ALGORITMA RUNUT-BALIK (*BACKTRACKING ALGORITHM*) PADA MASALAH *KNIGHT'S TOUR*

Fahmi Mumtaz

13506045
Teknik Informatika Institut Teknologi Bandung
e-mail: if16045@students.if.itb.ac.id; mumtaz_banget@yahoo.co.uk;

ABSTRAK

Makalah ini membahas tentang algoritma untuk menyelesaikan permasalahan pada aplikasi teori graf di permainan catur papan. Permasalahan yang dibahas pada makalah ini adalah masalah *knight's tour*. Ada banyak algoritma yang dapat dipakai untuk menyelesaikan permasalahan tersebut. Tapi dalam makalah ini, algoritma yang dipilih oleh penulis adalah algorima runut-balik (*backtracking algorithm*).

Kata kunci: *Knight's Tour*, Algoritma, Algoritma runut-balik (*backtracking algorithm*), strategi algoritmik, *graf*.

1. PENDAHULUAN

Di dalam kehidupannya, manusia selalu menemui masalah atau persoalan. Hal ini mungkin didasarkan dari sifat dasar manusia itu sendiri yang selalu ingin tahu tentang segala sesuatu. Deskripsi dari masalah menurut [NEA96] adalah pertanyaan atau tugas yang kita cari jawabannya.

Dalam menghadapi permasalahan, untuk menyelesaikannya manusia memerlukan langkah-langkah yang benar sehingga permasalahan tersebut dapat terselesaikan. Urutan langkah-langkah untuk memecahkan suatu masalah tersebutlah yang dinamakan dengan algoritma. Definisi lain dari algoritma adalah deretan langkah-langkah komputasi yang mentransformasikan data masukan menjadi keluaran [COR92].

Dalam menentukan langkah-langkah tersebut diperlukan suatu strategi agar langkah-langakah yang dipakai tersebut dapat menyelesaikan permasalahan secara mangkus (efisien). Strategi menurut Kamus Besar Bahasa Indonesia (KBBI) edisi tahun 1998 adalah rencana yang cermat mengenai kegiatan untuk mencapai sasaran khusus. Rencana itu sendiri dapat berisi suatu metode atau teknik yang digunakan untuk mencapai sasaran khusus tersebut.

Dengan pengertian algoritma dan strategi tersebut, kita dapat mendefinisikan strategi algoritmik (*algorithm strategies*) sebagai kumpulan metode atau teknik untuk memecahkan masalah guna mencapai tujuan yang ditentukan, yang dalam hal ini deskripsi metode atau teknik tersebut dinyatakan dalam suatu urutan langkah-langkah penyelesaian.

Secara umum, strategi pemecahan masalah dapat dikelompokan menjadi:

1. Strategi solusi langsung (*direct solution strategies*)
   Metode yang termasuk ke dalam strategi ini adalah:
   - Algoritma *Brute Force*
   - Algoritma G*reedy*
2. Strategi berbasis pencarian pada ruang status (*state-space bace strategies*)
   Metode yang termasuk ke dalam strategi ini adalah:
   - Algoritma *Backtracking*
   - Algoritma *Branch and Bound*
3. Strategi solusi atas-bawah (*top-down solution strategies*)
   Metode yang termasuk ke dalam strategi ini adalah algoritma *Divide and Conquer*.
4. Strategi solusi bawah-atas (*bottom-up solution strategies*)
   Metode yang termasuk ke dalam strategi ini adalah *Dynamic Programming*.

Strategi yang akan dipakai dalam pemecahan masalah dalam makalah ini adalah strategi berbasis pencarian pada ruang status (*state-space bace strategies*). Secara spesifik, metode yang digunakan adalah algoritma runut-balik (*Backtracking algorithm*).

Permasalahan yang dibahas disini adalah permasalahan dari aplikasi teori graf pada permainan catur papan yang bernama *Knight's Tour*. Permasalahan ini akan dibahas lebih lanjut dalam bab selanjutnya.

2. *KNIGHT'S TOUR*

Knight's Tour adalah suatu aplikasi dari teori graf pada permainan catur papan. Suatu *Knight's Tour* pada papan catur adalah rangkaian perjalanan kuda catur pada papan catur sehingga seluruh kotak terlewati kuda tepat satu kali. Aturan langkah kuda pada permainan catur adalah sebagai berikut :

- Melangkah dua persegi ke arah horisontal kemudian satu persegi ke arah vertikal, atau

- Melangkah dua persegi ke arah vertikal kemudian satu persegi ke arah horisontal, atau
- Melangkah dua persegi ke arah vertikal kemudian satu persegi ke arah horisontal, atau
- Melangkah satu persegi ke arah vertikal kemudian dua persegi ke arah horisontal.

Jika dalam *Knight's Tour* setiap persegi dari papan catur dapat dilewati tepat satu kali dan kuda kembali pada persegi semula maka disebut langkah kuda tertutup (*Closed Knight's Tour*). Namun, jika semua persegi telah dilewati dan kuda tidak dapat kembali ke posisi semula maka disebut langkah kuda yang terbuka (*Open Knight's Tour*)

4	1	6	55	10	51	46	49
7	56	3	64	45	48	11	52
2	5	58	9	54	13	50	47
57	8	63	14	59	44	53	12
28	37	60	43	62	15	24	41
31	34	29	38	25	42	21	18
36	27	32	61	16	19	40	23
33	30	35	26	39	22	17	20

Gambar 1. Contoh *Closed Knight's Tour* pada papan catur ukuran 8 x 8

Di atas adalah contoh dari *Knight's Tour*. Angka 1-64 menandakan urutan kotak yang dilewati oleh kuda catur tersebut. Langkah yang diambil oleh kuda urut dari 1 sampai dengan 64.

Banyak ilmuwan yang tertarik dengan permasalahan ini diantaranya adalah Leonhard Euler, Paul de Hijo, Sainte-Marie, Louis Posa, Allen Schwenk, Mark R. Keen, M. Kraitchik, Sloane's, Wegener, dan masih banyak ilmuan lainnya.

3. ALGORITMA RUNUT BALIK (BACKTRACK)

Algoritma runut-balik (*backtracking algorithm*) adalah algoritma yang berbasis pada *DFS* untuk mencari solusi persoalan secara lebih mangkus. Runut-balik, yang merupakan perbaikan dari algoritma *bruteforce*, secara sistematis mencari solusi persoalan di antara semua kemungkinan solusi yang ada. Dengan metode ini, kita tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan. Akibatnya, waktu pencarian dapat dihemat. Runut-balik lebih alami dinyatakan dalam algoritma rekursif. Kadang-kadang disebutkan pula bahwa runut-balik merupakan bentuk tipikal dari algoritma rekursif.

Istilah runut-balik pertama kali diperkenalkan oleh D.H. Lehmer pada tahun 1950. Selanjutnya, R. J. Walker, Golomb, dan Baumert menyajikan uraian umum tentang runut-balik dan penerapannya pada berbagai persoalan [HOR78].

3.1 Properti Umum Metode Runut-balik

Untuk menerapkan metode runut-balik, properti berikut didefinisikan:

1. Solusi persoalan.
   Solusi dinyatakan sebagai vektor *n-tuple*:

   X=(x_1, x_2, ..., x_n), x_i anggota himpunan berhingga S_i . Mungkin saja $S_1 = S_2 = ... = S_n$.

   Contoh: S_i = {0,1}
   S_i = 0 atau 1
2. Fungsi pembangkit nilai x_k
   Dinyatakan sebagai:
   $T(k)$
   $T(k)$ membangkitkan nilai untuk x_k, yang merupakan komponen vektor solusi
3. Fungsi pembatas (fungsi kriteria)
   Dinyatakan sebagai:
   B(x_1, x_2, ..., x_k)
   Fungsi pembatas menentukan apakah (x_1, x_2, ..., x_k) mengarah ke solusi. Jika ya, maka pembangkitan nilai untuk x_{k+1} dilanjutkan, tetapi jika tidak, maka (x_1, x_2, ..., x_k) dibuang dan tidak dipertimbangkan lagi dalam pencarian solusi.

3.2 Prinsip Pencarian Solusi dengan Metode Runut-balik

Di sini penulis hanya akan meninjau pencarian solusi pada pohon ruang status yang dibangun 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 metode pencarian mendalam (*DFS*). Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup (*live node*). Simpul hidup yang sedang diperluas dinamakan simpul-E (*Expand-node*). Simpul dinomori dari atas ke bawah sesuai dengan urutan kelahirannya (sesuai prinsip DFS).
2. Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi, maka simpul-E "dibunuh" sehingga menjadi simpul mati (*dead node*). Fungsi yang digunakan untuk membunuh simpul-E 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 pencatian solusi dilanjutkan dengan melakukan runut-balik ke simpul hidup terdekat (simpul orangtua). Selanjutnya simpul ini menjadi simpul-E yang baru. Lintasan baru dibangun kembali sampai lintasan tersebut membentuk solusi.
4. Pencarian dihentikan bila kita telah menemukan solusi atau tidak ada lagi simpul hidup untuk runut balik.

4. APLIKASI ALGORITMA RUNUT -BALIK PADA *KNIGHT'S TOUR*

Algoritma runut-balik (*backtracking algorithm*) dapat dipakai untuk menyelesaikan permasalahan dari *Knight's Tour*. Ide dasar dari algoritma runut balik adalah dengan membangun solusi-solusi parsial dari masalah (dalam hal ini jalan selangkah pada papan) lalu mencoba memperluas solusi tersebut. Jika solusi yang telah dikembangkan gagal (langkah selanjutnya tidak menuju solusi) maka akan dilakukan runut-balik dan mencoba solusi parsial lainnya.

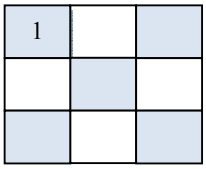
Algoritma runut-balik yang dipakai adalah:

1. Dari kotak tempat kuda tersebut berada, dibangkitkan langkah-langkah berikutnya yang memungkinkan dilalui oleh kuda tersebut.
2. Salah satu langkah (kotak) dipilih untuk diperluas.
3. Kuda melangkah ke kotak yang dipilih tersebut.
4. Kembali ke langkah satu sampai langkah yang diperluas tidak dapat mencapai solusi (kuda tidak dapat melangkah lagi).
5. Pencarian langkah dilakukan dengan melakukan runut balik ke langkah yang telah dilalui terdekat dan kuda melangkah balik ke kotak tersebut dan kembali ke langkah satu.
6. Pencarian langkah dihentikan bila telah melakukan solusi atau tidak ada langkah yang memungkinkan lagi bagi kuda catur tersebut.

Contoh pada papan 3x3:

Dengan menggunakan langkah-langkah yang sudah ditentukan pada algoritma runut-balik di atas maka dapat ditemukan:

Kondisi awal kuda berada pada (1,1)

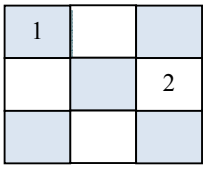


Gambar 2.

Pada tiap kotak pada papan catur, kita akan membuat daftar kotak yang dapat dilalui oleh kuda pada langkah selanjutnya.

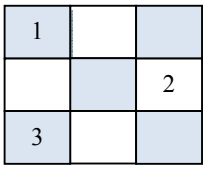
Pada kasus ini, ada dua kemungkinan langkah, yaitu kotak (2,3) dan (3,2).

Pertama kita coba (2,3). Jika langkah pada kotak ini tidak menuju solusi, maka kita akan runut-balik dan mencoba kotak (3,2).



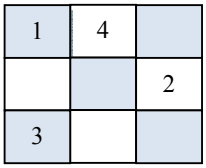
Gambar 3.

Dari kotak ini, ada dua kemungkinan langkah, yaitu (1,1) dan (3,1). Namun karena kotak (1,1) sudah pernah dilewati, maka kita akan mencoba kotak (3,1).



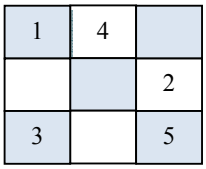
Gambar 4.

Dari kotak ini, ada dua kemungkinan langkah, yaitu (2,3) dan (1,2). Namun karena kotak (2,3) sudah pernah dilewati, maka kita akan mencoba kotak (1,2).



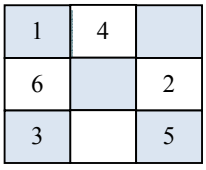
Gambar 5.

Dari kotak ini, ada dua kemungkinan langkah, yaitu (3,1) dan (3,3). Namun karena kotak (3,1) sudah pernah dilewati, maka kita akan mencoba kotak (3,3).



Gambar 6.

Dari kotak ini, ada dua kemungkinan langkah, yaitu (2,1) dan (1,2). Namun karena kotak (1,2) sudah pernah dilewati, maka kita akan mencoba kotak (2,1).



Gambar 7.

Dari kotak ini, ada dua kemungkinan langkah, yaitu (1,3) dan (3,3). Namun karena kotak (3,3) sudah pernah dilewati, maka kita akan mencoba kotak (1,3).

1	4	7
6		2
3		5

Gambar 8.

Dari kotak ini, ada dua kemungkinan langkah, yaitu (2,1) dan (3,2). Namun karena kotak (2,1) sudah pernah dilewati, maka kita akan mencoba kotak (3,2).

1	4	7
6		2
3	8	5

Gambar 9.

Dari kotak ini, ada dua kemungkinan langkah, yaitu (1,1) dan (1,3). Kotak (2,1) sudah pernah dilewati. Kotak (1,3) juga sudah pernah dilewati.

Tidak ada langkah yang memungkinkan. Maka kita akan melakukan runut-balik.

1	4	7
6		2
3		5

Gambar 10.

Tidak ada langkah yang memungkinkan. Maka kita kembali akan melakukan runut-balik.

1	4	
6		2
3		5

Gambar 11.

Langkah lainnya yang memungkinkan adalah kotak (3,3). Namun kotak tersebut sudah pernah dilalui oleh kuda. Maka runut-balik kembali dilakukan.

1	4	
		2
3		5

Gambar 12.

Tidak ada langkah yang memungkinkan. Maka kita kembali akan melakukan runut-balik.

1	4	
		2
3		

Gambar 13.

Tidak ada langkah yang memungkinkan. Maka kita kembali akan melakukan runut-balik.

1		
		2
3		

Gambar 14.

Tidak ada langkah yang memungkinkan. Maka kita kembali akan melakukan runut-balik.

1		
		2

Gambar 15.

Langkah lainnya yang memungkinkan adalah kotak (1,1). Namun kotak tersebut sudah pernah dilalui oleh kuda. Maka runut-balik kembali dilakukan.

1		

Gambar 16.

Kotak (3,2) adalah pilihan lain dari kotak (1,1) yang belum pernah dicoba. Maka kuda melangkah ke kotak tersebut.

Proses dilakukan lagi. Ketika telah berada di akhir proses, semua kemungkinan langkah telah dicoba. Dan setelah dicoba dengan langkah apapun papan dengan ukuran 3x3 selalu menghasilkan satu kotak yang tidak pernah dilalui oleh kuda. Maka proses dihentikan dan dapat diketahui bahwa pada papan 3x3 tidak ditemukan adanya *Closed Knight's Tour*.

Algoritma runut-balik ini dapat ditulis ke dalam *pseudo code* sebagai berikut:

```
{
    papan berukuran nxn
    (x,y) adalah koordinat (baris,kolom) dari kotak
    langkah adalah nomer kotak yang telah dikunjungi
    ok adalah Boolean yang menandakan sukses atau
    gagal
}
type papan_catur is array (1..n,1..n) of integer;
procedure benteng(papan: in out papan_catur;
                  x,y,langkah: in out integer;
                  ok: in out boolean) is w,z: integer;)
begin
    if langkah = n^2+1 then
       ok := ((x,y) = (1,1));
    elseif papan(x,y) /= 0 then
```

```
        ok := false;
      else
        papan(x,y) := langkah;
        loop
            (w,z) := Langkah berikutnya dari (x,y);
            benteng(papan, w, z, langkah+1, ok);
            exit when (ok or tidak ada langkah lain);
        end loop;
        if not ok then
            papan (x,y) := 0;  - - Backtracking
        end if;
      end if;
  end papan;
```

Algoritma runut-balik ini juga bisa diaplikasikan terhadap papan catur dengan ukuran yang lebih besar lagi dengan salah dua dari kemungkinan hasil yang didapat adalah sebagai berikut:

10	5	12	1
13	2	9	6
8	11	4	15
3	14	7	

Gambar 17. Papan 4 x 4 tidak pernah terjadi *Closed Knight Tour*

10	31	16	23	8	1
17	24	9	2	15	22
32	11	30	21	36	7
25	18	3	12	29	14
4	33	20	27	6	35
19	26	5	34	13	28

Gambar 18. Salah satu dari 6 *Closed Knight Tour* pada Papan 6 x 6 s

5. KESIMPULAN

Knight's Tour adalah salah satu masalah dari aplikasi teori graf pada permainan catur papan. Ada banyak strategi algoritmik yang dapat diterapkan untuk menentukan *Knight's Tour* pada permainan tersebut. Salah satunya dapat menggunakan algoritma runut-balik (*backtracking algorithm*).

REFERENSI

[1] Mumtaz, Fahmi. "Aplikasi Teori *Graf* pada *Knight's Tour*", 2007.

[2] Munir, Rinaldi. "Diktat Kuliah IF2251 Strategi Algoritmik", Program Studi Teknik Informatika STEI ITB, 2006.

[3] Lewandowski, Gary. "Project 1: *The Knight's Tour* Gary Lewandowski", CSCI 220, fall 2001

[4] http://www.csc.liv.ac.uk/%7Eped/algor_complete.html Tanggal akses: 19 Mei 2008 pukul 14:23

[5] http://www.daniweb.com/forums/post609524.html Tanggal akses: 19 Mei 2008 pukul 17:12

[6] http://web.telia.com/~u85905224/knight/eknight.htm Tanggal akses: 19 Mei 2008 pukul 21:45