Penerapan Algoritma Greedy dalam Strategi Permainan Monopoli

Kelompok 10

1. Sabrina Diza Melinda (G64180029)
2. Zahra Aulia Firdausi (G64180030)
3. Syukriyatul Hanifa (G64180062)
4. Hendrika Anggriawan (G64180088)

DEPARTEMEN ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT PERTANIAN BOGOR

2021

I. PENDAHULUAN

A. Latar Belakang

Permainan monopoli telah menjadi permainan umum di masyarakat. Monopoli merupakan permainan yang bertujuan mengumpulkan uang sebanyak-banyaknya dengan menguasai negara-negara yang tercantum pada papan dengan pembelian, penyewaan, dan pertukaran properti. Apabila dirunut dari pendekatan matematis, pemain dapat menghasilkan uang/kekayaan yang maksimum dengan modal yang minimum. Strategi untuk mendapat keuntungan bisa dicari dengan menggunakan algoritma greedy.

Algoritma greedy adalah salah satu algoritma yang dapat menyelesaikan permasalahan optimasi walaupun hasilnya tidak selalu merupakan solusi optimum. Pada dasarnya algoritma ini menyelesaikan permasalahan secara bertahap dimana pada setiap langkah terdapat banyak pilihan yang perlu di eksplorasi. Algoritma greedy disusun oleh elemen-elemen berikut, yaitu himpunan kandidat (C), himpunan solusi (S), fungsi seleksi, fungsi kelayakan dan fungsi objektif (Rachmawati *et al*, 2013).

II. ANALISIS MASALAH

Pada monopoli terdapat beberapa petak-petak negara yang memiliki harga beli dan harga sewa dengan nilai tertentu yang sudah ditentukan sebelumnya. Para pemain monopoli diberikan sejumlah uang sebagai modal untuk dapat membeli petak-petak tersebut sehingga mampu menguasai semua negara dan menjadi yang paling kaya. Permasalahan utama dari permainan monopoli ini adalah bagaimana pemain menentukan strategi terbaik untuk mendapatkan uang sebanyak-banyaknya dengan penggunaan modal seoptimal mungkin.

III. PEMBAHASAN

A. Pengumpulan Data

Data yang digunakan berasal dari simulasi permainan monopoli secara online dengan ketentuan jumlah pemain sebanyak 4 orang, setiap pemain melakukan 10 putaran, modal awal $2000, dan biaya *cost* dan *rent* sesuai ketentuan yang tersedia.

B. Algoritma Penyelesain

Persoalan yang ada dapat diselesaikan menggunakan algoritma greedy. Ada 3 pilihan yang dapat digunakan:

1. Greedy by density, yaitu permasalahan diselesaikan dengan melihat perbandingan antara harga dan profit serta melibatkan aspek frekuensi suatu petak dikunjungi
2. Greedy by profit, yaitu permasalahan diselesaikan dengan melihat keuntungan yang dapat diraih pada tiap petak dan melibatkan aspek frekuensi
3. Greedy by cost, yaitu permasalahan diselesaikan dengan melihat perbandingan antar harga petak pada monopoli

Inti kode program yang kami buat berisi algoritme sorting (kami menggunakan merge sort) dan algoritme greedy untuk memilih negara mana saja yang dapat dibeli.

Kode Program

```
int main() {
    double arr[NEG][COL]={0};
    int i, x, urutBy, a, b, c;
    for( i = 0; i < NEG; i++){
        scanf("%d %d %d", &a, &b, &c);
        arr[i][0] = a; // cost
        arr[i][1] = b; // rent
        arr[i][2] = c; // frek
    }
    for( i = 0; i < NEG; i++){
        arr[i][3]=arr[i][1]*arr[i][2];//profit
        arr[i][4]=arr[i][3]/arr[i][0];//density
        arr[i][5]=i;                  //idNegara
    }
```

Pada fungsi main, langkah pertama yang dilakukan adalah mendeklarasikan array 2 dimensi berukuran 27(NEG)x6(COL). Berikutnya, program memasukkan nilai cost,rent,frek,profit, density, dan idNegara.

```
    scanf("%d", &urutBy);
    if (urutBy==0){                 // by density
        mergeSort(arr, 0, NEG-1, 4);
        reverse(arr);
    } else if (urutBy == 1){        // by profit
        mergeSort(arr, 0, NEG-1, 3);
        reverse(arr);
    } else if (urutBy == 2){        // by cost
        mergeSort(arr, 0, NEG-1, 0);
    } else {
        printf("Input tidak valid\n");
    }
```

Langkah berikutnya adalah mengurutkan isi array berdasar density/profit/cost.

```
    int modal, totalCost, untung;
    scanf("%d", &modal);
    totalCost = 0;
    untung = 0;

    printf("Daftar negara yang dibeli:\n");
    i = 0;
    for (i = 0; i < NEG; i++){
        if (totalCost+arr[i][0] <= modal){
            x = (int) arr[i][5];
            printf("-%d %s\n", x+1 ,negara[x]);
            untung += (int)arr[i][3];
            totalCost += arr[i][0];
        }
    }
```

Setelah diurutkan, program mulai membelanjakan modal untuk membeli negara-negara berdasarkan array yang sudah diurutkan.

```
void mergeSort(double arr[][COL], int l,
               int r, int kolom) {
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m, kolom);
        mergeSort(arr, m + 1, r, kolom);
        merge(arr, l, m, r, kolom);
    }
}
```

Di luar fungsi main, didefinisikan fungsi untuk mengurutkan array. Untuk ini kami menggunakan algoritme merge sort. Algoritme merge sort yang kami buat adalah merge sort untuk array 2 dimensi. Foto diatas adalah potongan algoritme merge sort bagian divide.

```
void merge(double arr[][COL], int l, int m,
           int r,int kolom){
    int i, j, k, x;
    int n1 = m - l + 1;
    int n2 = r - m;
    double L[n1][COL],R[n2][COL]; //temp array

    // Copy data ke array L[] dan R[]
    for (i = 0; i < n1; i++){
        for (j = 0; j < COL; j++){
            L[i][j] = arr[l + i][j];
        }
    }
    for (i = 0; i < n2; i++){
        for (j = 0; j < COL; j++){
            R[i][j] = arr[m + 1 + i][j];
        }
    }

    // Menggabungkan temp array ke arr[l..r]
    i = 0; // Inisiasi index subarray L
    j = 0; // Inisiasi index subarray R
    k = l; // Inisiasi index subarray gabungan
    while (i < n1 && j < n2) {
        if (L[i][kolom] <= R[j][kolom]) {
            for (x = 0; x < COL; x++){
                arr[k][x] = L[i][x];
            }i++;
        } else {
            for (x = 0; x < COL; x++){
                arr[k][x] = R[j][x];
            }j++;
        }k++;
    }
    // Memasukkan sisa elemen array L[]
    while (i < n1) {
        for (x = 0; x < COL; x++){
            arr[k][x] = L[i][x];
        }i++;k++;
    }
    // Memasukkan sisa elemen array R[]
    while (j < n2) {
        for (x = 0; x < COL; x++){
            arr[k][x] = R[j][x];
        }j++;k++;
    }
}
```

Sedangkan foto diatas adalah potongan algoritme merge sort bagian conquer.

```
void reverse(double arr[][COL]){
    double bantu[1][COL];
    int i, j;
    for(i=0; i<NEG/2; i++){
        for(j=0; j<COL; j++){
            bantu[0][j] = arr[i][j];
            arr[i][j] = arr[NEG-1-i][j];
            arr[NEG-1-i][j] = bantu[0][j];
        }
    }
}
```

Karena merge sort yang kami buat hanya digunakan untuk mengurutkan dari kecil ke besar, sedangkan pada beberapa pilihan greedy memerlukan urutan besar ke kecil, maka kami membuat fungsi reverse untuk membalik urutan array.

C. Hasil Analisis

Sebelum memulai permainan, setiap pemain akan diberi modal sebesar $2000. Langkah pertama yang dapat dilakukan oleh pemain adalah mencari petak-petak yang berpotensial sehingga mampu unggul dalam permainan dengan mendapatkan uang/kekayaan yang paling tinggi. Pemilihan petak tersebut dapat dilakukan dengan menggunakan pendekatan algoritma *greedy by density, greedy by profit,* dan *greedy by cost*. Pendekatan dengan *greedy by density* didapat dengan mengurutkan nilai *density* setiap petak secara menurun lalu disesuaikan dengan modal yang dimiliki. Sama seperti *greedy by density*, *greedy by profit* juga didapat dengan mengurutkan nilai profit secara menurun. Sedangkan untuk *greedy by cost* didapat dengan mengurutkan negara berdasarkan nilai cost secara menaik. Berikut tabel hasil analisis pemilihan petak dengan keuntungan maksimum.

No	Negara	Cost	Rent	Frekuensi	Density	Greedy by		
						Density	Cost	Profit
1	Libya	50	5	1	0.10000	1	1	0
2	Sudan	60	10	6	1.00000	1	1	0
3	Station Japan	200	25	9	1.12500	1	1	0
4	Turkey	100	10	1	0.10000	0	1	0
5	Greece	100	10	7	0.70000	0	1	0
6	Bulgaria	120	16	5	0.66667	0	1	0
7	Poland	150	16	5	0.53333	0	1	0
8	Russia	150	16	10	1.06667	1	1	1
9	Ukraine	180	20	6	0.66667	0	1	0
10	Vatican	200	22	2	0.22000	0	1	0
11	Station Spain	200	25	7	0.87500	0	1	0
12	Lithuania	200	20	4	0.40000	0	1	0
13	Latvia	200	20	7	0.70000	0	1	0
14	Estonia	220	30	6	0.81818	0	0	0
15	Norway	220	30	6	0.81818	0	0	0
16	Sweden	220	30	5	0.68182	0	0	0
17	Finland	240	40	8	1.33333	1	0	1
18	Station USA	200	25	4	0.50000	0	0	0
19	Germany	280	40	7	1.00000	1	0	1
20	France	280	40	7	1.00000	1	0	1
21	United Kingdom	300	50	2	0.33333	0	0	0
22	Canada	300	50	5	0.83333	0	0	0
23	Mexico	300	50	8	1.33333	1	0	1
24	USA	320	60	5	0.93750	0	0	1
25	Station UK	200	25	7	0.87500	0	0	0
26	Qatar	360	60	2	0.33333	0	0	0
27	China	400	80	7	1.40000	1	0	1
					Total Cost	1960	1910	1970
					Total keuntungan	2290	1249	2300

Berdasarkan tabel diatas, algoritma *greedy by density* dan *greedy by profit* menghasilkan kekayaan yang sama besar, yakni sebesar $2330. Sedangkan dengan menggunakan algoritma *greedy by cost* kekayaan yang diperoleh hanya sebesar $1339. Kekayaan tersebut didapat dengan menjumlahkan total keuntungan dengan sisa uang dari modal. Berdasarkan besar kekayaannya, solusi yang optimal didapat dengan menggunakan algoritma g*reedy by density* atau *greedy by profit* sehingga dalam pelaksanaannya bisa menggunakan salah satu dari algoritma tersebut. Akan tetapi, tidak semua kasus mendapatkan solusi optimal dengan menggunakan algoritma *greedy by density* maupun *greedy by profit*. Bisa saja beberapa kasus akan mendapatkan solusi yang optimal dengan menggunakan algoritma *greedy* lainnya.

D. Analisis Kompleksitas Masalah

Dalam algoritma yang kami kembangkan terdapat 4 bagian utama yaitu:

1. *Fungsi Pembalik Array,* berfungsi untuk membalik urutan array dari ascending menjadi descending dengan kompleksitas O(n).
2. *Fungsi Merge Sort,* digunakan untuk mengurutkan data array dengan kompleksitas O(n log n).
3. *Fungsi Input,* digunakan untuk memasukkan data berupa cost, rent, frek, modal dengan kompleksitas O(n).

4. *Fungsi Selection,* berfungsi untuk memilih negara yang akan dibeli dengan kompleksitas O(n).

Sehingga total kompleksitas dari seluruh algoritma ini adalah O(n log n).

IV. KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan dari hasil implementasi algoritma greedy yang kami terapkan pada permainan monopoli ini dapat diambil kesimpulan bahwa, algoritma *greedy by density* dan *greedy by profit* memiliki uang/kekayaan tertinggi yaitu sebesar $2330. Namun tidak semua kasus mendapatkan keuntungan yang optimal dengan menggunakan algoritma by profit ini. Hal ini disebabkan karena adanya kemungkinan perbedaan biaya cost, rent, atau frekuensi pada kasus permainan monopoli lain. Solusi dari penerapan algoritma greedy ini hanya memberikan strategi yang berpotensi untuk memenangkan permainan ini.

B. Saran

Algoritma *greedy* belum merupakan algoritma yang paling optimal untuk menyelesaikan permasalahan ini. Untuk hasil yang lebih optimal, maka sebaiknya menggunakan algoritma *count sort* dalam melakukan fungsi *sorting*.

DAFTAR PUSTAKA

Rachmawati D, Candra A. 2013. Implementasi algoritma greedy untuk menyelesaikan masalah knapsack problem. Jurnal SAINTIKOM. 12(3): 187-188

Sunandar H, Pristiwanto. 2019. Optimalisasi implementasi algoritma greedy dalam fungsi penukaran mata uang rupiah. Jurnal Teknik Informatika Unika St. Thomas (JTIUST). 04(02) : 193-201.

Vignesh R, Tribikram P. 2016. Merge Sort Enhanced In Place Sorting Algorithm. International Conference on Advanced Communication Control and Computing Technologies (ICACCCT). 01(01): 698-704.

Wikipedia. 2019. Permainan Monopoly. https://id.wikipedia.org/wiki/Monopoli_(permainan). Diakses 21 Desember 2020.

Lampiran Kontribusi

NAMA	BAGIAN
Sabrina Diza Melinda (G64180029)	Analisis Masalah, Hasil Analisis
Zahra Aulia Firdausi (G64180030)	Analisis Kompleksitas, Kesimpulan dan Saran
Syukriyatul Hanifa (G64180062)	Pendahuluan, Pengumpulan Data
Hendrika Anggriawan (G64180088)	Algoritma Penyelesaian, Kode Program

Source code dapat diakses melalui tautan berikut: *ipb.link/kel10-greedymonopoli*