PEMANFAATAN ALGORITMA GREEDY DALAM APLIKASI PERMAINAN "WORMS"

LAPORAN TUGAS BESAR 1

Diajukan sebagai salah satu Tugas Besar 1

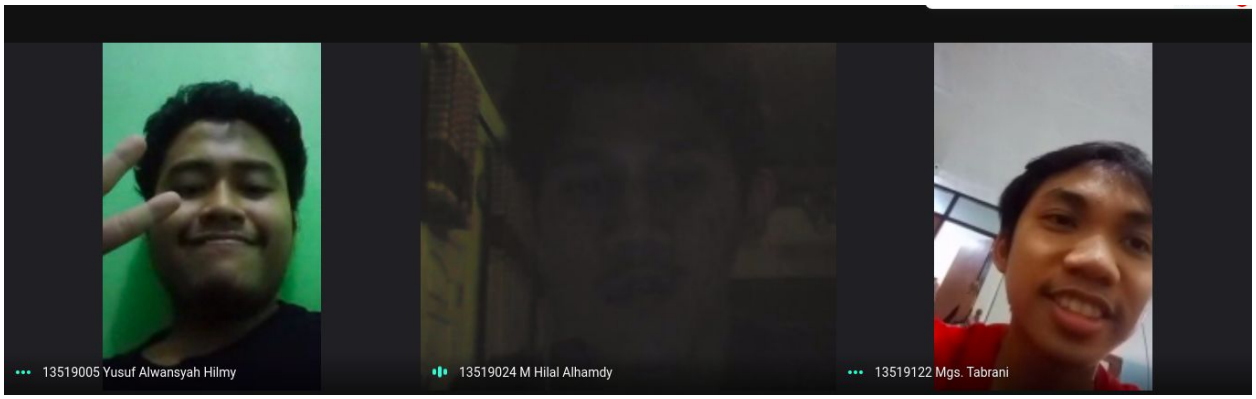
IF2211 Strategi Algoritma Semester II tahun 2020/2021

Oleh:

Yusuf Alwansyah Hilmy (13519005)

M. Hilal Alhamdy (13519024)

Mgs. Tabrani (13519122)



PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2021

BAB 1
DESKRIPSI TUGAS

Worms adalah sebuah turned-based game yang memerlukan strategi untuk memenangkannya. Setiap pemain akan memiliki 3 worms dengan perannya masing-masing. Pemain dinyatakan menang jika ia berhasil bertahan hingga akhir permainan dengan cara mengeliminasi pasukan worms lawan menggunakan strategi tertentu.

Pada tugas besar pertama Strategi Algoritma ini, gunakanlah sebuah game engine untuk mengimplementasikan permainan Worms. Game engine dapat diperoleh pada laman berikut https://github.com/EntelectChallenge/2019-Worms.

Tugas mahasiswa adalah mengimplementasikan seorang "pemain" Worms, dengan menggunakan strategi greedy untuk memenangkan permainan. Untuk mengimplementasikan seorang "pemain" tersebut, mahasiswa disarankan melanjutkan program yang terdapat pada starter bot di dalam starter pack pada laman berikut ini: (https://github.com/EntelectChallenge/2019-Worms/releases/tag/2019.3.2)

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

1. Peta permainan berukuran 33x33 cells. Terdapat 4 tipe cell, yaitu air, dirt, deep space, dan lava yang masing-masing memiliki karakteristik berbeda. Cell dapat memuat powerups yang bisa diambil oleh worms yang berada pada cell tersebut.

2. Di awal permainan, setiap pemain akan memiliki 3 pasukan worms dengan peran dan nilai health points yang berbeda, yaitu::

    a. Commando

    b. Agent

    c. Technologist

3. Pada setiap round, masing-masing pemain dapat memberikan satu buah command untuk pasukan worm mereka yang masih aktif (belum tereliminasi). Berikut jenis-jenis command yang ada pada permainan:

    a. Move

    b. Dig

    c. Shot

d. Do Nothing

e. Banana Bomb

f. Snowball

g. Select

4. Command dari kedua pemain akan dieksekusi secara bersamaan (bukan sekuensial) dan akan divalidasi terlebih dahulu. Command juga akan dieksekusi sesuai urutan prioritas tertentu.

5. Beberapa command, seperti shot dan banana bomb dapat memberikan damage pada worms target yang terkena serangan, sehingga mengurangi health pointsnya. Jika health points suatu worm sudah habis, maka worm tersebut dinyatakan tereliminasi dari permainan.

6. Permainan akan berakhir ketika salah satu pemain berhasil mengeliminasi seluruh pasukan worms lawan atau permainan sudah mencapai jumlah round maksimum (400 rounds).

Adapun peraturan yang lebih lengkap dari permainan Worms, dapat dilihat pada laman https://github.com/EntelectChallenge/2019-Worms/blob/develop/game-engine/game-rules.md.

Spesifikasi tugas:

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, kalian butuh beberapa requirement dasar sebagai berikut.

a. Java (minimal Java 8):
https://www.oracle.com/java/technologies/javase/javasejdk8-downloads.html

b. IntelIJ 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 2
LANDASAN TEORI

Algoritma *Greedy* merupakan metode yang paling populer dan sederhana untuk memecahkan persoalan optimasi. Persoalan optimasi tersebut biasanya beruapa persoalan mncari solusi optimal. Hanya ada dua macam persoalan optimasi, yaitu maksimasi (*maximation*) dan minimasi (*minimation*).

Algoritma *Greedy* adalah algoritma yang memecahkan persoalan secara langkah per langkah (*step by step*) sedemikian sehingga pada setiap langkah akan mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan dan berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Elemen-elemen pada algoritma *greedy* meliputi:

1. Himpunan kandidat, yang berisi kandidat yang akan dipilih pada setiap langkah.
2. Himpunan solusi, yang berisi kandidat yang sudah dipilih.
3. Fungsi solusi, yang berfungsi menentukan apakah himpunan kandidat yang dipilih (himpunan solusi) sudah memberikan solusi.
4. Fungsi seleksi, yang berfungsi memilih kandidat berdasarkan strategi *greedy* tertentu. Strategi *greedy* ini bersifat heuristik.
5. Fungsi kelayakan, yang berfungsi memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak).
6. Fungsi obyektif, yang berfungsi memaksimumkan atau meminimumkan.

Optimum global belum tentu merupakan solusi optimum (terbaik), bisa jadi merupakan solusi *sub-optimum* atau *pseudo-optimum* karena algoritma *greedy* tidak beroperasi secara menyeluruh terhadap semua kemungkinan solusi yang ada. Jadi, pada sebagian persoalan, algoritma *greedy* tidak selalu berhasil memberikan solusi yang optimal, namun sub-optimal. Jika solusi terbaik mutlak tidak terlalu diperlukan, maka algoritma *greedy* dapat digunakan untuk menghasilkan solusi hampiran, daripada menggunakan algoritma yang kebutuhan waktunya eksponensial untuk menghasilkan solusi yang eksak. Contoh-contoh persoalan yang dapat diselesaikan dengan algoritma *greedy* yaitu persoalan penukaran uang, persoalan memilih aktivitas, minimasi waktu di dalam sistem, persoalan *knapsack*, penjadwalan *job* dengan tenggat waktu, pohon merentang minimum, lintasan terpendek, kode Huffman, dan pecahan Mesir.

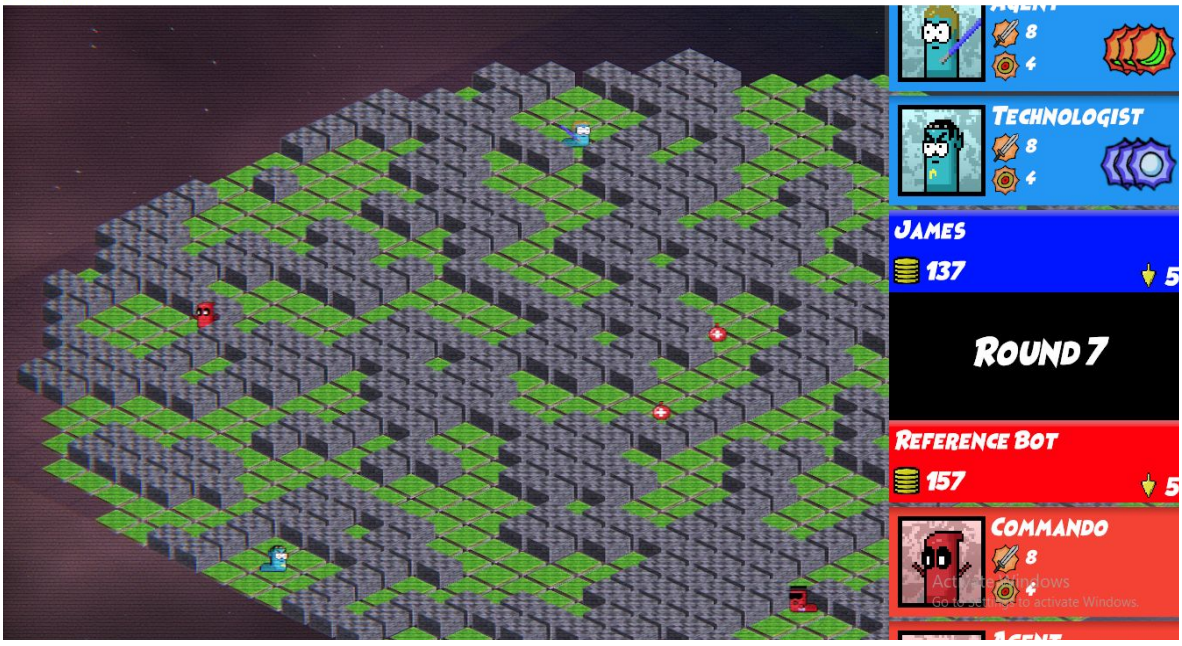
BAB 3

PEMANFAATAN STRATEGI GREEDY

Pada permainan Worms, pemenang ditentukan berdasarkan siapa yang paling lama bertahan. Para pemain harus saling mengalahkan. Pemain diberikan 3 *worms* masing-masing. Strategi *greedy* dapat diterapkan dalam permainan ini karena sifat algoritma *greedy* yang mencari solusi berupa optimasi, baik minimum ataupun maksimum. Berikut beberapa strategi *greedy* yang dapat diterapkan dalam permainan Worms.

a. Greedy by Location

Greedy by location, yaitu strategi *greedy* yang memanfaatkan panjang lintasan atau jarak dari suatu permasalahan. Pada kasus permainan Worms ini, strategi *greedy by location* digunakan dengan cara mencari panjang lintasan atau jarak terpendek antara *worms* pemain dengan *worms* lawan. Pemain akan memilah setiap musuh yang ada di hadapannya. Apabila salah satu musuh memiliki jarak yeng lebih pendek kepada pemain daripada musuh lain, pemain akan menyerang musuh itu terlebih dahulu.



b. Greedy by Farming

Greedy by farming, yaitu strategi *greedy* yang memanfaatkan Dirt sebanyak mungkin untuk dihancurkan, yang mana akan menambah poin *worms* sehingga dapat mengungguli poin lawan. Permainan memiliki batas ronde. Jika batas ronde telah habis dan permainan berakhir tanpa ada yang tereliminasi, permainan akan ditentukan berdasarkan pemain yang memiliki poin lebih banyak. Hal inilah yang membuat strategi ini dapat dimanfaatkan jika ingin bermain bertahan.

c. Greedy by Weapon Choices

Greedy by weapon, yaitu strategi *greedy* yang melihat senjata yang akan digunakan dalam menghadapi musuh. Pemain akan memilih senajata apa yang akan digunakan berdasarkan *health point* musuh. Apabila musuh memiliki *health point* yang lebih besar dari pemain, pemain akan memilih senjata yang berjumlah lebih banyak. Sedangkan, apabila musuh memiliki *health point* yang lebih kecil dariapada pemain, pemain akan memilih senjata yang berjumlah lebih banyak. Dengan demikian, pemakaian senjata dalam permainan dapat dilakukan dengan efektif.

d. Greedy by Health Point

Greedy by health point, yaitu strategi yang *greedy* yang melihat jumlah *health point* lawan. Dalam strategi ini, pemain akan mencari musuh yang memiliki *health point* paling sedikit dan akan menyerang musuh tersebut. Dengan menyerang musuh yang memiliki *health point* lebih kecil terlebih dahulu, kemungkinan untuk menyerang lebih banyak musuh menjadi lebih besar.

BAB 4

IMPLEMENTASI DAN PENGUJIAN

A. Analisis & Implementasi

Algoritma *Greedy* yang kami gunakan adalah *greedy by location* yaitu strategi yang memprioritaskan berdasarkan posisi-posisi cacing. Kami mengincar lokasi tengah karena lokasi tengah paling terakhir menjadi lava.

Pada Implementasinya, kami menggunakan kode berikut:

```
public Command run() {

        MyWorm myTeam = getCurrentWorm(gameState);
        List<Cell> surround = getSurroundingCells(myTeam.position.x, myTeam.position.y);
        if (gameState.map[myTeam.position.x][myTeam.position.y].type == CellType.LAVA) {
            for (Cell cell : surround) {
                if (cell.type != CellType.LAVA) {
                    if (cell.type == CellType.DIRT) {
                        return new DigCommand(cell.x, cell.y);
                    }
                    return new MoveCommand(cell.x, cell.y);
                }
            }
        }

        Worm enemy = getFirstWormInRange();
        if (enemy != null) {
            Direction dir = resolveDirection(currentWorm.position, enemy.position);
            return new ShootCommand(dir);
        }

        for (Cell num : surround) {
            if (num.powerUp != null) {
                return new MoveCommand(num.x, num.y);
            }
        }

        Direction now = toCenter(myTeam.position);
        if (now != null) {
                Cell c = surround.stream()
                        .filter(w -> (w.x == myTeam.position.x + now.x) && (w.y == myTeam.position.y + now.y))
                        .findFirst()
                        .get();
                if (c.type.equals(CellType.DIRT)) {
                    return new DigCommand(myTeam.position.x + now.x, myTeam.position.y + now.y);
                }
                return new MoveCommand(myTeam.position.x + now.x, myTeam.position.y + now.y);
        }

        return new DoNothingCommand();
```

Algoritma *greedy* yang kami pakai ada pada bagian berikut:

```
Direction now = toCenter(myTeam.position);
        if (now != null) {
                Cell c = surround.stream()
                        .filter(w -> (w.x == myTeam.position.x + now.x) && (w.y == myTeam.position.y +
now.y))
                        .findFirst()
                        .get();
                if (c.type.equals(CellType.DIRT)) {
                    return new DigCommand(myTeam.position.x + now.x, myTeam.position.y + now.y);
                }
                return new MoveCommand(myTeam.position.x + now.x, myTeam.position.y + now.y);
        }
```

Pada bagian tersebut akan dilakukan pencarian arah untuk berjalan ke tengah melalui variabel *now*. Variabel *now* akan bernilai null jika sudah ditengah, selain itu akan mengembalikan arah yang menuju ke tengah. Jika sudah didapat arahnya maka akan dicek *tile* yang dituju apakah DIRT atau AIR. Jika DIRT, maka akan melakukan DigCommand yang melakukan perintah *dig* sedangkan jika AIR, akan melakukan MoveCommand yang melakukan perintah *move*.

Elemen dalam algoritma *greedy*:

1. Himpunan Kandidat: d

Selain itu terdapat strategi tambahan untuk mendukung berjalannya permainan sebagai berikut:

```
MyWorm myTeam = getCurrentWorm(gameState);
        List<Cell> surround = getSurroundingCells(myTeam.position.x, myTeam.position.y);
        if (gameState.map[myTeam.position.x][myTeam.position.y].type == CellType.LAVA) {
            for (Cell cell : surround) {
                if (cell.type != CellType.LAVA) {
                    if (cell.type == CellType.DIRT) {
                        return new DigCommand(cell.x, cell.y);
                    }
                    return new MoveCommand(cell.x, cell.y);
                }
            }
        }
```

Bagian ini akan menjalankan program ketika *worms* berada dalam *cell* LAVA. Saat *worms* berada di LAVA, maka program akan memerintahkan *worms* untuk beranjak dari tempatnya.

```
Worm enemy = getFirstWormInRange();
if (enemy != null) {
    Direction dir = resolveDirection(currentWorm.position, enemy.position);
    return new ShootCommand(dir);
}
```

Bagian ini menjalankan perintah *shoot* saat ada *worm* musuh yang dekat.

```
for (Cell num : surround) {
    if (num.powerUp != null) {
        return new MoveCommand(num.x, num.y);
    }
}
```

Bagian ini akan mengarahkan *worms* menuju *Power Up* jika ada disekitarnya.

B. Pengujian

Pada pengujian ini program yang kami buat sudah cukup sesuai keinginan kecuali terdapat masalah pada *worms* dengan id 3. *Worms* tersebut tidak dapat menemukan arah pada setiap gilirannya. Kami tidak tau pastinya namun kemungkinan terbesar terdapat kesalahan pada algoritma penentuan sekitarnya (getSurrounding).

BAB 5
KESIMPULAN DAN SARAN

A. Kesimpulan

Strategi *greedy* dapat dimanfaatkan untuk menyelesaikan permasalahan optimas dalam kehidupan sehari-hari. Salah satu penerapan strategi *greedy* dapat ditemukan dalam sebuah permainan, seperti permainan Worms. Beberapa strategi *greedy* yang dapat diterapkan adalah *greedy by location*, *greedy by farming*, *greedy by weapon choice*, *greedy by health point*.

B. Saran

Cakupan dalam strategi *greedy* sangatlah luas, sehingga tidak seluruh strategi *greedy* dapat dimasukkan dalam laporan ini. Penulisan laporan ini diharapkan dapat memberikan referensi kepada peneliti lain agar dapat memanfaatkan strategi *greedy* dalam permasalahan lain.

DAFTAR PUSTAKA

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Besar-1-IF2211-Strategi-Algoritma-2021.pdf (diakses pada 14 Februrari 2021).

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf (diakses pada 17 Februari 2021).

Rahasta, Diani, 2009. Aplikasi Algoritma Greedy untuk Menyelesaikan Permainan Hedgewars, Program Studi Teknik Informatika ITB. (diakses pada 19 Februari 2021).