Tugas Besar 1

Greedy Treasure Hunter

IF2211 Strategi Algoritma

Disusun Oleh :

Viktor Trimulya Buntoro / 13512038

Michael Alexander Wangsa / 13512046

Dariel Valdano / 13512079

Program Studi Teknik Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

I. Deskripsi Masalah

Dalam permainan Greedy Treasure Hunter, pemain akan mengumpulkan koin dari peti harta karun. Area permainan digambarkan sebagai graf dengan jarak antar simpul yang terdefinisi. Simpul dapat merepresentasikan posisi pemain, posisi semua peti, atau posisi kosong. Waktu yang dibutuhkan untuk berpindah dari satu simpul ke simpul lainnya sesuai dengan jarak yang dibutuhkan, misalnya jarak pemain ke suatu peti 5, artinya membutuhkan 5 satuan waktu.

Terdapat 3 warna yang berbeda untuk peti harta karun. Peti merah berisi 5 koin, peti kuning berisi 3 koin, dan peti hijau berisi 1 koin. Untuk mendapatkan koin dari suatu peti, pemain harus menggunakan tipe peralatan yang tepat untuk membuka peti karena peti akan meledak jika tidak terbuka dalam waktu tertentu. Semakin banyak koin di dalam peti, semakin cepat peti tersebut akan meledak. Peti merah akan meledak dalam 2 satuan waktu, peti kuning akan meledak dalam 3 satuan waktu, dan peti hijau akan meledak dalam 4 satuan waktu.

Di awal permainan, pemain mendapat modal uang untuk membeli sejumlah peralatan yang tersedia. Peralatan dibagi menjadi 3 tipe yaitu normal, 2x, 4x. Peralatan tipe normal akan membuka peti selama 4 satuan waktu (hanya bisa untuk peti hijau), sedangkan peralatan tipe 2x atau 4x akan membuka peti lebih cepat 2x (2 satuan waktu; untuk peti kuning atau hijau) atau 4x (1 satuan waktu; untuk semua peti). Peralatan yang sudah dipakai tidak dapat digunakan lagi untuk membuka peti yang lain, sehingga pemain perlu memilih jumlah peralatan yang akan dibeli untuk setiap tipe peralatan. Harga peralatan normal adalah satu koin, sedangkan peralatan 2x adalah dua koin, dan harga peralatan 4x adalah tiga koin.

Dalam permainan, didefinisikan area permainan, modal awal pemain, target koin yang harus dikumpulkan oleh pemain, jumlah peti untuk setiap tipenya. Posisi pemain dan posisi semua peti dibangkitkan secara random. Skor pada permainan ini adalah sisa modal awal ditambah jumlah koin yang didapatkan untuk durasi waktu tertentu. Pemain harus berusaha menggunakan modal sesedikit mungkin tetapi memperoleh koin sebanyak-banyaknya.

Di dalam tugas ini, Anda diminta mengaplikasikan algoritma greedy untuk memenangkan permainan Greedy Treasure Hunter ini. Program yang dibuat harus memperlihatkan permainan dengan alternatif sebagai berikut:

- Pemain komputer sebagai greedy hunter. Pemain dianggap menang jika skor yang didapatkan melebihi batas koin yang harus dikumpulkan.
- Pemain manusia sebagai hunter, dengan pemain komputer sebagai hunter saingan (greedy hunter), dan sejumlah penumpang dibangkitkan secara random. Pemain yang menang memiliki skor yang lebih tinggi.

Untuk alternatif 1, tidak ada pemain manusia, tetapi hanya memperlihatkan simulasi permainan saja. Ketika komputer sebagai greedy hunter, komputer akan mengaplikasikan algoritma greedy untuk melakukan penentuan peti mana saja yang akan didatangi dan jenis peralatan yang akan digunakan untuk membuka peti tersebut, termasuk menentukan rute

yang akan dilalui. Penentuan ini mempertimbangkan posisi setiap peti, posisi pemain, peralatan yang dimiliki, waktu untuk membuka peti, deadline ledakan setiap jenis peti, dan jumlah koin yang akan didapatkan.

Anda harus merancang minimal masing-masing dua buah strategi greedy yang berbeda (greedy by X dan greedy by Y) untuk penyusunan rute pemain mengumpulkan koin dari setiap peti yang akan didatangi. Selain itu, penentuan lintasan terpendek dari suatu posisi ke posisi yang lain juga dapat menggunakan strategi greedy. Deskripsi algoritma greedy harus dapat memperlihatkan properti algoritmanya, yaitu himpunan kandidat, himpunan solusi, fungsi seleksi, fungsi kelayakan, dan fungsi obyektif.

II. Dasar Teori

Algoritma *greedy* membentuk solusi langkah per langkah (*step by step)*. Terdapat banyak pilihan yang perlu dieksplorasi pada setiap langkah solusi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya. Sebagai contoh, jika kita menggunakan algoritma *greedy* untuk menempatkan komponen di atas papan sirkuit, sekali sebuah komponen telah ditetapkan posisinya, komponen tersebut tidak dapat dipindahkan lagi.

Pendekatan yang digunakan di dalam algoritma *greedy* adalah membuat pilihan yang "tampaknya" memberikan perolehan terbaik, yaitu dengan membuat pilihan **optimum lokal** pada setiap langkah dengan harapan bahwa sisanya mengarah ke solusi **optimum global.**

Pada setiap langkah di dalam algoritma *greedy* kita baru memperoleh optimum lokal. Bila algoritma berakhir, kita berharap optimum lokal menjadi optimum global. Algoritma *greedy* mengasumsikan bahwa optimum lokal merupakan bagian dari optimum global.

III. Analisis Pemecahan Masalah

Pada permainan ini terdapat dua macam algoritma *Greedy* yang digunakan:

1. *Greedy by Ratio(SYLPH)*

   Algoritma ini menggunakan perbandingan antara nilai dengan jarak antar peti sebagai acuan utamanya. Dari kumpulan perbandingan-perbandingan tersebut diambil rute dari titik awal dengan rasio tertinggi melalui algoritma *Greedy.*

   Elemen-elemen algoritma:

   - Himpunan kandidat : Himpunan perbandingan nilai peti terhadap jarak menuju peti.
   - Himpunan solusi : sebuah lintasan dalam graf permainan
   - Fungsi seleksi : Pilih peti yang memiliki rasio nilai:jarak tertinggi

- Fungsi kelayakan : Peti-peti yang terpilih dapat membentuk suatu lintasan dalam graf
- Fungsi obyektif : Harta yang didapat maksimum

2. *Greedy by Time, Distance, and Worth (MAGI):*

   Algoritma ini menggunakan array perbandingan nilai dengan jarak, yang diurutkan menurun berdasarkan rasio yang paling besar ke kecil. Lalu dilakukan pengecekan apakah jika menuju kandidat pertama memungkinkan secara waktu, jika tidak maka kandidat kedua dicek, jika memungkinkan secara waktu, dicek lagi apakah peti akan meledak jika dicoba dibuka dengan tool dan waktu yang tersisa, jika bisa maka kandidat diterima, jika semua percobaan pembukaan peti meledak atau tidak punya alat, maka kandidat pertama yang diterima oleh pengecekan waktu yang akan diterima sebagai hasil akhir.

   Elemen-elemen algoritma:

   - Himpunan kandidat : Himpunan perbandingan nilai peti terhadap jarak menuju peti.
   - Himpunan solusi : sebuah lintasan dalam graf himpunan kandidat
   - Fungsi seleksi : 1. Peti diurutkan menurut rasio nilai:jarak

     2. Pengecekan apakah sisa waktu memungkinkan untuk pergi ke tempat peti

     3. Pengecekan apakah peti akan meledak ketika akan dicoba dibuka
   - Fungsi kelayakan : Peti-peti yang terpilih dapat membentuk suatu lintasan dalam graf
   - Fungsi obyektif : Harta yang didapat maksimum

IV. Implementasi dan Pengujian

a. Spesifikasi Teknis

i. IDE : Visual Studio 2013 dengan SFML

ii. Fungsi dan Prosedur

1. Graf

```
class Graf {
    public:
            Graf();
            ~Graf();
```

```
            bool Load(std::string filename);
            int GetModal();
            int GetNPetiMerah();
            int GetNPetiKuning();
            int GetNPetiHijau();
            int GetMinSkor();
            int GetWaktuPermainan();
            int GetJumlahSimpul();
            int* GetArrPosPeti();
            int GetPetiPosKe(int Berapa);
            int GetPosPemain();
            int** GetMatrixJarakSimpul();
            int GetJarakSimpulKeKe(int Asal, int
Tujuan);
            bool IsMerah(int Pos);
            bool IsKuning(int Pos);
            bool IsHijau(int Pos);
            bool IsKosong(int Pos);
            void SetArrPosPeti(int Posisi, int
Value);
            void SetArrPosPetiOpen(int Posisi);
            void Print();

            void TulisOutput(int Simpul, int Peti);
    private:
            int modal;
            int merah,kuning,hijau;
            int skor;
            int waktumain;
            int n;
            int *posisi; //array of posisi peti
            int pemain;
            int **jarak; // matrix dari jarak
```

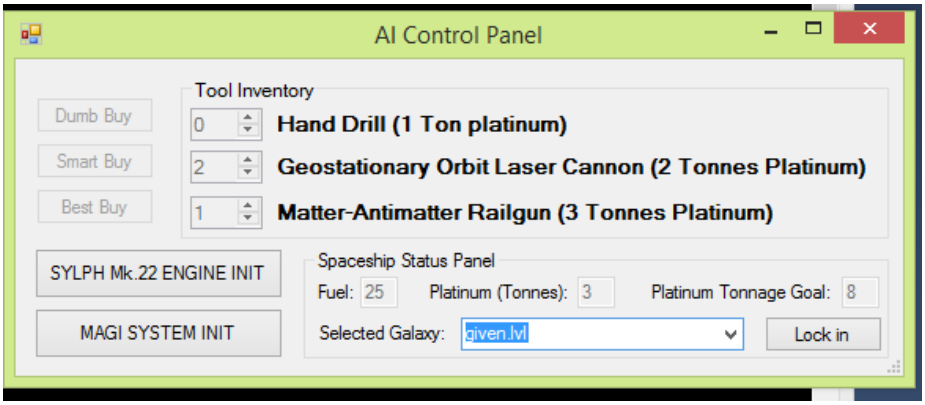
2. Algoritma

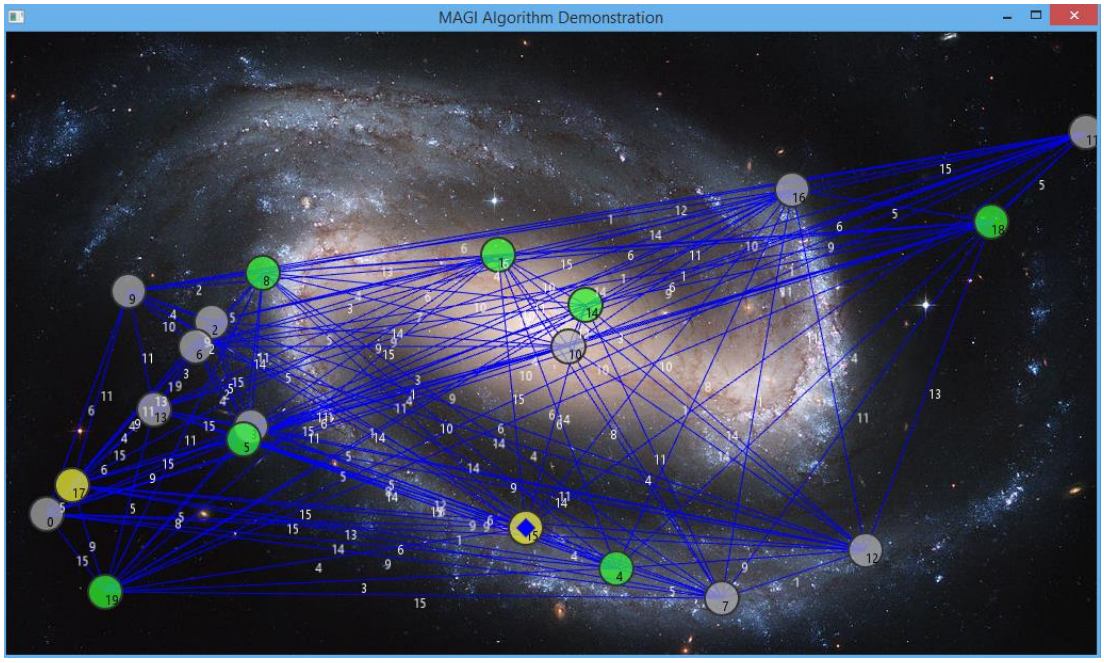
```
int SYLPH(Graf G, Player P, int pos);

int MAGI(Graf G, Player P, int TimeLeft);

int* Melchior(Graf G, Player P,int* ArrayJarak);
/* Mengusulkan kandidat simpul yang dikunjungi
berikutnya */
bool Balthazar(int* ArrayJarak, int Simpul, int
TimeLeft);
/* Menolak atau menerima kandidat tujuan simpul
berdasarkan waktu tempuh*/
bool Casper(Graf G, Player P, int TimeLeft, int*
ArrayJarak, int Simpul);
/* Menolak atau menerima kandidat tujuan
berdasarkan ketersediaan alat pembuka peti dan
mengecek apakah peti meledak jika dicoba
dibuka*/
```

b. *Capture* layar





c. Analisis hasil pengujian

Uji coba kasus dari contoh pada soal memberikan hasil algoritma MAGI lebih baik daripada algoritma SYLPH. Uji coba pada kasus simpul graf dengan banyak batasan keadaan menunjukkan bahwa algoritma MAGI lebih efektif dibandingkan SYLPH.

Kedua algoritma menggunakan uji perbandingan harta terhadap jarak sebagai basis algoritma. Namun, algoritma MAGI menambahkan faktor batasan waktu dalam memilih rutenya. Atas dasar ini, *greedy* milik MAGI dapat memilih solusi yang lebih optimal disbanding SYLPH.

Selain pencarian rute, algoritma *greedy* juga digunakan dalam pembelian alat serta penggunaannya. Terdapat tiga macam algoritma *greedy* yang digunakan untuk pembelian alat, masing-masing dengan pendekatan yang berbeda. Hasil terbaik didapat pada Smart Buy.

V. Kesimpulan dan Saran

Algoritma *Greedy* dapat digunakan dalam permainan "Treasure Hunter". Namun, penggunaan algoritma ini harus mempertimbangkan banyak faktor yang dapat mempengaruhi hasil dari algoritma tersebut.

VI. Daftar Referensi

Munir, Rinaldi. 2009. Diktat Kuliah IF2211 Strategi Algoritma.

www.sfml-dev.org/

Microsoft MSDN

http://holowczak.com

http://www.bogotobogo.com

http://stackoverflow.com

http://www.cplusplus.com