Penerapan Algoritma *Backtracking* pada *Game The Lonely Knight*

Ananda Kurniawan Pramudiono - 13511052
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13511052@std.stei.itb.ac.id

***Abstract*—*Game The Lonely Knight* adalah permainan dimana pemain menjadi seorang ksatria untuk menyebrang dari kastil 1 ke kastil 2 dengan menggunakan langkah Kuda pada permainan catur. Untuk dapat menyebrang dan memenangkan game ini maka langkah/pijakan pertama harus sama dengan langkah/pijakan terakhir. Terdapat banyak algoritma untuk menyelesaikan permainan ini. Makalah ini membahas penerapan algoritma *backtrracking* untuk menyelesaikan permainan ini**

Kata Kunci—*The Lonely Knight, Backtracking*

I. PENDAHULUAN

Seiring dengan perkembangan globalisasi, manusia tidak lepas dari perkembangan teknologi yang terus berkembang. Untuk menghadapi perkembangan teknologi tersebut, manusia selalu menemukan masalah-masalah yang dapat menghambat perkembangan teknologi tersebut.

Untuk menghadapi sebuah masalah, seseorang membutuhkan strategi untuk menyelesaikan masalah tersebut. Strategi adalah langkah-langkah dari sebuah rencana yang sudah dipersiapkan untuk memecahkan masalah tertentu.

Beberapa strategi pencarian solusi yang telah dipelajari dalam perkuliahan strategi algoritma ini adalah :

1. Algoritma Brute Force
2. Algoritma Greedy
3. Algoritma Divide and Conguer
4. DFS dan BFS
5. Algoritma Runut-balik (backtracking)
6. Algoritma Branch and Bound
7. Program Dinamis
8. Pencocokan String

Permasalahan yang ada dalam makalah ini adalah dalam permainan *The Lonely Knight*. Bagaimana cara menyelesaikan permasalahan aplikasi teori graf pada permainan *The Lonely Knight*. Sedangkan strategi yang akan digunakan penulis dalam memecahkan masalah ini adalah algoritma runutbalik (*backtracking algorithm*).

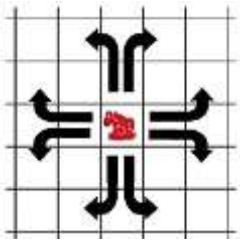
A. *The Lonely Knight*

The Lonely Knight adalah permainan logika dimana menantang pemain sebagai ksatria untuk menyebrang ke kastil seberang dengan melewati jembatan yang terdiri atas 14 kotak. Ksatria menyebrang menunggang sebuah kuda dimana harus berjalan sesuai dengan jalan kuda pada permainan catur

Gambar 1 *Games The Lonely Knight*

Langkah kuda yang dapat ditempuh adalah :

a. Dua langkah ke arah kanan dan satu langkah ke arah bawah
b. Dua langkah ke arah kanan dan satu langkah ke arah atas
c. Dua langkah ke arah bawah dan satu langkah ke arah kanan
d. Dua langah ke arah bawah dan satu langkah ke arah kiri
e. Dua langkah ke arah kiri dan satu langkah ke arah bawah
f. Dua langkah ke arah kiri dan satu langkah ke arah atas
g. Dua langkah ke arah atas dan satu langkah ke arah kiri
h. Dua langkah ke arah atas dan satu langkah ke arah kanan



Gambar 2 Langkah Kuda dalam Permainan Catur

Untuk menyelesaikan permainan, ksatria tersebut harus mengunjungi tiap kotak tepat satu kali dan kembali ke kotak awal.

B. Algoritma *Backtracking*

Algoritma *backtracking* adalah algoritma pemecahan masalah berbasis pada DFS yang mencari solusi persoalan secara lebih mangkus. *Backtracking* merupakan perbaikan dari algoritma *Brute Force* yang mencari solusi persoalan diantara semua kemungkinan solusi yang ada, pada algoritma *Backtracking* tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya perlu mempertimbangkan proses pencarian yang sudah mengara ke solusi saja. Sehingga waktu pencarian menjadi lebih cepat.

Istilah runut balik pertama kali dikenalkaan pada tahun 1950 oleh D.H Lehmer. Kemudian R.J Walker, Golomb dan Baumert menyajikan uraian umum tentang algoritma ini dan penerapanya terhadap berbagai persoalan.

Algoritma Backtracking telah banyak diterapkan untuk berbagai macam hal seperti:

- Games (tic-tac-toe, maze-based game)
- Berbagai permasalahan pada catur (eight queens puzzle, Knigth's Tour Problem)
- Permasalahan dalam parsing seperti knapsack problem
- Puzzle seperti Sudoku, maupun crossword puzzle

Backtracking dapat diterapkan hanya untuk masalah masalah yang mempunyai konsep "calon parsial solusi" dan tes yang relatif cepat. *Backtracing* seringkali lebih cepat daripada *brute force* dari semua calon solusi yang banyak, hal ini dikarenakan algoritma *backtracking* dapat menghilangkan sejumlah besar kandidat dengan sebuah tes.

Properti Umum Metode Algoritma *Backtracing*

1. Solusi persoalan

   Solusi dinyatakan sebagai vektor dengan n-tuple: X = (x1, x2, …, xn), xi Î Si. Mungkin saja S1 = S2 = … = Sn.

   Contoh: Si = {0, 1}

   xi = 0 atau 1

2. Fungsi pembangkit nilai xK

   Dinyatakan sebagai: T(k)

   T(k) membangkitkan nilai untuk xk, yang merupakan komponen vektor solusi.

3. Fungsi pembatas

   Dinyatakan sebagai B(x1, x2, …, xk)

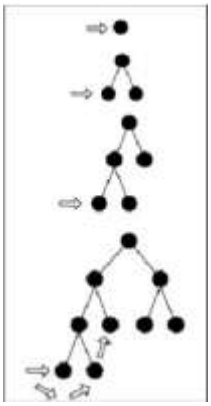
   B bernilai true jika (x1,x2,…,xk) mengarah ke solusi. Jika true, maka pembangkitan nilai untuk xk+1dilanjutkan, tetapi jika false, maka (x1, x2, …, xk) dibuang.

   Semua kemungkinan solusi dari persoalan disebut ruang solusi (solution space). Secara formal dapat dinyatakan, bahwa xi ∈ Si, maka S1 x S2 x … x Sn disebut ruang solusi. Jumlah anggota di dalam ruang solusi adalah | S1|·| S2| ·… · | Sn |.

Pencarian Solusi dengan *Backtracking*

Dari pohon ruang status yang telah dibangun maka dapat ditinjau pencarian solusi secara dinamis. Langkah langkah pencarian solusi adalah sebagai berikut:

1. Solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai adalah mengikuti aturan pencarian mendalam (DFS). Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup (live node). Simpul hidup yang sedang diperluas dinamakan simple (Expand-node).
2. Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi, maka simpul-E tersebut "dibunuh" sehingga menjadi simpul mati (dead node). Fungsi yang digunakan untuk membunuh simpulE adalah dengan menerapkan fungsi pembatas (bounding function). Simpul yang sudah mati tidak akan pernah diperluas lagi.
3. Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian diteruskan dengan membangkitkan simpul anak yang lainnya. Bila tidak ada lagi simpul anak yang dapat dibangkitkan, maka pencarian solusi dilanjutkan dengan melakukan runut-balik ke simpul hidup terdekat (simpul orangtua) Selanjutnya simpul ini menjadi simpul-E yang baru.
4. Pencarian dihentikan bila kita telah menemukan solusi atau tidak ada lagi simpul hidup untuk runut-balik.



Gambar 3 Ilustrasi pembentukan simpul dalam *backtracking*

II. PENERAPAN *BACKTRACKING* PADA *GAMES THE LONELY KNIGH*

Algoritma *backtracking* digunakan untuk menemukan solusi yang tepat. Algoritma *backtracking* akan membangun solusi parsial dari semua kemungkinan yang ada dan akan mengembangkan solusi tersebut. Dalam permainan ini algoritma *backtracking* akan membangun beberapa solusi dari step pertama yang dipilih dan diperluas tiap tiap solusi tersebut. Jika solusi yang diperluas tidak mencapai hasil maka algoritma ini akan melakukan backtracking dan mencari solusi dari solusi parsial lainnya.

Tahap tahap algoritma *backtracking* dalam persoalan *The Lonely Knight* adalah sebagai berikut :

1. Dari kotak pertama yang dipilih ksatria dibangun solusi-solusi parsial yang berisi langkah ke kotak berikutnya yang dapat memungkinkan dilalui ksatria tersebut.
2. Salah satu dari solusi tersebut dipilih dan kemudian dibangkitkan langkah berikutnya.
3. Ksatria jalan sesuai degan langkah yang dibangkitkan.
4. Tahap 1 dan tahap 2 dilakukan sampai semua solusi ditemukan (semua kotak terlewati dan kembali ke kotak pertama) atau solusi tidak ditemukan (sampai langkah berikutnya tidak dapat diperluas lagi).
5. Apabila solusi tidak ditemukan, dilakukan runut balik ke langkah sebelumnya, kemudian dari kotak tersebut dilakukan tahap 1 dan tahap 2 lagi.
6. Pencarian solusi akan berhenti ketika solusi ditemukan atau tidak ditemukan satupun solusi.

Kotak – kotak yang harus disebangi diberi urutan abjad sebagai berikut :

A	B	C	
D	E	F	G
H	I	J	K
L	M	N	

Gambar 4

Dengan menggunakan langkah-langkah yang sudah ditentukan pad algortima *backtracking* di atas maka dapat ditemukan :

Kotak pertama yang dipilih ksatria adalah kotak C.

		1	

Gambar 5

Pada tiap kotak, akan dibuat daftar kotak yang dapat dilalui oleh kuda pada langkah selanjutnya.

Pada kasus ini, ada dua kemungkinan langkah, yaitu kotak I, K dan D.

Pertama kita coba kotak I. Jika langkah pada kotak ini tidak menuju solusi, maka kita akan runut-balik dan mencoba kotak K.

		1	
	2		

Gambar 4

Dari kotak ini, langkah yang mungkin adalah kotak A C dan G, Kotak C sudah pernah dilewati. Coba ambil kotak A Jika langkah pada kotak ini tidak menuju solusi, maka kita akan runut-balik dan mencoba kotak G.

3		1	
	2		

Gambar 6

Dari kotak ini, langkah yang mungkin adalah kotak I dan F, Kotak I sudah pernah dilewati. Coba ambil kotak F.

3		1	
		4	
	2		

Gambar 7

Dari kotak ini, langkah yang mungkin adalah kotak A M dan H, Kotak A sudah pernah dilewati. Coba ambil kotak H Jika langkah pada kotak ini tidak menuju solusi, maka kita akan runut-balik dan mencoba kotak M.

3		1	
		4	
5	2		

Gambar 8

Dari kotak ini, langkah yang mungkin adalah kotak B, F dan N, Kotak F sudah pernah dilewati. Coba ambil kotak B Jika langkah pada kotak ini tidak menuju solusi, maka kita akan runut-balik dan mencoba kotak N.

3	6	1	
		4	
5	2		

Gambar 9

Dari kotak ini, langkah yang mungkin adalah kotak H, J dan G, Kotak H sudah pernah dilewati. Coba ambil kotak G Jika langkah pada kotak ini tidak menuju solusi, maka kita akan runut-balik dan mencoba kotak J.

3	6	1	
		4	7
5	2		

Gambar 10

Dari kotak ini, langkah yang mungkin adalah kotak N, B dan I, Kotak B dan I sudah pernah dilewati. Kita ambil kotak N.

3	6	1	
		4	7
5	2		
		8	

Gambar 11

Dari kotak ini, langkah yang mungkin adalah kotak E, G dan H, Kotak G dan H sudah pernah dilewati. Kita ambil kotak E.

3	6	1	
	9	4	7
5	2		
		8	

Gambar 12

Dari kotak ini, langkah yang mungkin adalah kotak L, N dan K, Kotak N sudah pernah dilewati. Coba ambil kotak K Jika langkah pada kotak ini tidak menuju solusi, maka kita akan runut-balik dan mencoba kotak L.

3	6	1	
	9	4	7
5	2		10
		8	

Gambar 13

Dari kotak ini, langkah yang mungkin adalah kotak C, E dan M, Kotak C dan E sudah pernah dilewati. Kita ambil kotak M.

	6	1	
	9	4	7
5	2		10
	11	8	

Gambar 14

Dari kotak ini, langkah yang mungkin adalah kotak D, F dan K, Kotak F dan K sudah pernah dilewati. Kita ambil kotak D.

3	6	1	
12	9	4	7
5	2		10
	11	8	

Gambar 15

Dari kotak ini, langkah yang mungkin adalah kotak M, C dan J, Kotak M dan C sudah pernah dilewati. Kita ambil kotak J.

3	6	1	
12	9	4	7
5	2	13	10
	11	8	

Gambar 16

Dari kotak ini, langkah yang mungkin adalah kotak B, D dan L, Kotak B dan D sudah pernah dilewati. Kita ambil kotak L.

3	6	1	
12	9	4	7
5	2	13	10
14	11	8	

Gambar 17

Solusi tidak ditemukan langkah berikutnya tidak kembali ke kotak pertama, maka dilakukan runut balik.

3	6	1	
12	9	4	7
5	2	13	10
	11	8	

Gambar 18

Tidak ada langkah yang memungkinkan. Maka kembali dilakukan runut balik.

3	6	1	
12	9	4	7
5	2		10
	11	8	

Gambar 19

Tidak ada langkah yang memungkinkan. Maka kembali dilakukan runut balik.

3	6	1	
	9	4	7
5	2		10
	11	8	

Gambar 20

Tidak ada langkah yang memungkinkan. Maka kembali dilakukan runut balik.

3	6	1	
	9	4	7
5	2		10
		8	

Gambar 21

Tidak ada langkah yang memungkinkan. Maka kembali dilakukan runut balik.

3	6	1	
	9	4	7
5	2		
		8	

Gambar 22

Dari kotak ini, langkah yang mungkin adalah kotak L, N dan K, Kotak N sudah pernah dilewati. Kotak K sudah coba diambil dan tidak menuju solusi, maka kita ambil kotak L .

3	6	1	
	9	4	7
5	2		
10		8	

Gambar 23

Dari kotak ini, langkah yang mungkin adalah kotak E dan J, Kotak E sudah pernah dilewati. Kita ambil kotak J.

3	6	1	
	9	4	7
5	2	11	
10		8	

Gambar 24

Dari kotak ini, langkah yang mungkin adalah kotak B, D dan L, Kotak B dan L sudah pernah dilewati. Kita ambil kotak D.

3	6	1	
12	9	4	7
5	2	11	
10		8	

Gambar 25

Dari kotak ini, langkah yang mungkin adalah kotak M, C dan J, Kotak M dan C sudah pernah dilewati. Kita ambil kotak M.

3	6	1	
12	9	4	7
5	2	11	
10	13	8	

Gambar 26

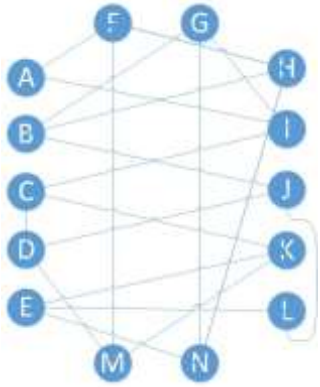
Dari kotak ini, langkah yang mungkin adalah kotak D, F dan K, Kotak D dan F sudah pernah dilewati. Kita ambil kotak K.

3	6	1	
12	9	4	7
5	2	11	14
10	13	8	

Gambar 27

Solusi ditemukan langkah berikutnya kembali ke kotak pertama.

Graf solusi dari kotak kotak yang harus disebrangi tersebut adalah



Gambar 28 Graf Solusi

III KESIMPULAN

Algoritma backtracking dapat digunakan dalam pemecahan masalah dalam permainan knight's tour. Bila dibandingkan dengan algoritma lainnya seperti brute force dan exhaustive search, algoritma inimempermudah pencarian langkah-langkah yang harus ditempuh untuk menyelesaikan permainan knight's tour tanpa harus mencoba semua kemungkinan layaknya brute force.

REFERENSI

[1] Munir, Rinaldi. "Diktat Kuliah IF2251 Strategi Algoritmik", Program Studi Teknik Informatika STEI ITB, 2009.

[2] http://www.plastelina.net
Diakses tanggal 19 Desember 2013 pukul 20.00

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.

Bandung, 19 Desember 2013

Ananda Kurniawan
13511052