

Tugas Besar 1 IF2211 Strategi Algoritma

Semester II Tahun 2022/2023

Pemanfaatan Algoritma Greedy dalam Aplikasi Permainan “Galaxio”

Disusun Oleh:

Manuella Ivana Uli Sianipar 13521051

Asyifa Nurul Shafira 13521125

Ferindya Aulia Berlianty 13521161



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

DAFTAR ISI

DAFTAR ISI	1
DAFTAR GAMBAR	2
BAB I DESKRIPSI TUGAS	3
BAB II LANDASAN TEORI	6
BAB III APLIKASI STRATEGI GREEDY	10
3.1 Pemetaan Elemen/Komponen Algoritma Greedy pada Bot Permainan Galaxio	10
3.1.1 Pemetaan Elemen/Komponen Algoritma Greedy pada Permasalahan General Bot	10
3.1.2 Pemetaan Elemen/Komponen Algoritma Greedy pada Permasalahan Memilih Arah Tujuan Bot	11
3.1.2.1 Pemetaan Elemen/Komponen Algoritma Greedy pada Permasalahan Menghindari Objek yang Merugikan	11
3.1.2.2 Pemetaan Elemen/Komponen Algoritma Greedy pada Permasalahan Respons Terhadap Musuh	12
3.1.2.3 Pemetaan Elemen/Komponen Algoritma Greedy pada Permasalahan Terlalu Dekat ke Ujung Map	13
3.2 Eksplorasi alternatif solusi greedy yang mungkin dipilih dalam persoalan Galaxio	14
3.3 Analisis efisiensi dan efektivitas dari kumpulan alternatif solusi greedy yang dirumuskan	15
3.3.1 Analisis terhadap greedy dengan prioritas merata	15
3.3.2 Analisis terhadap greedy dengan prioritas menghindari objek pengganggu	16
3.3.3 Analisis terhadap greedy dengan prioritas menghindari musuh	16
3.3.4 Analisis terhadap greedy dengan prioritas menghindari ujung asap	16
3.4 Strategi greedy yang dipilih	16
BAB IV IMPLEMENTASI DAN PENGUJIAN	17
4.1 Implementasi algoritma greedy	17
4.1.1 Implementasi kelas computeNextPlayerAction	17
4.3 Analisis dari desain solusi algoritma greedy	21
BAB V KESIMPULAN, SARAN, KOMENTAR DAN REFLEKSI	22
5.1 Kesimpulan	22
5.2 Saran	22
5.3 Komentar	22
5.4 Refleksi	23
DAFTAR PUSTAKA	24
LINK REPOSITORI	24
LINK VIDEO	24

DAFTAR GAMBAR

Gambar 4.1.1 Flowchart dari solusi algoritma Greedy	20
Gambar 4.2.1 Bot berhasil menang dalam permainan	21
Gambar 4.2.2 Bot terperangkap di antara asteroid dan gas cloud	21

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. Pada tugas besar pertama Strategi Algoritma ini, gunakanlah sebuah game engine yang mengimplementasikan permainan Galaxio. Tugas mahasiswa adalah mengimplementasikan bot kapal dalam permainan Galaxio dengan menggunakan strategi greedy untuk memenangkan permainan.

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.

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 slavo 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.
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:

- FORWARD : *Command* ini akan menggerakkan kapal Anda ke arah yang disediakan dalam derajat.
- STOP : *Command* ini menghentikan pergerakan kapal Anda pada saat detak permainan ini sampai perintah pergerakan lain dikeluarkan.
- START_AFTERBURNER : *Command* ini mengaktifkan afterburner kapal Anda.
- STOP_AFTERBURNER : *Command* ini menonaktifkan afterburner kapal Anda.
- FIRE_TORPEDOES : *Command* ini menghabiskan 1 muatan salvo dan 5 ukuran untuk mengirimkan salvo torpedo di pos yang diberikan.
- FIRE_SUPERNOVA : *Command* ini berperan dalam pengambilan supernova untuk mengirimkan supernova dalam judul yang diberikan.
- DETONATE_SUPERNOVA : *Command* ini akan meledakkan supernova yang ada.
- FIRE_TELEPORTER : *Command* ini akan menghabiskan 1 biaya teleportasi dan 20 ukuran untuk mengirim teleporter di pos yang diberikan.
- TELEPORTUSE_SHIELD : *Command* ini akan memindahkan Anda ke teleporter yang sudah ada, jika ada.

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.

BAB II

LANDASAN TEORI

2.1 Gambaran Algoritma *Greedy* secara Umum

Algoritma *greedy* merupakan sebuah algoritma yang mengimplementasikan konsep “*greedy*” dalam pendefinisian solusinya. Algoritma greedy biasanya digunakan untuk mencari solusi dari persoalan optimasi (*optimization problem*) untuk memaksimumkan atau meminimumkan suatu parameter. Algoritma greedy dibentuk dengan cara pemecahan masalah (*problem solving*) dan pembentukan solusi secara langkah per langkah (*step by step*). Pada setiap langkah tersebut, terdapat banyak langkah yang dapat dievaluasi. Pada algoritma *greedy*, pembuat program harus dapat menentukan keputusan terbaik pada setiap langkahnya. Namun, di dalam aturan algoritma *greedy* tidak diperbolehkan adanya *backtracking* (tidak dapat mundur ke solusi sebelumnya untuk menentukan solusi langkah sekarang). Oleh karena itu, diharapkan pembuat program memilih solusi yang merupakan solusi optimum lokal (*local optimum*) pada setiap langkahnya. Hal tersebut bertujuan agar langkah-langkah optimum lokal tersebut mengarah pada solusi dengan optimum global (*global optimum*) yang cakupannya lebih luas daripada optimum lokal.

Sebuah persoalan dapat diselesaikan dengan menggunakan algoritma *greedy* apabila persoalan tersebut memiliki dua sifat berikut:

- Solusi optimal dari sebuah persoalan dapat ditentukan dari solusi optimal sub persoalan tersebut.
- Pada setiap persoalan, terdapat suatu langkah yang dapat dilakukan dimana langkah tersebut menghasilkan solusi optimal pada sub persoalan tersebut. Langkah ini disebut juga sebagai *greedy choice*.

Terdapat beberapa elemen/komponen yang perlu didefinisikan di dalam algoritma greedy. Beberapa elemen/komponen algoritma greedy tersebut adalah:

- Himpunan Kandidat (C): Berisi kandidat yang akan dipilih pada setiap langkah.
- Himpunan Solusi (S): Berisi kandidat yang sudah terpilih sebagai solusi.
- Fungsi solusi (solution function): Menentukan apakah himpunan solusi yang dipilih sudah memberikan solusi.

- Fungsi seleksi (selection function): Memilih kandidat berdasarkan strategi greedy tertentu. Fungsi ini memiliki sifat heuristik yaitu fungsi dirancang untuk mencari solusi optimum dengan mengabaikan apakah fungsi tersebut terbukti paling optimum secara matematis.
- Fungsi kelayakan (feasible): Memeriksa kelayakan apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi
- Fungsi objektif (objective function): Memaksimumkan atau meminimumkan suatu parameter pada suatu persoalan.

Skema umum algoritma greedy menggunakan pseudocode dengan pendefinisian elemen/komponennya sebagai berikut:

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

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
```

Gambar 2.1 Pseudocode algoritma greedy

Pada akhir tiap iterasi, solusi yang terbentuk adalah optimum lokal, dan pada akhir loop while-do akan ditemukan optimum global, namun optimum global yang ditemukan ini belum tentu merupakan solusi yang terbaik karena algoritma greedy tidak melakukan operasi secara menyeluruh kepada semua kemungkinan yang ada.

2.2 Garis Besar Cara Kerja Bot Permainan Galaxio

Ada beberapa komponen yang perlu diketahui untuk memahami cara kerja game ini, diantaranya yaitu:

- Engine

Engine merupakan komponen yang berperan dalam mengimplementasikan logic dan rules game.

- Runner

Runner merupakan komponen yang berperan dalam menggelar sebuah match serta menghubungkan bot dengan engine.

- Logger

Logger merupakan komponen yang berperan untuk mencatat log permainan sehingga kita dapat mengetahui hasil permainan. Log juga akan digunakan sebagai input dari visualizer.

Garis besar cara kerja program game Galaxio adalah sebagai berikut:

1. Runner –saat dijalankan– akan meng-*host* sebuah *match* pada sebuah hostname tertentu. Untuk koneksi lokal, runner akan meng-host pada localhost:5000.
2. Engine kemudian dijalankan untuk melakukan koneksi dengan runner. Setelah terkoneksi, Engine akan menunggu sampai bot-bot pemain terkoneksi ke runner.
3. Logger juga melakukan hal yang sama, yaitu melakukan koneksi dengan runner.
4. Pada titik ini, dibutuhkan beberapa bot untuk melakukan koneksi dengan runner agar *match* dapat dimulai. Jumlah bot dalam satu pertandingan didefinisikan pada atribut BotCount yang dimiliki file JSON "appsettings.json". File tersebut terdapat di dalam folder "runner-publish" dan "engine-publish".
5. Permainan akan dimulai saat jumlah bot yang terkoneksi sudah sesuai dengan konfigurasi.
6. Bot yang terkoneksi akan mendengarkan event-event dari runner. Salah satu event yang paling penting adalah RecieveGameState karena memberikan status game.
7. Bot juga mengirim event kepada runner yang berisi aksi bot.
8. Permainan akan berlangsung sampai selesai. Setelah selesai, akan terbuat dua file json yang berisi kronologi *match*.

2.3 Implementasi Algoritma *Greedy* ke dalam Bot Permainan Galaxio

Pada mulanya, kita perlu untuk mengunduh versi terbaru dari starter pack tersebut, yakni pada situs <https://github.com/EntelectChallenge/2021-Galaxio/releases/tag/2021.3.2>. Dalam membuat bot,

diperlukan Java (minimal Java 11), IntelliJ IDEA sebagai code editor, NodeJS, serta .Net Core 3.1. Strategi greedy yang diimplementasikan dikaitkan dengan fungsi objektif permainan Galaxio. Parameter yang menjadi penentu kemenangan pemain adalah waktu. Pemain yang paling lama bertahan adalah pemenangnya. Algoritma greedy dalam bot dapat diimplementasikan secara modular di setiap kelas tertentu. Implementasi digunakan dengan bahasa Java dan berparadigma object-oriented, sehingga dibentuk kelas-kelas tertentu dalam pelaksanaannya.

2.4 Garis Besar Game Engine Permainan Galaxio

Game engine merupakan suatu framework perangkat lunak yang didesain sebagai kumpulan tools dan features yang digunakan untuk membantu development dari suatu game.

Dalam menjalankan program ini, terdapat file “main.bat”. Pemain dengan sistem operasi berbasis windows dapat melakukan double-click dalam menjalankan permainannya, sedangkan pemain dengan sistem operasi Linux/Mac dapat menjalankan command “make main”. File “BotService.java” yang terdapat di dalam direktori “starter-bots” dan file “appsettings.json” yang terdapat di dalam direktori “runner-publish” dapat diedit untuk mengatur informasi terkait bot yang dibuat. Pada dasarnya, game-engine dijalankan dengan mode command-line interface (CLI). Untuk mempermudah visualisasi, dapat digunakan visualizer yang terdapat pada link [Tubes1_Glaxrid/visualiser at main · manuellaiv/Tubes1_Glaxrid \(github.com\)](https://github.com/Tubes1_Glaxrid/visualiser)

BAB III

APLIKASI STRATEGI GREEDY

3.1 Pemetaan Elemen/Komponen Algoritma Greedy pada Bot Permainan Galaxio

3.1.1 Pemetaan Elemen/Komponen Algoritma Greedy pada Permasalahan General Bot

Dalam permainan Galaxio, tujuan dari setiap bot pemain yaitu berusaha untuk memenangkan permainan dengan cara mempertahankan kapal pemain paling terakhir untuk hidup. Terdapat banyak cara agar tujuan tersebut dapat tercapai, sebagai contohnya yaitu menghindari objek yang dapat mengurangi ukuran kapal dan memperlambat gerakan, menghindari musuh dan lain-lain.

Nama Elemen/Komponen	Definisi Elemen/Komponen
Himpunan kandidat	Seluruh kemungkinan <i>command</i> yang dapat dilakukan pada suatu <i>gamestate</i> .
Himpunan solusi	Seluruh kemungkinan <i>command</i> agar bot menang.
Fungsi solusi	Melakukan pengecekan apakah permutasi dari <i>command</i> tersebut dapat membuat bot mempertahankan kapal pemain paling terakhir untuk hidup.
Fungsi seleksi	Memilih <i>command</i> berdasarkan data keadaan <i>game state</i> saat itu serta fungsi heuristik tingkat prioritas <i>command</i> yang harus diikuti. Bentuk dari fungsi heuristik tersebut mengikuti fungsi-fungsi seleksi sub bagian dekomposisi untuk setiap kasus pada game state. Tingkat prioritas tersebut bersifat statik selama permainan berlangsung (tidak ada perubahan kondisi pada fungsi heuristik).
Fungsi kelayakan	Memeriksa apakah <i>command</i> yang dituliskan oleh bot merupakan <i>command</i> yang valid.

Fungsi objektif	Mencari <i>command</i> yang membuat bot pemain memenangkan permainan dengan mempertahankan kapal pemain paling terakhir untuk hidup.
-----------------	--

3.1.2 Pemetaan Elemen/Komponen Algoritma Greedy pada Permasalahan Memilih Arah Tujuan Bot

Pemilihan arah tujuan bot merupakan permasalahan utama dalam membangun bot. Arah yang dipilih harus menguntungkan bot. Faktor-faktor yang mempengaruhi keuntungan suatu arah pada permainan Galaxio adalah:

1. Objek-objek yang merugikan seperti Gas Cloud yang mengurangi ukuran kapal dan Asteroid Field yang membuat pergerakan kapal menjadi lebih lambat.
2. Musuh yang dengan ukuran yang lebih kecil harus dikejar, sedangkan musuh dengan ukuran yang lebih besar harus dijauhi.
3. Menghindari ujung map agar ukuran kapal tidak berkurang.

3.1.2.1 Pemetaan Elemen/Komponen Algoritma Greedy pada Permasalahan Menghindari Objek yang Merugikan

Objek-objek yang dikategorikan merugikan pada bot adalah Gas Cloud, Asteroid Field dan Wormhole. Gas Cloud dapat mengurangi ukuran kapal sebesar 1 per tick. Asteroid field mengurangi kecepatan kapal menjadi setengahnya. Wormhole memindahkan kapal menuju wormhole pasangannya, namun karena pasangan wormhole tidak diketahui, lebih tidak berisiko apabila bot menghindari wormhole.

Nama Elemen/Komponen	Definisi Elemen/Komponen
Himpunan kandidat	Arah dalam bentuk integer dari angka 0 sampai 360.
Himpunan solusi	List arah yang memiliki <i>food</i> dan <i>superfood</i> .
Fungsi solusi	Fungsi <code>nearGasCloud</code> , <code>nearAsteroidfield</code> , <code>nearWormhole</code> digunakan untuk mendeteksi apakah bot sedang berada di dekat objek-objek tersebut. Objek-objek tersebut berpotensi

	untuk merugikan bot dari segi ukuran atau kecepatan.
Fungsi seleksi	Memilih <i>food</i> atau <i>superfood</i> terdekat dari bot dengan mengambil elemen pertama dari list karena list sudah diurutkan menaik berdasarkan jarak.
Fungsi kelayakan	Memeriksa apakah <i>command</i> yang dituliskan oleh bot merupakan command yang valid. Daftar <i>command</i> yang valid pada strategi ini adalah bot melakukan <i>command FORWARD</i> atau bot tidak melakukan <i>command FORWARD</i> . Namun, yang membedakan pemetaan elemen/komponen algoritma greedy hanyalah tujuan kapal yang akan mengarah kemana.
Fungsi objektif	Pengurangan/penambahan nilai arah untuk mendapatkan nilai arah yang meminimalisir kerugian akibat objek-objek tersebut memanfaatkan fungsi <i>isLeft</i> untuk mengetahui apakah bot harus diputar lebih ke kiri atau lebih ke kanan. Jika objek penganggu ada di sebelah kiri (<i>isLeft</i> bernilai true) maka nilai arah akan dikurangkan sebesar 70 sementara jika objek ada di sebelah kanan (<i>isLeft</i> bernilai false) maka nilai arah akan ditambahkan sebesar 70.

3.1.2.2 Pemetaan Elemen/Komponen Algoritma Greedy pada Permasalahan Respons Terhadap Musuh

Pada permainan Galaxio, kapal yang lebih besar dapat memakan kapal yang lain. Fungsi *nearestPlayer* digunakan untuk menentukan respons bot terhadap musuh terdekatnya. Pada bot ini ada tiga klasifikasi respons yaitu:

1. Kategori bahaya apabila musuh dekat dengan bot dan ukurannya lebih besar dari bot.
2. Kategori bisa dikejar apabila musuh dekat dengan bot dan ukurannya lebih kecil dari bot.
3. Kategori biasa apabila musuh cukup jauh, tidak berpotensi apa-apa.

Nama Elemen/Komponen	Definisi Elemen/Komponen
Himpunan kandidat	Arah dalam bentuk integer dari angka 0 sampai 360.
Himpunan solusi	List arah yang memiliki <i>food</i> dan <i>superfood</i> .
Fungsi solusi	Fungsi nearestPlayer dapat menentukan apakah musuh terdekat berpotensi berbahaya, dapat dikejar atau tidak berpotensi apa-apa.
Fungsi seleksi	Memilih <i>food</i> atau <i>superfood</i> terdekat dari bot dengan mengambil elemen pertama dari list karena list sudah diurutkan menaik berdasarkan jarak.
Fungsi kelayakan	Memeriksa apakah <i>command</i> yang dituliskan oleh bot merupakan command yang valid. Daftar <i>command</i> yang valid pada strategi ini adalah bot melakukan <i>command FORWARD</i> atau bot tidak melakukan <i>command FORWARD</i> . Namun, yang membedakan pemetaan elemen/komponen algoritma greedy hanyalah tujuan kapal yang akan mengarah kemana.
Fungsi objektif	Apabila musuh terdekat berbahaya, bot akan berputar sejauh 160 derajat ke kiri atau ke kanan tergantung fungsi isLeft yang menentukan musuh berada di posisi kiri atau kanan. Jika musuh dapat dikejar, maka bot akan mengarah ke musuh tersebut.

3.1.2.3 Pemetaan Elemen/Komponen Algoritma Greedy pada Permasalahan Terlalu Dekat ke Ujung Map

Kapal yang keluar dari map akan berkurang ukurannya sebesar 1 per tick. Pengarahan bot ke tengah map harus dilakukan secara berkala karena map akan mengecil terus-menerus.

Nama Elemen/Komponen	Definisi Elemen/Komponen
Himpunan kandidat	Arah dalam bentuk integer dari angka 0 sampai 360.
Himpunan solusi	List arah yang memiliki <i>food</i> dan <i>superfood</i> .
Fungsi solusi	Pengecekan posisi bot apakah terlalu ke ujung atau tidak.
Fungsi seleksi	Memilih <i>food</i> atau <i>superfood</i> terdekat dari bot dengan mengambil elemen pertama dari list karena list sudah diurutkan menaik berdasarkan jarak.
Fungsi kelayakan	Memeriksa apakah <i>command</i> yang dituliskan oleh bot merupakan command yang valid. Daftar <i>command</i> yang valid pada strategi ini adalah bot melakukan <i>command FORWARD</i> atau bot tidak melakukan <i>command FORWARD</i> . Namun, yang membedakan pemetaan elemen/komponen algoritma greedy hanyalah tujuan kapal yang akan mengarah kemana.
Fungsi objektif	Pengurangan/penambahan nilai arah untuk mendapatkan nilai arah yang meminimalisir kerugian akibat objek-objek tersebut memanfaatkan fungsi <i>isLeft</i> untuk mengetahui apakah bot harus diputar lebih ke kiri atau lebih ke kanan. Jika objek pengganggu ada di sebelah kiri (<i>isLeft</i> bernilai true) maka nilai arah akan dikurangkan sebesar 70 sementara jika objek ada di sebelah kanan (<i>isLeft</i> bernilai false) maka nilai arah akan ditambahkan sebesar 70.

3.2 Eksplorasi alternatif solusi *greedy* yang mungkin dipilih dalam persoalan Galaxio

Terdapat banyak alternatif solusi algoritma greedy yang dapat diimplementasikan pada bot permainan Galaxio. Hal ini dikarenakan terdapat banyak elemen dari game engine yang

dapat diakses oleh bot dan kemudian diubah state pada elemen tersebut. Selain itu, elemen-elemen tersebut saling berinteraksi dengan elemen lainnya sehingga jika terdapat perubahan pada suatu state elemen, terdapat peluang bahwa state lainnya berubah juga.

Untuk mengambil keputusan arah bot terdapat beberapa kemungkinan. Defaultnya bot akan selalu mengarah ke makanan terdekat, namun prioritas respons terhadap parameter-parameter lain perlu dilakukan. Parameter-parameter lain dikategorikan menjadi tiga yaitu:

1. Objek pengganggu seperti Gas Cloud, Asteroid Field dan Wormhole
2. Musuh
3. Jarak ke ujung map

Eksplorasi strategi greedy yang dilakukan divariasikan menjadi empat yaitu:

1. Greedy dengan prioritas merata
2. Greedy dengan prioritas menghindari objek pengganggu
3. Greedy dengan prioritas menghindari musuh
4. Greedy dengan prioritas menghindari ujung map

3.3 Analisis efisiensi dan efektivitas dari kumpulan alternatif solusi *greedy* yang dirumuskan

Pada permainan Galaxio, banyak gamestate yang dapat diketahui dengan mudah seperti posisi pemain, posisi lawan, objek pengganggu seperti Gas Cloud, Asteroid Field, Wormhole, dan lain-lain. Hal tersebut memudahkan program agar dapat berjalan dengan efektif.

3.3.1 Analisis terhadap greedy dengan prioritas merata

Algoritma ini membuat bot menuju ke arah yang paling aman dari objek-objek pengganggu, musuh dan map. Arah yang dituju adalah arah yang paling aman. Namun, karena bot akan selalu menjauh dari gangguan, bot tidak akan fokus terhadap makanan sehingga bot tidak akan efektif untuk menangkap makanan.

3.3.2 Analisis terhadap greedy dengan prioritas menghindari objek pengganggu

Algoritma ini membuat bot menuju ke arah yang menghindari objek pengganggu sehingga pergerakan bot akan menjadi lebih efektif. Namun, kemungkinan bot akan lebih mengutamakan menghindari objek gangguan daripada musuh yang berbahaya.

3.3.3 Analisis terhadap greedy dengan prioritas menghindari musuh

Algoritma ini membuat bot menuju ke arah yang menghindari musuh berbahaya sehingga bot akan aman dari musuh yang akan memakan. Namun, kemungkinan bot akan lebih memilih untuk mengejar musuh daripada memperhatikan objek sekitarnya yang mungkin saja membuat kapal bot menjadi lebih kecil atau bergerak lebih lambat.

3.3.4 Analisis terhadap greedy dengan prioritas menghindari ujung asap

Algoritma ini membuat bot menuju ke tengah sehingga bot tidak akan terlempar keluar map. Namun, bot kemungkinan tidak akan memperhatikan musuh dan objek sekitarnya.

3.4 Strategi greedy yang dipilih

Strategi greedy yang dipilih adalah greedy yang memprioritaskan menghindari musuh. Pertimbangannya adalah ketika bot dalam kasus harus memilih dimakan musuh atau ukuran yang mengecil, bot lebih baik memilih untuk mengecilkan ukuran karena bot masih bisa untuk berkembang lagi dengan mencari makanan. Apabila bot memilih untuk dimakan musuh, bot akan langsung kalah dalam permainan.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi algoritma *greedy*

4.1.1 Implementasi kelas computeNextPlayerAction

```
class computeNextPlayerAction
Deklarasi
playerAction = PlayerAction
foodList = list of GameObject // dengan tipe food
playerList = list of GameObject // dengan tipe player
superfoodList = list of GameObject // dengan tipe superfood
gascloudList = list of GameObject // dengan tipe gascloud
asteroidfieldList = list of GameObject // dengan tipe asteroidfield
wormholeList = list of GameObject // dengan tipe wormhole

Body
// Defaultnya arahkan ke makanan
playerAction.heading = getHeadingBetween(bot, foodList[1])

// Jika ada superfood arahkan ke superfood
if (!superfoodList.isEmpty() and getDistanceBetween(bot,
superfoodList[1]) < getDistanceBetween(bot, foodList[1])) then
    playerAction.heading = getHeadingBetween(bot, superfoodList[1])
endif

// Menjauh jika ada player lain yang berpotensi memakan
if (!playerList.isEmpty() and nearestPlayer(playerList[1]) = 1) then
    if(isLeft(bot,playerList[1],playerAction.heading)) then
        playerAction.heading = playerAction.heading - 160
    else
        playerAction.heading = playerAction.heading + 160
    endif
endif

// Jika terlalu dekat ke ujung
if (getDistanceBetween(center, bot) + 1.7*bot.getSize() >
gameState.getWorld().getRadius()) then
    playerAction.heading = getHeadingBetween(bot, center) - 40
endif

// Mendekat jika ada player yang bisa dimakan
if(!playerList.isEmpty() and nearestPlayer(playerList[1]) == 2)
    playerAction.heading = getHeadingBetween(bot, playerList[1])
endif

// Menjauhi asteroidfield
```

Laporan Tugas Besar 1 IF2211 Strategi Algoritma

```
if (!asteroidfieldList.isEmpty() and
nearAsteroidfield(asteroidfieldList[1])) then
    if (isLeft(bot, asteroidfieldList[1], playerAction.heading)) then
        playerAction.heading = playerAction.heading - 70
    else
        playerAction.heading = playerAction.heading + 70;
    endif
endif

// Menjauhi gascloud
if (!gascloudList.isEmpty() and nearGasCloud(gascloudList[1])) then
    if (isLeft(bot, gascloudList[1], playerAction.heading)) then
        playerAction.heading = playerAction.heading - 70
    else
        playerAction.heading = playerAction.heading + 70
    endif
endif

// Menjauhi wormhole
if (!wormholeList.isEmpty() and nearWormhole(wormholeList[1])) then
    if (isLeft(bot, wormholeList[1], playerAction.heading)) then
        playerAction.heading = playerAction.heading - 70
    else
        playerAction.heading = playerAction.heading + 70
    endif
endif

class nearGasCloud
Deklarasi
gascloud = GameObject
-> 1.2*bot.getSize() + 1.2*gascloud.getSize() > getDistanceBetween(bot,
gascloud)

class nearAsteroidfield
Deklarasi
asteroidfield = GameObject
-> 1.2*bot.getSize() + asteroidfield.getSize() > getDistanceBetween(bot,
asteroidfield)

class nearWormhole
Deklarasi
wormhole = GameObject
-> 1.2*bot.getSize() + wormhole.getSize() > getDistanceBetween(bot,
wormhole)

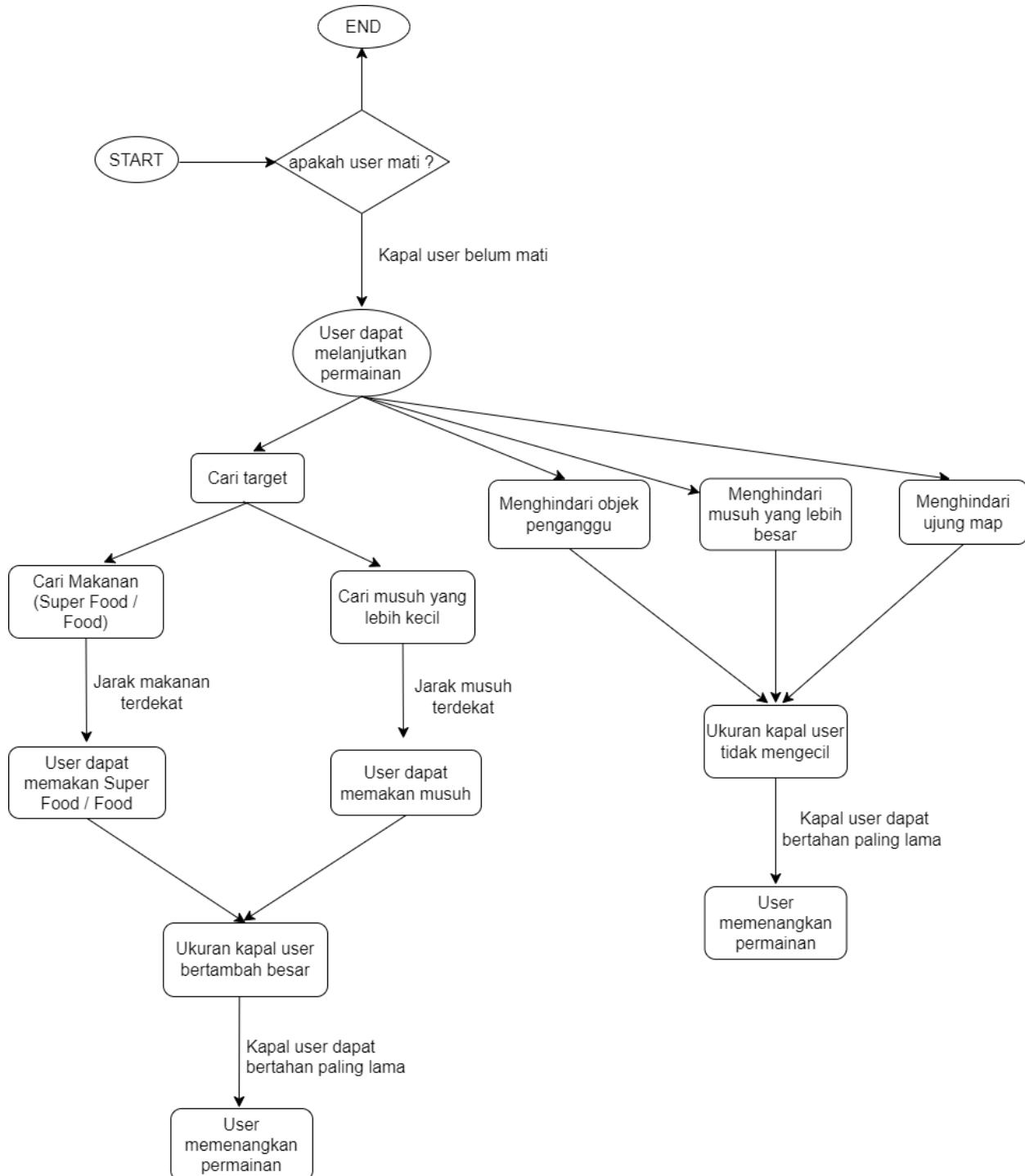
class nearestPlayer
Deklarasi
player = GameObject
```

Laporan Tugas Besar 1 IF2211 Strategi Algoritma

```
if (0.5*player.getSize() > 2.7*bot.getSize()) and
2.6*getDistanceBetween(bot, player) < 0.7*player.getSpeed()) then
    -> 1 // bahaya player lain
else if (0.5*player.getSize() < 2.7*bot.getSize()) and
2.6*getDistanceBetween(bot, player) < 0.7*bot.getSpeed()) then
    -> 2 // bisa dikejar
else
    -> 3 // tidak ada apa-apa
endif

class isLeft
Deklarasi
dif = int
Body
obj1, obj2 = GameObject
dif = getHeadingBetween(obj1, obj2)
if (heading > dif) then
    dif = dif + 360
endif
if (dif - heading <= 180) then
    -> true
else
    -> false
endif
```

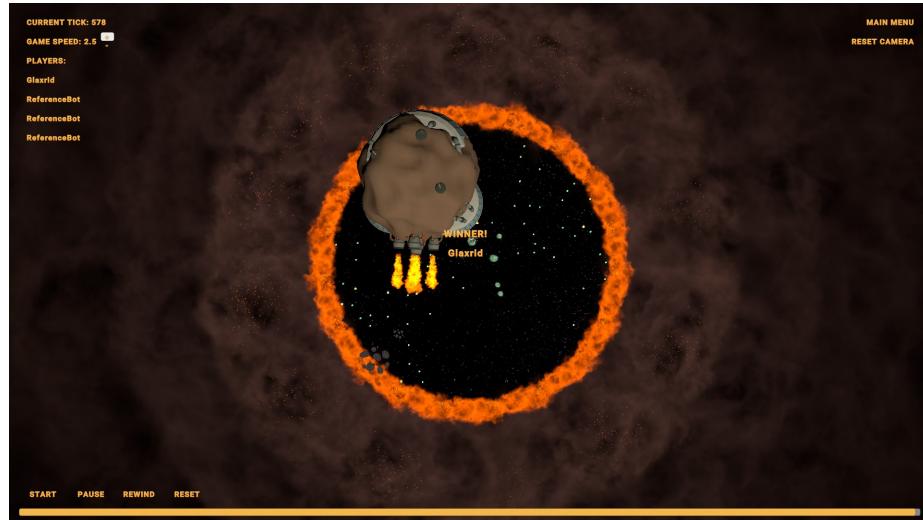
4.2 Flowchart dari desain solusi algoritma *greedy*



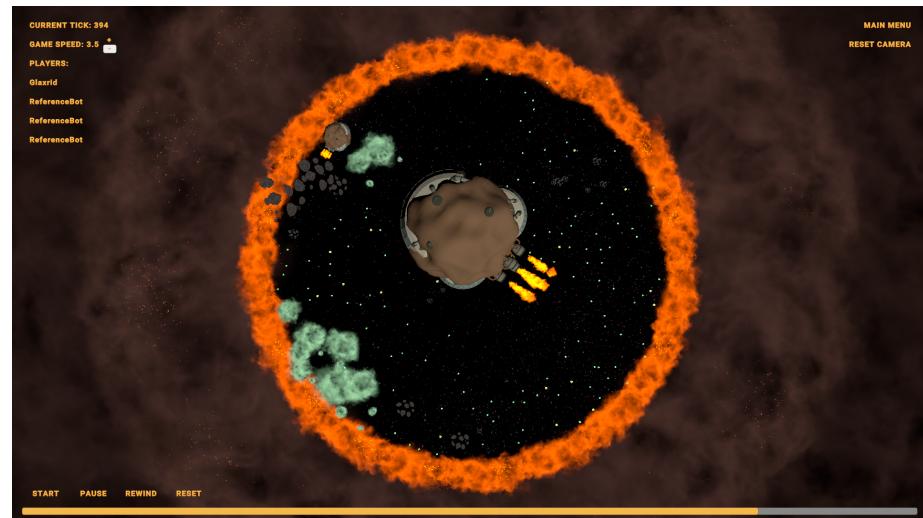
Gambar 4.1.1 Flowchart dari solusi algoritma Greedy

4.3 Analisis dari desain solusi algoritma *greedy*

Pengujian dilakukan menggunakan *reference bot* default dan bot Glaxrid cukup efisien dalam memainkan permainan Galaxio. Namun, pada kondisi-kondisi tertentu bot bisa tidak optimal seperti pada saat dikelilingi objek-objek yang dihindari.



Gambar 4.2.1 Bot berhasil menang dalam permainan



Gambar 4.2.2 Bot terperangkap di antara asteroid dan gas cloud

BAB V

KESIMPULAN, SARAN, KOMENTAR DAN REFLEKSI

5.1 Kesimpulan

Dari tugas besar IF2211 Strategi Algoritma semester 2 2022/2023 berjudul “Pemanfaatan Algoritma Greedy dalam Aplikasi Permainan Galaxio”, kami berhasil membuat sebuah bot dalam bahasa Java yang memanfaatkan algoritma greedy. Program yang dibuat tidak hanya menggunakan satu algoritma greedy saja melainkan beberapa algoritma greedy yang dikombinasikan untuk menghasilkan strategi yang terbaik. Program berhasil dijalankan tanpa bug dan sesuai dengan spesifikasi tugas besar 1 IF2211 yang diberikan.

5.2 Saran

Saran-saran yang dapat kami berikan untuk tugas besar IF2211 Strategi Algoritma semester 2 2022/2023 adalah:

1. Untuk memperjelas spesifikasi tugas yang diberikan (seperti apakah strategi yang digunakan sudah benar bentuk greedy atau belum) mungkin diperlukan asistensi beberapa kali sebelum pengumpulan.
2. Penulisan pseudocode tampak kurang perlu dikarenakan program yang lumayan panjang dan membaca program lebih mudah daripada membaca pseudocode dengan asumsi program sudah well commented.

5.3 Komentar

Dari tugas besar IF2211 Strategi Algoritma yang telah diberikan, sudah berjalan dengan lancar dan baik untuk koordinasi antara anggota kelompok. Program yang dibuat tidak hanya menggunakan satu algoritma greedy saja, namun juga mengkombinasikan beberapa algoritma greedy lainnya sehingga diperoleh hasil yang optimum. Untuk spesifikasi tugas yang diberikan sudah cukup jelas, namun untuk menentukan apakah strategi yang digunakan sudah benar berbentuk greedy atau tidak, mungkin diperlukan asistensi sebelum pengumpulan untuk memperjelas.

5.4 Refleksi

Setelah menyelesaikan tugas besar pertama IF2211 Strategi Algoritma, kami dapat merefleksikan bahwa komunikasi antar anggota kelompok berjalan cukup baik sehingga tidak terjadi miskomunikasi dan kesalahpahaman dalam penggerjaan tugas besar ini, sebelum dimulainya penggerjaan tugas besar ini, sudah dilakukan diskusi untuk membahas pembagian kerja untuk setiap anggota kelompok, dan terakhir kami menyadari perlunya mempelajari strategi algoritma greedy yang berguna dalam menghasilkan hasil yang optimum.

DAFTAR PUSTAKA

[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) diakses pada tanggal 7 Februari 2023

[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) diakses pada tanggal 7 Februari 2023

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-\(2022\)-Bag3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf) diakses pada tanggal 7 Februari

[Maven – Running Maven \(apache.org\)](https://maven.apache.org/) diakses pada tanggal 10 Februari 2023

LINK REPOSITORY

https://github.com/manuellaiv/Tubes1_Glaxrid

LINK VIDEO

<https://youtu.be/o1Os8mT-IOI>