

**LAPORAN TUGAS BESAR
STRATEGI ALGORITMA - IF2211**

Kelas Mahasiswa K03

Dosen: Rila Mandala

**Diajukan sebagai tugas besar Mata Kuliah IF2211 Strategi Algoritma pada Semester II
Tahun Akademik 2022/2023**



Disusun Oleh:

Kelompok greedisgood

Haikal Ardzi Shofiyurrohman	13521012
Eunice Sarah Siregar	13521013
M. Malik I. Baharsyah	13521029

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2023**

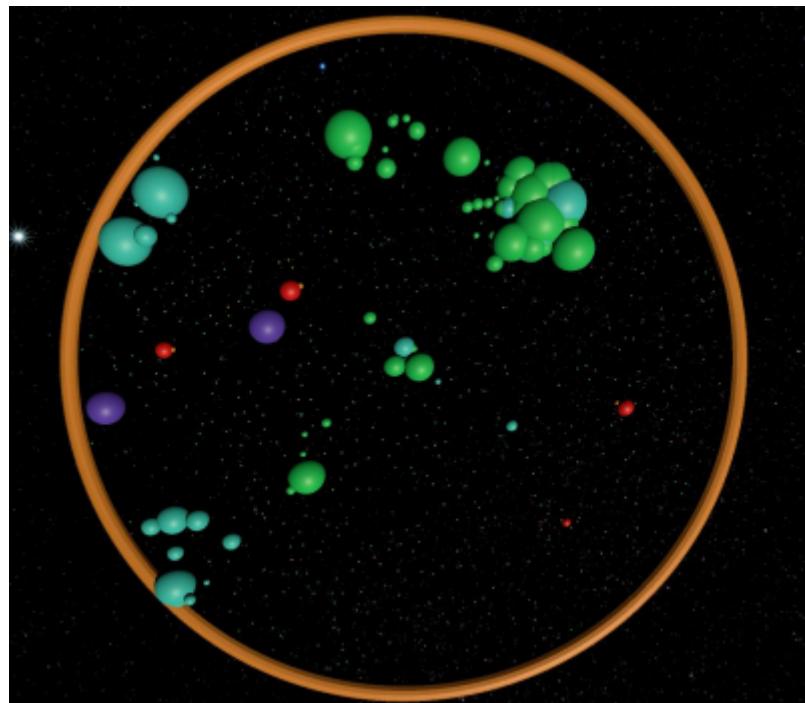
DAFTAR ISI

BAB I DESKRIPSI TUGAS	1
BAB II LANDASAN TEORI	5
2.1. Dasar Teori	5
2.2. Cara Kerja Program Bot Permainan Galaxio	6
2.3. Implementasi Algoritma Greedy ke dalam Bot Permainan Galaxio	6
2.4. Proses Pembuatan Bot dan Cara Menjalankan Game Engine Galaxio	7
2.4.1. Proses Pembuatan Bot	7
2.4.2. Cara Menjalankan Game Engine Galaxio	7
BAB III APLIKASI STRATEGI GREEDY	9
3.1. Mapping Persoalan Menjadi Algoritma Greedy	9
3.1.1 Mapping Komponen Algoritma Greedy pada Permasalahan General Bot	9
3.1.2 Mapping Komponen Algoritma Greedy pada Permasalahan Point	9
3.1.3 Mapping Komponen Algoritma Greedy pada Permasalahan Menghindari Obstacle dan Projectile	10
3.1.4 Mapping Komponen Algoritma Greedy pada Permasalahan Penyerangan	11
3.2. Eksplorasi Alternatif	12
3.2.1 Greedy by Point	12
3.2.2 Strategi menghindari projectile dan obstacle	13
3.2.3 Strategi dalam menyerang	13
3.3. Analisis Efisiensi dan Efektivitas	13
BAB IV IMPLEMENTASI DAN PENGUJIAN	15
4.1. Implementasi Algoritma Greedy	15
4.2. Penjelasan Struktur Data Bot Permainan Galaxio	21
4.3. Analisis Desain Solusi Algoritma Greedy Pada Setiap Pengujian	31
BAB V KESIMPULAN, SARAN DAN REFLEKSI	34
5.1. Kesimpulan	34
5.2. Saran	34
5.3. Refleksi	34
DAFTAR PUSTAKA	35
LAMPIRAN	36

BAB I

DESKRIPSI TUGAS

Galaxio adalah sebuah *game battle royale* yang mempertandingkan bot kapal anda dengan beberapa bot kapal yang lain. Setiap pemain akan memiliki sebuah bot kapal dan tujuan dari permainan adalah agar bot kapal anda yang tetap hidup hingga akhir permainan. Penjelasan lebih lanjut mengenai aturan permainan akan dijelaskan di bawah. Agar dapat memenangkan pertandingan, setiap *bot* harus mengimplementasikan strategi tertentu untuk dapat memenangkan permainan.



Gambar 1. Ilustrasi permainan Galaxio

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

<https://github.com/EntelectChallenge/2021-Galaxio>

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

<https://github.com/EntelectChallenge/2021-Galaxio/releases/tag/2021.3.2>

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

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

1. Peta permainan berbentuk kartesius yang memiliki arah positif dan negatif. Peta hanya menangani angka bulat. Kapal hanya bisa berada di integer x,y yang ada di peta. Pusat peta adalah 0,0 dan ujung dari peta merupakan radius. Jumlah ronde maximum pada game sama dengan ukuran radius. Pada peta, akan terdapat 5 objek, yaitu *Players*, *Food*, *Wormholes*, *Gas Clouds*, *Asteroid Fields*. Ukuran peta akan mengecil seiring batasan peta mengecil.
2. Kecepatan kapal dilambangkan dengan x . Kecepatan kapal akan dimulai dengan kecepatan 20 dan berkurang setiap ukuran kapal bertambah. Ukuran (radius) kapal akan dimulai dengan ukuran 10. Heading dari kapal dapat bergerak antar 0 hingga 359 derajat. Efek *afterburner* akan meningkatkan kecepatan kapal dengan faktor 2, tetapi mengecilkan ukuran kapal sebanyak 1 setiap tick. Kemudian kapal akan menerima 1 salvo charge setiap 10 tick. Setiap kapal hanya dapat menampung 5 *salvo charge*. Penembakan *salvo torpedo* (ukuran 10) mengurangkan ukuran kapal sebanyak 5.
3. Setiap objek pada lintasan punya koordinat x,y dan radius yang mendefinisikan ukuran dan bentuknya. *Food* akan disebarluaskan pada peta dengan ukuran 3 dan dapat dikonsumsi oleh kapal player. Apabila player mengkonsumsi *Food*, maka *Player* akan bertambah ukuran yang sama dengan *Food*. *Food* memiliki peluang untuk berubah menjadi *Super Food*. Apabila *Super Food* dikonsumsi maka setiap makan *Food*, efeknya akan 2 kali dari *Food* yang dikonsumsi. Efek dari *Super Food* bertahan selama 5 tick.
4. *Wormhole* ada secara berpasangan dan memperbolehkan kapal dari player untuk memasukinya dan keluar di pasangan satu lagi. *Wormhole* akan bertambah besar setiap *tick game* hingga ukuran maximum. Ketika *Wormhole* dilewati, maka *wormhole* akan mengecil sebanyak setengah dari ukuran kapal yang melewatinya dengan syarat *wormhole* lebih besar dari kapal player.
5. *Gas Clouds* akan tersebar pada peta. Kapal dapat melewati *gas cloud*. Setiap kapal bertabrakan dengan *gas cloud*, ukuran dari kapal akan mengecil 1 setiap tick game. Saat kapal tidak lagi bertabrakan dengan *gas cloud*, maka efek pengurangan akan hilang.
6. *Torpedo Salvo* akan muncul pada peta yang berasal dari kapal lain. *Torpedo Salvo* berjalan dalam lintasan lurus dan dapat menghancurkan semua objek yang berada pada lintasannya. *Torpedo Salvo* dapat mengurangi ukuran kapal yang ditabraknya. *Torpedo Salvo* akan mengecil apabila bertabrakan dengan objek lain sebanyak ukuran yang dimiliki dari objek yang ditabraknya.
7. *Supernova* merupakan senjata yang hanya muncul satu kali pada permainan di antara quarter pertama dan quarter terakhir. Senjata ini tidak akan bertabrakan dengan objek lain pada lintasannya. *Player* yang menembakkannya dapat meledakannya dan memberi damage ke *player* yang berada dalam zona. Area ledakan akan berubah menjadi *gas cloud*.
8. *Player* dapat meluncurkan *teleporter* pada suatu arah di peta. *Teleporter* tersebut bergerak dalam direksi dengan kecepatan 20 dan tidak bertabrakan dengan objek apapun. *Player* tersebut dapat berpindah ke tempat *teleporter* tersebut. Harga setiap peluncuran *teleporter* adalah 20. Setiap 100 tick *player* akan mendapatkan 1 *teleporter* dengan jumlah maximum adalah 10.

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

9. Ketika kapal *player* bertabrakan dengan kapal lain, maka kapal yang lebih besar akan dikonsumsi oleh kapal yang lebih kecil sebanyak 50% dari ukuran kapal yang lebih besar hingga ukuran maximum dari ukuran kapal yang lebih kecil. Hasil dari tabrakan akan mengarahkan kedua dari kapal tersebut lawan arah.
10. Terdapat beberapa *command* yang dapat dilakukan oleh player. Setiap *tick*, player hanya dapat memberikan satu *command*. Berikut jenis-jenis dari command yang ada dalam permainan:
 - a. FORWARD
 - b. STOP
 - c. START_AFTERRUNNER
 - d. STOP_AFTERRUNNER
 - e. FIRE_TORPEDOES
 - f. FIRE_SUPERNOVA
 - g. DETONATE_SUPERNOVA
 - h. FIRE_TELEPORTER
 - i. TELEPORT
 - j. USE_SHIELD
11. Setiap *player* akan memiliki *score* yang hanya dapat dilihat jika permainan berakhir. *Score* ini digunakan saat kasus *tie breaking* (semua kapal mati). Jika mengonsumsi kapal player lain, maka *score* bertambah 10, jika mengonsumsi *food* atau melewati *wormhole*, maka *score* bertambah 1. Pemenang permainan adalah kapal yang bertahan paling terakhir dan apabila *tie breaker* maka pemenang adalah kapal dengan *score* tertinggi.

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

<https://github.com/EntelectChallenge/2021-Galaxio/blob/develop/game-engine/game-rules.md>

BAB II

LANDASAN TEORI

2.1. Dasar Teori

Dalam memenangkan permainan Galaxio, dibutuhkannya algoritma yang dapat mendukung performa kapal. Algoritma yang akan dipakai adalah algoritma Greedy, dengan mencari solusi yang paling optimal. Algoritma Greedy merupakan metode yang sangat populer dan sederhana untuk memecahkan permasalahan optimasi. Algoritma Greedy memiliki prinsip setiap langkah yang dapat dilakukan, lakukan saat itu juga. Namun, langkah tersebut tidak bisa dilakukan secara asal, diperlukan keputusan yang terbaik untuk menentukan langkah yang akan diambil karena tidak dapat kembali ke langkah sebelumnya. Oleh sebab itu, dengan menggunakan algoritma Greedy diharapkan menghasilkan solusi optimum lokal di setiap langkahnya. Hal tersebut ditujukan untuk mengarah pada solusi dengan optimum global.

Pada algoritma Greedy, terdapat komponen yang perlu ditinjau kembali, seperti:

- himpunan kandidat merupakan kandidat yang mungkin terpilih di setiap langkahnya,
- himpunan solusi merupakan kumpulan langkah yang dipilih,
- fungsi solusi merupakan kumpulan solusi yang memberikan solusi,
- fungsi seleksi merupakan seleksi untuk mendapatkan solusi yang terbaik,
- fungsi kelayakan merupakan pemeriksaan terhadap kandidat yang dipilih oleh fungsi seleksi dan dapat dimasukkan ke dalam himpunan solusi, dan
- fungsi objektif merupakan memaksimalisasi atau meminimalisasi parameter terhadap suatu persoalan.

Berdasarkan komponen tersebut, maka dapat disimpulkan bahwa algoritma Greedy melibatkan pencarian himpunan bagian dari himpunan kandidat, dengan himpunan bagian harus memenuhi syarat seperti himpunan bagian menyatakan suatu solusi dan dioptimasi dengan fungsi objektif. Skema umum algoritma Greedy seperti berikut.

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

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

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

Pada akhir setiap iterasi menghasilkan solusi dalam bentuk optimum lokal, tetapi pada akhir loop while-do akan dibentuk solusi optimum global. Solusi optimum global pada algoritma Greedy, belum tentu menjadi solusi terbaik. Hal itu disebabkan oleh operasi algoritma Greedy tidak menyeluruh terhadap semua kemungkinan solusi yang ada. Algoritma Greedy juga memiliki fungsi seleksi yang berbeda, sehingga pengguna harus memilih fungsi yang paling tepat untuk menghasilkan solusi yang optimal. Namun, algoritma Greedy memiliki kelebihan seperti kebutuhan waktunya eksponensial dibandingkan dengan algoritma Brute Force. Meskipun algoritma Greedy tidak menghasilkan solusi terbaik, tetapi solusi algoritma Greedy yang dihasilkan dapat dianggap sebagai solusi hampiran (*approximation*).

2.2. Cara Kerja Program Bot Permainan Galaxio

Secara garis besar, cara kerja *bot* permainan Galaxio ini adalah pertama-tama *bot* mengambil setiap informasi yang tersedia pada *state* setiap *tick*-nya. Data-data yang terkandung dalam *state* tersebut beserta penjelasannya adalah sebagai berikut:

- *bot*: informasi detail bot.
 - *id*: id bot.
 - *size*: ukuran bot.
 - *speed*: kecepatan bot.
 - *currentHeading*: arah bot pada *tick* saat ini.
 - *position*: posisi bot berupa titik untuk *tick* saat ini.
 - *x*: posisi bot di koordinat sumbu x.
 - *y*: posisi bot di koordinat sumbu y.
 - *gameObjectType*: tipe objek bot, yakni *player*.
- *playerAction*: aksi bot yang akan dikirim ke *log* untuk *tick* saat ini.
 - *playerId*: identitas player.
 - *action*: aksi bot untuk *tick* saat ini.
 - *heading*: arah bot ketika melakukan aksi, berupa derajat.
- *gameState*: informasi detail permainan pada *tick* saat ini.
 - *World*: informasi detail *map*.
 - *centerPoint*: titik tengah *map*.
 - *x*: titik tengah *map* di koordinat sumbu x.
 - *y*: titik tengah *map* di koordinat sumbu y.
 - *radius*: jari-jari peda pada saat inisialisasi awal permainan.
 - *currentTick*: informasi *tick* saat ini.
 - *gameObjects*: sebuah list berupa objek-objek aktif yang terdapat di *map* untuk *tick* saat ini.
 - *id*: id objek.
 - *size*: ukuran objek.
 - *speed*: kecepatan objek.
 - *currentHeading*: arah objek.
 - *position*: posisi objek berupa titik.
 - *x*: titik objek di sumbu x.
 - *y*: titik objek di sumbu y.
 - *gameObjectType*: tipe objek, dapat berupa *food*, *wormhole*, *gas cloud*, dll.

- *playerGameObjects*: sebuah list berupa semua player aktif yang terdapat pada *tick* saat ini.
 - *id*: *id player*.
 - *size*: ukuran *player*.
 - *speed*: kecepatan *player*.
 - *currentHeading*: arah *player*.
 - *position*: posisi berupa titik *player*.
 - *x*: titik *player* pada sumbu x.
 - *y*: titik *player* pada sumbu y.
 - *gameObjectType*: tipe objek, yaitu *player*.
- *isTeleport*: informasi bernilai 1 jika *bot* telah melakukan penembakan *teleporter* dan *teleporter* sedang berjalan mengarah ke target, sebaliknya bernilai 0.
- *teleportTick*: informasi *tick* saat *bot* ingin melakukan *teleport* setelah menembak *teleporter*.
- *countTeleport*: jumlah *teleporter* yang dimiliki *bot* agar bisa ditembakkan.
- *target*: target *bot* saat melakukan aksi menembak *teleporter*.

Setelah *bot* memperoleh data-data di atas, maka *bot* akan melakukan analisis dengan algoritma *greedy* yang telah dibuat berdasarkan seluruh informasi *state* dari permainan untuk mendapatkan aksi yang akan dikirim ke *hub* permainan untuk setiap *tick*-nya. Pada kasus permainan Galaxio, sintaks perintah aksi dan *heading* *bot* yang dikirim ke *hub* harus sesuai dengan perintah yang telah disediakan. Apabila tidak sesuai, maka konsekuensinya adalah *bot* tidak dapat *di-build* menjadi file berformat *jar*. Selain itu, jika aksi yang dikirim ke *hub* tidak sesuai dengan peraturan permainan, maka *bot* tidak akan melakukan apa-apa. Sebagai contoh, ketika *bot* sedang tidak menembakkan *teleporter* dan aksi yang dikirim ke *hub* adalah *TELEPORT*, maka aksi tersebut tidak valid dan mengakibatkan *bot* untuk diam tidak melakukan aksi apa-apa.

2.3. Implementasi Algoritma Greedy ke dalam Bot Permainan Galaxio

Setelah *bot* mendapatkan seluruh informasi detail *state* yang terdapat pada permainan Galaxio, selanjutnya *bot* dapat mengirim aksi dan *heading* kepada *hub* permainan setiap *tick*-nya dengan algoritma *greedy* melalui pendekatan heuristik yang didefinisikan oleh pemrogram. Seperti yang telah dijelaskan pada poin sebelumnya, terdapat banyak kemungkinan solusi algoritma *greedy* yang dapat diperoleh. Meskipun pada implementasinya solusi yang didapat dari algoritma *greedy* bisa jadi bukan merupakan optimum global, tetapi algoritma *greedy* masih dapat memberikan solusi yang cukup baik dalam banyak kasus. Dalam konteks permainan Galaxio, meskipun solusi dari algoritma *greedy* mungkin tidak selalu optimal, *bot* yang dibuat dengan algoritma ini masih mampu memberikan performa yang cukup baik dalam memenangkan permainan.

Oleh karena itu, dalam proses pembuatan *bot* permainan Galaxio, penulis mencari algoritma *greedy* yang saat digunakan dapat memberikan performa terbaik dalam memenangkan permainan sesuai dengan peraturan yang berlaku. Selain itu, penulis juga memperhitungkan berbagai aspek seperti pergerakan lawan, kondisi permainan, dan strategi yang dapat dilakukan oleh *bot*. Dalam hal ini, penggunaan pendekatan heuristik yang tepat

oleh penulis dapat membantu *bot* untuk mengambil keputusan yang cerdas dalam setiap situasi permainan untuk mencapai kemenangan.

2.4. Proses Pembuatan Bot dan Cara Menjalankan Game Engine Galaxio

2.4.1. Proses Pembuatan Bot

Pembuatan *bot* dapat dimulai dengan mengunduh *starter-pack* versi 2021.3.2 yang didapat di *repository* Github pemilik resmi permainan Galaxio (<https://github.com/EntelectChallenge/2021-Galaxio/releases/tag/2021.3.2>) dan pembuat *bot* mengekstraknya terlebih dahulu. Pada permainan Galaxio, *bot* dapat dibangun dengan berbagai macam bahasa pemrograman seperti Java, C++, Kotlin, Python, dll. Pada kesempatan kali ini, fokus pembahasan penulis hanya pada pembuatan *bot* dalam bahasa pemrograman Java.

Basis *folder* pengembangan *bot* berbahasa Java terdapat di *folder* *starter-bots/JavaBot/src/main/java*. Sebelum memulai pengembangan *bot* Galaxio dapat terlebih dahulu memperbarui *file* ObjectTypes.java dan PlayerActions.java di *folder* Enums dengan menambahkan objek *TORPEDO_SALVO*, *SUPERFOOD*, *SUPERNOVA_PICKUP*, *SUPERNOVA_BOMB*, *TELEPORTER*, *SHIELD* untuk file ObjectTypes.java dan aksi *FIRETORPEDOES*, *FIRESUPERNOVA*, *DETONATESUPERNOVA*, *FIRETELEPORT*, *TELEPORT*, *ACTIVATESHIELD* pada file PlayerActions.java. Setelah dilakukan pembaruan, pemrogram dapat memulai pengembangan *bot* Galaxio berbasis bahasa Java.

Pengembangan *bot* dilakukan di *file* bernama BotService.java yang terdapat pada direktori basis *folder* java yang telah disebutkan sebelumnya lalu menuju ke *folder* Services. Setelah pengembangan selesai, *source code* perlu dilakukan *build* sebuah *file* jar, yaitu dengan menggunakan maven atau intelliJ. Pada kasus *build* *file* jar menggunakan maven, dilakukan pemanggilan *command* “mvn clean package” pada terminal di *folder* JavaBot sehingga *build* berhasil dilakukan dan akan muncul *folder* bernama target yang berisi JavaBot.jar.

2.4.2. Cara Menjalankan Game Engine Galaxio

Sebelum menjalankan *game engine*, jumlah *bot* yang ingin dijalankan pada *game engine* dapat diatur dengan cara memodifikasi *file* appsettings.json di dalam folder runner-publish dan engine-publish.

Cara menjalankan Game Runner, Engine, atau Logger pada UNIX-based OS dapat memodifikasi atau langsung menjalankan “run.sh” yang tersedia pada *folder* starter-pack.

Untuk kasus pengguna Windows, dapat menggunakan atau memodifikasi batch script seperti berikut:

```
@echo off  
:: Game Runner
```

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

```
cd ./runner-publish/
start "" dotnet GameRunner.dll

:: Game Engine
cd ../engine-publish/
timeout /t 1
start "" dotnet Engine.dll

:: Game Logger
cd ../logger-publish/
timeout /t 1
start "" dotnet Logger.dll

:: Bots
cd ../reference-bot-publish/
timeout /t 3
start "" dotnet ReferenceBot.dll
cd ../

pause
```

dengan “dotnet ReferenceBot.dll” adalah perintah untuk menjalankan *reference bot*, yaitu *template bot* bawaan yang diberikan oleh Galaxio. Dalam menjalankan *bot* milik sendiri yang telah dibuat, dapat mengubah direktori terlebih dahulu ke *folder target* setelah dilakukan *build source code* Java lalu menambahkan perintah “java -jar JavaBot.jar”. Setelah itu, simpan batch script di dalam folder starter-pack. *Bot* siap untuk dijalankan dan bertarung dengan *bot* lain untuk mencapai kemenangan sesuai dengan algoritma yang dibuat dengan bahasa Java.

BAB III

APLIKASI STRATEGI GREEDY

3.1. *Mapping* Persoalan Menjadi Algoritma Greedy

3.1.1 *Mapping* Komponen Algoritma *Greedy* pada Permasalahan *General Bot*

Permainan Galaxio memiliki tujuan untuk menjadi pemain terakhir yang dapat bertahan dalam permainan atau yang biasa disebut *battle royale*. Untuk mencapai tujuan tersebut terdapat berbagai cara, seperti berusaha untuk memakan sebanyak-banyaknya, menyerang bot lawan dengan bermacam-macam *projectile*, dan lain-lain.

Komponen	Definisi
Himpunan kandidat	Semua perintah yang tersedia
Himpunan solusi	Semua perintah yang dipilih
Fungsi solusi	Mengecek perintah yang dipilih agar memberikan pertahanan yang terbaik dan dapat dilakukan saat itu juga
Fungsi seleksi	Memilih perintah yang menghasilkan pertahanan terbaik
Fungsi kelayakan	Memeriksa perintah yang dipilih valid dan dapat dilakukan atau tidak
Fungsi objektif	Memaksimumkan pertahanan yang dilakukan dalam sebuah ronde

3.1.2 *Mapping* Komponen Algoritma *Greedy* pada Permasalahan *Point*

Permainan Galaxio untuk meningkatkan jumlah point yang dimiliki dapat diperoleh dari foodList yang berisikan *object type FOOD*, *object type SUPERFOOD*, dan *object type SUPERNOVA_PICKUP*. Selain itu, dapat juga memperoleh point dengan memakan *ship* lawan yang berukuran lebih kecil dan disimpan dalam satu variable.

Komponen	Definisi
Himpunan kandidat	Permutasi dari perintah <i>FORWARD</i> , perintah <i>STOP</i> , perintah <i>START_AFTERRBURNER</i> , perintah <i>STOP_AFTERRBURNER</i> , perintah <i>FIRE_TORPEDOES</i> , perintah <i>FIRE_SUPERNOVA</i> , perintah <i>DETONATE_SUPERNOVA</i> , perintah <i>FIRE_TELEPORT</i> , perintah <i>TELEPORT</i> ,

	<i>atau</i> tidak melakukan semuanya. Selain itu, terdapat aksi yang dapat dilakukan oleh <i>ship</i> , berputar ke arah suatu objek dalam satuan derajat.
Himpunan solusi	Kemungkinan permutasi dari kandidat yang dapat meningkatkan point <i>ship</i> .
Fungsi solusi	Mengecek permutasi dari perintah tersebut membuat <i>bot</i> menambah point berjalan seperti seharusnya (tidak memakan <i>bot</i> yang berukuran lebih besar dari dirinya)
Fungsi seleksi	Memilih perintah yang menghasilkan point terbanyak
Fungsi kelayakan	Memeriksa perintah yang dipilih valid dan dapat dilakukan atau tidak. Dalam hal ini, perintah yang dipilih adalah perintah <i>FORWARD</i> dengan mengarah ke <i>foodList</i> atau <i>enemies</i> dengan ukuran lawan lebih kecil.
Fungsi objektif	Memaksimalkan point yang didapat dengan mencari <i>food</i> terdekat

3.1.3 Mapping Komponen Algoritma *Greedy* pada Permasalahan Menghindari *Obstacle* dan *Projectile*

Dalam permainan Galaxio, terdapat *object* yang dapat mengurangi point *ship* dengan membuat list *obstacle* dan *projectile*. Obstacle list berisi *object type GAS_CLOUD* dan *object type ASTEROID_FIELD*. Sedangkan dalam *projectile list* berisi *object type TORPEDO_SALVO*, *object type TELEPORT*, dan *object type SUPERNOVA_BOMB*.

Komponen	Definisi
Himpunan kandidat	Permutasi dari perintah <i>FORWARD</i> , perintah <i>STOP</i> , perintah <i>START_AFTERSHOCK</i> , perintah <i>STOP_AFTERSHOCK</i> , perintah <i>FIRE_TORPEDOES</i> , perintah <i>FIRE_SUPERNOVA</i> , perintah <i>DETONATE_SUPERNOVA</i> , perintah <i>FIRE_TELEPORT</i> , perintah <i>TELEPORT</i> , <i>atau</i> tidak melakukan semuanya. Selain itu, terdapat aksi yang dapat dilakukan oleh <i>ship</i> , berputar ke arah suatu objek dalam

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

	satuan derajat.
Himpunan solusi	Kemungkinan permutasi dari kandidat yang dapat menghindari dari <i>obstacle</i> dan <i>projectile</i> lain.
Fungsi solusi	Mengecek permutasi dari perintah tersebut membuat bot terhindar dari <i>obstacle</i> dan <i>projectile</i> berjalan seperti seharusnya.
Fungsi seleksi	Memilih perintah berdasarkan data keadaan permainan dengan memperhatikan tingkat prioritas perintah yang harus dilakukan. Tingkat prioritas tersebut dapat berubah-ubah sesuai dengan keadaan permainan.
Fungsi kelayakan	Mengecek perintah yang dituliskan adalah perintah yang valid sesuai dengan daftar perintah. Pada strategi menghindari <i>obstacle</i> dan <i>projectile</i> adalah perintah <i>FORWARD</i> , atau perintah <i>ACTIVATESHIELD</i> dan dengan berputar ke arah <i>projectile</i> atau <i>foodList</i> .
Fungsi objektif	Mencari permutasi dari perintah yang dapat mendukung <i>bot</i> dalam menghindari <i>obstacle</i> dan <i>projectile</i> dengan pengurangan point paling sedikit.

3.1.4 Mapping Komponen Algoritma *Greedy* pada Permasalahan Penyerangan

Untuk bertahan hidup sampai akhir permainan, diperlukan adanya serangan agar lawan dapat mati lebih cepat. Lawan yang dimaksud disimpan dalam variabel *enemies* dengan mendapatkan *id* semua *player* yang tidak sama dengan *id* pemain.

Komponen	Definisi
Himpunan kandidat	Permutasi dari perintah <i>FORWARD</i> , perintah <i>STOP</i> , perintah <i>START_AFTERSHIELD</i> , perintah <i>STOP_AFTERSHIELD</i> , perintah <i>FIRE_TORPEDOES</i> , perintah <i>FIRE_SUPERNOVA</i> , perintah <i>DETONATE_SUPERNOVA</i> , perintah <i>FIRE_TELEPORT</i> , perintah <i>TELEPORT</i> , atau tidak melakukan semuanya. Selain itu, terdapat aksi yang dapat dilakukan oleh <i>ship</i> , berputar ke arah suatu objek dalam

	satuan derajat.
Himpunan solusi	Kemungkinan permutasi dari kandidat yang dapat menyerang lawan.
Fungsi solusi	Mengecek permutasi dari perintah tersebut membuat bot menyerang lawan (memastikan jarak lawan agar tepat sasaran)
Fungsi seleksi	Memilih perintah berdasarkan data keadaan permainan dengan memperhatikan tingkat prioritas perintah yang harus dilakukan. Tingkat prioritas tersebut dapat berubah-ubah sesuai dengan keadaan permainan.
Fungsi kelayakan	Mengecek perintah yang dituliskan adalah perintah yang valid sesuai dengan daftar perintah. Pada strategi menyerang lawan adalah perintah <i>FIRETORPEDOES</i> , perintah <i>FIRETELEPORT</i> , perintah <i>TELEPORT</i> dan dengan memutar kearah lawan.
Fungsi objektif	Mencari permutasi dari perintah yang dapat mendukung <i>bot</i> dalam menyerang lawan dengan tingkat keakuratan yang tinggi.

3.2. Eksplorasi Alternatif

Implementasi algoritma *greedy* pada *bot* permainan Galaxio memiliki banyak alternatif solusinya. Hal tersebut merupakan dampak dari komponen pada *game engine* yang jumlahnya sangat banyak dan saling berinteraksi satu sama lain sehingga memunculkan cabang-cabang solusi lain yang angkanya tidak sedikit. Oleh karena itu, strategi yang dipilih penulis memiliki interaksi dengan strategi algoritma *greedy* yang berbeda. Berikut ini beberapa alternatif yang dapat diimplementasikan pada *bot* pemain:

3.2.1 *Greedy by Point*

Greedy by point adalah pendekatan algoritma *greedy* yang mengutamakan penambahan point dengan memakan makanan atau lawan yang ada disekitar *ship*. Pada setiap permainan, *ship* akan mencari makan terus-menerus sampai ukurannya membesar. Ada beberapa strategi untuk meningkatkan point agar memperbesar ukuran *ship*.

Strategi yang dipilih, yaitu ketika ukuran lawan lebih besar dari ukuran *bot* dan jarak antara *bot* dengan lawan kurang dari 100, maka *bot* akan mencari makan dengan perintah FORWARD dan ke arah *foodList*. Strategi ini bertujuan untuk menghindari lawan yang berukuran lebih besar dan mengutamakan untuk memperoleh point yang lebih besar dari

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

lawan. Hal ini dikarenakan ketika ukuran lawan lebih besar, maka bot akan kalah dalam permainan.

3.2.2 Strategi menghindari *projectile* dan *obstacle*

Saat terkena *projectile* dan *obstacle*, point *ship* akan berkurang sehingga mengakibatkan mengecilnya ukuran *ship*. Ukuran *ship* yang kecil mengakibatkan risiko terserang lawan semakin besar sehingga dapat kalah dalam waktu yang singkat. Strategi ini terjadi ketika terdapat object type *obstacleList* dan *projectileList* di dekat ship dan akan melakukan perintah *FORWARD* dan perintah *ACTIVATESHIELD* serta memutar ke arah *projectileList* atau *foodList*.

3.2.3 Strategi dalam menyerang

Untuk mampu bertahan sampai akhir permainan, dibutuhkan strategi untuk menyerang lawan. Ketika menyerang lawan, akan terjadi pengurangan point pada bot, tetapi saat serangan tepat kena akan mendapatkan point sebesar point. Penyerangan dapat terjadi ketika disekitar bot terdapat lawan. Dengan perintah *FIRETORPEDOES*, perintah *FIRETELEPORT*, atau perintah *TELEPORT* dengan memutar ke arah lawan.

3.3. Analisis Efisiensi dan Efektivitas

Alternatif Solusi Greedy	Efisiensi	Efektivitas
Greedy by Point	Best Case dan Worst Case: $O(1)$. Ketika terdapat musuh di sekitar bot yang lebih besar, maka bot akan mencari makan di dekatnya.	Dengan mengutamakan dalam mencari makan, akan menambah point dari bot sehingga memperbesar ukuran dari bot. Ukuran bot yang besar mampu meningkatkan potensi untuk menang lebih banyak lagi.
Strategi menghindari <i>projectile</i> dan <i>obstacle</i>	Best Case dan Worst Case: $O(n)$ Ketika tidak terdapat obstacle dan projectile, sehingga melakukan <i>ACTIVATESHIELD</i> atau mengarah ke <i>foodList</i>	Untuk menghindari <i>projectile</i> atau <i>obstacle</i> , algoritma ini memanfaatkan <i>foodList</i> agar tidak terkena efek projectile atau obstacle.
Strategi dalam menyerang	Best Case dan Worst Case: $O(1)$	Algoritma ini memanfaatkan perintah untuk menyerang lawan. Serangan yang dilakukan dapat mengurangi point lawan sehingga

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

	Saat terdapat bot lawan mendekat, akan aktif perintah <i>FIRETORPEDOES</i> , perintah <i>FIRETELEPORT</i> , atau <i>TELEPORT</i> dan bergerak ke arah lawan.	dapat memperbesar kemungkinan untuk bertahan hidup.
--	--	---

Berdasarkan penjelasan dari algoritma diatas, penulis memutuskan untuk memilih algoritma *greedy by point* dengan mengutamakan ukuran *bot* sampai besar.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi Algoritma Greedy

```
PUBLIC CLASS BotService

    PRIVATE GameObject bot

    PRIVATE PlayerAction thisPlayerAction

    PRIVATE GameState gameState

    PRIVATE boolean isTeleport = false

    PRIVATE int teleportTick = 0

    PRIVATE int countTeleport = 1

    PRIVATE GameObject target


    PUBLIC PROCEDURE BotService()

        playerAction = NEW PlayerAction()

        gameState = NEW GameState()

    END PROCEDURE


    PUBLIC FUNCTION GameObject getBot()

        RETURN bot

    END FUNCTION


    PUBLIC PROCEDURE setBot (GameObject newBot)

        bot=newBot

    END PROCEDURE


    PUBLIC PROCEDURE setPlayerAction(PlayerAction newPlayerAction)

        playerAction=newPlayerAction

    END PROCEDURE

    //Prosedur pengimplementasian algoritma greedy
```

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

```
PUBLIC PROCEDURE computeNextPlayerAction(PlayerAction playerAction)
    //Jika kapal tidak ada gameObject yang bisa dituju kapal
    playerAction.action = PlayerActions.FORWARD
    playerAction.heading = NEW Random().nextInt(360)

    //Pengecekan ketersediaan gameObject
    IF NOT gameState.getGameObjects().isEmpty() THEN
        /*
            * Inisiasi dengan pengambilan segala informasi dalam game
            setiap tick
            * dengan memasukkan GameObject kedalam list ke masing-masing
            kategori
        */
        GameObject[] foodList = gameState.getGameObjects()
            .stream().filter(item -> item.getGameObjects() ==
ObjectTypes.FOOD OR
                                item.getGameObjects() ==
ObjectTypes.SUPERFOOD OR
                                item.getGameObjects() ==
ObjectTypes.SUPERNOVA_PICKUP)
            .sorted(Comparator.comparing(item ->
getDistanceBetween(bot, item)))
            .collect(Collectors.toList())

        GameObject[] enemies = gameState.getPlayerGameObjects()
            .stream().filter(enemy -> enemy.id != bot.id)
            .sorted(Comparator.comparing(enemy ->
getDistanceBetween(bot, enemy)))
            .collect(Collectors.toList())

        GameObject[] foodList = gameState.getGameObjects()
            .stream().filter(item -> item.getGameObjects() ==
ObjectTypes.TORPEDO_SALVO OR
                                item.getGameObjects() ==
ObjectTypes.TELEPORTER OR
```

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

```
        item.getGameObjects() ==  
ObjectTypes.SUPERNOVA_BOMB)  
  
                .sorted(Comparator.comparing(item ->  
getDistanceBetween(bot, item)))  
  
                .collect(Collectors.toList())  
  
  
        GameObject[] obstacleList = gameState.getGameObjects()  
  
                .stream().filter(item -> item.getGameObjects() ==  
ObjectTypes.GAS_CLOUD OR  
  
                        item.getGameObjects() ==  
ObjectTypes.ASTERIOD_FIELD)  
  
                .sorted(Comparator.comparing(item ->  
getDistanceBetween(bot, item)))  
  
                .collect(Collectors.toList())  
  
  
        GameObject[] wormhole = gameState.getGameObjects()  
  
                .stream().filter(item -> item.getGameObjects() ==  
ObjectTypes.WORMHOLE)  
  
                .sorted(Comparator.comparing(item ->  
getDistanceBetween(bot, item)))  
  
                .collect(Collectors.toList())  
  
/*  
 * Secara default akan terus mencari makanan terdekat  
 * jika tidak ada kondisi apapun yang terpenuhi  
 */  
  
playerAction.heading = getHeadingBetween(foodList[0])  
playerAction.action = PlayerActions.FORWARD  
  
  
        //Jika food terdekat terdapat musuh dengan size yang lebih  
besar  
  
        IF enemies[0].size > bot.getSize AND getDistanceBetween(bot,  
enemies[0]) < 100 THEN  
  
                playerAction.heading = getHeadingBetween(foodList[0]+180  
MOD 360)  
  
                playerAction.action = PlayerActions.FORWARD  
  
        END IF
```

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

```
//Jika food terdekat terhalang oleh obstacle
IF getDistanceBetween(obstacleList[0], bot) < 100 THEN
    int i=0
    WHILE getDistanceBetween(foodList[i], obstacleList[0]) <
obstacleList[0].getSize() DO
        i++
    END WHILE
    playerAction.heading = getHeadingBetween(foodList[i])
    playerAction.action = PlayerActions.FORWARD
END IF

/*
 * Mekanisme penembakan dengan syarat bot size 30 atau
 * lebih dan jarak antara bot tidak terlalu jauh agar
 * akurasi lebih tepat
 */
IF bot.size >= 30 AND (getDistanceBetween(bot, enemies[0]) <
125 + bot.getSize() + enemies[0].getSize()) THEN
    playerAction.heading = getHeadingBetween(enemies[0])
    playerAction.action = PlayerAction.FIRETORPEDOES
END IF

//Mekanisme pertahanan jika ada projektil yang datang
IF projectileList.size() > 0 AND
getDistanceBetween(projectileList[0], bot) < 100 AND bot.getSize() > 30
THEN
    playerAction.heading = getHeadingBetween(projectileList[0])
    playerAction.action = PlayerActions.ACTIVATESHIELD
END IF

//Manuver untuk mencegah kapal keluar zona/map
DOUBLE distanceFromCenter = getDistanceBetween(bot,
gameState.world)
IF distanceFromCenter + (1.5 * bot.size) >
gameState.world.radius THEN
```

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

```
        playerAction.heading = getHeadingBetween(gameState.world)

        playerAction.action = PlayerActions.FORWARD

    END IF

    IF gameState.world.getCurrentTick() % 100 == 0 THEN

        countTeleport++

    END IF

    GameObject[] availableTargets = gameState

        .getPlayerGameObjects()

        .stream().filter(enemy -> enemy.id != bot.id AND enemy.size
< bot.size)

            .sorted(Comparator.comparing(enemy ->
getDistanceBetween(this.bot, enemy), Comparator.reverseOrder()))

            .collect(Collectors.toList());

    PRINT("target")

    PRINT(target)

    //Mekanisme penembakan projektil teleporter

    IF (NOT availableTargets.isEmpty()) AND bot.size > 50 +
availableTargets[0].getSize() AND (NOT isTeleport) AND countTeleport > 0
THEN

        target = availableTargets[0]

        isTeleport = TRUE

        playerAction.heading = getHeadingBetween(target)

        playerAction.action = FIRETELEPORT

        countTeleport--

        teleportTick = gameState.world.getCurrentTick +
(INT)getDistanceBetween(target, bot) / 20

        PRINT("Berhasil menembakkan teleport")

    END IF

    //Mekanisme teleportasi ke projektil teleporter

    IF isTeleport AND gameState.world.getCurrentTick() ==
teleportTick THEN

        playerAction.action = PlayerActions.TELEPORT
```

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

```
        playerAction.heading = getHeadingBetween(target)

        isTeleport = false

        target = NULL

        teleportTick = 0

        PRINT("Berhasil terlaport")

    END IF

    END IF

    thisPlayerAction = PlayerAction

END PROCEDURE

PUBLIC FUNCTION GameState getGameState()

    RETURN gameState

END FUNCTION

PUBLIC PROCEDURE setGameState(GameState newGameState)

    gameState = newGameState

    updateSelfState()

END PROCEDURE

PRIVATE PROCEDURE updateSelfState()

    GameObject[] optionalBot = gameState.getPlayerGameObjects().stream()

        .filter(gameObject ->
gameObject.id.equals(bot.id)).findAny()

    optionalBot.ifPresent(conBot -> bot = conBot)

END PROCEDURE

PRIVATE FUNCTION DOUBLE getDistanceBetween(GameObject object1,
GameObject object2)

    int triangleX = Math.abs(object1.getPosition().x -
object2.getPosition().x)

    int triangleY = Math.abs(object1.getPosition().y -
object2.getPosition().y)

    RETURN Math.sqrt(triangleX * triangleX + triangleY * triangleY)
```

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

```
END FUNCTION

PRIVATE FUNCTION DOUBLE getDistanceBetween(GameObject object1, World
object2)
    int triangleX = Math.abs(object1.getPosition().x -
object2.getPosition().x)
    int triangleY = Math.abs(object1.getPosition().y -
object2.getPosition().y)
    RETURN Math.sqrt(triangleX * triangleX + triangleY * triangleY)

END FUNCTION

PRIVATE FUNCTION INT getHeadingBetween(GameObject otherObject)
    int direction = toDegrees(Math.atan2(otherObject.getPosition().y =
bot.getPosition().y),
        otherObject.getPosition().x - bot.getPosition().x))
    RETURN (direction + 360) % 360

END FUNCTION

PRIVATE FUNCTION toDegrees(DOUBLE v)
    RETURN (int) (v * (180 / Math.PI))

END FUNCTION

END CLASS
```

4.2. Penjelasan Struktur Data Bot Permainan Galaxio

Struktur data pada permainan Galaxio berbasis *class*, yang berarti data dan perilaku yang terkait didefinisikan sebagai class dan objek-objek yang diinstansiasi dari class tersebut. Secara garis besar, *class* tersebut dapat dibagi menjadi 3 kategori utama, yaitu Enums yang berisi objek-objek konstan yang terdapat pada permainan, Models yang mengandung semua objek yang bisa diinstansiasi ke permainan, Services yang merupakan tempat utama logika *bot* untuk melakukan keputusan berdasarkan analisis *state* permainan dengan menginstansiasi *class* yang berada pada kategori Models. Selain ketiga kategori tersebut, terdapat *class* Main.java yang merupakan program utama untuk menjalankan *bot* sehingga dapat terhubung ke *game engine* dan *logger* melalui *hub*.

Berikut penjelasan lebih lanjut setiap *class* yang ada pada *bot* permainan Galaxio:

1. Kategori Enums

Kategori ini berisi objek-objek yang anggotanya merupakan konstanta yang dapat digunakan sepanjang permainan. Kategori ini hanya memiliki dua anggota, yaitu `ObjectTypes.java` dan `PlayerActions.java`. Secara garis besar, kategori ini isinya adalah enumerasi untuk objek yang diinisialisasi. Berikut penjelasan lebih lanjutnya:

i. `ObjectTypes.java`

Berisi enumerasi objek-objek konstan yang terdapat pada permainan dan dapat digunakan oleh `class bot`. Enumerasi tersebut adalah `PLAYER`, `FOOD`, `WORMHOLE`, `GAS_CLOUD`, `ASTEROID_FIELD`, `TORPEDO_SALVO`, `SUPERFOOD`, `SUPERNOVA_PICKUP`, `SUPERNOVA_BOMB`, `TELEPORTER`, dan `SHIELD`. Selain berisi enumerasi, `class` ini juga memiliki atribut dan method yang penjelasannya adalah sebagai berikut:

- Atribut

Atribut	Deskripsi
<code>public final Integer value</code>	Menandakan value pada objek

- Methods

Methods	Deskripsi
<code>ObjectTypes(Integer value)</code>	Method yang mengatur tipe objek menjadi value tertentu
<code>public static ObjectTypes valueOf(Integer value)</code>	mereturn tipe objek dengan value tertentu

ii. `PlayerActions.java`

Berisi enumerasi aksi-aksi konstan pada permainan yang dapat dipanggil oleh `bot` untuk melakukan aksi. Enumerasi tersebut adalah `FORWARD`, `STOP`, `STARTAFTERBURNER`, `STOPAFTERBURNER`, `FIRETORPEDOES`, `FIRESUPERNOVA`, `DETONATESUPERNOVA`, `FIRETELEPORT`, `TELEPORT`, dan `ACTIVATESHIELD`. Selain berisi enumerasi, `class` ini juga memiliki atribut dan method yang penjelasannya adalah sebagai berikut:

- Atribut

Atribut	Deskripsi
<code>public final Integer value</code>	Menandakan value pada aksi

- Methods

Methods	Deskripsi
private PlayerActions(Integer value)	Method yang mengatur tipe aksi menjadi value tertentu

2. Kategori Models

Secara garis besar, kategori Models berisi semua komponen yang dapat diinstansiasi oleh *class* lain. Berikut adalah rincian dari kategori Models:

i. GameObject.java

GameObject.java merupakan *class* yang memiliki atribut dan method untuk informasi-informasi yang terkandung dalam suatu objek permainan. Penjelasan atribut dan method-nya adalah sebagai berikut:

- Atribut

Atribut	Deskripsi
public UUID id	Representasi dari id objek
public Integer size	Representasi dari ukuran objek
public Integer speed	Representasi dari kecepatan objek setiap <i>tick</i>
public Integer currentHeading	Representasi arah hadapan objek
public Position position	Representasi posisi titik objek
public ObjectTypes gameObjectType	Representasi tipe objek berdasarkan enumerasi objek konstan

- Methods

Methods	Deskripsi
public GameObject(UUID id, Integer size, Integer speed, Integer currentHeading, Position position, ObjectTypes gameObjectType)	Method yang merupakan konstruktor/pengatur setiap atribut objek <i>game</i>

public UUID getId()	Method yang mereturn id objek
public void setId(UUID id)	Method yang mengatur id objek menjadi suatu id tertentu berupa tipe data UUID
public int getSize()	Method yang mereturn ukuran dari objek berupa integer
public void setSize(int size)	Method yang mengatur ukuran dari objek berdasarkan parameter integer
public int getSpeed()	Method yang mereturn kecepatan dari objek
public void setSpeed(int speed)	Method yang mengatur kecepatan objek
public Position getPosition()	Method yang mereturn posisi dari objek pada <i>game</i> berupa titik koordinat
public void setPosition(Position position)	Method yang mengatur posisi objek <i>game</i> pada koordinat tertentu
public ObjectTypes getGameObjectType()	Method yang mereturn tipe objek untuk suatu objek tertentu
public void setGameObjectType(ObjectTypes gameObjectType)	Method yang mengatur tipe objek dari suatu objek menjadi tipe objek tertentu
public static GameObject FromStateList(UUID id, List<Integer> stateList)	Method yang mereturn objek permainan yang sesuai dengan parameter id dan list di stateList

ii. GameState.java

GameState.java merupakan *class* yang memiliki atribut dan method untuk mengolah informasi-informasi yang terkandung dalam suatu *state* tertentu di permainan. Penjelasan atribut dan method-nya adalah sebagai berikut:

- Atribut

Atribut	Deskripsi
public World world	Representasi informasi <i>class</i>

	world yang terdapat di satu <i>state</i>
public List<GameObject> gameObjects	Representasi informasi list objek pada <i>game</i> di satu <i>state</i>
public List<GameObject> playerGameObjects	Representasi informasi list player pada <i>game</i> di satu <i>state</i>

- Methods

Methods	Deskripsi
public GameState()	Method yang merupakan konstruktor <i>class</i> GameState
public GameState(World world , List<GameObject> gameObjects, List<GameObject> playerGameObjects)	Method yang merupakan konstruktor <i>class</i> GameState dengan atribut sesuai dengan parameter masukan
public World getWorld()	Method yang mereturn informasi <i>class</i> world pada satu <i>state</i> tertentu
public void setWorld(World world)	Method yang mengatur atribut world untuk satu <i>state</i> tertentu
public List<GameObject> getGameObjects()	Method yang mereturn list objek pada <i>game</i> untuk satu <i>state</i> tertentu
public void setGameObjects(List<GameObje ct> gameObjects)	Method yang mengatur list objek pada <i>game</i> untuk satu <i>state</i> tertentu
public List<GameObject> getPlayerGameObjects()	Method yang mereturn list player pada <i>game</i> untuk satu <i>state</i> tertentu
public void setPlayerGameObjects(List<Ga meObject> playerGameObjects)	Method yang mengatur list player pada <i>game</i> untuk satu <i>state</i> tertentu

iii. GameStateDto.java

GameStateDto.java merupakan *class* yang memiliki atribut dan method untuk mengolah informasi-informasi yang terkandung dalam suatu *state* tertentu

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

yang datanya akan *di-transfer* ke *hub* dan dikirim ke *log* permainan. Penjelasan atribut dan method-nya adalah sebagai berikut:

- Atribut

Atribut	Deskripsi
private World world	Representasi informasi <i>class</i> <i>World</i> yang terdapat di satu <i>state</i>
private Map<String, List<Integer>> gameObjects	Representasi informasi objek permainan yang terdapat di satu <i>state</i> tertentu
private Map<String, List<Integer>> playerObjects	Representasi informasi <i>player</i> permainan yang terdapat di satu <i>state</i> tertentu

- Methods

Methods	Deskripsi
public Models.World getWorld()	Method yang mereturn atribut <i>class</i> <i>World</i> berdasarkan <i>state</i> permainan tertentu
public void setWorld(Models.World world)	Method yang mengatur <i>class</i> <i>World</i> berdasarkan <i>state</i> permainan tertentu
public Map<String, List<Integer>> getGameObjects()	Method yang mereturn objek permainan berdasarkan <i>state</i> tertentu
public void setGameObjects(Map<String, List<Integer>> gameObjects)	Method yang mengatur objek permainan berdasarkan <i>state</i> tertentu
public Map<String, List<Integer>> getPlayerObjects()	Method yang mereturn <i>player</i> permainan berdasarkan <i>state</i> tertentu
public void setPlayerObjects(Map<String, List<Integer>> playerObjects)	Method yang mereturn <i>player</i> permainan berdasarkan <i>state</i> tertentu

iv. PlayerAction.java

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

PlayerAction.java merupakan *class* yang memiliki atribut dan method untuk mengolah informasi-informasi yang terkandung dalam suatu aksi *bot* tertentu di permainan. Penjelasan atribut dan method-nya adalah sebagai berikut:

- Atribut

Atribut	Deskripsi
public UUID playerId	Representasi informasi id dari player
public PlayerActions action	Representasi informasi aksi dari player
public int heading	Representasi informasi arah aksi dari player

- Methods

Methods	Deskripsi
public UUID getPlayerId()	Method yang mereturn id dari player
public void setId(UUID playerId)	Method yang mengatur id dari player berdasarkan parameter id
public PlayerActions getAction()	Method yang mereturn aksi dari player
public void setAction(PlayerActions action)	Method yang mengatur aksi player berdasarkan parameter aksi
public int getHeading()	Method yang mereturn heading/arrah aksi player berupa integer, yaitu sumbu
public void setHeading(int heading)	Method yang mengatur heading player saat melakukan aksi tertentu dengan parameter berupa integer

v. Position.java

Position.java merupakan *class* yang memiliki atribut dan method untuk mengolah informasi-informasi yang terkandung dalam suatu posisi titik tertentu di permainan. Penjelasan atribut dan method-nya adalah sebagai berikut:

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

- Atribut

Atribut	Deskripsi
public int x	Representasi informasi posisi titik di sumbu x
public int y	Representasi informasi posisi titik di sumbu y

- Methods

Methods	Deskripsi
public Position()	Method yang merupakan konstruktor <i>class</i> Position
public Position(int x, int y)	Method yang merupakan konstruktor <i>class</i> Position berdasarkan atribut parameter
public int getX()	Method yang mereturn posisi titik di sumbu x
public void setX(int x)	Method yang mengatur posisi titik di sumbu x dengan parameter tertentu
public int getY()	Method yang mereturn posisi titik di sumbu y
public void setY(int y)	Method yang mengatur posisi titik di sumbu y dengan parameter tertentu

vi. World.java

World.java merupakan *class* yang memiliki atribut dan method untuk mengolah informasi-informasi yang terkandung dalam peta di satu permainan. Penjelasan atribut dan method-nya adalah sebagai berikut:

- Atribut

Atribut	Deskripsi
public Position centerPoint	Representasi informasi posisi titik tengah <i>map</i> berupa <i>class</i> Position

public Integer radius	Representasi jari-jari peta pada saat inisialisasi awal permainan dimulai
public Integer currentTick	Representasi nilai <i>tick</i> untuk ronde yang dimaksud

- Methods

Methods	Deskripsi
public Position getCenterPoint()	Method yang mereturn posisi titik tengah peta berupa <i>class</i> Position
public void setCenterPoint(Position centerPoint)	Method yang mengatur posisi titik tengah peta dengan parameter berupa <i>class</i> Position
public Integer getRadius()	Method yang mereturn jari-jari peta pada saat inisialisasi awal permainan
public void setRadius(Integer radius)	Method yang mengatur jari-jari peta pada saat inisialisasi awal permainan
public Integer getCurrentTick()	Method yang mereturn <i>tick</i> untuk ronde yang dimaksud
public void setCurrentTick(Integer currentTick)	Method yang mengatur <i>tick</i> sesuai dengan ronde yang dimaksud

3. Kategori Services

Pada umumnya, kategori Services berisi *class* yang akan menjadi tempat logika *bot* berada. Algoritma yang diterapkan perlu analisis lebih lanjut terkait informasi pergerakan lawan dan situasi kondisi permainan. Oleh karena itu, *class* ini memerlukan *class* lain yang berada di kategori Models dan Enums untuk memperoleh informasi terkait. Berikut adalah rincian *class* yang berada pada kategori Services:

i. BotService.java

BotService.java merupakan *class* yang menjadi komponen utama logika milik *bot*. Di dalam *class* ini, dapat diimplementasikan algoritma *greedy*. *Class* ini memiliki atribut dan method untuk mengolah informasi-informasi yang

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

terkandung dalam *bot* di permainan. Penjelasan atribut dan method-nya adalah sebagai berikut:

- Atribut

Atribut	Deskripsi
private GameObject bot	Representasi informasi <i>bot</i> yang menjadikan <i>bot</i> sebagai objek permainan dengan <i>class</i> GameObject
private PlayerAction playerAction	Representasi informasi aksi yang akan dilakukan <i>bot</i> pada saat <i>tick</i> tertentu sebagai <i>class</i> PlayerAction
private GameState gameState	Representasi informasi suatu <i>state</i> pada permainan untuk memperoleh setiap informasi yang terkandung di gameState berupa <i>class</i> GameState
private boolean isTeleport	Representasi informasi <i>bot</i> telah menembakkan <i>teleporter</i>
private int teleportTick	Representasi informasi <i>tick</i> tertentu agar <i>bot</i> melakukan aksi <i>teleport</i>
private int countTeleport	Representasi informasi jumlah <i>teleporter</i> yang sedang dimiliki oleh <i>bot</i>
private GameObject target	Representasi informasi target yang dikunci oleh <i>bot</i> untuk menembak dan melakukan <i>teleport</i>

- Methods

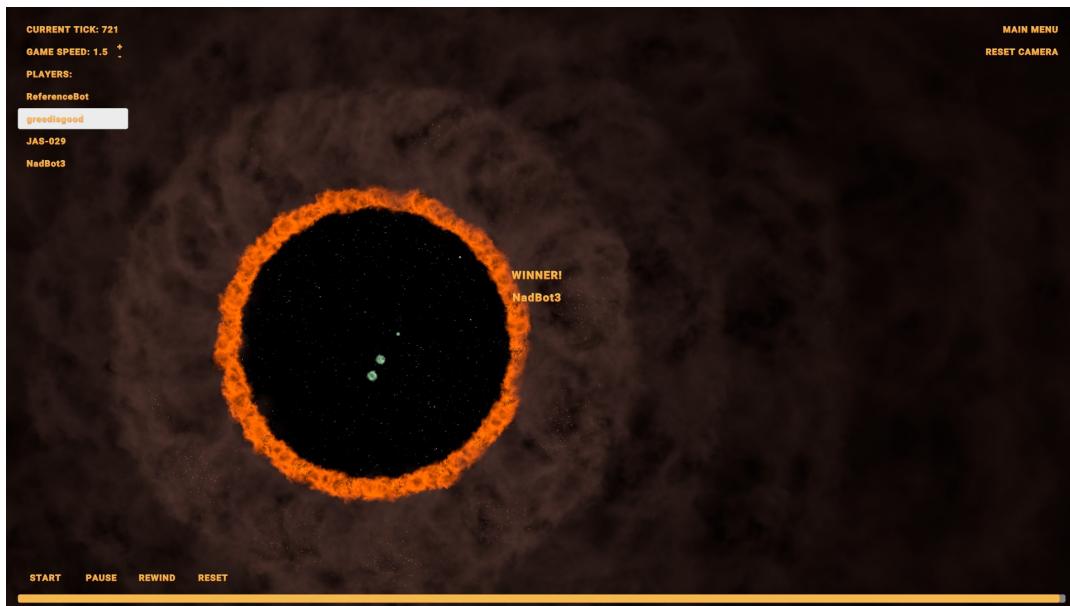
Methods	Deskripsi
public BotService()	Method yang merupakan konstruktor dari <i>bot</i>
public GameObject getBot()	Method yang mereturn <i>bot</i> berupa <i>class</i> GameObject

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023

public void setBot(GameObject bot)	Method yang mengatur atribut objek <i>bot</i> menjadi <i>bot</i> dengan atribut tertentu
public PlayerAction getPlayerAction()	Method yang mereturn aksi <i>bot</i> yang sedang dilakukan berupa <i>class PlayerAction</i>
public void setPlayerAction(PlayerAction playerAction)	Method yang mengatur aksi <i>bot</i> menjadi aksi sesuai dengan parameter <i>class PlayerAction</i> tertentu
public void computeNextPlayerAction(PlayerAction playerAction)	Method yang berisi logika <i>bot</i> sehingga aksi dan <i>heading</i> <i>bot</i> berubah sesuai dengan algoritma <i>greedy</i> yang telah dibuat
public GameState getGameState()	Method yang mereturn <i>state</i> permainan yang sedang berlangsung di ronde tertentu
public void setGameState(GameState gameState)	Method yang mengatur <i>state</i> permainan yang sedang berlangsung
private void updateSelfState()	Method yang memperbarui keadaan <i>bot</i>
private double getDistanceBetween(GameObject object1, GameObject object2)	Method yang mereturn jarak antara satu objek ke objek lain
private double getDistanceBetween(GameObject object1, World object2)	Method yang mereturn jarak antara satu objek terhadap peta
private int getHeadingBetween(GameObject otherObject)	Method yang mereturn <i>heading</i> atau arah <i>bot</i> terhadap objek permainan berupa sumbu
private int getHeadingBetween(World otherObject)	Method yang mereturn <i>heading</i> atau arah <i>bot</i> terhadap peta permainan berupa sumbu
private int toDegrees(double v)	Method yang mereturn pengubahan parameter <i>v</i> menjadi derajat

4.3. Analisis Desain Solusi Algoritma Greedy Pada Setiap Pengujian

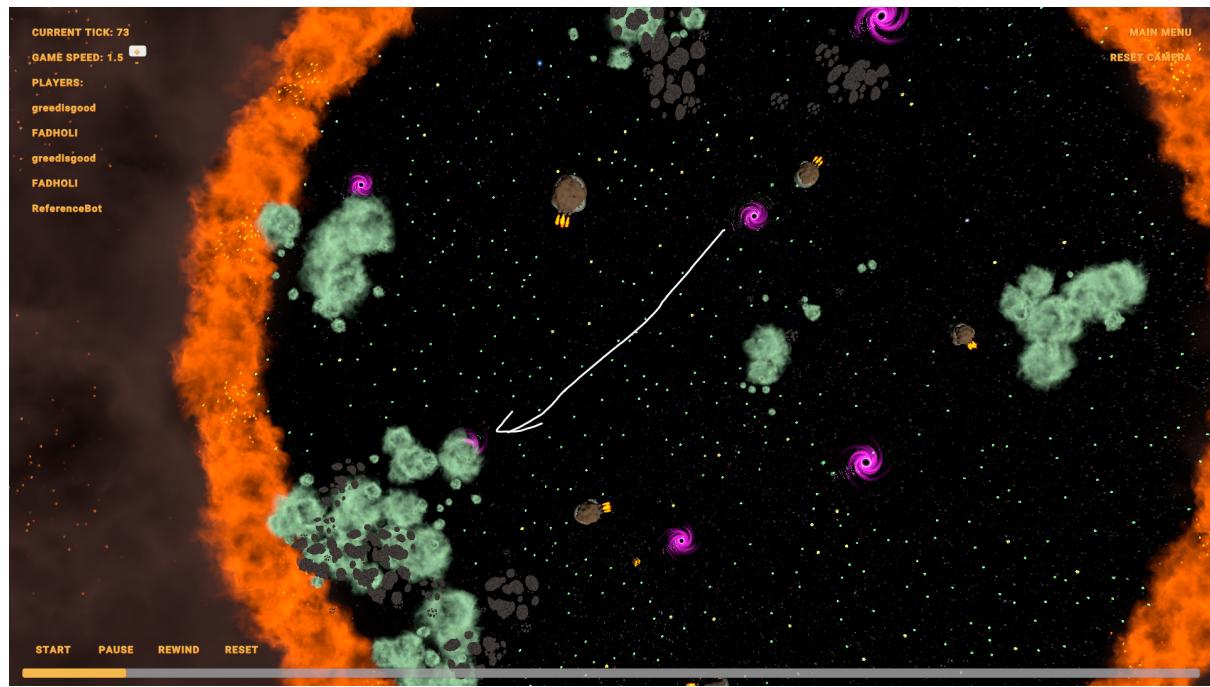
Algoritma greedy yang penulis implementasikan pada permainan Galaxio cukup optimal. Algoritma yang dipilih mampu mencari *food* yang tidak dalam zona berbahaya seperti *food* di dekat musuh atau *food* berada atau di seberang suatu *obstacle* (*Asteroid* ataupun *gas cloud*), Namun ada beberapa kendala yang pertama seperti Gambar 4.3.1 ini, karena *food* sudah sangat sedikit dan algoritma penulis memprioritaskan untuk menjatuhkan musuh karena size musuh lebih kecil dan jarak musuh sangat dekat maka bot pun malah mengejar musuh tersebut dan melewati batas map dan bot tersebut mengalami kekalahan.



Gambar 4.3.1

Lalu jika bot mengenai wormhole yang menjebak, seperti gambar 4.2.2 bot penulis yang ada dekat wormhole diawal tanda panah tidak mengetahui bahwa *wormhole* tersebut terhubung dengan *wormhole* yang berada tepat di *gas cloud* sehingga bot penulis mengalami pengurangan size yang cukup signifikan. Penulis tidak dapat membuat kondisi jika *wormhole* terdekat terhubung dengan *wormhole trap* dikarenakan koneksi antar *wormhole* pada game itu acak.

Tugas Besar I
IF2211 Strategi Algoritma
2022/ 2023



Gambar 4.3.2

Selanjutnya, seperti pada gambar 4.3.3 jika food aman terdekat terlalu jauh maka bot akan tetap menembus obstacle dan mendapatkan pengurangan size.



Sisanya, perilaku bot sesuai dengan ekspektasi dari penulis, dan kemenangan dan kekalahan yang didapat di tiap game yang dijalani juga faktor keberuntungan mempengaruhi, seperti pada saat spawn food lebih banyak bertebaran di sekitaran musuh dan pada bot penulis dikelilingi oleh obstacle.

BAB V

KESIMPULAN, SARAN DAN REFLEKSI

5.1. Kesimpulan

Dalam tugas besar mata kuliah Strategi Algoritma, membuat penulis untuk menciptakan algoritma greedy yang dapat diimplementasikan pada permainan Galaxio. Algoritma greedy yang dipilih adalah memperbesar ukuran bot sebesar-besarnya dengan mencari makan sebanyak-banyaknya.

5.2. Saran

Saran dari penulis seperti memahami cara kerja dari Galaxio agar dapat mudah dalam mengembangkan performa dari *bot*. Selain itu, dalam penulisan laporan dilakukan lebih detail dan lebih cepat lagi.

5.3. Refleksi

Setelah menyelesaikan tugas besar Strategi Algoritma, penulis belajar banyak tentang algoritma greedy. Terdapat beberapa hal yang penulis refleksikan selain pembelajaran algoritma greedy, yaitu lebih mengerti dengan pemrograman berorientasi objek menggunakan bahasa Java.

DAFTAR PUSTAKA

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-B
ag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf)

[https://github.com/EntelectChallenge/2021-Galaxio/blob/develop/game-engine/game-rules.m
d#command-structure](https://github.com/EntelectChallenge/2021-Galaxio/blob/develop/game-engine/game-rules.md#command-structure)

LAMPIRAN

Link Github: https://github.com/AlphaThrone/Tubes1_greedisgood.git

Link Youtube: https://www.youtube.com/watch?v=BolJQn_6tKc