BAB 2

LANDASAN TEORI

2.1 Algoritma Greedy

Algoritma *Greedy* merupakan jenis algoritma yang menggunakan pendekatan penyelesaian masalah dengan mencari nilai maksimum sementara pada setiap langkahnya. Nilai maksimum sementara ini dikenal dengan istilah *maximum local*. Kebanyakan kasus, algoritma *Greedy* tidak akan menghasilkan solusi paling optimal, algoritma *Greedy* biasanya memberikan solusi yang mendekati nilai optimum dalam waktu yang cukup tepat.

Dalam banyak kasus, strategi algoritma *Greedy* tidak menghasilkan solusi optimal, tetapi *Greedy* dapat menghasilkan solusi optimal lokal yang mendekati solusi optimal global dalam waktu yang wajar. Secara bahasa *Greedy* artinya "rakus". Algoritma *Greedy* membentuk solusi langkah per langkah. Untuk setiap langkah, terdapat banyak pilihan yang perlu dieksplorasi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Setiap langkah mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsip "*take what you can get now!")*, berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

2.1.1 Skema Umum Algoritma Greedy

Persoalan optimasi dalam konteks algoritma greedy disusun oleh elemen-elemen sebagai berikut :

1. Himpunan Kandidat=C

   Himpunan ini berisi elemen-elemen pembentuk solusi, pada setiap langkahnya satu buah kandidat dari himpunannya.

2. Himpunan Solusi=S

   Merupakan himpunan dan kandidat-kandidat yang terpilih sebagai solusi persoalan. Himpunan solusi adalah himpunan bagian dari himpunan kandidat.

3. Fungsi Seleksi=*select*()

Fungsi yang memilih kandidat yang paling memungkinkan dari himpunan kandidat untuk dimasukkan ke dalam himpunan solusi agar solusi optimal terbentuk. Kandidat yang sudah terpilih pada suatu langkah tidak akan dipertimbangkan lagi pada langkah selanjutnya.

4. Fungsi Kelayakan=*feasible*()

Fungsi yang memeriksa apakah suatu kandidat yang terpilih akan menimbulkan solusi yang layak, yaitu kandidat tersebut bersama dengan himpunan solusi yang terpilih tidak akan melanggar kendala yang berlaku pada masalah.

5. Fungsi Solusi=*solution*()

Merupakan fungsi yang mengembalikan nilai *Boolean*. *True* jika himpunan yang sudah terbentuk merupakan solusi yang lengkap, *false* jika himpunan solusi belum lengkap.

6. Fungsi Objektif=*objective*()

Fungsi yang memaksimalkan atau meminimalkan nilai solusi.

Berikut adalah pseudocode algoritma *Greedy* secara umum :

Tabel 1.1 Pseudocode Umum Algoritma *Greedy*[1]

```
procedure greedy(input C: himpunan_kandidat;
output S : himpunan_solusi)
{ menentukan solusi optimum dari persoalan optimasi dengan algoritma greedy
  Masukan: himpunan kandidat C
  Keluaran: himpunan solusi S
}
Deklarasi
x : kandidat;

Algoritma:
S¬{}   { inisialisasi S dengan kosong }
while (belum SOLUSI(S)) and (C ¹ {} ) do
       x¬SELEKSI(C); { pilih sebuah kandidat dari C}
       C¬ C - {x}      { elemen himpunan kandidat berkurang satu }
if LAYAK(S È {x}) then
    S¬S È {x}
endif
endwhile
{SOLUSI(S) sudah diperoleh or C = {} }
```

Skema umum dari algoritma *Greedy* dapat dituliskan sebagai berikut :

1. Inisiasi S dengan kosong.
2. Pilih sebuah kandidat dari C (dengan *select*()).

3. Kurangi C dengan kandidat yang terpilih.
4. Periksa apakah kandidat yang dipilih tersebut bersama-sama dengan S membentuk solusi yang layak (dengan *feasible*()). Jika ya, masukkan kandidat ke S; jika tidak, buang kandidat tersebut dan tidak perlu ditelaah lagi.
5. Periksa apakah S yang sudah terbentuk telah memberikan solusi yang lengkap (dengan *solution*()). Jika ya, berhenti; jika tidak, ulangi dari langkah dua.

Pembahasan pada penelitian ini adalah beberapa algoritma *Greedy* yang dapat memecahkan persoalan. Terdapat empat algoritma *Greedy* yang akan diimplementasikan untuk menentukan langkah pion, yaitu :

1. *Greedy by Hitter*

Pion pemain akan melangkah sesuai dengan angka yang dihasilkan oleh dadu, apabila di depan pion pemain terdapat pion lawan maka pion pemain tersebut akan memakan pion lawan yang berada di depannya.

2. *Greedy No One Left Behind*

Pion pemain akan melangkah sesuai dengan angka yang dihasilkan oleh dadu, apabila posisi pion pemain berada di paling belakang, maka pion pemain tersebut akan melangkah agar tidak berada di paling belakang.

3. *Greedy by Quick Move*

Pion pemain akan melangkah sesuai dengan angka yang dihasilkan oleh dadu, apabila posisi pion pemain berada dekat dengan *finish*, maka pion pemain tersebut akan melangkah agar segera mencapai *finish*.

4. *Greedy by Blocking*

Pion pemain akan melangkah sesuai dengan angka yang dihasilkan oleh dadu, apabila di deoan pion pemain terdapat pion pemain lain maka pion tersebut akan melangkah dan bertumpuk dengan pion kawan yang berada di depannya.

2.2 Permainan Ludo

Ludo adalah sejenis papan permainan yang dimainkan oleh 2-4 pemain, dimana para pemain berlomba menjalankan empat pion mereka dari start sampai *finish* berdasarkan lemparan dadu.

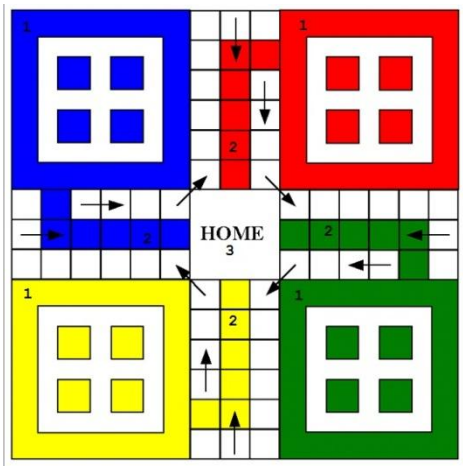
2.2.1 Sejarah Permainan Ludo

Ludo berasal dari kata Latin ”*ludus*”yang artinya permainan (game)[1]. Permainan ini biasanya dimainkan dua sampai empat orang. Ludo sebenarnya adalah permainan yang berasal dari orang India, Pachisi. Pachisi sendiri diperkirakan sudah dimainkan sejak 500 tahun sebelum masehi. Tetapi, Ludo yang muncul sekitar tahun 1896 telah dipatenkan di Inggris.

Area khusus pada papan Ludo biasanya berwarna kuning, hijau, merah dan biru. Setiap pemain mendapatkan empat buah pion sesuai dengan warna yang dipilih. Papan Ludo berbentuk persegi dengan jalur mengikuti arah jarum jam, bagian tengah terdapat kolom berwarna yang merupakan kolom “rumah” pemain, pada bagian ujung kolom berwarna terdapat kolom berbentuk persegitiga, itulah *finish* dari permainan ini.

2.2.2 Bagian Bagian Pada Papan Ludo

Bagian-bagian ludo dapat dilihat pada gambar 2.1, dimana terdapat nomor dan akan dijelaskan per-poin.



Gambar 2.1 Papan Permainan Ludo

Papan permainan Ludo berbentuk persegi empat, di setiap sudutnya terdapat *home base* dengan empat warna berbeda yaitu merah, hijau, kuning, dan biru, pada gambar 2.1 diberi tanda dengan angka 1 (satu). Setiap *home base* memiliki empat buah pion dengan warna yang sama dengan *home base* masing-masing.

Pion berjalan memutar sesuai arah jarum jam, bisa dilihat pada gambar 2.1 arah laju permainan Ludo ditandai dengan tanda panah. Apabila pion telah

berputar sebanyak satu purtaran maka pion tersebut masuk ke dalam jalur yang berada di tengah sesuai dengan warna *home base* dan pion masing-masing, seperti pada gambar 2.1 yang ditandai dengan angka 2 (dua).

Pada gambar 2.1 yang ditandai dengan angka 3 (tiga) adalah *finish* atau tujuan pion setelah berputar dan masuk ke *line* berwarna yang berada di tengah.

2.2.3 Aturan Permainan Ludo

Permainan Ludo dimainkan oleh dua hingga empat orang. Setiap pemain mendapatkan empat buah pion dengan warna sesuai dengan pilihan masing-masing, pion-pion tersebut digerakkan searah dengan jarum jam, berjalan memutar sesuai dengan jalur yang ada pada papan, untuk mencapai kotak tujuan yang berada di tengah papan, seperti yang telah dijelaskan pada gambar 2.1.

Setiap daerah atau Negara memiliki aturan ludo yang berbeda-beda, namun pada penelitian ini peraturan pada permainan Ludo adalah sebagai berikut.

1. Pemain memilih warna yang akan dimainkan, kemudian meletakkan empat pion yang sesuai dengan warna *home base* yang dipilih.
2. Pemain melemparkan dadu secara bergantian.
3. Untuk dapat mengeluarkan pion ke jalur permainan, pemain harus mendapatkan angka enam dari dadu yang dilemparkan.
4. Pemain menggerakkan pion yang sudah berada di jalur permainan sesuai dengan angka yang dihasilkan dadu.
5. Jika pemain mendapatkan angka enam lagi, maka pemain dapat memilih untuk menggerakkan pion yang berada di jalur atau mengeluarkan pion yang berada di *home base*.
6. Apabila angka enam muncul pemain dapat melempat dadu lagi, dan tidak ada batasan apabila dadu yang dilempar menghasilkan angka enam berturut-turut.
7. Jika pion pemain berhenti di tempat yang telah terisi pion lawan maka pion milik lawan tersebut mati dan masuk kembali ke *home base* milik lawan.
8. Jika pion pemain berhenti di tempat yang terisi pion milik sendiri, maka pion tersebt akan bertumpuk membuat benteng dan pion lawan tidak dapat lewat berapapun angka yang dihasilkan oleh dadu.

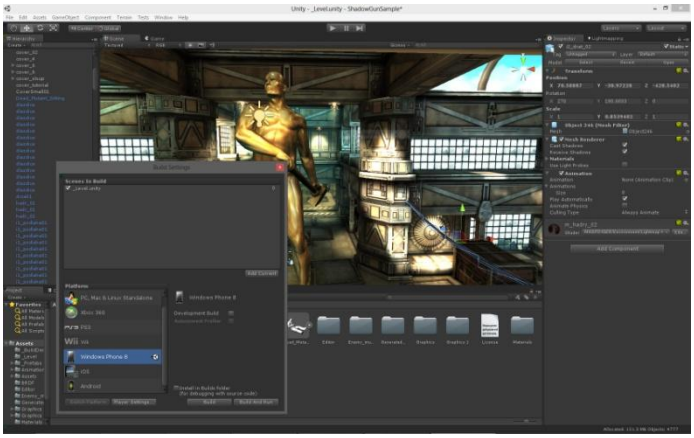
9. Pion yang sudah melewati satu putaran masuk ke dalam jalur berwarna sesuai dengan warna pion.
10. Pion tidak akan bergerak apabila angka yang dihasilkan dadu belum sesuai untuk masuk ke dalam *finish*. Contoh: pion hanya perlu tiga langkah lagi menuju *finish*, namun angka yang dihasilkan dadu adalah empat, maka pion akan maju tiga langkah dan mundur satu langkah. Begitu seterusnya hingga pion berhasil masuk *finish*.

2.3 *Game Engine*

Game Engine adalah sistem perangkat lunak yang dirancang untuk menciptakan dan pengembangan video game. Ada banyak mesin permainan yang dirancang untuk bekerja pada konsol permainan video dan sistem operasi desktop seperti Microsoft Windows, Linux, dan Mac OS X. Proses pengembangan permainan sering dihemat oleh sebagian besar menggunakan kembali mesin permainan yang sama untuk menciptakan permainan yang berbeda.

2.3.1 Unity

Unity adalah *game engine* dengan built-in IDE yang dikembangkan oleh Unity Technologies. *Game engine* ini digunakan untuk mengembangkan *game* dengan platform desktop, konsol, dan perangkat *mobile*. Tampilan antarmuka Unity dapat dilihat pada gambar 2.2.



Gambar 2.2 Unity

Scripting dari *game engine* Unity adalah implementasi dari NET Framework. Pemrograman dapat menggunakan UnityScript, C#, atau Boo (yang terinspirasi dari bahasa Python). Dimulai dengan dirilisnya versi 3.0. Unity

mendukung penyebaran ke berbagai platform. Pengembang dapat membuat ke perangkat mobile, web browser, desktop, atau konsol. Unity juga memungkinkan spesifikasi kompresi tekstur dan pengaturan resolusi setiap platform.

2.4 Bahasa Pemrograman JavaScript

JavaScript adalah bahasa skrip yang populer di internet dan dapat bekerja di sebagian besar penjelajah web populer. JavaScript pertama kali dikembangkan oleh Brendan Eich dari Netscape dibawah nama Mocha, yang nantinya namanya diganti menjadi LiveScript, dan akhirnya menjadi JavaScript.

JavaScript bisa digunakan untuk banyak tujuan, misalnya untuk membuat efek rollover baik di gambar maupun teks, dan yang penting juga adalah untuk membuat AJAX. JavaScript adalah bahasa yang digunakan untuk AJAX.