Tugas Besar 1 IF2211

Strategi Algoritma Semester II tahun 2021/2022

Pemanfaatan Algoritma Greedy dalam Aplikasi Permainan "Galaxio"

Disusun oleh:

Kelompok 12 – "artinya apa tuh bang messi"

Mutawally Nawwar 13521065

Vieri Fajar Firdaus 13521099

Rizky Abdillah Rasyid 13521109

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

DAFTAR PUSTAKA

BAB 1
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. Bahasa Pemrograman yang digunakan adalah Java. Bahasa Java tersebut digunakan untuk membuat algoritma pada *bot*. IDE yang digunakan untuk membantu projek ini adalah Intellij IDEA atau menggunakan *text editor* yang digunakan adalah Visual Studio Code, serta menggunakan kakas maven untuk melakukan build source code menjadi file *jar*. Untuk menjalankan permainan, digunakan game engine yang diciptakan oleh entellect challenge yang terdapat pada repository githubnya.

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 disebarkan pada peta dengan ukuran 3 dan dapat dikonsumsikan 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 menembakannya 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 mengonsumsi 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*. Untuk daftar *commands* yang tersedia, bisa merujuk ke tautan panduan di spesifikasi tugas
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 2
LANDASAN TEORI

2.1. Dasar Teori

Pada tugas ini dikhususkan untuk mengimplementasikan salah satu algoritma, yaitu *greedy*. Algoritma greedy adalah metode sederhana yang digunakan untuk memecahkan masalah optimasi (oprimization problem) untuk mendapatkan maksimasi dan minimasi. Prinsip algoritma greedy adalah "take what you can get now". Algortima ini membentuk solusi per langkah (step by step). Pada setiap langkah akan dilakukan evaluasi terhadap pilihan-pilihan yang ada. Sehingga, perlu ditentukan keputusan terbaik dari pilihan-pilihan yang tersedia pada setiap langkahnya. Tetapi pada algoritma greedy tidak diperbolehkan adanya backtracking (mundur kembali ke langkah sebelumnya). Sehingga tiap langkah akan terpilih optimum local yang diharapkan optimum lokal yang terpilih mengarah ke solusi optimum global. Terdapat beberapa elemen/komponen yang perlu didefinisikan pada algoritma greedy. Berikut adalah elemen/komponen algortima greedy:

- Himpunan Kandidat (C): Berisi kandidat yang mungkin dipilih pada setiap langkahnya.
- Himpunan Solusi (S): Berisi kandidat yang sudah dipilih sebagai solusi
- Fungsi solusi: Menentukan apakah himpunan solusi yang dikumpulkan sudah memeberikan solusi.
- Fungsi seleksi: Memilih kandidat berdasarkan strategi greedy tertentu. Fungsi ini bersifat heuristik (fungsi yang dirancang untuk menentukan solusi optimum dengan mengabaikan apakah pilihan tersebut terbukti paling optimum secara matematis).
- Fungsi kelayakan: Memerikas apakah kandidat yang dipilih dari fungsi seleksi dapat dimasukan ke dalam himpunan solusi.
- Fungsi objektif: Memaksimumkan atau meminimumkan suatu parameter pada suatu persoalan.

Dengan menggunakan elemen/komponen diatas dapat dikatakan bahwa Algoritma greedy melibatkan pencarian himpunan bagian S dari himpunan kandidat C, yang mana S harus memenuhi beberapa kriteria yang ditetukan, yaitu S emnyatakan suatu solusi dan S dioptimasi oleh obyektif.

Berikut adalah skema umum dari algortima greedy menggunakan pseudocode:

```
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 iterai, solusi yang dikembalikan adalah optimum lokal, dan pada akhir iterasi keseluruhan akan diperoleh optimum global. Optimum global yang didapat belum tentu merupakan solusi optimum (terbaik) karena algoritma greedy tidak beroperasi menyeluruh terhadap semua kemungkinan solusi yang ada.

2.2. Garis Besar Cara Kerja Bot Permainan *Galaxio*

Secara garis besar, cara kerja bot dari permainan Galaxio ini adalah mengambil data(*object)* yang tersedia pada State. Data-data yang terdapat pada game adalah sebagai berikut:

- Object: Semua objek yang direpresentasikan sebagai bantuk lingkaran dan memiliki tititk pusat X dan Y koordinat dan radius yang mendefinisikan ukuran dan bentuknya.
    - Food: Makanan yang akan meningkatkan ukuran pemain yang tersebar dalam map, tidak bergerak, serta akan dihapus jika keluar dari map.
    - SuperFood: Makanan yang dapat meningkatkan ukuran pemain dua kali lipat disbanding Food (makanan biasa).
    - Wormholes: Lubang yang membuat pemain bisa berpindah dari satu posisi ke posisi lain.
    - GasCloud: Objek yang membuat size pemain berkurang setiap satu tick dalam permainan jika suatu pemain berada di dalamnya.
    - AsteroidFields: Objek yang memebuat kecepatan pemain berkurang dengan faktor dua apabila pemain melewatinya.
    - TorpedoSalvo: Rudal yang dapat ditembakan ke musuh lainnya dengan efek tertentu.
    - Supernova: Senjata terkuat yang hanya akan muncul di waktu tertentu antara kuarter pertama dan kuarter terakhir permainan.
    - Teleport: Objek yang dapat digunakan pemain untuk berpindah posisi dengan cara menembakannya.
    - Id: Kode unik tertentu yang dimiliki tiap game object.

- Player: Pemain akan berupa sebuah kapal berbentuk bulat yang akan bertumbuh seiring dengan kegiatan memakan food.
    - Speed: Variabel yang menentukan seberapa jauhnya sebuah kapal pemain akan bergerak dalam satu tick.
    - Size: Ukuran/radius dari suatu objek.
    - Heading: Ukuran derajat radian yang merepresentasikan orientasi objek tersebut.
    - AfterBurner: Item yang akan meningkatkan kecepatan pemain dengan faktor dua.
    - DarkMatterTorpedos: Senjata yang dapat digunakan pemain untuk merekayasa area di sekitarnya.
    - Shield: Item pertahanan yang dapat digunakan untuk mencegah dampak dari serangan pemain lain.
    - X position: Posisi pada grafik kartesius permainan berdasarkan sumbu-X.
    - Y position: Posisi pada grafik kartesius permainan berdasarkan sumbu-Y.
- World: Kumpulan dari beberapa komponen tertentu yang terkait dengan perubahan faktor-faktor esensial pada permainan.
    - CenterPoint: Titik tengah suatu objek pada map.
    - tick: Ukuran waktu yang berlaku pada permainan Galaxio.
- Command: Pada tiap tick setiap pemain dapat melakukan satu buah perintah/command untuk kapal mereka.
    - FORWARD: Perintah untuk bergerak maju sesuai arahnya pada saat tersebut.
    - STOP: Perintah untuk berhenti hingga ada perintah lain yang dikeluarkan.
    - START_AFTERBURNER: Perintah untuk menyalakan afterBurner.
    - STOP_AFTERBURNER: Perintah untuk mematikan afterBurner.
    - FIRE_TORPEDOS: Perintah untuk menembakkan torpedo.
    - FIRE_SUPERNOVA: Perintah untuk menembakkan supernova.
    - DETONATE_SUPERNOVA: Perintah untuk meledakkan supernova.
    - FIRE_TELEPORTER: Perintah untuk menembakkan teleporter.
    - TELEPORT: Perintah untuk berpindah pada posisi teleporter saat itu.
    - USE_SHIELD: Perintah untuk mengaktifkan shield.

Setelah *bot* mengambil data-data diatas, maka bot dapat melakukan analisis menggunakan algoritma yang telah dituliskan sebelumnya untuk mencari *command* paling tepat untuk dieksekusi pada ronde selajutnya. Allgoritma yang dituliskannya tersebut berbentuk fungsi objektif dari algoritma *greedy*. Perlu ditekankan pula, untuk setiap keadaan data, bot harus mengirimkan command yang bersesuaian.

2.3. Implementasi Algoritma Greedy ke Dalam Bot Permainan *Galaxio*

Terdapat berbagai algortima yang dapat diimplementasikan ke dalam bot permainan. Salah satunya algoritma greedy, Algortima Greedy tidak membutuhkan waktu dan resource yang banyak untuk mengevaluasi kemungkinan command yang dapat terjadi. Tetapi dibutuhkan Teknik huristik yang didefinisikan dengan logika pemrogram yag disesuaikan dengan logika dan aturan permainan. Hasil algoritma ini mungkin tidak menghasilkan solusi yang optimum global, karena algoritma greedy tidak beroperasi menyeluruh terhadap semua kemungkinan solusi yang ada.

Pada bot permainan yang kami buat, kami mengaplikasikan algortima greedy karea bersifat intuitif serta tidak memerlukan waktu dan resource yang banyak. Karena menggunakan Teknik heuristik kami dapat membuat banyak kemungkinan solusi greedy yang disesuaikan dengan aturan dan logika permainan Galaxio. Meskipun terdapat peluang mendapatkan solusi yang tidak optimum secara global, tetapi setidaknya besar kemungkinan untuk mencapai solusi optimum lokal yang nilainya mendekati solusi optimum global.

2.4. Garis Besar Game Engine Permainan *Galaxio*

GameEngine pada permainan Galaxio adalah sebuah kerangka perangkat lunak yang dirancang sebagai kumpulan alat dan fitur yang membantu pengembangan game. Game engine memiliki beberapa tools inti seperti rendering engine, physics engine, sound engine, scripting language, animation engine, artificial intelligence, networking engine, streaming engine, memory management, threading support, localization support, scene graph, dan video support. Namun, tidak semua tools disediakan oleh pemrogram pada permainan ini, hanya beberapa tools penting yang tersedia di repository, seperti rendering engine untuk command prompt, collision engine, dan scripting language. Untuk beberapa tools tambahan, seperti 2D rendering engine, dapat diunduh dari third party software.

Pada permainan Galaxio, game engine yang digunakan dibuat menggunakan bahasa C# dan sudah tersedia di repository yang diunggah oleh Entelect, penyelenggara dari challenge ini. Game engine sudah berbentuk file .jar, sehingga pemrogram tidak perlu melakukan compile dan build ulang project secara manual. Untuk menjalankan game engine, pengguna perlu memenuhi prerequisites dengan memiliki JDK pada mesin eksekusi. Setelah prerequisites terpenuhi, pengguna dapat menjalankan game engine dengan menjalankan file run.bat untuk pengguna dengan OS Windows atau menjalankan perintah shell script pada file Makefile dengan mengetikkan perintah "make run" pada command prompt untuk pengguna dengan OS Linux atau macOS.

Selain file game-engine, terdapat beberapa file dan folder yang perlu ditambahkan pada stater-pack. Beberapa file dan folder yang wajib ada agar game-engine dapat dijalankan adalah:

- Game-engine.jar & game-engine-jar-with-dependencies.jar: source code serta dpendencides dari game engine.
- Game-config.json: berisi aturan yang harus diikuti oleh game engine ketika mengeksekusi permainan.
- Game-runner-config.json: berisi informasi directory dari file bot serta parameter-parameter lainnya yang dibutuhkan oleh game engine.
- Run.sh/Run.bat: file scripting language yang digunakan untuk menjalankan game engine pada OS (Run.sh untuk OS Linux dan MacOS serta run.bat untuk OS Windows).

Setelah algoritma greedy dari starter bot dibuat, kompilasi dilakukan agar perubahan tersebut dapat diterapkan pada file jar dari starter bot. Dalam program ini, telah disediakan build tools dari Apache Maven Project, yang memungkinkan pemrogram menjalankan beberapa

command dari build lifecycle seperti validate, compile, test, package, verify, install, dan deploy. Namun, pada permainan Galaxio ini, pemrogram hanya perlu menjalankan perintah compile dan install untuk melakukan kompilasi perubahan source code dan mengubahnya ke dalam bentuk file jar, tanpa perlu melakukan semua perintah pada build lifecycle.

BAB 3
APLIKASI STRATEGI GREEDY

3.1. Pemetaan Komponen/Elemen pada Permainan Galaxio

Algoritma Greedy dapat dibagi menjadi enam bagian fungsi dan himpunan yang mendefiniskan algoritma tersebut, yaitu:

- Himpunan Kandidat
  Himpunan semua command yang mungkin dilakukan oleh bot yaitu :

  a. FORWARD

  b. STOP

  c. START_AFTERBURNER

  d. STOP_AFTERBURNER

  e. FIRE_TORPEDOES

  f. FIRE_SUPERNOVA

  g. DETONATE_SUPERNOVA

  h. FIRE_TELEPORTER

  i. TELEPORT

- Himpunan Solusi Himpunan command-command yang sudah terpilih.
- Fungsi Solusi Fungsi yang mengecek apakah sudah sampai finish line.
- Fungsi Seleksi Fungsi dan algoritma yang digunakan untuk memilih command yang dipilih, akan dibahas pada poin berikutnya.
- Fungsi Kelayakan Fungsi yang digunakan untuk mengecek apakah command yang digunakan valid. Sebelum menggunakan akan dicek, apakah kita memiliki power up tersebut. Sebelum belok kiri atau kanan akan dicek lane kita berada, apakah memungkinkan.
- Fungsi Objektif Mencapai finish line sebelum musuh dengan round yang minimum.

3.2. Alternatif Solusi Algoritma Greedy

Pada bagian ini, supaya analisis efisiensi lebih terlihat akan digunakan kompleksitas waktu. Agar perbedaan antar algoritma juga terlihat lebih jelas akan digunakan variable berikut.

1. Greedy by Food

   Strategi greedy pada alternatif ini adalah memilih command sehingga calon *food* atau *superfood* berada di dalam area aman dan calon makanan berada di luar jangkauan gas. Pada strategi ini kita selalu memperhitungkan jarak antar makanan yang akan kita ambil. Kita akan selalu mengambil makanan yang telah di filter lalu disorting berdasarkan jarak terdekat.

Selain itu kita juga memprioritaskan adanya supernova, apabila supernova berada di sekitar kita maka supernova akan diprioritaskan terlebih dahulu

Analisis Efisiensi :

Kita harus mengecek seluruh makanan yang akan kita makan dan mengecek kondisi-kondisi tertentu (terdapat halangan didepan). Oleh karena itu kompleksitasnya adalah O(N) dengan N adalah banyak makanan yang akan kita ambil

Analisis Efektivitas :

Pada dasarnya jika makanan berada di dalam radius tertentu dan tidak berada didekat gas, kita pasti akan mengambil makanan tersebut. Akan tetapi, kita tidak selalu dapat memprediksi apakah ada beberapa makanan yang memiliki jarak yang sama, apabila terdapat beberapa makanan dengan jarak yang sama, bot kita akan kebingungan untuk memilih makanan mana yang akan kita pilih. Apabila seluruh makanan telah habis maka langkah yang akan dilakukan dalam *greedByFood* adalah bergerak random dengan jangkauan yang aman dari musuh.

2. Greedy by Defense

Pada greedy ini, difokuskan untuk menghindari segala bahaya yang dapat membuat ukuran bot berkurang. Dalam greedy ini pun masih dibagi menjadi beberapa bagian.

a. Menghindar teleport

Untuk menghindari teleport pertama akan dicari dulu apakah terdapat teleport pada tick saat ini. Ketika ada teleport yang jaraknya dekat dengan bot dan jarak tersebut lebih kecil dari player terbesar, maka bot akan kabur dari teleport. Hal ini merupakan penerapan greedy dimana asumsi teleport tersebut merupakan milik player terbesar.

Pada algoritma ini membutuhkan kompleksitas O(N) untuk menfilter N objek pada peta. Kmeudian membutuhkan juga O(T log T) untuk mengurutkan T teleport berdasarkan jarak.

b. Menghindar torpedo

Untuk menghindari torpedo ada beberapa cara, yaitu mengeluarkan shield, tembak torpedo, atau kabur. Kapan paling optimal mengeluarkan shield, yaitu ketika jarak bot dengan torpedo terdekat sangat dekat dan torpedo mengarah ke bot. Ketika bot tidak memiliki shield, maka torpedo akan menembakan torpedo untuk menghancurkan torpedo lawan. Ketika bot terlalu kecil dan tidak memiliki torpedo, maka bot akan kabur dengan arah 90 derajat terhadap arah torpedo terdekat.

Pada algoritma ini membutuhkan kompleksitas O(N) untuk menfilter N objek di peta dan O(T log T) untuk mengurutkan T torpedo berdasarkan jarak.

c. Menghindar musuh

Untuk menghindari musuh, bot akan menembakkan torpedo Ketika memiliki torpedo. Jika bot tidak memiliki torpedo, maka akan kabur dengan arah 90 derajat terhadap heading lawan terdekat.

Pada algoritma ini membutuhkan kompleksitas O(N log N) untuk mengurutkan N lawan berdasarkan jarak.

d. Mengindari gas cloud

Dalam mengindari gas cloud bot akan mengecek gas cloud terdekat. Jika dirasa sudah sangat dekat, bot akan berbelok berlawanan dengan arah datang bot. Pada algoritma ini membutuhkan O(N log N) untuk mengurutkan N gas cloud berdasarkan jaraknya.

3. Greedy by Offense

Pada greedy ini, difokuskan untuk menyerang lawan dengan harapan dapat mengurangi ukuran lawan atau bahkan mengeliminasi lawan. Dalam greedy ini juga masih dibagi menjadi beberapa bagian.

a. Menyerang dengan teleport

Pada strategi ini, bot akan menyerang menggunakan teleport. Bot akan memilih terlebih dahulu lawan dengan ukuran terdekat dan terkecil. Setelah bot menembakan teleport, bot akan mengukur apakah jarak teleport dengan lawan terdekat lebih kecil disbanding ukuran bot. Ketika terpenuhi kondisi tersebut, bot akan langsung teleport kesana.

Pada algoritma ini membutuhkan O(N log N) untuk mengurutkan lawan dengan ukuran terkecil dan jarak terpendek.

b. Menyerang dengan torpedo

Pada strategi ini, bot akan menyerang musuh terdekat dengan menembakkan torpedo. Hal ini tidak sembarangan, tetapi harus memastikan bahwa ukuran bot cukup untuk menembakan torpedo.

Algoritma ini membutuhkan O(N log N) untuk mengurutkan lawan berdasarkan jarak.

c. Menyerang dengan supernova

Pada strategi ini, Ketika bot tidak memiliki torpedo dan tidak memiliki teleport, bot akan menyerang menggunakan supernova. Namun, harus dicek terlebih dahulu apakah memiliki supernova. Bot akan menyerang lawan terdekat.

Algoritma ini membutuhkan O(N log N) untuk mengurutkan lawan berdasarkan jaraknya.

3.3. Solusi Yang Dipilih

Jika dilihat dari efisiensi algoritma, setiap algoritma memiliki kompleksitas waktu yang berbeda-beda. Akan tetapi, nilai variabel yang ada cukup kecil sehingga tidak terlalu memengaruhi runtime. Oleh karena itu, bagian efisiensi tidak menjadi pertimabangan utama kami.

Solusi yang kami pilih secara umum, kita memprioritaskan untuk bertahan terlebih dahulu,lalu *greedByOffense* dan secara default akan melakukan *gredByFood* . Namun hal ini tidak selalu kita gunakan karena terdapat kondisi-kondisi tertentu sehingga kita tidak bisa menggunakan salah satu dari ketiga *greedy* tersebut.

Alternatif pertama yang kita gunakan ketika ukuran bot kita kurang dari 15, kita akan memprioritaskan untuk mencari makanan, mencari makanan merupakan hal yang penting dalam permainan kali ini apabilla kita terkena *torpedo* dengan ukuran kita kurang dari 15 *bot* akan cepat mati

Alternatif kedua, apabila ukuran *bot* kurang dari 35, kita akan memprioritaskan untuk bertahan terlebih dahulu, lalu apabila telah aman *bot* akan mencari makanan. Dalam hal ini kita memilih ukuran 25 karena dalam bertahan kita akan menggunakan *shield* ataupun *torpedos* , dimana shield dan torpedo memankan ukuran kita sebanyak 10 dan 20

Alternatif ketiga, apabila ukuran *bot* lebih dari sama denga 35, *bot* akan memprioritaskan untuk bertahan, menyerang dan secara *default* akan mencari makanan. Untuk proses menyerang telah dijelaskan dalam *greedByOffense* sehingga dalam kondisi

Dalam prioritas perintah, bot akan memprioritaskan secara urut dari proses diatas, mulai dari alternatif pertama, alternatif kedua, alternatif ketiga. Apabila syarat alternatif pertama tidak tercapai bot akan langsung memilih alternatif kedua, dan ketika syarat alternatif kedua tidak tercapai bot akan langsung memilih alternatif ketiga, dan apabila seluruh syarat dari alternatif pertama, kedua dan ketiga tidak tercapai bot secara default akan memilih untuk melakukan *greedByFood*

BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1 Pseudocode

Pada subbab ini akan diberikan pseudocode dari fungsi-fungsi utama pada file `BotService.java`. Fungsi selain jalan kerja utama tidak kami masukkan karena akan sangat panjang. Untuk kode lengkap bisa dilihat di *source code* .

```
function greedByFood(PlayerAction : playerAction) -> PlayerAction
kamus lokal
   playerelse,gaslist,food,supernovalist : list<GameObject>
Algoritma
   // persiapan data
   Gaslist <- gameState.GAS_CLOUD
   food <- gameState.FOOD
   supernovalist <- gameState.SUPERNOVA
   // untuk food kita memilih sehingga objek berada didalam radius dan
   //mengurutkan berdasarkan jarak terdekat dengan bot


   if(Gaslist.size()>0) then
      food <- gameState.FOOD.filter(Gaslist)
      // memfilter sehingga makanan jauh dari gas

   if(supernovalist.size()>1) then
      if(distance(supernova)<200) then
         -> move(supernova)
      else
         -> move(food)
   else if(food.size()>=1) then
      -> move(food)
   else
      -> move(mid)
```

```
Procedure computeNextPlayerAction(Player Action : playerAction)
Kamus lokal

Algoritma
   if(gameState.world.currenTick>400) then
   else if(bot.getSize()<15) then
      playerAction <- greedByFood(playerAction)

   else if(bot.getSize()<35) then
      if(inWorld()) then
         if(!teleList.isEmpty()) then
            playerAction <- avoidTele()

         else if(!torpedoList.isEmpty()) then
            playerAction <- avoidTorpedo()

         else if(!gasList.isEmpty()) then
```

```
            playerAction <- avoidGas()

         else then
            playerAction <- greedByFood(playerAction)


      else then
         playerAction <- move(mid)
   else
      if(inWorld()) then
         if(!teleList.isEmpty()) then
            playerAction <- avoidTele()

         else if(!torpedoList.isEmpty()) then
            playerAction <- avoidTorpedo()

         else if(!gasList.isEmpty()) then
            playerAction <- avoidGas()

         else if(bot.getSize()>35) then
            playerAction <- greedByOffense(playerAction)

         else then
            playerAction <- greedByFood(playerAction)


      else then
         playerAction <- move(mid)
```

```
function avoidGas() -> PlayerAction
kamus local
   tmp : PlayerAction
   telelist : list<GameObject>
   differenceHeading : integer
Algoritma
   Telelist <- gameState.TELEPORT
   differenceHeading <- bot.currentHeading-getHeadingBetween(telelist[0])

   If(differenceHeading>0) then
      tmp.action <- PlayerActions.FORWARD
      tmp.heading <- (telelist[0].getCurrentHeading()+270) mod 360
   else
      tmp.action <- PlayerActions.FORWARD
      tmp.heading <- (telelist[0].getCurrentHeading()+90) mod 360

  -> tmp
```

```
function avoidTorpedo() -> PlayerAction
kamus local
   tmp : PlayerAction
   torpedolist : list<GameObject>
```

```
   differenceHeading : integer
Algoritma
   torpedolist <- gameState.TORPEDO
   differenceHeading <- bot.currentHeading-getHeadingBetween(torpedolist[0])

   If(getDistance(bot.torpedolist[0])<=120 AND bot.getSize()>50) then
      tmp.action <- PlayerActions.ACTIVATE_SHIELD
      tmp.heading <- (torpedolist[0].getCurrentHeading()+270) mod 360
   else if(getDistance(bot.torpedolist[0])<=180 AND bot.getSize()>50) then
      tmp.action <- PlayerActions.FIRE_TORPEDOES
      tmp.heading <- getHeading(torpedolist[0])
   else if(differenceHeading>0)
      tmp.action <- PlayerActions.FORWARD
      tmp.heading <- (torpedolist[0].getCurrentHeading()+270) mod 360
   else
      tmp.action <- PlayerActions.FORWARD
      tmp.heading <- (torpedolist[0].getCurrentHeading()+90) mod 360

  -> tmp
```

```
function avoidGas() -> PlayerAction
kamus local
   tmp : PlayerAction
   gaslist : list<GameObject>
   differenceHeading : integer
Algoritma
   gaslist <- gameState.TORPEDO
   differenceHeading <- bot.currentHeading-getHeadingBetween(gaslist[0])

   if(differenceHeading>0) then
      tmp.action <- PlayerActions.FORWARD
      tmp.heading <- bot.getCurrentHeading()+90-Math.abs(differenceHeading)
+360) mod 360
   else
      tmp.action <- PlayerActions.FORWARD
      tmp.heading <- bot.getCurrentHeading()-90+Math.abs(differenceHeading)
+360) mod 360


  -> tmp
```

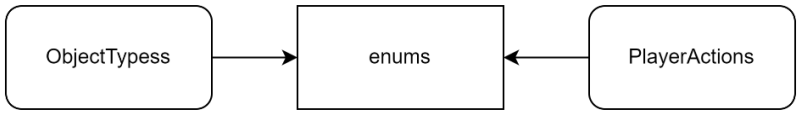
```
function greedByOffense(PlayerAction : playerAction) -> PlayerAction
kamus lokal
   playerlist, playerlisttele, tele: list<GameObject>
Algoritma
   // persiapan data
   playerlist <- gameState.PLAYER
   //tele heading membelakangi bot, dianggap milik bot
   telelist <- gameState.TELEPORT
   //paling dekat dengan tele
```

```
playerlisttele <- gameState.PLAYER

if(bot.getTorpedoCount()>0 AND bot.getSize()>25) then
   playerAction.action = FIRE_TORPEDOES
   playerAction.heading = getHeadingBetween(playerlist[0])
else if(bot.getTeleCount()>0) then
   playerAction.action = FIRE_TELEPORTER
   playerAction.heading = getHeadingBetween(playerlist[0])
else if(bot.getSupernovaAvailable()>0) then
   playerAction.action = FIRE_SUPERNOVA
   playerAction.heading = getHeadingBetween(playerlist[0])

if(telelist.size()>0 AND playerlisttele[0].getSize()<bot.getSize()) then
   playerAction.action = TELEPORT
   playerAction.heading = bot.getCurrentHeading()

-> playerAction
```

4.2 Struktur Data

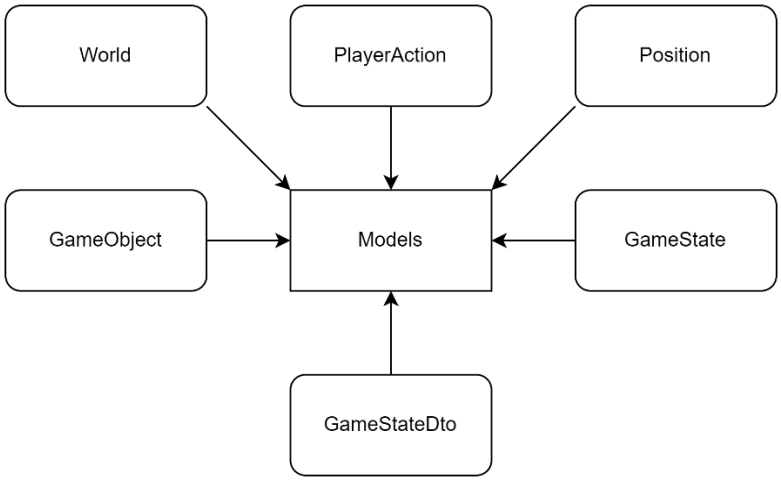
Terdapat tiga struktur data pada game galaxio yang berisikan enumerasi dari *command* dan *object type.* Ada juga layanan dari bot itu sendiri, serta informasi mengenai dunia di dalam game ini. Berikut adalah struktur datanya:

1. Enums



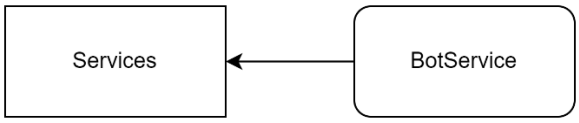
Pada enums, terdapat enumerasi dari *command* pada file PlayerActions dan tipe dari objek yang ada pada file ObjectTypes. Dari file-file tersebut dapat diketahui apa saja objek yang ada pada sunia game dan apa yang bisa dilakukan oleh player dalam game ini.

2. Models



Pada models terdapat tipe data yang digunakan untuk memodelkan informasi dari game ke bahasa java. Terdapat GameObject yang merupakan tipe data dari objek yang ada pada game. Terdapat pula GameState yang berisi semua informasi dari game pada state tertentu. Selain itu, terdapat PlayerAction yang merupakan tipe aksi yang dikirimkan oleh bot ke game. Position merupakan tipe posisi sebuah objek di dalam game. Terkahir terdapat World yang merupakan tipe dari dunia pada game ini.

3. Services

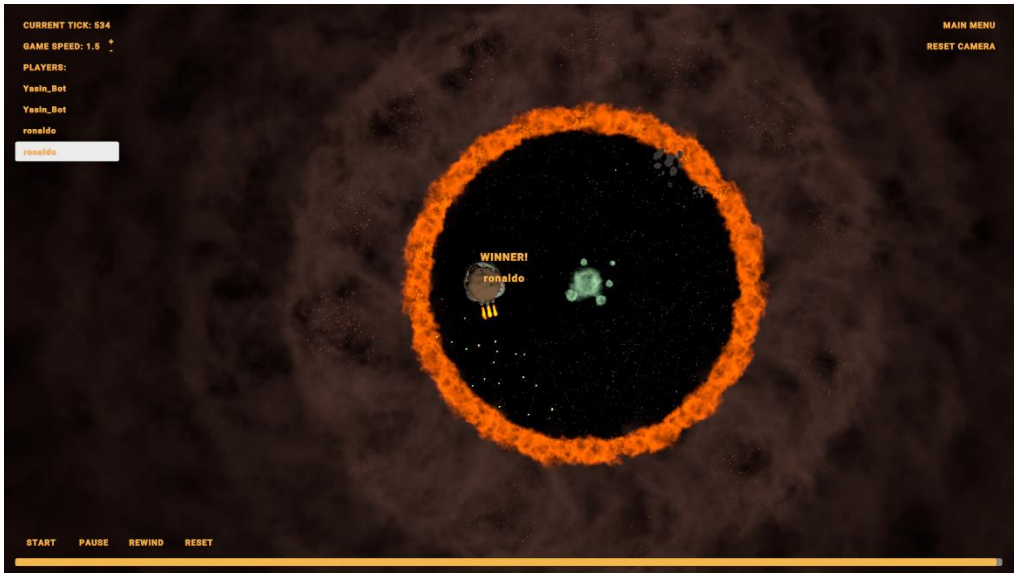


Pada services merupakan tempat algoritma greedy diterapkan. Disini semua yang bisa dilakukan oleh bot bersumber.

4.3 Analisis dan Pengujian

e. Pengujian 1

Keadaan ketika menang

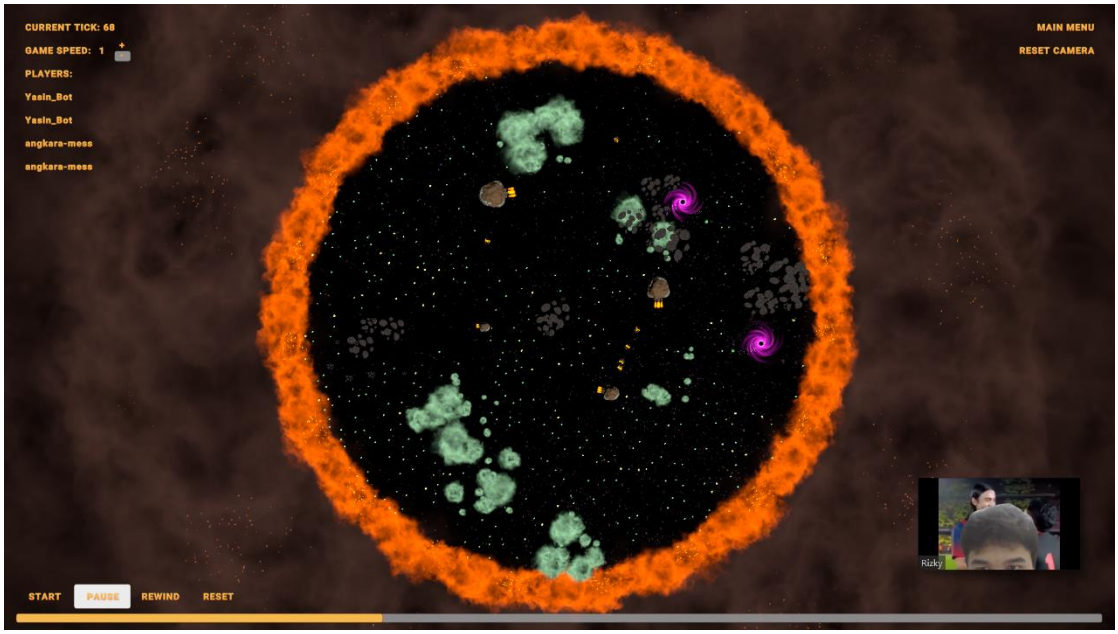



```
{"TotalTicks":534,"Players":[{"Placement":1,"Seed":17105,"Score":66,"Id":"e344f033-6471-4619-ac11-0c063b5e9174","Nickname":"ronaldo","MatchPoints":8},{"Placement":3,"Seed":23291,"Score":47,"Id":"89e06e62-3429-48ee-b8c1-d9a491b50e7a","Nickname":"Yasin_Bot","MatchPoints":4},{"Placement":4,"Seed":6139,"Score":42,"Id":"e263dd3b-a2e7-4f12-b76e-1b9024a92746","Nickname":"Yasin_Bot","MatchPoints":2},{"Placement":2,"Seed":17454,"Score":41,"Id":"e056d20b-4ef5-40c7-a83f-5f176c3d5e78","Nickname":"ronaldo","MatchPoints":6}],"WorldSeeds":[49833],"WinningBot":{"Id":"e344f033-6471-4619-ac11-
```

```
0c063b5e9174","Size":69,"Speed":3,"GameObjectType":1,"CurrentHeadi
ng":102,"Position":{"X":-300,"Y":8}}}
```


f. Pengujian 2

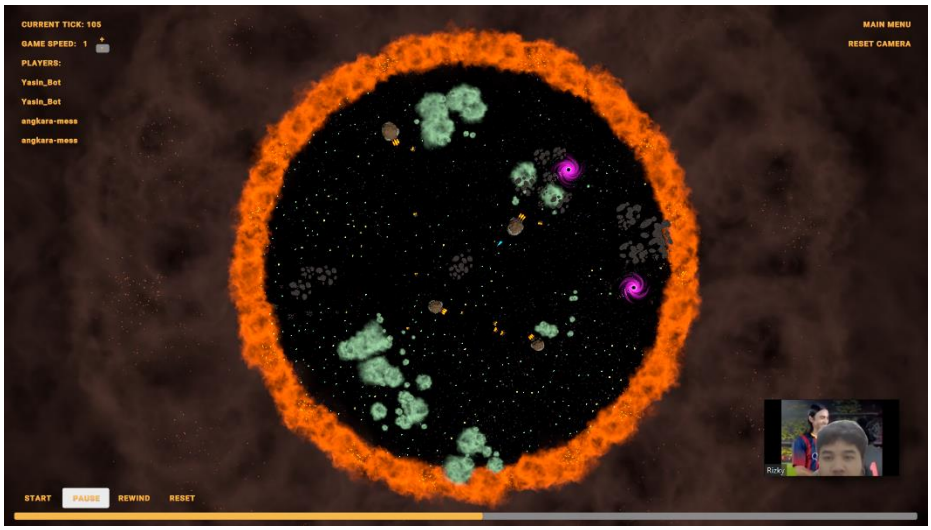
Keadaan ketika menyerang dan menghindar dari serangan(torpedo)




```
{"TotalTicks":202,"Players":[{"Placement":1,"Seed":13501,"Score":3
6,"Id":"473db553-54cb-4df4-bebb-
65863d7a07cf","Nickname":"Yasin_Bot","MatchPoints":8},{"Placement"
:4,"Seed":27767,"Score":29,"Id":"0af857d1-e920-472b-a74f-
f8c7f40b3480","Nickname":"angkara-
mess","MatchPoints":2},{"Placement":2,"Seed":20273,"Score":35,"Id"
:"58172777-004f-4553-8505-519e56f23384","Nickname":"angkara-
mess","MatchPoints":6},{"Placement":3,"Seed":11990,"Score":33,"Id"
:"938fbf09-c970-4ba1-80fc-
9dde07e82261","Nickname":"Yasin_Bot","MatchPoints":4}],"WorldSeeds
":[17634],"WinningBot":{"Id":"473db553-54cb-4df4-bebb-
65863d7a07cf","Size":90,"Speed":2,"GameObjectType":1,"CurrentHeadi
ng":342,"Position":{"X":-14,"Y":-82}}}
```

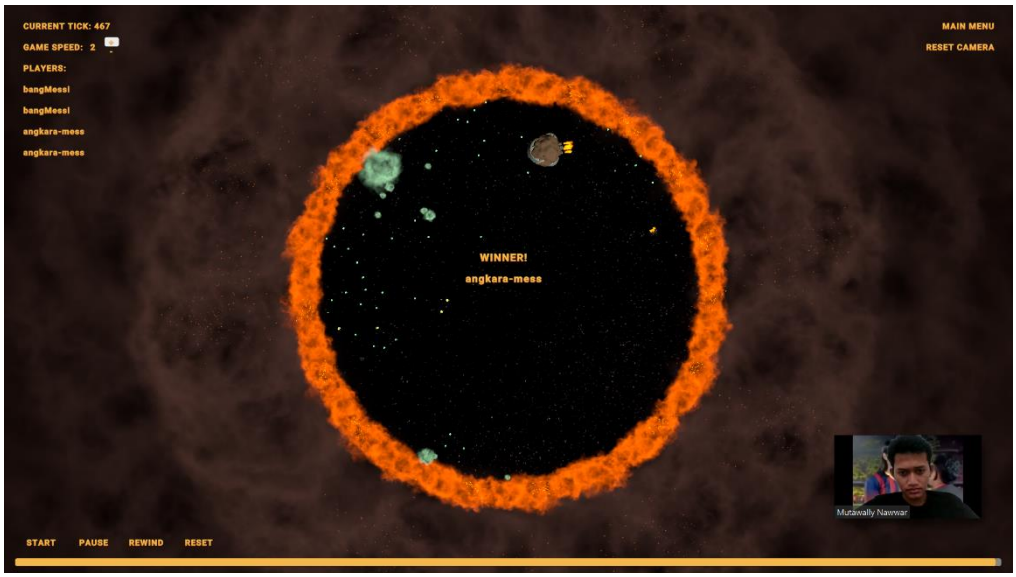
g. Pengujian 3

Menyalakan teleport ketika terdapat musuh dengan ukuran lebih kecil dari kita




```
{"TotalTicks":202,"Players":[{"Placement":1,"Seed":13501,"Score":36,"Id":"473db553-54cb-4df4-bebb-65863d7a07cf","Nickname":"Yasin_Bot","MatchPoints":8},{"Placement":4,"Seed":27767,"Score":29,"Id":"0af857d1-e920-472b-a74f-f8c7f40b3480","Nickname":"angkara-mess","MatchPoints":2},{"Placement":2,"Seed":20273,"Score":35,"Id":"58172777-004f-4553-8505-519e56f23384","Nickname":"angkara-mess","MatchPoints":6},{"Placement":3,"Seed":11990,"Score":33,"Id":"938fbf09-c970-4ba1-80fc-9dde07e82261","Nickname":"Yasin_Bot","MatchPoints":4}],"WorldSeeds":[17634],"WinningBot":{"Id":"473db553-54cb-4df4-bebb-65863d7a07cf","Size":90,"Speed":2,"GameObjectType":1,"CurrentHeading":342,"Position":{"X":-14,"Y":-82}}}
```

h. Pengujian 4




```
{"TotalTicks":467,"Players":[{"Placement":1,"Seed":27724,"Score":52,"Id":"559134ea-d61f-4ab0-93b2-d06a6290dee7","Nickname":"angkara-mess","MatchPoints":8},{"Placement":2,"Seed":14568,"Score":50,"Id":"7843e4b8-c86a-44dd-b6d4-3f75e9e4cd92","Nickname":"angkara-mess","MatchPoints":6},{"Placement":4,"Seed":15126,"Score":18,"Id":"e33d871a-eac7-43a5-80f4-7bc8cca63a41","Nickname":"bangMessi","MatchPoints":2},{"Placement":3,"Seed":2798,"Score":1,"Id":"6700c5c1-fcbf-4795-aac3-e667700226a8","Nickname":"bangMessi","MatchPoints":4}],"WorldSeeds":[21816],"WinningBot":{"Id":"559134ea-d61f-4ab0-93b2-d06a6290dee7","Size":50,"Speed":4,"GameObjectType":1,"CurrentHeading":188,"Position":{"X":100,"Y":382}}}
```

i. Pengujian 5


```
12.{"TotalTicks":530,"Players":[{"Placement":1,"Seed":24269,"Score":92,"Id":"4dedccc9-f4c5-437e-bb41-bd6fb10587fb","Nickname":"bangMessi","MatchPoints":8},{"Placement":4,"Seed":22726,"Score":25,"Id":"b8bad7c6-c921-4c15-8810-dd6f91f49cd4","Nickname":"bangMessi","MatchPoints":2},{"Placement":3,"Seed":5840,"Score":21,"Id":"c4621638-74a1-41de-806d-da54f0079dce","Nickname":"angkara-mess","MatchPoints":4},{"Placement":2,"Seed":12192,"Score":24,"Id":"923f7d99-6c65-4bd4-8ba9-a268596af231","Nickname":"angkara-mess","MatchPoints":6}],"WorldSeeds":[34007],"WinningBot":{"Id":"4dedccc9-f4c5-437e-bb41-bd6fb10587fb","Size":7,"Speed":30,"GameObjectType":1,"CurrentHeading":333,"Position":{"X":-370,"Y":257}}}
13.
```


Berdasarkan pengujian yang telah kami laksanakan melawan berbagai bot. Bot kami menang sebanyak 8 kali dari 10 kpercobaan. Sehingga, bisa dikatakan bahwa strategi greedy yang kami jalankan cukup optimal. Namun, jika dibandingkan dengan bot-bot buatan peserta Entelect, akan tidak seoptimal bot mereka karena mereka menggunakan machine learning yang menggunakan komputer untuk mendapat hasil yang optimal

BAB 5
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Kami berhasil mengimplementasikan algoritma greedy untuk membuat bot permainan Galaxio dengan Bahasa pemrograman Java yang bisa mencapai tujuan utama yaitu menjadi satu-satunya pemain yang bertahan hidup. Dari hasil yang didapat, penggunaan strategi greedy cukup optimal untuk memenangkan permainan ini dengan kemungkinan mendapat pilihan optimum yang cukup besar. Dari program yang dibuat perlu ditentukan teknik hurstik sehingga strategi menjadi efektif dan cocok untuk berbagai kasus. Jadi, bot yang kami program dengan algoritma greedy cukup baik dan berhasil melawan bot lain yang ada.

5.2. Saran

Kami menyadari bahwa dalam pengerjaan ini dapat dikembangkan lebih baik lagi, hal yang dapat diperbaiki adalah sebagai berikut:

1. Sebaiknya melakukan dekomposisi masalah terlabiih dahulu agar masalah dapat dipecah menjadi sub masalah sehingga pembagian tugas dapat dilakukan segera.
2. Perlu didalami dan dipelajari lebih lanjut sehingga mengurangi kendala padda saat perancangan strategi algoritma.
3. Laporan sebaiknya dicicil dari awal tugas mengingat isi dari laporan cukup banyak sehingga laporan dapat lebih sempurna dari sebelumnya.
4. Pengujian lebih baik dilakukan dengan banyak bot lain sehingga dapat mengukur kemampuan bot lebih jauh lagi.

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)-Bag2.pdf

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf