PERMAINAN KNIGHT'S TOUR DENGAN ALGORITMA BACKTRACKING DAN ATURAN WARNSDORFF

Fransisca Cahyono (13509011) [1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*13509011@std.stei.itb.ac.id*

Abstract*—Permainan catur adalah permainan yang unik dan dapat mengasah ketajaman otak pemainnya. Permainan ini juga menarik karena terdapat berbagai jenis bidak yang memiliki karakteristik pergerakan berbeda-beda. Permainan knight's tour adalah perkembangan dari permainan catur yang hanya menggunakan bidak kuda untuk melewati setiap kotak pada papan permainan tanpa pernah menginjak kotak yang sama dua kali. Untuk memecahkan permasalahan pada permainan knight's tour tersebut dengan cepat, digunakan algoritma backtracking yang memungkinkan pencarian untuk "kembali" ke langkah sebelumnya dan menemukan langkah lain untuk memecahkan masalah. Langkah yang paling optimal dalam pencarian solusi permainan ini dapat dicari dengan menggunakan aturan Warnsdorff (Warnsdorff rules).

***Index Terms*—backtracking, catur, knight's tour, Warnsdorff.**

1. PENDAHULUAN

Permainan catur pertama kali ditemukan di masyarakat Persia dan Arab. Menurut H. J. R. Murray, penulis buku History of Chess (1913), catur berasal dari India dan mulai ada pada abad ke-6. Di sana catur dikenal dengan nama *chaturanga*, yang artinya empat unsur yang terpisah. Awalnya, bidak catur memang hanya empat jenis, yaitu raja, benteng, ksatria (kuda), dan uskup (gajah). Menurut mistisisme India kuno, catur dianggap mewakili alam semesta ini, sehingga sering dihubungkan dengan empat unsur kehidupan, yaitu api, udara, tanah dan air karena dalam permainannya, catur menyimbolkan cara-cara hidup manusia. Dalam permainannya, catur mengandalkan analisa dan ketajaman otak pemain, disertai keterampilan strategi dalam menentukan langkah, rencana, risiko, dan menentukan kapan harus berkorban agar menang.

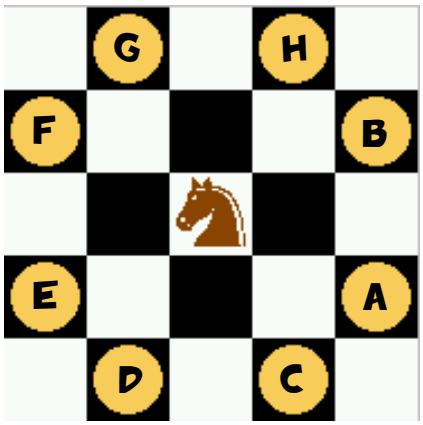
Salah satu bidak catur yang memiliki pergerakan unik adalah bidak kuda. Bidak kuda ini hanya dapat bergerak dengan arah pergerakan membentuk huruf L. Karena keunikannya itulah akhirnya terbentuk sebuah permasalahan yang berhubungan dengan pergerakan bidak kuda pada papan catur. Permasalahan tersebut menarik untuk dipecahkan dan akhirnya membentuk sebuah permainan pengasah otak yang disebut *knight's tour*.

2. KNIGHT'S TOUR

Knight's tour adalah salah satu permainan catur klasik yang menggunakan langkah bidak kuda pada papan catur. Aturan dari permainan ini sederhana. Permainan dimulai dengan meletakkan bidak kuda pada sebuah titik di atas papan catur kosong. Papan yang digunakan dapat berukuran 5x5, 6x6, 8x8, dan sebagainya. Ukuran papan minimum yang dapat digunakan adalah 5x5. Untuk papan berukuran 3x3, 4x4, atau yang lebih kecil, tidak akan ditemukan solusi permainannya. Jika papan berbentuk persegi panjang, ukuran minimum papan adalah 3x4.

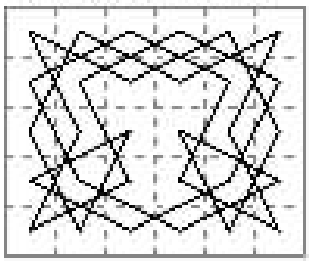
Bidak kuda yang tadi telah diletakkan tersebut harus berjalan sesuai dengan langkah kuda pada permainan catur. Langkah yang dapat ditempuh antara lain:

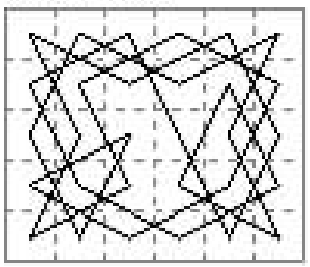
a. Dua langkah ke arah kanan dan satu langkah ke arah bawah
b. Dua langkah ke arah kanan dan satu langkah ke arah atas
c. Dua langkah ke arah bawah dan satu langkah ke arah kanan
d. Dua langah ke arah bawah dan satu langkah ke arah kiri
e. Dua langkah ke arah kiri dan satu langkah ke arah bawah
f. Dua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



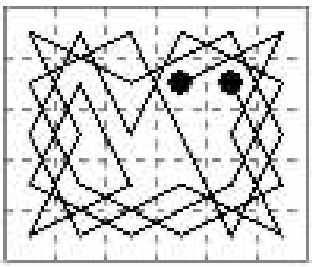
Gambar 2.1 Langkah Kuda dalam Permainan Catur

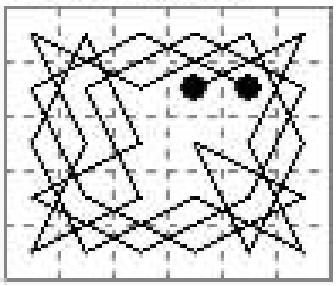
Permainan knight's tour ini akan selesai jika bidak kuda telah menginjak semua titik pada papan catur tanpa pernah menginjak titik yang sama dua kali.
Ada dua jenis permainan knight's tour, yaitu *open knight's tour* dan *closed knight's tour*. *Open knight's tour* terjadi jika bidak kuda telah melalui seluruh titik pada papan dan tidak dapat kembali ke titik awal mulai permainan. Jika bidak kuda telah melalui seluruh titik pada papan dan pada akhirnya kembali ke titik awal permainan, permainan tersebut merupakan *closed knight's tour.*





Gambar 2.2 Closed Knight's Tour





Gambar 2.3 Open Knight's Tour

1	14	9	20	3
24	19	2	15	10
13	8	25	4	21
18	23	6	11	16
7	12	17	22	5

Gambar 2.4 Contoh Penyelesaian dalam Knight's Tour 5x5

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

Gambar 2.5 Contoh Penyelesaian dalam Knight's Tour 6x6

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

Gambar 2.6 Contoh Penyelesaian dalam Knight's Tour 8x8

Untuk menyelesaikan *knight's tour* ini, dapat digunakan berbagai cara. Apabila digunakan konsep matematika secara umum, yaitu konsep faktorial, maka kemungkinan yang terjadi untuk papan berukuran 8x8 adalah 64! = 1.27×10^{89}, sehingga membutuhkan waktu yang sangat lama untuk menyelesaikannya. Jika digunakan konsep eksponensial, kemungkinan yang terjadi sekitar 64×4^{63}. Dengan konsep simetri, kemungkinannya paling sedikit adalah 8.5×10^{38}.

Salah satu cara yang mempersingkat waktu pencarian solusi dalam permainan *knight's tour* ini adalah dengan menggunakan algoritma *backtracking* dan aturan Warnsdorff.

3. ALGORITMA BACKTRACKING

Runut-balik (*backtracking*) adalah algoritma yang berbasis pada DFS untuk mencari solusi persoalan secara lebih mangkus. Runut-balik, yang merupakan perbaikan dari algoritma *brute-force*, secara sistematis mencari solusi persoalan di antara semua kemungkinan solusi yang ada. Istilah *backtracking* pertama kali diperkenalkan oleh D. H. Lehmer pada tahun 1950.

Algoritma *backtracking* banyak diterapkan untuk program *games* :

– permainan *tic-tac-toe*,
– maze,
– Catur,
– Knight's tour,
– 8 queen
– masalah-masalah kecerdasan buatan (*artificial intelligence*).

Properti Umum Metode *Bactracking*

1. Solusi persoalan.
   Solusi dinyatakan sebagai vektor dengan *n-tuple*: $X = (x1, x2, \ldots, xn)$, $xi \hat{I} Si$. Mungkin saja $S1 = S2 = \ldots = Sn$.
   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
   $B(x1, x2, \ldots, xk)$
   B bernilai *true* jika $(x1, x2, \ldots, xk)$ mengarah ke solusi. Jika *true*, maka pembangkitan nilai untuk *xk*+1dilanjutkan, tetapi jika *false*, maka $(x1, x2, \ldots, xk)$ 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 $| S1| \cdot | S2| \cdot \ldots \cdot | Sn |$.

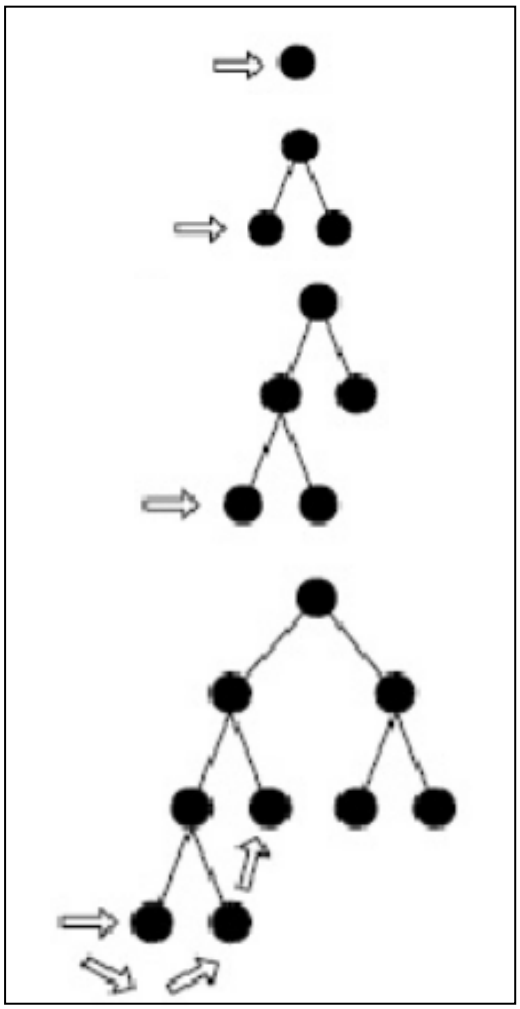
Pencarian Solusi dengan Backtracking

Solusi dari metode backtracking ini dicari dengan membentuk lintasan dari akar ke daun. Aturan yang dipakai mengikuti aturan pencarian DFS (*Depth First Search*).

Simpul-simpul yang telah dihasilkan dinamakan simpul hidup (*live node*) dan simpul yang sedang diperluas dinamakan simpul E (*expand node*). Setiap kali dilakukan perluasan pada simpul E, lintasan yang terbentuk akan semain panjang. Jika lintasan yang dibentuk oleh simpul E tidak mengarah pada solusi, maka simpul tersebut akan dibuang, tidak dipakai lagi, dan menjadi simpul mati (*dead node*). Fungsi yang dipakai untuk membuang simpul E tersebut dinamakan fungsi pembatas (*bounding function*). Bila pencarian berakhir di simpul mati, proses pencarian akan dilakukan ke simpul anak lainnya. Namun, bila sudah tidak ada lagi simpul anak yang dapat digunakan, maka pencarian akan dilakukan *backtracking* ke simpul orang tuanya. Simpul ini akan menjadi simpul E yang baru. Pencarian akan dihentikan jika solusi telah ditemukan atau tidak dapat lagi melakukan *backtracking* ke simpul sebelumnya (simpul hidup sudah tidak ada lagi).

Algoritma *Backtracking* dalam *Knight'sTour* adalah sebagai berikut :

1. Permainan dimulai dari titik (kotak) awal kuda ditempatkan dan membangkitkan langkah-langkah (simpul) yang mungkin dilalui oleh kuda.
2. Memilih salah satu langkah (titik). Langkah tersebut kemudian diperluas.
3. Menempatkan bidak kuda pada kotak yang telah dipilih.
4. Mengulangi langkah pertama untuk titik yang sedang ditempati.
5. Jika belum ditemukan solusi, kembali ke langkah sebelumnya (*backtracking*).
6. Pencarian berhenti jika telah ditemukan solusi atau tidak ada lagi langkah yang memungkinkan.



Gambar 3.1 Ilustrasi Pembentukan Simpul dalam *backtracking*

Berikut adalah contoh *pseudocode* dari algoritma *backtracking* untuk kasus *Knight's Tour:*

- board adalah matriks n x n (ukuran dari papan)
- (x,y) adalah koordinat letak kotak
- move adalah nomor kotak yang telah dilewati
- ok adalah boolean apakah sukses atau gagal

```
type chess_board is array (1..n,1..n)
      of integer;

procedure knight (board : in out
      chess_board;
      x,y,move : in out integer;
      ok : in out Boolean) is

w, z : integer;
begin
   if move = n^2+1 then
      ok := ( (x,y) = (1,1) );
   elsif board(x,y) /= 0 then
      ok := false;
   else
      board(x,y) := move;
      loop
         (w,z) := Next position from
(x,y);
         knight(board, w, z, move+1,
ok );
      exit when (ok or No moves
remain);
      end loop;
      if not ok then
         board ( x,y ) :=0;
         Backtracking
      end if;
   end if;
end knight;
```

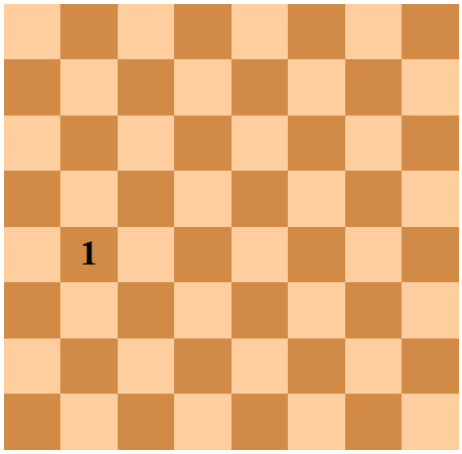
Gambar 3.2 Salah satu *pseudocode* untuk memecahkan permainan *Knight's Tour*

4. ATURAN WARNSDORFF

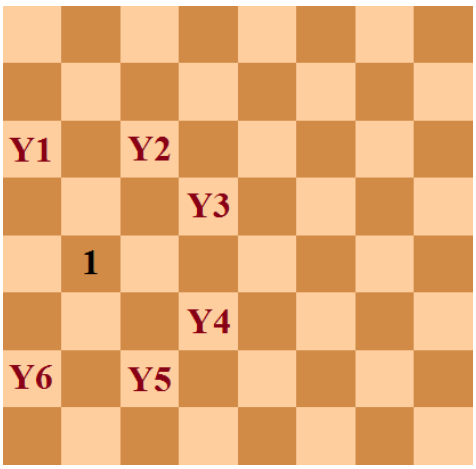
Aturan Warnsdorff adalah salah satu metode heuristik yang ditemukan oleh seorang matematikawan bernama H. C. von Warnsdorff pada tahun 1823 dalam karyanya yang berjudul "*Des Rösselsprungs einfachste und allgemeinste Lösung*" (yang artinya : *The Knight's Simplest and Most General Move Solution*).

Langkah-langkah yang digunakan dalam algoritma Warnsdorff untuk memecahkan permainan *knight's tour* ini adalah sebagai berikut:

1. Pilih posisi X secara random pada papan permainan dan tandai posisi tersebut sebagai posisi awal (ditandai dengan angka 1).



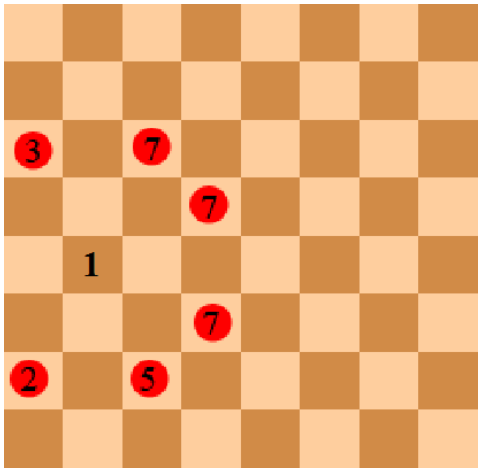
2. Misalkan posisi-posisi selanjutnya yang dapat diakses dari posisi 1 disimpan dalam *array* Y. Pada contoh di bawah ini, posisi yang dapat diakses dari posisi 1 adalah Y1, Y2, Y3, Y4, Y5, dan Y6.



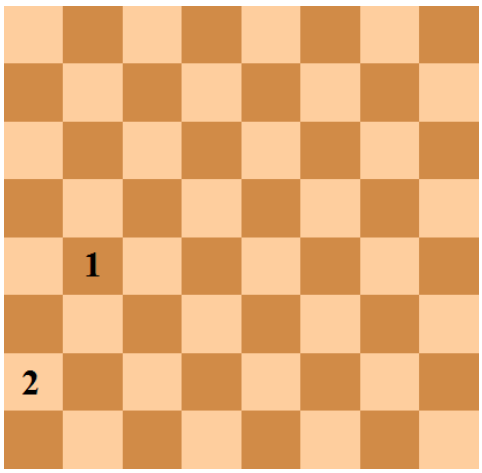
3. Untuk setiap posisi yang tersimpan dalam Y, hitung berapa langkah maksimal yang dapat diakses dari posisi pada tiap Y. Posisi yang dihitung haruslah posisi yang belum pernah dilewati sebelumnya. Mengacu pada Gambar 2.1, langkah yang dapat dilakukan dari masing-masing posisi Y1-Y6 adalah:

   Y1 : A, B, H (3 langkah)
   Y2 : A, B, C, E, F, G, H (7 langkah)
   Y3 : A, B, C, D, F, G, H (7 langkah)
   Y4 : A, B, C, D, E, G, H (7 langkah)
   Y5 : A, B, E, F, H (5 langkah)
   Y6 : A, B (2 langkah)

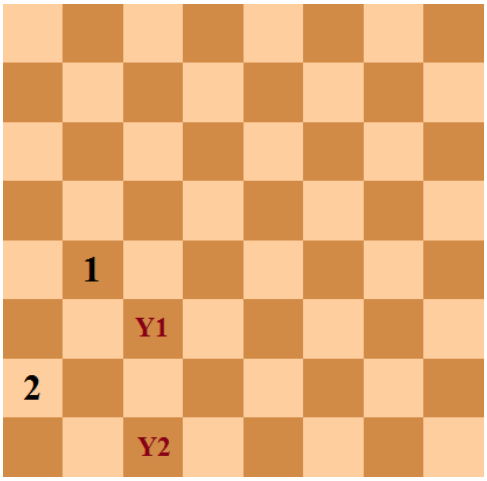
   Masukkan jumlah langkah maksimal tersebut ke dalam *array* Y.

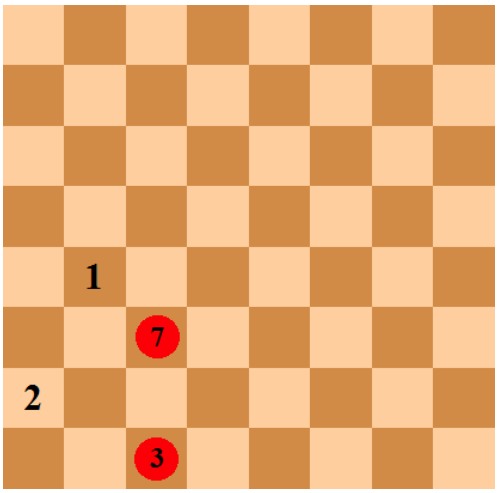


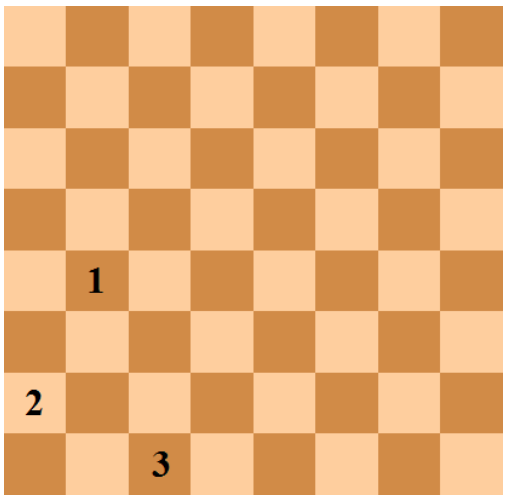
4. Posisi selanjutnya dipilih berdasarkan nilai pada *array* Y yang paling kecil. Pada contoh ini, posisi selanjutnya (posisi 2) adalah Y6 yang memiliki langkah sebanyak 2. Masukkan angka 2 untuk menandakan bahwa posisi tersebut telah dilalui.



5. Pencarian dilanjutkan dengan mengulangi langkah nomor 2-4 untuk X = posisi 2 dan seterusnya sampai seluruh papan permainan terisi. Berikut adalah langkah untuk memilih posisi nomor 3.







6. Hasil permainan yang telah selesai dapat dilihat pada gambar di bawah ini.

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

Dengan menggunakan algoritma Warnsdorff, simpul yang harus dipilih untuk mencapai solusi permainan akan optimal dan kemungkinan terjadinya kesalahan dalam pemilihan simpul juga menjadi semakin kecil. Hal ini dikarenakan dengan memilih posisi yang memiliki langkah terkecil, maka peluang bahwa langkah itu benar semakin besar. Contohnya pada kasus di atas :

Peluang sebuah langkah di Y benar adalah:

$$P(Y) = \frac{1}{jumlah\ langkah\ maksimum}$$

Pada posisi 1 :

Y1 : 3 langkah → P (Y1) = $\frac{1}{3}$

Y2 : 7 langkah → P (Y2) = $\frac{1}{7}$

Y3 : 7 langkah → P(Y3) = $\frac{1}{7}$

Y4 : 7 langkah → P (Y4) = $\frac{1}{7}$

Y5 : 5 langkah → P (Y5) = $\frac{1}{5}$

Y6 : 2 langkah → P (Y6) = $\frac{1}{2}$

Karena peluang kebenaran pada Y6 bernilai paling besar, maka langkah ke Y6 dari posisi 1 adalah langkah yang paling optimal.

5. 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 ini mempermudah pencarian langkah-langkah yang harus ditempuh untuk menyelesaikan permainan *knight's tour* ini.

Algoritma Warnsdorff adalah salah satu algoritma yang dapat mempercepat proses pencarian simpul secara *backtracking* yang akan dipilih dalam proses pencarian solusi pada permainan *knight's tour* ini. Algoritma ini akan memperkecil terjadinya kesalahan pemilihan simpul dengan memilih rute dengan peluang kebenaran yang paling besar.

REFERENSI

[1] Munir, Rinaldi. 2009. "Diktat Kuliah IF3051 Strategi Algoritma", Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung

[2] http://en.wikipedia.org/wiki/Knight%27s_tour (Waktu akses : 4 Desember 2011)

[3] http://en.wikipedia.org/wiki/Knight%27s_Tour#Warnsdorff.27s_rule (Waktu akses : 5 Desember 2011)

[4] http://gamatika.wordpress.com/2011/04/26/knight%E2%80%99s-tour/ (Waktu akses : 5 Desember 2011)

[5] http://www.scribd.com/doc/54860174/ian-Knight-Tour (Waktu akses : 4 Desember 2011)

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, 5 Desember 2011

Fransisca Cahyono (13509011)