Algoritma BFS dalam Pencarian Jumlah Simpul Minimum untuk Memenangkan Permainan Ular Tangga

Qurrata A'yuni and 13518004
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13518004@std.stei.itb.ac.id

***Abstract*—Algoritma BFS (Breadth First Search) adalah salah satu algoritma pencarian jalur sederhana yang mana pencarian dimulai dari titik awal, kemudian dilanjutkan ke semua cabang titik tersebut secara berurut. Algoritma ini dapat dimanfaatkan dalam permainan ular tangga dengan setiap kotak pada permainan direpresentasikan dengan sebuah simpul. Pencarian simpul paling akhir akan dilakukan dengan membangkitkan simpul-simpul dimulai dari simpul awal. Dalam pencarian jalur sederhana dengan menggunakan algoritma BFS ini diharapkan akan banyak simpul terkecil yang ditempuh untuk menyelesaikan permainan.**

Keywords—BFS; Breadth First Search; ular tangga

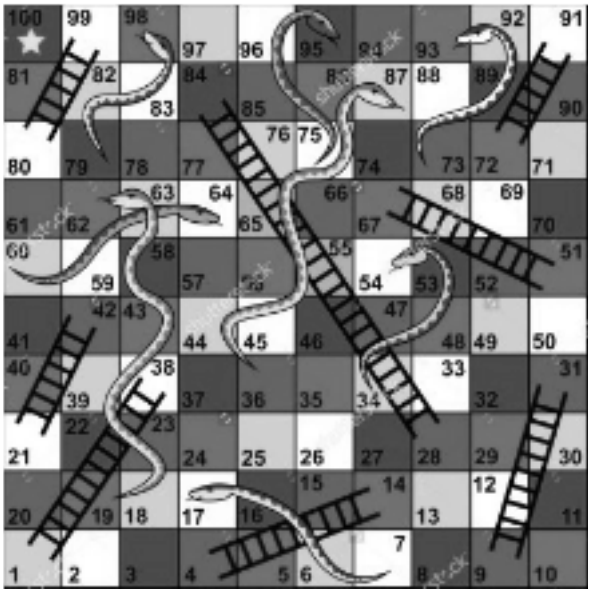
I. Pendahuluan

Algoritma Breadth First Search (BFS) adalah sebuah algoritma yang digunakan untuk melakukan pencarian solusi dari sebuah graf yang setiap nodenya saling terhubung satu sama lain dengan metode melewati semua daerah terdekat dengan daerah asal untuk mencari jalan menuju tujuan. Breadth First Search memiliki sifat komplit bila nilai b terbatas, optimal jika langkah sama dengan biaya, serta memiliki kompleksitas $O(b^d)$, dan memiliki kompleksitas ruang $O(b^d)$ dengan b adalah jumlah graf terdekat dari simpul sebelumnya dan d adalah kedalaman yang harus dilakukan.

Menurut Melsi (2015:10) ular tangga adalah permainan papan yang dimainkan oleh dua orang atau lebih. Biasanya permainan ini dimainkan oleh anak-anak. Papan permainan dibagi menjadi kotak-kotak kecil dan pada beberapa kotak terdapat sejumlah tangga dan ular yang menghubungkan dengan kotak yang lainnya. Permainan ini menggunakan dadu untuk menentukan berapa langkah yang harus dijalani bidak. Permainan ini termasuk kategori *"board game"* atau permainan papan sejenis dengan monopoli, ludo, dan sebagainya.

II. Ular Tangga

Permainan ular tangga dimainkan dalam kotak berukuran nxn, misalnya pada pembahasan kali ini ukuran papan adalah 10x10. Tujuan dari permainan ini adalah pemain menaruh paling banyak bidak pada kotak tujuan yaitu kotak terakhir yang diberi angka 100. Akan terdapat informasi mengenai tangga dan ular dalam permainan ini.



Gambar 1. Papan permainan ular tangga

Informasi tangga dalam permainan akan menguntungkan pemain, karena dengan adanya tangga, bidak akan dapat berpindah dari satu kotak ke kotak selanjutnya dengan

perpindahan yang cukup besar. Tergantung dari informasi yang akan diberikan.

Sedangkan informasi ular dalam permainan akan merugikan pemain, karena bidak dapat berpindah dari satu kotak ke kotak lain dengan cara mundur yang artinya akan menghalang tujuan dari permainan ini yaitu meletakkan sebanyak mungkin bidak pada kotak terakhir dengan cepat.

Permainan ini juga membutuhkan sebuah dadu untuk menentukan berapa kotak yang dapat dilewati oleh sebuah bidak. Kemunculan angka pada dadu adalah dari satu sampai enam. Diasumsikan pengocokan hanya dilakukan satu kali saja, karena terdapat aturan tambahan bila pemain mendapatkan dadu berangka enam maka pemain boleh mengocok dadu sekali lagi. Untuk menyederhanakan pemanfaatan algoritma Breadth First Search dalam pembahasan kali ini pengocokan dadu hanya dilakukan satu kali dalam setiap pembangkitan simpul anak.

Permainan dimulai dari pemain belum memasuki kotak yang direpresentasikan dengan angka nol. Pemain akan melakukan pengocokan dadu. Kemungkinan angka yang muncul adalah dari satu hingga enam. Sehingga kita mendapatkan informasi bahwa dari 0 bidak dapat berpindah ke kotak 1, kotak 2, kotak 3, kotak 4, kotak 5, dan kotak 6. Informasi tambahan mengenai tangga akan dapat merubah informasi perpindahan yang dapat dilakukan. Misalnya terdapat informasi bahwa kotak 1 terhubung dengan kotak 38 menggunakan tangga. Informasi dari 0 ke kotak berikutnya akan berubah menjadi ke kotak 38, kotak 2, kotak 3, kotak 4, kotak 5, dan kotak 6.

Pemanfaatan BFS (Breadth First Search) akan dilakukan dengan mebangkitkan simpul anak dari simpul akar yaitu nol, kemudian mencari kotak-kotak yang terhubung dengannya. Kotak yang memberi informasi bahwa terdapat ular yang dapat memundurkan lokasi bidak akan dihindari. Apabila dalam pencarian ditemukan kotak terakhir yaitu 100, maka pencarian diberhentikan.

Dalam setiap simpul akan disimpan informasi mengenai jarak yang sudah ditempuh. Jarak di sini bukanlah jarak dari suatu kotak ke kotak lain. Melainkan banyaknya kotak yang sudah ditempuh dari simpul asal. Untuk menyederhanakan penamaan di dalam program akan diberi nama jarak minimal. Diharapkan dengan adanya informasi mengenai tangga dan pemanfaatan BFS (Breadth First Search) simpul tujuan dapat ditemukan dengan perpindahan yang tidak begitu banyak.

Sebagai contoh informasi tangga dan ular dalam permainan ini akan disimpan dalam sebuah map kemudian akan dimasukkan sebagai tambahan informasi dalam graf, berikut informasinya.

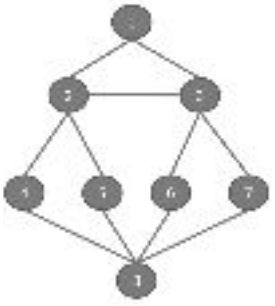
TABLE I. TABEL INFORMASI ULAR DAN TANGGA

Tangga	Ular
(1,38)	(17,7)
(4,14)	(54,34)
(9,31)	(62,19)
(21,42)	(64,60)
(28,84)	(87,36)
(51,57)	(93,73)
(72,91)	(95,75)
(80,99)	(98, 79)

III. ALGORITMA BFS

Algoritma Breadth First Search (BFS) adalah algoritma yang melakukan pencarian jalur dengan secara melebar. Pencarian dimulai dari simpul akar dan akan dilakukan pencarian melebar dari satu simpul ke simpul lain yang berada satu tingkat yang sama terlebih dahulu. Struktur data yang digunakan dalam algoritma ini adalah antrian. Pencarian solusi menggunakan Breadth First Search (BFS) dapat digambarkan melalui algoritma di bawah ini:

1. Tambahkan simpul akar ke dalam antrian Q. Jika akar merupakan solusi pencarian akan dihentikan.
2. Jika antrian Q kosong, maka berhenti.
3. Dengan menggunakan metode FIFO (First In First Out) pilih antrian pertama misalnya simpul n, kemudian bangkitkan semua anaknya dan tambahkan ke dalam antrian Q.
4. Jika n merupakan solusi maka berhenti, jika tidak lakukan kembali dari langkah kedua.



Gambar 2. Urutan pencarian algoritma BFS

IV. IMPLEMENTASI

Implementasi algoritma Breadth First Search pada permainan ular tangga akan dibahas pada bagian ini. Mulai dari pemain berangkat dari simpul akar hingga menuju simpul tujuan. Pencarian diharapkan akan menghasilkan jarak terdekat yang diperoleh dengan algoritma Breadth First Search.

A. *Sisi*

Pada permainan ular tangga ini, terdapat sebuah struktur data kelas yang menyimpan informasi asal dan tujuan. Sebuah kotak direpresentasikan dengan sebuah angka. Sebuah angka merupakan informasi asal. Setiap kotak pada papan memiliki keterhubungannya dengan kotak lain. Misalkan, terdapat kotak dengan nomor satu pada papan, terhubung dengan kotak nomor dua. Sehingga sisi yang dapat dibentuk dari pernyataan tersebut adalah sisi(1,2) yang asalnya dari satu menuju dua.

Diasumsikan dalam permainan ini dari setiap kotak asal, pemain hanya dapat bergerak sejumlah satu sampai enam kotak. Karena dalam pengocokan dadu hanya ada enam kemungkinan angka yang muncul. Misalnya kotak nol dapat memiliki:

sisi: (0,1), (0,2), (0,3), (0,4), (0,5), dan (0,6)

Informasi tangga yang diberikan dalam permainan dapat merubah informasi dari sisi itu sendiri. Misalkan terdapat informasi bahwa terdapat tangga yang menghubungkan kotak 1 dengan kotak 38 dan kotak 4 terhubung dengan kotak 14. Sehingga informasi sisi yang dapat dibentuk dari kotak nol berubah menjadi:

sisi: (0,38), (0,2), (0,3), (0,14), (0,5), dan (0,6)

Implementasi kelas sisi dalam bahasa Java

```
class Sisi {
  int asal;
  int tujuan;
}
```

B. Simpul

Simpul akan menyimpan informasi vertex dan jarak yang sudah ditempuh dalam artian banyaknya simpul yang sudah ditempuh dimulai dari simpul akar.

Implementasi kelas simpul dalam bahasa Java

```
class Node {
  int vertex; // posisi kotak
  int minJarak; // banyak kotak yang ditempuh
}
```

C. Graf

Graf yang digunakan dalam permainan ular tangga ini adalah graf berarah. Graf ini direpresentasikan dalam bentuk matriks adjasensi yang memperlihatkan setiap simpul terhubung dengan simpul mana saja.

Dalam permainan ini terdapat 100 simpul yang merepresentasikan 100 kotak yang terdapat di dalam papan. Informasi setiap kotak terhubung ke kotak mana saja akan disimpan di dalam sebuah graf dengan matriks adjasensi berukuran 100x100 yang memberikan informasi bahwa setiap simpul terhubung ke simpul yang mana saja.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**dst.**
0	-	-	1	1	1	1	1	-	-	
1	-	-	1	1	-	1	1	1	-	
2	-	-	-	1	-	1	1	1	1	
3	-	-	-	-	-	1	1	1	1	
4	-	-	-	-	-	1	1	1	1	
5	-	-	-	-	-	-	1	1	1	
6	-	-	-	-	-	-	-	1	1	
7	-	-	-	-	-	-	-	-	-	
8	-	-	-	-	-	-	-	-	-	
dst.										

Gambar 3. Graf yang direpresentasikan dengan matriks adjasensi

Implementasi kelas graf dalam bahasa Java

```
class Graf {
  List<List<Integer>> listAdj; // matriks
}
```

D. *BFS*

Pembangkitan simpul dimulai dari simpul akar yaitu simpul 0, setiap simpul anak akan dibangkitkan dan ditambahkan ke dalam sebuah antrian. Berikut ini algoritma BFS dalam permainan ini.

```
procedure BFS (input g: Graf, input n: integer, input asal: integer)
{ Traversal graf dengan algoritma pencarian BFS }
{ Masukan: g adalah graf, n adalah banyaknya simpul, dan asal adalah simpul akar
  Keluaran: banyak kotak yang harus ditempuh dari kotak asal }
Deklarasi:
  q : Queue { Queue of Node }
  dikunjungi : array of Boolean
  node, node_ : Node
Algoritma
Queue(q) { buat Queue of Node kosong }
Boolean(dikunjungi) { simpul sudah dikunjungi/tidak }
Node node ←Node(asal, 0)
q.add(node) { tambahan simpul akar ke antrian }
while not isEmpty(q)  do
  node ← q.poll() { mengambil antrian pertama }
  if node.vertex = n then break
  for i in g.listAdjensi.get(node.vertex) do
        if  not dikunjungi[i] then
            dikunjungi[i] ← true
           Node node_ ← Node(i, node.minJarak+1)
           q.add(node_)
```

Berikut ini ilustrasi BFS dalam pencarian solusi

TABLE II. TABEL ILUSTRASI BFS

Iterasi	Node	Q
1	0	{0}
2	0	{38,2,3,14,5,6}
3	38	{2,3,14,5,6,39,40,41,42,43,44}
4	2	{3,14,5,6,39,40,41,42,43,44,7,8}
5	3	{14,5,6,39,40,41,42,43,44,7,8,31}
6	14	{5,6,39,40,41,42,43,44,7,8,31,15,16,18,19,20}
7	5	{6,39,40,41,42,43,44,7,8,31,15,16,18,19,20,10,11}
8	6	{39,40,41,42,43,44,7,8,31,15,16,18,19,20,10,11,12}
9	39	{40,41,42,43,44,7,8,31,15,16,18,19,20,10,11,12,45}
10	40	{41,42,43,44,7,8,31,15,16,18,19,20,10,11,12,45,46}
…..		
69	73	{94,75 96 70 71 74 97 79 76 77 78}
70	94	{75,96,70,71,74,97,79,76,77,78,99,100}
71	75	{96,70,71,74,97,79,76,77,78,99,100,81}
72	96	{70,71,74,97,79,76,77,78,99,100,81}
73	70	{71,74,97,79,76,77,78,99,100,81}
74	71	{74,97,79,76,77,78,99,100,81}
75	74	{97,79,76,77,78,99,100,81}
76	97	{79,76,77,78,99,100,81}
77	79	{76,77,78,99,100,81,82,83}
78	76	{77,78,99,100,81,82,83}
79	77	{78,99,100,81,82,83}
80	78	{99,100,81,82,83}
81	99	{100,81,82,83}
82	100	

Solusi: 14-16-22-84-88-94-100

Output program: 7

Program yang diimplementasikan akan memberikan output ke layar. Ini diperoleh dari simpul akhir yaitu simpul bernomor 100 dengan nilai jarak terkecilnya bernilai 7. Nilai 7 diperoleh melalui pencarian BFS dengan membangkitkan simpul anak dari simpul akar.

Dalam program yang diimplementasikan tidak akan diperlihatkan simpul mana saja yang memberikan output 7, karena tujuan utama dari program yang akan diimplementasikan adalah memberikan jumlah minimum

kotak yang dilalui oleh bidak pada papan permainan untuk bisa memenangkan permainan. Apabila ingin menemukan simpul mana yang menjadi solusi maka kita harus melakukan *backtrack* dari simpul tujuan. Simpul tujuan tersebut dibangkitkan dari simpul mana. Tetapi program yang diimplementasikan tidak melakukan itu, lain kesempatan mungkin dapat ditambahkan implementasi yang dibutuhkan. Tetapi pembahasan berikut ini akan menunjukkan dari mana angka 7 ini diperoleh dengan menggambarkan pohon pencariannya, dengan catatan tidak semua simpul akan digambarkan dalam pohon karena akan cukup memakan tempat, sehingga pohon dengan simpul solusinya saja yang akan ditampilkan.

Gambar 4. Pohon solusi

Berikut ini penjelasan output 7 diperoleh.

- Simpul akar (simpul 0) - ketika pemain belum meletakkan bidak. Setiap kali pemain mengocok dadu ada enam kemungkinan nilai. Nomor angka yang dipilih pada dadu adalah 4. Sehingga bidak akan berpindah dari simpul 0 ke simpul 4. Simpul 4 terhubung ke simpul 14 sehingga bidak akan berpindah ke kotak 14.
- Simpul 14 - angka yang muncul pada dadu adalah 2, sehingga bidak akan pindah ke kotak 16.
- Simpul 16 - angka yang muncul pada dadu adalah 6, sehingga bidak akan pindah ke kotak 22.
- Simpul 22 - angka yang muncul pada dadu adalah angka 6 sehingga bidak akan pindah ke kotak 28. Simpul 28 terhubung dengan simpul 84 melalui tangga sehingga bidak akan pindah ke kotak 84.
- Simpul 84 - angka yang muncul adalah pada dadu adalah 4 sehingga bidak akan pindah ke kotak 88.
- Simpul 88 - angka yang muncul pada dadu adalah 6 sehingga bidak akan pindah ke kotak 94.
- Simpul 94 - angka yang muncul pada dadu adalah 6 sehingga bidak akan pindah ke kotak 100. Kotak 100 (simpul 100) merupakan simpul tujuan

Dalam program yang diimplementasikan akan dihasilkan jumlah kotak yang dikunjungi dalam papan. Dalam contohnya akan dihasilkan 7. Angka ini bukan menunjukkan bahwa hanya dibutuhkan 7 langkah untuk memenangkan permainan. Melainkan terdapat setidaknya ada tujuh kotak/simpul yang harus dilewati agar dapat memenangkan permainan.

Setiap kali pembangkitan simpul tidak diperhitungkan angka pada dadu yang mana yang akan dibangkitkan melainkan dengan algoritma BFS (Breadth First Search) semua simpul anak akan dibangkitkan. Sehingga ketika simpul anak tersebut merupakan simpul tujuan maka pencarian langsung diberhentikan. Setiap simpul menyimpan informasi berapa kotak yang sudah dilalui dari simpul akar. Sehingga kita dapat mengetahui ada berapa kotak yang harus dilewati agar mencapai simpul *goal.*

Untuk mengetahui simpul mana saja yang menjadi simpul solusi dapat dilihat dari pohon pencariannya. Dari simpul tujuan yang sudah dicapai dapat dilihat simpul *parent*nya hingga mencapai ke simpul awal lagi. Maka kita dapat mencatat simpul mana saja yang harus dilewati sekaligus dalam setiap kali giliran permainan berapa jumlah dadu yang harus didapatkan sehingga dapat memainkan jumlah dadu/banyak kotak yang dilalui seminimal mungkin.

KESIMPULAN

Algoritma BFS (Breadth First Search) dapat dimanfaatkan dalam permainan ular tangga yaitu untuk mencari banyak kotak yang harus dilalui dalam papan permainan agar dapat dimenangkan oleh pemain. Setiap kotak direpresentasikan dengan sebuah angka. Ukuran papan berupa nxn. Sehingga akan terdapat nxn kotak/simpul. Pencarian jumlah minimum kotak yang dilalui untuk mencapai simpul tujuan dimulai dari simpul akar. Simpul akar dalam permainan ditandai dengan angka nol. Dari simpul akar kita memiliki kesempatan untuk mengocok dadu, kemungkinan angka yang muncul ada enam. Maka ada enam kemungkinan lokasi perpindahan bidak pemain. Setiap lokasi ini adalah anak dari simpul akar. Metode pembangkitan simpul-simpul memanfaatkan struktur data antrian yang mengambil elemen yang paling dulu berada di dalam antrian. Begitu seterusnya selama antrian tidak kosong atau simpul tujuan sudah ditemukan. Pada akhir program akan ditampilkan banyaknya kotak minimum untuk mencapai simpul *goal.*

VIDEO LINK AT YOUTUBE

https://youtu.be/fH4e5RE7aR4

REFERENCES

[1] Fatkhan Amirul Huda, 'Pengertian Media Pembelajaran Permainan Ular Tangga', 2019. [Online]. Available: http://fatkhan.web.id/pengertian-media-pembelajaran-permainan-ular-tangga/. [Accessed: 01-Mei-2020]

[2] Siddharth, 'Snake and Ladder Problem'. [Online]. Available: https://www.geeksforgeeks.org/snake-ladder-problem-2/ [Accessed: 01-Mei-2020]

[3] Rinaldi Munir, 'Breadth First Search dan Depth First Search'. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/BFS-dan-DFS-(2020).pdf. [Accessed 01-Mei-2020]

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bukittinggi, 1 Mei 2020

Qurrata A'yuni - 13518004