greedy by powerups

   Pada strategi ini, dalam memilih perintah terbaik, akan dipertimbangan perintah dengan cara memprioritaskan menggunakan *powerups* yang dimiliki. Powerups yang terambil akan segera digunakan. Strategi ini dapat diimplementasikan dengan memprioritaskan *powerups* tertentu terlebih dahulu yang dimiliki atau menggunakan *powerups* pertama yang ada di senarai powerups yang dimiliki.

   Kompleksitas algoritma ini akan bergantung pada strategi mana yang diimplementasikan. Jika menggunakan prioritas powerups tertentu, maka perlu melakukan pencarian pada senarai powerups yang dimiliki, dengan kompleksitas O(n), dengan n merupakan banyaknya *powerups* yang dimiliki. Jika menggunakan *powerups* pertama yang ada di senarai, maka kompleksitasnya O(1) karena kita hanya perlu mengecek elemen pertama atau terakhir dari senarai *powerups*.

   Efektivitas dari algoritma ini kurang baik karena pemakaian *powerups* tidak langsung berpengaruh pada kemenangan, tetapi hanya dapat membantu mobil kita mencapai garis finish atau menghalangi mobil musuh mencapai garis finish. Di samping itu, pemakaian *powerups* harus disesuaikan dengan keadaan game saat itu.

4. Greedy by speed

   Pada strategi ini, dalam memilih perintah terbaik, akan dikalkulasi perintah yang akan memberikan speed tertinggi pada ronde tersebut. Speed tertinggi dapat dicapai dengan meningkatkan speed menggunakan command tertentu, seperti ACCELERATE atau BOOST.

Algoritma ini akan mengecek apakah memiliki BOOST atau tidak, jika iya maka digunakan, jika tidak, maka gunakan ACCELERATE. Oleh karena itu, kompleksitas algoritma ini O(n) dengan n adalah banyaknya *powerups* yang dimiliki. Hal ini dikarenakan kita perlu mencari apakah BOOST ada dalam senarai *powerups* yang dimiliki.

Efektivitas dari algoritma ini kurang baik jika tidak memperhitungkan batasan-batasan lain, seperti kecepatan maksimum. Kecepatan maksimum dipengaruhi oleh *damage* yang diterima. Namun, jika ditambah dengan perhitungan batasan lain, maka efektivitas algoritma ini cukup bagus karena semakin besar kecepatan mobil kita, semakin cepat mobil kita mencapai garis finish, yaitu hal yang paling menentukan kemenangan.

C. Solusi greedy terbaik

Dari seluruh solusi greedy yang ada, kelompok kami memilih menggunakan algoritma greedy by damage dan greedy by speed. Pada strategi ini, mobil kita akan mencoba sebisa mungkin untuk mengambil perintah yang tidak memberikan *damage* pada mobil kita. Setelah mempertimbangkan jalur yang tidak memberikan *damage*, kita dapat mempertimbangkan untuk semakin memperbesar kecepatan atau setidaknya mempertahankan kecepatan yang ada. Hal tersebut dapat dicapai dengan menggunakan command BOOST jika tidak ada damage yang diterima dengan kecepatan BOOST, atau menggunakan command ACCELERATE jika tidak ada damage yang diterima dengan tingkat kecepatan selanjutnya. Jika tidak memenuhi, maka pertimbangkan untuk menggunakan *powerups* lain. Jika semua jalur memberikan damage maka pertimbangkan menggunakan BOOST jika *damage* masih bisa ditolerir, jika tidak maka ambil jalur yang memberikan *damage* paling kecil.

Pemilihan strategi ini dikarenakan damage akan membatasi kecepatan maksimum dengan semakin besar *damage*, maka semakin kecil kecepatan maksimum, serta kita perlu mengorbankan 1 ronde untuk mengurangi *damage* sebesar 2 unit dengan command FIX tanpa berpindah tempat. Selain itu, untuk memenangkan pertandingan, diperlukan kecepatan yang tinggi sehingga dapat meminimalkan jumlah ronde yang diperlukan untuk mencapai garis finish terlebih dahulu.

Algoritma yang digunakan dalam mengimplementasikan strategi greedy yang dipilih ialah sebagai berikut.

1. run

Mengembalikan Command yang paling optimal dengan algoritma Greedy berdasarkan GameState saat ini.

2. hasPowerUp

   Fungsi yang mengecek kepemilikikan sebuah powerUp. Mengembalikan trus jika punya, false jika tidak.

3. laneOneDecision

   Fungsi yang menganalisis kemungkinan solusi ketika mobil berada di lane 1. Fungsi ini menerima parameter berupa hashtable yang berisi obstacle dan powerup di jalur depan dan kanannya kemudian mengembalikan string decision yang paling optimal.

4. middleDecision

   Fungsi yang menganalisis kemungkinan solusi ketika mobil berada di lane 2 atau 3. Fungsi ini menerima parameter berupa hashtable yang berisi obstacle dan powerup di jalur depan, kiri, dan kanan serta lane mobil saat ini. kemudian mengembalikan string decision yang paling optimal.

5. laneFourDecision

   Fungsi yang menganalisis kemungkinan solusi ketika mobil berada di lane 4. Fungsi ini menerima parameter berupa hashtable yang berisi obstacle dan powerup di jalur depan dan kirinya kemudian mengembalikan string decision yang paling optimal.

Bab 4

Implementasi Dan Pengujian

Implementasi program yang kami buat dapat dilihat pada laman berikut ini
https://github.com/hafidznrq/Stima-Entelect-Overdrive-2020

A. Pseudocode

```
function run(gameState : GameState) → himpunan_solusi
{ Mengembalikan Command yang paling optimal dengan algoritma Greedy berdasarkan
GameState saat ini }
```

KAMUS

```
constant BOOST_SPEED : int = 15
myCar : Car <id : integer, position : <lane : integer, block : integer> speed
   : integer, damage : integer, state : State, powerups : list of PowerUps,
   boosting : boolean, boostCounter : integer>
opponent : Car <id : integer, position : <lane : integer, block : integer>
   speed : integer, damage : integer, state : State, powerups : list of
   PowerUps, boosting : boolean, boostCounter : integer>
maxSpeed : int
decision : string
obstacleLeft : Hashtable of <string,integer>
obstacleRight : Hashtable of <string,integer>
obstacleStraight : Hashtable of <string,integer>
obstacleAccel : Hashtable of <string,integer>
obstacleBoost : Hashtable of <string,integer>

hasPowerUp(powerUp : PowerUp) → boolean
{Fungsi yang mengecek kepemilikan sebuah powerUp. Mengembalikan trus jika punya,
false jika tidak}

laneOneDecision(obstacleStraight : Hashtable<string,integer>, obstacleRight :
   Hashtable<string,integer>) → string
{Fungsi yang menganalisis kemungkinan solusi ketika mobil berada di lane 1. Fungsi
ini menerima parameter berupa hashtable yang berisi obstacle dan powerup di jalur
depan dan kanannya kemudian mengembalikan string decision yang paling optimal}

middleDecision(obstacleLeft : Hashtable<string,integer>, obstacleStraight :
   Hashtable<string,integer>, obstacleRight : Hashtable<string,integer>, lane
   : int) → string
```

{Fungsi yang menganalisis kemungkinan solusi ketika mobil berada di lane 2 atau 3. Fungsi ini menerima parameter berupa hashtable yang berisi obstacle dan powerup di jalur depan, kiri, dan kanan serta lane mobil saat ini. kemudian mengembalikan string decision yang paling optimal}

```
laneFourDecision(obstacleLeft : Hashtable<string,integer>, obstacleStraight :
   Hashtable<string,integer>) → string
```
{Fungsi yang menganalisis kemungkinan solusi ketika mobil berada di lane 4. Fungsi ini menerima parameter berupa hashtable yang berisi obstacle dan powerup di jalur depan dan kirinya kemudian mengembalikan string decision yang paling optimal}

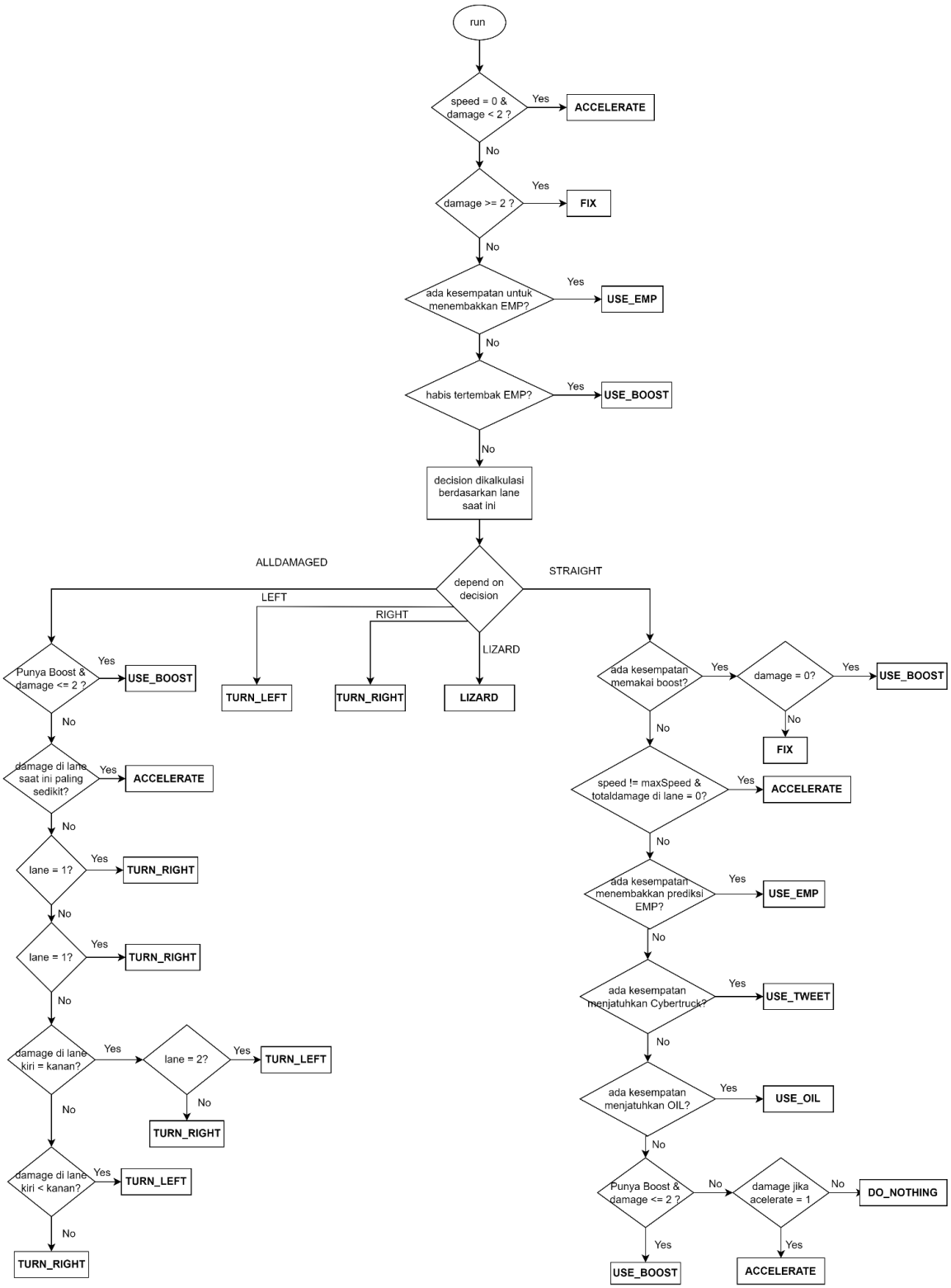
ALGORITMA

```
{Kondisi mendesak dan darurat}
if (myCar.speed = 0 and myCar.damage < 2) then
      {darurat kecepatan = 0}
      → ACCELERATE
else if (myCar.damage 2) then
      {mendesak untuk segera diperbaiki}
      → FIX

if (hasPowerUp(EMP) and myCar.position.block < opponent.position.block and
   (myCar.position.lane = opponent.position.lane or (myCar.position.lane = 2
   and opponent.position.lane = 1) or (myCar.position.lane = 3 and
   opponent.position.lane = 4)) then
   {mengeluarkan EMP ketika dibelakang lawan dan sudah pasti mengenai lawan}
      → USE_EMP

if (hasPowerUp(BOOST) and myCar.speed = 3 and
   obstacleBoost("TOTALDAMAGE")<=3) then
   {mengeluarkan boost ketika menerima serangan EMP dari lawan}
      → USE_BOOST

if (myCar.position.lane = 1) then
      {decision ketika mobil berada di lane 1}
      decision ← laneOneDecision(obstacleStraight, obstacleRight)
else if (myCar.position.lane = 2 or myCar.position.lane = 3) then
      {decision ketika mobil berada di lane 2 atau 3}
      decision ← middleDecision(obstacleLeft, obstacleStraight,
                obstacleRight, myCar.position.lane)
else {myCar.position.lane = 4}
      {decision ketika mobil berada di lane 4}
      decision ← laneFourDecision(obstacleLeft, obstacleStraight)
```

```
depend on decision
      decision = “STRAIGHT”   :
            {decision paling optimal adalah jalan lurus}
               if (hasPowerUp(BOOST) and !mycar.boosting and myCar.speed <=
               maxSpeed and obstacleBoost.get(TOTALDAMAGE) = 0) then
                  if (myCar.damage = 0) then
                  {menggunakan boost ketika tidak ada damage}
                  → USE_BOOST
                  else
                  {melakukan perbaikan terlebih dahulu sebelum boost}
                  → FIX

               if (myCar.speed  maxSpeed and
               obstacleAccel.get(TOTALDAMAGE)=0) then
                  {melakukan accelerate ketika damage yang diterima = 0}
                  → ACCELERATE

               if (hasPowerUp(EMP) and myCar.position.block <
               opponent.position.block and abs(myCar.position.lane –
               opponent.position.block) <= 1) then
                  {melakukan tembakan prediksi EMP (tidak selalu mengenai lawan)}
                  → USE_EMP

               if (hasPowerUp(TWEET)) then
                  {mengeluarkan cybertruck di depan lawan}
                  → USE_TWEET(opponent.position.lane,
                  opponent.position.block + BOOST_SPEED + 1)

               if (hasPowerUp(OIL) and myCar.position.block >
               opponen.position.block) then
                  {mengeluarkan oil jika berada di depan lawan}
                  → USE_OIL

               if (hasPowerUp(BOOST) and obstacleBoost.get(TOTALDAMAGE) <=2)
               then
                  {mengeluarkan boost walau menabrak beberapa obstacle}
                  → USE_BOOST
               if (obstacleAccel.get(TOTALDAMAGE) <= 1) then
                  {melakukan accelerate walau menabrak beberapa obstacle}
                  → ACCELERATE
               else
```

```
            → DO_NOTHING

decision = ”LEFT”       :
      {decision paling optimal adalah belok kiri}
      → TURN_LEFT

decision = “RIGHT”      :
      {decision paling optimal adalah belok kanan}
      → TURN_RIGHT

decision = “LIZARD”     :
      {decision paling optimal adalah menggunakan LIZARD}
      → USE_LIZARD

decision = “ALLDAMAGED” :
   {tidak ditemukan decision paling optimal}
   if (hasPowerUp(BOOST) and obstacleBoost(“TOTALDAMAGE”) <= 2) then
      {mengeluarkan boost walau menabrak beberapa obstacle}
      → USE_BOOST
   else
         if ((myCar.position.lane = 1 or obstacleAccel(“TOTALDAMAGE”)
         obstacleLeft(“TOTALDAMAGE”) and (mycar.position.lane = 4
         orobstacleAccel(“TOTALDAMAGE”)  obstacleRight(“TOTALDAMAGE”))
         then
            {melakukan accelerate walau menabrak beberapa obstacle}
            → ACCELERATE
         else
            if (myCar.position.lane = 1) then
            {belok kanan ketika berada di lane paling kiri}
            → TURN_RIGHT
            else if (myCar.position.lane = 4) then
            {belok kiri ketika berada di lane paling kanan}
            → TURN_LEFT
            else {myCar berada di lane 2 atau 3}
            {belok ke lane yang berada di tengah}
               if (obstacleLft(“TOTALDAMAGE”)
               obstacleRight(“TOTALDAMAGE”) then
                  → TURN_LEFT
               else
                  → TURN_RIGHT
```

B. Flowchart


run
speed = 0 & damage < 2 ?
Yes
ACCELERATE
No
damage >= 2 ?
Yes
FIX
No
ada kesempatan untuk menembakkan EMP?
Yes
USE_EMP
No
habis tertembak EMP?
Yes
USE_BOOST
No
decision dikalkulasi berdasarkan lane saat ini
depend on decision
ALLDAMAGED
LEFT
RIGHT
LIZARD
STRAIGHT
Punya Boost & damage <= 2 ?
Yes
USE_BOOST
No
damage di lane saat ini paling sedikit?
Yes
ACCELERATE
No
lane = 1?
Yes
TURN_RIGHT
No
lane = 1?
Yes
TURN_RIGHT
No
damage di lane kiri = kanan?
Yes
lane = 2?
Yes
TURN_LEFT
No
TURN_RIGHT
No
damage di lane kiri < kanan?
Yes
TURN_LEFT
No
TURN_RIGHT
TURN_LEFT
TURN_RIGHT
LIZARD
ada kesempatan memakai boost?
Yes
damage = 0?
Yes
USE_BOOST
No
FIX
No
speed != maxSpeed & totaldamage di lane = 0?
Yes
ACCELERATE
No
ada kesempatan menembakkan prediksi EMP?
Yes
USE_EMP
No
ada kesempatan menjatuhkan Cybertruck?
Yes
USE_TWEET
No
ada kesempatan menjatuhkan OIL?
Yes
USE_OIL
No
Punya Boost & damage <= 2 ?
No
damage jika acelerate = 1
No
DO_NOTHING
Yes
USE_BOOST
Yes
ACCELERATE

C. Struktur Data

Pada implementasi program ini, kami menggunakan beberapa struktur data, yaitu:

1. GameState

   Struktur data GameState berisi informasi penting dalam keberjalanan game Overdrive. Struktur data ini terdiri dari ronde game saat ini, ronde maksimal yang dapat dimainkan, player berupa struktur dara Car, opponent berupa struktur data Car, dan peta permainan.

2. Bot

   Struktur data bot merupakan struktur data untuk mengendalikan mobil selama permainan. Di dalam struktur data ini terdapat kumpulan Command dan method run untuk menjalankan mobil sesuai dengan algoritma yang ditentukan.

3. Car

   Struktur data Car berisi data-data player yang sedang bermain. Struktur data Car terdiri dari id pemain, position berupa koordinat lane dan block, speed mobil saat ini, damage yang diterima mobil, state mobil pada ronde sebelumnya, list powerups yang dimiliki pemain, penanda boosting, dan boostCounter.

4. Lane

   Struktur data Lane berisi kumpulan block dalam satu lajur permainan. Struktur data ini terdiri dari position, terrain setiap block, penanda block ditempati oleh player, dan penanda block ditempati oleh cybertruck.

5. Position

   Struktur data Position berisi koordinat posisi x dan y dalam peta pertandingan.

6. Command

   Struktur data Command berisi perintah yang akan diproses untuk keberjalanan permainan. Di dalam game Overdrive ini terdapat 11 command yang dapat dijalankan oleh pemain, yaitu:

   a. ACCELERATE : Command untuk melakukan peningkatan kecepatan menjadi status kecepatan di atasnya.

b. USE_BOOST : Command untuk melakukan boosting hingga kecepatan maksimal
c. DECELERATE : Command untuk melakukan penurunan kecepatan ke status kecepatan di bawahnya
d. TURN_LEFT : Command untuk berpindah ke jalur sebelah kiri
e. TURN_RIGHT : Command untuk berpindah ke jalur sebelah kanan
f. NOTHING : Command untuk tidak melakukan apapun selama ronde tersebut
g. USE_EMP : Command untuk menembakkan EMP ke jalur di depan player
h. FIX : Command untuk memperbaiki mobil dari damage yang diterima
i. USE_LIZARD : Command untuk melompat menghindari beberapa block di dalam jangkauan kecepatannya.
j. USE_OIL : Command untuk menumpahkan oli di block yang dipijak saat ini
k. USE_TWEET : Command untuk menjatuhkan cybertruck ke dalam posisi tertentu di dalam permainan

7. LookupPowerups

   Struktur data LookupPowerups berisi hashtable yang keynya berupa PowerUps dan valuenya bernilai banyaknya PowerUps yang dimiliki. Di dalam Struktur data ini terdapat methods untuk mengecek ketersediaan PowerUps

8. FileMaker

   Struktur data FileMaker berisi data unit testing dan methods untuk mencetak log permainan. Struktur data ini bertujuan untuk mencatat aktivitas bot selama bertanding dan digunakan untuk analisis yang dapat meningkatkan performa bot yang dibuat

D. Pengujian

Untuk menguji keoptimalan bot yang sudah kami buat, kami mencoba bertanding dengan *reference bot* yang telah disediakan oleh Entellect. Kami melakukan 30 kali pertandingan dengan 100% kemenangan. Secara rata-rata, bot kami mampu menyelesaikan pertandingan dengan 130 ronde. Dalam 30 pertandingan tersebut, kami mencatatkan ronde paling minimum sebanyak 119 ronde dan paling maksimum sebanyak 144 ronde. Total poin rata-rata yang kami raih mencapai 513 poin dengan poin minimum

sebesar 429 poin dan paling maksimum sebanyak 637 poin. Hasil pertandingan dapat berubah dengan sangat dinamis karena map dibangun secara otomatis ketika permainan dimulai. Hasil terbaik dapat diraih ketika keberadaan obstacle di map tidak terlalu banyak dan lawan jarang menembakkan EMP. Kasus terburuk ketika ada begitu banyak obstacle yang tersebar ke semua lane dan lawan sering menggunakan EMP untuk menembak ke arah player. Secara umum, strategi Greedy yang kami implementasikan dapat berjalan dengan baik dan mencapai tujuan dari permainan ini. Berdasarkan uji coba di atas, solusi yang kami buat sudah optimal dalam memenangkan permainan Overdrive ini.

Bab 5

Kesimpulan dan Saran

Kesimpulan

Permainan Overdrive dapat diselesaikan dengan strategi algoritma Greedy. Kelompok kami membuat strategi greedy berdasarkan damage dan speed sehingga pemain dapat melaju dengan kencang sekaligus menghindari potensi bertabrakan dengan rintangan yang ada. Bot yang kami buat dapat menyelesaikan permainan Overdrive dengan cukup optimal dan mendapatkan score yang tinggi. Kami juga memanfaatkan powerups yang ada untuk meningkatkan kecepatan player dan menghalangi laju dari mobil lawan. Implementasi program yang kami buat dapat dilihat pada laman ini

Saran

Terkait dengan topik terkait, kami menyarankan beberapa hal untuk diperhatikan sebagai berikut:

- Dalam algoritma greedy yang dipakai, penulis menyarankan untuk memprioritaskan menghindari obstacle yang dapat merusak dan memperlambat mobil, terutama seperti wall, oil, dan mud dikarenakan apabila mobil menabrak obstacle, maka mobil akan mendapat kerusakan yang akan mengurangi kecepatan maksimum dari mobil.
- Penulis juga menyarankan untuk membuat strategi khusus untuk mengurangi kecepatan mobil atau decelerate saat mendapati block di depan mobil terdapat rintangan namun hanya di bagian akhir dari block yang dapat dicapai.
- Penulis juga mengamati bahwa potensi dari boost sangat lah besar sehingga penulis menyarankan untuk memaksa menggunakan boost apabila semua jalur terdapat obstacle dan damage yang akan diterima apabila melewati jalur tengah <= 2.
- Penulis menyarankan untuk memperoleh bot yang lebih baik, cobalah dengan bertanding dengan bot teman dan lakukan analisis terkait pergerakan dari bot kita dan bot musuh, lalu ambil pilihan yang terbaik untuk diterapkan di bot kita.

Daftar Pustaka

https://github.com/EntelectChallenge/2020-Overdrive diakses pada 18 Februari 2022.

Rinaldi Munir. 2022. *Algoritma Greedy Bag. 1.*
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf