LAPORAN TUGAS BESAR 1 IF 2211 STRATEGI ALGORITMA PENERAPAN ALGORITMA GREEDY DALAM PERMAINAN WORMS

Diajukan sebagai salah satu tugas mata kuliah Strategi Algoritma pada Semester II Tahun Akademik 2020-2021



oleh

Fakhri Nail	**13519035**
Arsa Daris Gintara	**13519037**
Nizamixavier Rafif Lutvie	**13519085**

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2021

DAFTAR ISI

BAB I
DESKRIPSI MASALAH

Pada tugas besar kali ini, anda diminta untuk membuat sebuah *bot* untuk bermain permainan Worms yang telah dijelaskan sebelumnya. Untuk memulai, anda dapat mengikuti panduan singkat sebagai berikut.

1. Download *latest release starter pack zip* dari tautan berikut https://github.com/EntelectChallenge/2019-Worms/releases/tag/2019.3.2
2. Untuk menjalankan permainan, kaliah butuh beberapa requirement dasar sebagai berikut
   a. Java (minimal Java 8): https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html
   b. IntellIJ IDEA: https://www.jetbrains.com/idea/
   c. NodeJS: https://nodejs.org/en/download/
3. Untuk menjalankan permainan, kalian dapat membuka file "run.bat" (Untuk Windows/Mac dapat buka dengan double-click, Untuk Linux dapat menjalankan command "make run").
4. Secara default, permainan akan dilakukan diantara reference bot (default-nya berbahasa JavaScript) dan starter bot yang disediakan. Untuk mengubah hal tersebut, silahkan edit file "game-runner-config.json". Anda juga dapat mengubah file "bot.json" untuk mengatur informasi terkait bot anda.
5. Silahkan bersenang-senang dengan memodifikasi bot yang disediakan di starter-bot. Ingat bahwa bot kalian harus menggunakan bahasa Java dan di-build menggunakan IntelIJ. Dilarang menggunakan kode program tersebut untuk pemainnya atau kode program lain yang diunduh dari Internet. Mahasiswa harus membuat program sendiri, tetapi belajar dari program yang sudah ada tidak dilarang.
6. (Optional) Anda dapat melihat hasil permainan dengan menggunakan visualizer berikut https://github.com/dlweatherhead/entelect-challenge-2019-visualiser/releases/tag/v1.0f1
7. Untuk referensi lebih lanjut, silahkan eksplorasi di tautan berikut.

Strategi greedy yang diimplementasikan tiap kelompok harus dikaitkan dengan fungsi objektif dari permainan itu sendiri, yaitu memenangkan permainan dengan cara mengeliminasi seluruh worms lawan dengan senjata dan skill yang sudah disediakan dalam permainan. Salah satu contoh pendekatan greedy yang bisa digunakan (pendekatan tak terbatas pada contoh ini saja) adalah menyerang pasukan lawan dengan senjata dengan hitpoint / damage terbesar. Buatlah strategi greedy terbaik, karena setiap "pemain" dari masing-masing kelompok akan diadu satu sama lain dalam suatu kompetisi Tubes 1 (secara daring).

Strategi greedy harus dituliskan secara eksplisit pada laporan, karena akan diperiksa pada saat demo apakah strategi yang dituliskan sesuai dengan yang diimplementasikan. Tiap kelompok dapat menggunakan kreativitas mereka dalam menyusun strategi greedy untuk memenangkan permainan. Implementasi pemain harus dapat dijalankan pada game engine yang telah disebutkan pada spesifikasi

tugas besar, serta dapat dikompetisikan dengan pemain dari kelompok lain.

BAB II
LANDASAN TEORI

Algoritma Greedy

Algoritma greedy merupakan salah satu teknik untuk memecahkan persoalan optimasi agar menemukan solusi yang optimal. Algoritma memecahkan persoalan secara langkah per langkah, sehingga algoritma akan mengambil pilihan yang terbaik pada setiap langkah dan tidak bisa kembali pada langkah sebelumnya. Terdapat 6 komponen dalam algoritma greedy, yaitu:

1. Himpunan kandidat (C)

   Himpunan kandidat adalah himpunan yang berisi kandidat yang akan dipilih pada setiap langkah.

2. Himpunan solusi (S)

   Himpunan solusi adalah himpunan yang berisi kandidat yang sudah dipilih.

3. Fungsi solusi

   Fungsi solusi adalah fungsi untuk menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi.

4. Fungsi seleksi (selection function)

   Fungsi seleksi adalah fungsi untuk memilih kandidat berdasarkan strategi greedy tertentu.

5. Fungsi kelayakan (feasible)

   Fungsi kelayakan adalah fungsi untuk memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak).

6. Fungsi obyektif

   Fungsi obyektif adalah fungsi untuk mengoptimasi solusi, biasanya memaksimumkan atau meminimumkan.

Terdapat beberapa langkah untuk menjalankan algoritma greedy. Pertama, perlu dibuat sebuah himpunan solusi kosong untuk menyimpan pilihan atau langkah yang telah diambil. Kemudian, perlu dicek apakah himpunan belum merupakan solusi permasalahan dan masih terdapat kandidat pilihan yang dapat diambil. Jika benar, akan dipakai fungsi seleksi untuk mencari kandidat pilihan yang akan diambil. Jika sudah ditemukan, keluarkan kandidat pilihan tersebut dari himpunan kandidat pilihan. Setelah itu, akan digunakan fungsi kelayakan untuk mengetahui apakah kandidat pilihan yang akan diambil dapat dimasukkan ke dalam himpunan solusi. Jika layak, kandidat akan dimasukkan ke dalam himpunan solusi. Proses tersebut akan berulang sampai himpunan solusi sudah merupakan solusi permasalahan atau sudah tidak terdapat kandidat yang dapat diambil. Jika himpunan solusi merupakan solusi permasalahan, maka solusi telah ditemukan. Sedangkan jika himpunan solusi belum merupakan solusi permasalahan tetapi tidak terdapat solusi kandidat yang dapat diambil, maka tidak terdapat solusi bagi permasalahan.

Algoritma greedy bersifat heuristik, sehingga hanya akan membentuk solusi yang paling optimal secara lokal tetapi tidak selalu menghasilkan solusi yang paling optimal secara global. Hal ini disebabkan oleh strategi algoritma yang tidak beroperasi secara menyeluruh terhadap seluruh kemungkinan solusi yang

ada serta adanya beberapa kemungkinan fungsi seleksi yang berbeda sehingga harus memilih fungsi yang tepat untuk mendapatkan solusi yang paling optimal.Algoritma greedy dapat digunakan untuk menghasilkan solusi hampiran atau solusi yang mendekati solusi optimal jika waktu komputasi yang lebih pendek lebih penting daripada mendapatkan solusi yang terbaik.

Pemanfaatan Game Engine

Menambahkan pemain

Untuk menambahkan pemain pada game engine (starter-pack), pada file game-runner-config.json diubah player-a atau player-b Dalam kasus penulis, player-a perlu diubah untuk bisa menggunakan bahasa Java melawan reference bot yang menggunakan bahasa javascript. Bagian player-a diubah menjadi "./starter-bots/java" dan player-b tetap. Bot.json juga dimodifikasi untuk mengubah tampilan nama pada visualizer. Bagian nickName diubah menjadi nama yang diinginkan, dalam kasus penulis diubah menjadi “Master Tanur”.

Menambahkan strategi *greedy* pada permainan

Untuk menambahkan strategi greedy yang ingin diimplementasikan pada permainan, perlu dimodifikasi dan ditambah file-file java yang terletak pada folder src milik bot/pemain (folder starter-bots pada kasus penulis) yang ingin dimodifikasi. File utama yang diubah adalah bot.java dengan beberapa tambahan perubahan juga pada file-file pada folder command, entities, dan enums. Ditambahkan juga file-file java baru sebagai pendukung algoritma strategi greedy yang diletakkan pada folder command dan entities.

Cara menjalankan game engine

Untuk menjalankan game engine, folder java yang sudah dibuat di-*build* menggunakan Jetbrains IntelliJ IDEA. Setelah itu dijalankan run.bat itu menjalankan permainan.

BAB III

PEMANFAATAN STRATEGY *GREEDY*

Mapping Permasalahan Greedy

Sebelum mengaplikasikan strategi greedy pada permainan Worms, penulis harus memetakan elemen-elemen algoritma greedy terlebih dahulu. Terdapat enam elemen yang harus dipetakan, yaitu himpunan kandidat, himpunan solusi, fungsi solusi, fungsi seleksi, fungsi kelayakan dan fungsi objektif.

Himpunan kandidat dalam permasalahan ini adalah *command* yang bisa dilakukan oleh worm. Fungsi seleksi di sini adalah strategi yang ingin diterapkan dan yang akan dieksplorasi pada tugas besar ini. Fungsi solusi merupakan fungsi untuk menentukan apakah kandidat yang ada memberikan solusi dan menghasilkan *true* atau *false*, dalam hal ini yaitu mengecek apakah game masih berlanjut atau tidak. Kemudian, himpunan solusi adalah hasil dari fungsi solusi. Fungsi kelayakan adalah fungsi untuk mengecek *command* yang digunakan ilegal atau tidak.

Eksplorasi Alternatif Greedy

Greedy by Lowest Health

Eksplorasi fungsi seleksi yang pertama adalah greedy by health. Seleksi worm musuh yang akan diserang dilakukan berdasarkan jumlah *health point* yang dimilikinya. Worm musuh yang memiliki jumlah *health point* terkecil akan diserang terlebih dahulu. Worm dengan HP (*health point*) terkecil di awal game adalah Technologist dan Agent dengan 100 sehingga jika ingin menerapkan strategi ini diperlukan faktor selain HP atau pemilihan dilakukan secara random jika jumlah HP sama.

Pertimbangan untuk strategi ini adalah jika penulis membunuh worm musuh dengan cepat karena HP-nya kecil, maka akan bisa menyerang worm musuh lain tanpa dampak dan risiko tinggi.

Kelebihan strategi ini adalah worm milik penulis akan fokus terhadap satu worm musuh terlebih dahulu, lalu berganti target setelah musuh mati. Kekurangan strategi ini adalah karena letak worm-worm yang terpencar sehingga butuh waktu untuk semua worm berkumpul di satu titik dan ada resiko di tengah perjalanan diserang oleh worm musuh.

Greedy by Nearest Worm

Fungsi seleksi ini memperhitungkan jarak antara masing-masing worm milik penulis dengan masing-masing worm milik musuh dan memilih yang memiliki jarak terdekat antar keduanya. Dengan kondisi awal game berupa worm-worm yang terpencar, maka kemungkinan besar akan terbentuk tiga pasangan worm milik penulis dan musuh.

Kelebihannya adalah memprediksi hasil dari masing-masing pasangan akan lebih mudah karena kondisi satu lawan satu, sedangkan kekurangannya adalah jika worm milik penulis mendapatkan pasangan worm musuh yang bisa diprediksi akan menang maka kemungkinan besar worm milik penulis akan mati karena tidak ada yang bisa menginterupsi.

Greedy by Nearest Power Up

Fungsi seleksi ini mirip dengan fungsi seleksi berdasarkan jarak worm hanya saja yang dihitung jarak dengan power up. Dalam permainan Worms power up akan menambah HP worm yang mengambilnya dengan jumlah yang sudah ditentukan sebelumnya. Namun, jika HP worm yang mengambilnya masih penuh tidak akan terjadi apa-apa.

Fungsi seleksi ini lebih baik diterapkan dengan menggabungkan fungsi seleksi yang lain juga karena setelah powerup habis maka tidak ada target lagi yang bisa diperhitungkan.

Greedy by Weapon

Dengan fungsi seleksi ini, pilihan serangan worm diurutkan berdasarkan senjata khusus yang dimiliki worm. Senjata dengan *damage* terbesar adalah Banana Bomb dengan *damage* sejumlah 20, selanjutnya Shoot dengan 8 dan Snowball dengan 0.

Selama ada worm musuh di dalam *range* senjata akan diserang dengan senjata khusus jika ada, jika tidak ada maka akan menggunakan Shoot. Khusus untuk Snowball dan Banana Bomb, worm akan menyerang meskipun dihalangi oleh tembok. Fungsi seleksi ini akan dijalankan setiap kali ada worm musuh yang berada di dalam *range* worm penulis.

Greedy by Nearest Entity

Fungsi seleksi ini mirip dengan greedy by nearest worm dan power up, hanya saja lebih umum. Entity yang dimaksud disini adalah worm dan powerup, sehingga fungsi seleksi akan menghitung mana entity terdekat dari worm dan memilih aksi berdasarkan informasi tersebut.

Dengan fungsi seleksi ini, kelemahan fungsi seleksi worm terdekat dan powerup terdekat akan saling tertutupi. Power up akan dapat digunakan dengan lebih baik dan worm penulis akan bisa menginisiasi serangan tanpa harus menunggu worm musuh datang.

Pemilihan Strategi Greedy

Strategi yang dipilih oleh penulis adalah Greedy by Nearest Entity untuk pergerakan dan Greedy by Highest Attack untuk penyerangan. Dengan membagi greedy untuk pergerakan dan penyerangan maka diharapkan akan tercapai hasil yang lebih optimal.

Pertimbangan memilih Greedy by Nearest Entity adalah karena algoritmanya yang paling komplit dalam hal menyerang dan bertahan, worm akan mendapat kesempatan menambah HP miliknya jika berada di dekat powerup dan akan bisa menginisiasi penyerangan jika berada di dekat worm musuh.

Sedangkan, pertimbangan dalam memilih Greedy by Weapon adalah karena dengan menyerang dengan senjata khusus sejak awal maka dampak yang ditimbulkan menjadi lebih besar, selain itu juga untuk menghindari kondisi dimana *damage* yang bisa dikeluarkan melebihi sisa dari HP musuh sehingga akan sia-sia.

Algoritma greedy gabungan ini dijalankan dengan prioritas menyerang lebih dahulu, setelah itu jika tidak memungkinkan akan bergerak menuju entitas terdekat. Dengan algoritma ini, perhitungan kompleksitas fungsi seleksi adalah $O(n^2)$. Dalam mencari worm musuh yang berada di dalam *range* akan dilakukan perbandingan paling banyak sebesar n^2 karena menyusuri sekeliling worm berbentuk luas kotak. Lalu, dalam proses mendapatkan posisi worm dan power up terdekat pun juga menggunakan cara yang sama.

Namun, efektivitas algoritma penulis masih belum optimal. Dengan membuat perbandingan dengan ReferenceBot yang ada, algoritma greedy penulis tidak bisa mendapatkan solusi optimum global dengan frekuensi tinggi yang berakibat pada kekalahan game.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

Pseudocode

function run() → Command

Deklarasi

inRangeWorm, nearestWorm : Worm

nearestPowerUp : Cell

Algoritma

if (inRangeWorm != null) then

→ attackNearest(inRangeWorm)

else

→ moveNearestEntity(nearestWorm, nearestPowerUp)

endif

function isOccupied(cell: Cell) → boolean

{menghasilkan boolean apakah suatu cell telah ditempati worm atau tidak}

Deklarasi

Algoritma

→ cell.ocupier != null

function euclideanDistance(aX, aY, bX, bY: integer) → integer

{menghasilkan integer berupada jarak antar titik}

Deklarasi

Algoritma

→ integer(((aX-bX)**2 + (aY-bY)**2)**0.5)

function isValidCoordinate(x, y: integer) → boolean

{menghasilkan boolean apakah suatu koordinat itu valid atau tidak (didalam mapSize)}

Deklarasi

Algoritma

→ x ≥ 0 and x < gameState.mapSize and y ≥ 0 and y < gameState.mapSize

function canSnowball() → boolean

{Menghasilkan boolean apakah worm dapat melakukan snowball atau tidak}

Deklarasi

{currentWorm telah terdefinisi secara global di program utama}

Algoritma

→ (currentworm.id = 3 and currentWorm.snowballs.count > 0)

function canBananaBombs() → boolean

{Menghasilkan boolean apakah worm dapat melakukan banana bombs atau tidak}

Deklarasi

{currentWorm telah terdefinisi secara global di program utama}

Algoritma

→ (currentworm.id = 2 and currentWorm.bananaBombs.count > 0)

function getNearestWorm() → Worm

{Menghasilkan worm musuh terdekat dari currentWorm}

Deklarasi

{currentWorm, opponent telah terdefinisi secara global di program utama}

nearestWorm, enemyWorm : Worm

nearestDistance, tempDistance : integer

Algoritma

nearestWorm ← null

nearestDistance ← 999999

foreach enemyWorm **in** opponent.worms

tempDistance ← euclideanDistance(enemyWorm.position.x, enemyWorm.position.y, currentWorm.position.x, currentWorm.position.y)

if (tempDistance < nearestDistance and enemyWorm.health > 0) **then**

nearestDistance ← tempDistance

nearestWorm ← enemyWorm

endif

endforeach

→ nearestWorm

function getNearestPowerUp() → Cell

{Menghasilkan Cell terdekat yang terkandung powerup atau bernilai null jika tidak ada powerup yang tersisa}

Deklarasi

{currentWorm, gameState telah terdefinisi secara global}

tempDistance, nearestDistance, i, j : integer

nearestPowerUp : Cell

Algoritma

nearestPowerUp ← null

nearestDistance ← 999999

i **traversal** [0 … gameState.mapSize]

j **traversal** [0 … gameState.mapSize]

if (gamestate.map[i][j] != null) **then**

tempDistance ← euclideanDistance(gameState.map[i][j].x, gameState.map[i][j].y, currentWorm.position.x, currentWorm.position.y)

if (tempDistance < nearestDistance) **then**

nearestDistance ← tempDistance

nearestPowerUp = gamestate.map[i][j]

endif

endif

→ nearestPowerUp

function attackNearest(enemyWorm: Worm) → Command

{Menghasilkan command untuk attack worm terdekat dari currentWorm}

Deklarasi

{currentWorm telah terdefinisi secara global}

Algoritma

```
    if (canSnowball()) then
        → SnowballCommand(enemyWorm.position.x, enemyWorm.position.y)
    else if (canBananaBombs()) then
        → BananaBombCommand(enemyWorm.position.x, enemyWorm.position.y)
    else
        direction ← resolveDirection(currentWorm.position, enemyWorm.position)
        → ShootCommand(direction)
    endif


function moveNearest(x, y: integer) → Command
{Menghasilkan command untuk worm bergerak}
Deklarasi
    {currentWorm telah terdefinisi secara global}
    moveX, moveY, subCurrX, subCurrY : integer
    surroundingBlocks : List<Cell>
    cell : Cell
Algoritma
    moveX ← currentWorm.position.x
    moveY ← currentWorm.position.y
    subCurrX ← x - currentWorm.position.x
    subCurrY ← y - currentWorm.position.y

    if (subCurrX > 0) then
        moveX ← moveX + 1
    else if (subCurrX < 0) then
        moveX ← moveX - 1
    endif

    if (subCurrX > 0) then
        moveX ← moveX + 1
    else if (subCurrY < 0) then
        moveY ← moveY + 1
    endif
```

```
        surroundingBlocks ← getSurroundingCells(currentWorm.position.x, currentWorm.position.y)
        foreach cell in surroundingBlocks
            if (cell.x = moveX and cell.y = moveY) then
                if (cell.type = CellType.DIRT) then
                    → DigCommand(moveX, moveY)
                else if (cell.type = CellType.AIR or cell.type = CellType.LAVA) then
                    if (not(isOccupied(cell))) then
                        → MoveCommand(moveX, moveY)
                break
                endif
            endif
        endforeach

        → DoNothingCommand()

function moveNearestEntity(nearestWorm: Worm, nearestPowerUp: Cell) → Command
{Menghasilkan command untuk worm bergerak terdekat nearestPowerUp atau nearestWorm}
Deklarasi
        isWormNearest: boolean
        nearestWormDistance, nearestPowerUpDistance: integer
Algoritma
        if (nearestPowerUp != null) then
            nearestWormDistance ← euclideanDistance(enemyWorm.position.x,
            enemyWorm.position.y, currentWorm.position.x, currentWorm.position.y)
            nearestPowerUpDistance ← euclideanDistance(nearestPower.x, nearestPower.y,
            currentWorm.position.x, currentWorm.position.y)
            isWormNearest ← nearestWormDistance > nearestPowerUpDistance

            if (isWormNearest) then
                → moveNearest(nearestWorm.position.x, nearestWorm.position.y)
            else
                → moveNearest(nearestPowerUp.x, nearestPowerUp.y)
```

→ moveNearest(nearestWorm.position.x, nearestWorm.position.y)

Struktur Data

Pada program ini banyak struktur data yang didefinisikan yaitu Gamestate, Position, Cell, Worm, Myplayer, MyWorm, Opponent, PowerUp, Weapon, BananaBombs, Snowballs.

Struktur data Gamestate merepresentasikan state secara umum dalam program ini yang terdiri dari tipe data integer currentRound, maxRounds, mapSize, currentWormId, consecutiveDoNothingCount, lalu bertipe data bentukan Myplayer yaitu myPlayer, array of Opponent yaitu opponents, dan matriks of Cell yaitu map.

Struktur data Position merepresentasikan posisi suatu entitas di dalam map/cell, struktur ini terdiri dari dua anggota bertipe data integer yaitu x dan y.

Struktur data Cell merepresentasikan setiap sel yang ada dalam map pada program ini, struktur ini terdiri dari tipe data integer x dan y, tipe data bentukan CellType yaitu type, PowerUp yaitu powerUp, dan Worm yaitu occupier.

Struktur data Worm merepresentasikan worm pada game ini yang terdiri dari tipe data integer id, health, diggingRange, movementRange, dan roundUntilUnfrozen, lalu ada tipe data bentukan Position yaitu position.

Struktur data MyPlayer merepresentasikan penulis termasuk worm yang terdiri dari tipe data integer id, score, dan health dan array of MyWorm yaitu worms.

Struktur data MyWorm merepresentasikan worm penulis yang merupakan pewarisan dari struktur data Worm dengan tambahan tipe data bentukan Weapon yaitu weapon, Snowballs yaitu snowballs, BananaBombs yaitu bananaBombs.

Struktur data Opponent merepresentasikan lawan termasuk worm yang terdiri dari tipe data integer id dan score dan array of Worm yaitu worms.

Struktur data PowerUp merepresentasikan powerup pada game ini yang terdiri dari tipe data PowerUpType yaitu type dan integer value.

Struktur data Weapon merepresentasikan senjata default yang dimiliki tiap worm terdiri dari tipe data integer yaitu damage dan range.

Struktur data BananaBombs merepresentasikan Banana Bombs yang dimiliki worm bertipe Agent terdiri dari tipe data integer yaitu damage, range, count, dan damageRadius.

Struktur data Snowballs merepresentasikan Snowballs yang dimiliki worm bertipe Technologist terdiri dari tipe data integer yaitu freezeDuration, range, count, dan freezeRadius.

Analisis Desain

Strategi greedy yang diimplementasi belum bisa mencapai nilai optimum global dengan frekuensi tinggi. Strategi membutuhkan kondisi khusus agar bisa mendapatkan nilai optimum, yaitu ketika Snowball dari worm penulis mengenai dua worm musuh. Sedangkan, jika kondisi tersebut tidak terpenuhi seringkali berakibat kekalahan.

Selain itu, algoritma greedy tidak memperhitungkan jarak antar worm sehingga terkadang Snowball atau Banana Bomb yang digunakan oleh sebuah worm terkena worm itu sendiri ataupun worm

teman. Pemakaian Snowball dan Banana Bomb juga tidak bisa mengenai lebih dari 1 worm musuh dengan konsisten.

Dengan desain algoritma seperti ini juga tidak memperhitungkan penggunaan Snowball secara optimal, yaitu dengan menunggun enemy lepas dari status *frozen* terlebih dahulu lalu menggunakan kembali Snowball.

BAB V

KESIMPULAN DAN SARAN

Kesimpulan

Algoritma greedy merupakan salah satu algoritma yang dapat diimplementasikan pada permainan Worms. Setelah beberapa percobaan, penerapan algoritma tersebut memberikan hasil yang kurang optimal. Hal ini disebabkan oleh sifat algoritma yang hanya memberikan solusi terbaik secara lokal sehingga tidak selalu memberikan solusi yang terbaik secara global.

Saran

Penulis menyadari bahwa masih terdapat banyak kekurangan dalam pembuatan laporan ini. Penulis berharap bahwa kemungkinan-kemungkinan solusi lain dari strategi algoritma greedy dapat dijelajahi lebih lanjut agar dapat menemukan solusi yang terbaik untuk permainan.

DAFTAR PUSTAKA

1. http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf diakses 18 Februari 2021
2. http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Besar-1-IF2211-Strategi-Algoritma-2021.pdf diakses 10 Februari 2021