Aplikasi Algoritma *Brute Force* Pada *Knight's Tour Problem*

Sahat Nicholas Simangunsong - 13509095
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509095@std.stei.itb.ac.id

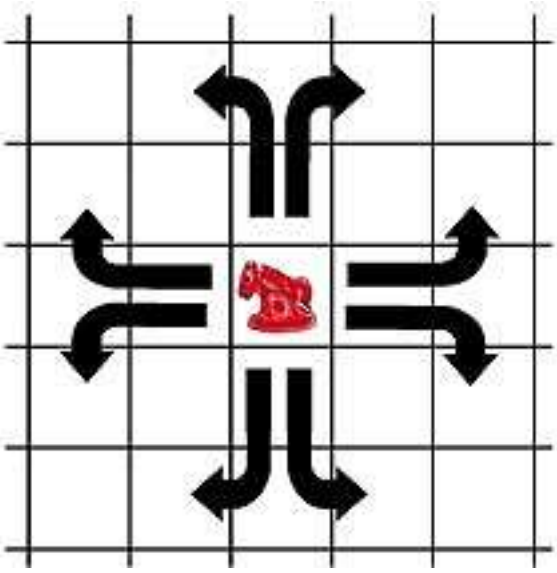
***Abstract*—*Knight's tour* merupakan persoalan matematis bagaimana bidak kuda melewati seluruh petak yang ada di papan catur tepat hanya sekali. Ada dua jenis *knight's tour*, yaitu *open knight's tour* dan *closed knight's tour*. Pada *open knight's tour*, bidak kuda harus melewati seluruh petak pada papan catur, tetapi tidak kembali ke posisi awalnya, sedangkan pada *closed knight's tour*, bidak kuda melewati seluruh petak pada papan catur dan kembali ke posisi awalnya. Terdapat berbagai solusi untuk memecahkan persoalan *knight's tour*. Salah satunya adalah dengan menggunakan algoritma *brute force*. Hal inilah yang akan dibahas pada makalah ini.**

Kata kunci*—*knight's tour*, *brute force

1. PENDAHULUAN

1.1 *Knight's Tour*

Knight's tour merupakan persoalan matematis yang melibatkan sebuah bidak kuda pada sebuah papan catur yang kosong. Bidak kuda memiliki gerakan seperti huruf L dan tidak seperti bidak yang lain, bidak kuda dapat melompati bidak-bidak yang lain.



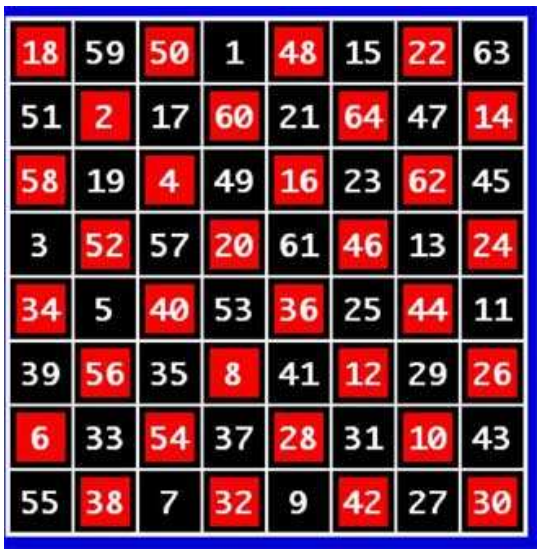
Gambar 1 – Arah Pergerakan Bidak Kuda

Pada *knight's tour*, bidak kuda diletakkan pada papan catur kosong dan harus melewati seluruh petak pada papan catur sebanyak satu kali. Terdapat dua jenis *knight's tour*, yaitu *open knight's tour* dan *closed knight's tour*. Pada *open knight's tour*, bidak kuda tidak kembali ke posisi awalnya setelah melewati seluruh petak pada papan catur. Sebaliknya, pada *closed knight's tour*, bidak kuda kembali ke posisi awalnya setelah melewati seluruh petak pada papan catur.

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

Gambar 2 – *Open Knight's Tour*

Pada gambar di atas, dapat dilihat sebuah *open knight's tour*. Bidak kuda bergerak dari petak 1 dan berakhir di petak 64. Disini jelas terlihat bahwa bidak kuda telah melewati seluruh petak pada papan catur, tetapi tidak kembali ke posisi awalnya. Karena itulah dinamakan *open knight's tour*.



Gambar 3 – *Closed Knight's Tour*

Pada gambar di atas, dapat dilihat sebuah *closed knight's tour*. Bidak kuda bergerak dari petak 1 dan berakhir di petak 1 juga. Disini jelas terlihat bahwa bidak kuda telah melewati seluruh petak pada papan catur dan kembali ke posisi awalnya. Karena itulah dinamakan *closed knight's tour*.

1.2 Algoritma *Brute Force*

Algoritma *brute force* merupakan sebuah pendekatan yang *straightforward* untuk menyelesaikan suatu persoalan. Algoritma ini bekerja dengan mencari seluruh kemungkinan solusi yang ada. *Brute force* termasuk jenis algoritma yang sederhana jika dibandingkan dengan algoritma-algoritma lainnya seperti algoritma *greedy,* algoritma *breadth first search*, algoritma *divite and conquer*, dan sebagainya. Jika dilihat dari sisi kemangkusan algoritma, algoritma *brute force* merupakan algoritma yang kurang mangkus dan kurang cocok digunakan untuk memecahkan persoalan dengan waktu yang cepat.

Salah satu kelebihan dari algoritma *brute force* yaitu dalam pembuatannya, tidak diperlukan pemikiran yang terlalu mendalam. Untuk persoalan dengan skala yang kecil, algoritma ini cocok untuk digunakan. Alasannya adalah karena skala programnnya kecil, *execution time*-nya tentu tidak akan lama. Jika algoritma ini digunakan untuk membuat suatu *game* yang memiliki tingkat kompleksitas yang tinggi, tentu saja *game* akan berjalan dengan sangat lambat.

2. METODE

Persoalan *knight's tour* dapat diselesaikan dengan menggunakan algoritma *brute force*. Berikut merupakan langkah-langkah penyelesaian persoalan *knight's tour* pada papan catur berukuran 8x8 dengan menggunakan algoritma *brute force*.

1. Letakkan kuda di sembarang petak.
2. Lakukan pergerakan ke suatu arah secara terus-menerus. Jika bidak kuda tidak dapat lagi bergerak ke arah tersebut, gerakkan bidak kuda ke arah yang lain secara terus-menerus juga. Begitu seterusnya.
3. Untuk setiap pergerakan bidak kuda, lakukan pemeriksaan terhadap hal berikut :
    a. Periksa apakah pergerakan selanjutnya tidak keluar dari papan catur.
    b. Periksa apakah petak tempat pergerakan selanjutnya sudah pernah dilewati atau tidak.
4. Jika bidak kuda tidak dapat bergerak lagi sebelum seluruh petak dilewati, kembalikan bidak kuda ke posisi sebelumnya, kemudian gerakkan bidak kuda ke arah yang lain.
5. Ulangi langkah 2 sampai seluruh petak telah dilewati.

Berikut merupakan potongan *source code* dalam bahasa java yang digunakan untuk menyelesaikan persoalan *knight's tour* dengan menggunakan algoritma *brute force*.

KnightTour.java

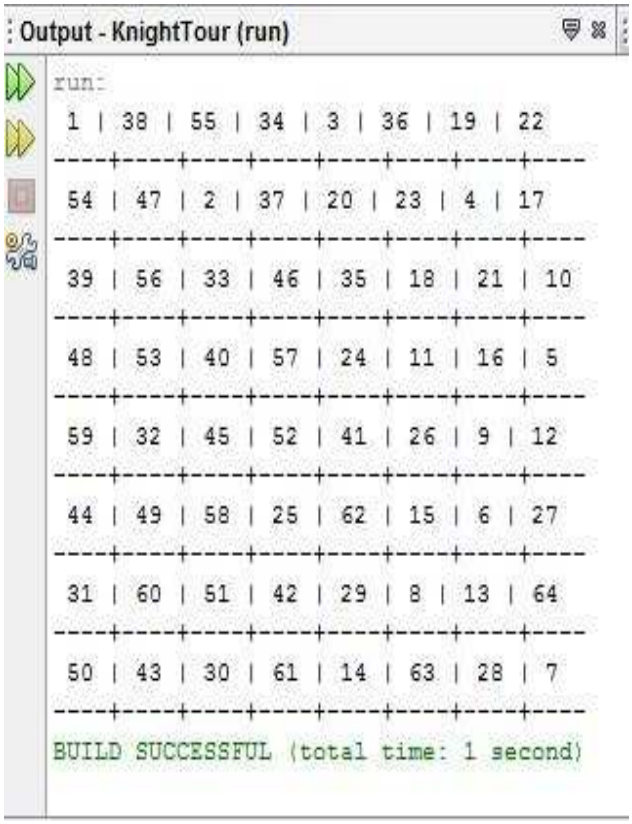
```
package knighttour;

import java.util.*;

public class KnightTour {
   /* VARIABLE */

   /* FUNCTION */
   public void initKnightPosition() {
   /*
         meletakkan bidak kuda pada papan catur secara acak
   */
   }

   public void moveRight1Up2() {
   /*
         menggerakkan bidak kuda ke kanan sebanyak satu langkah dan ke atas sebanyak dua langkah
   */
   }

   public void moveRight2Up1() {
   /*
         menggerakkan bidak kuda ke kanan sebanyak dua langkah dan ke atas sebanyak satu langkah
   */
   }

   public void moveRight2Down1() {
   /*
```

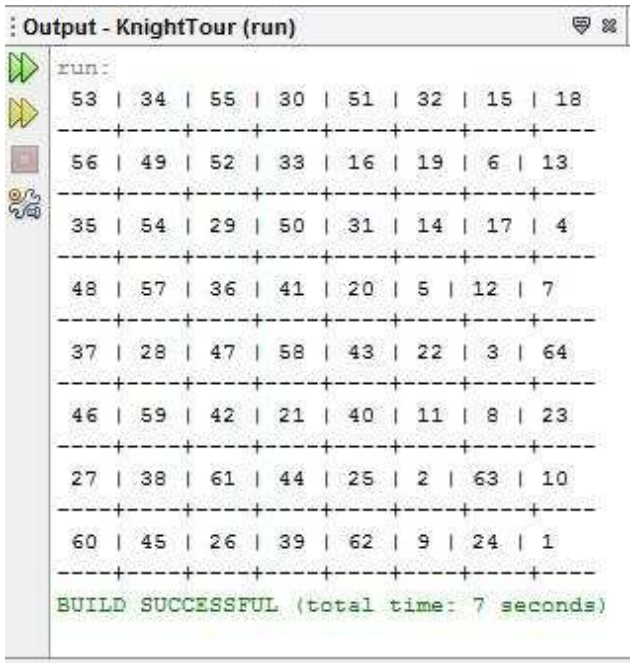
```
        menggerakkan bidak kuda ke kanan sebanyak dua langkah dan ke bawah sebanyak satu langkah
    */
    }

    public void moveRight1Down2() {
    /*
        menggerakkan bidak kuda ke kanan sebanyak satu langkah dan ke bawah sebanyak dua langkah
    */
    }

    public void moveLeft1Down2() {
    /*
        menggerakkan bidak kuda ke kiri sebanyak satu langkah dan ke bawah sebanyak dua langkah
    */
    }

    public void moveLeft2Down1() {
    /*
        menggerakkan bidak kuda ke kiri sebanyak dua langkah dan ke bawah sebanyak satu langkah
    */
    }

    public void moveLeft2Up1() {
    /*
        menggerakkan bidak kuda ke kiri sebanyak dua langkah dan ke atas sebanyak satu langkah
    */
    }

    public void moveLeft1Up2() {
    /*
        menggerakkan bidak kuda ke kiri sebanyak satu langkah dan ke atas sebanyak dua langkah
    */
    }

    public void setDirection(int moveTo) {
    /*
        menentukan arah pergerakan bidak kuda
    */
    }

    public void checkCell() {
    /*
        mengecek petak pada papan catur apakah sudah pernah dilewati atau belum
    */
    }

    public void reverseMovement(int moveTo) {
    /*
        mengembalikan bidak kuda ke posisi sebelumnya
    */
    }

    public void solveKnightTour() {
    /*
        fungsi utama untuk menyelesaikan persoalan knight's tour
    1. Letakkan kuda di sembarang petak.
    2. Lakukan pergerakan ke suatu arah secara terus-menerus. Jika bidak kuda tidak dapat lagi bergerak ke arah tersebut, gerakkan bidak kuda ke arah yang lain secara terus-menerus juga. Begitu seterusnya.
    3. Untuk setiap pergerakan bidak kuda, lakukan pemeriksaan terhadap hal berikut :
        a. Periksa apakah pergerakan selanjutnya tidak keluar dari papan catur.
        b. Periksa apakah petak tempat pergerakan selanjutnya sudah pernah dilewati atau tidak.
    4. Jika bidak kuda tidak dapat bergerak lagi sebelum seluruh petak dilewati, kembalikan bidak kuda ke posisi sebelumnya, kemudian gerakkan bidak kuda ke arah yang lain.
    5. Ulangi langkah 2 sampai seluruh petak telah dilewati.
    */
    }
}
```

Main.java

```
package knighttour;

public class Main {

    public static void main(String[] args) {
            KnightTour knightTour = new KnightTour();
            knightTour.solveKnightTour();

            for (int i=0; i<knightTour.chessBoard.length; i++) {
                    for (int j=0; j<knightTour.chessBoard.length; j++) {
                            if (j==7) {
                                    System.out.println(" " + knightTour.chessBoard[j][i]);
                                    System.out.println("-----+----+----+----+----+----+----+----");
                                    }
                            else {
                                    System.out.print(" " + knightTour.chessBoard[j][i] + " |");
                            }
                    }
            }
    }
}
```

Berikut merupakan *screenshot* dari hasil eksekusi *source code* di atas.





Gambar 4 – Solusi Persoalan *Knight's Tour* dengan menggunakan algoritma *brute force*

Dari gambar-gambar di atas, dapat dilihat bahwa bidak kuda telah melewati seluruh petak pada papan catur, tetapi tidak kembali ke posisi awal (*open knight's tour*). Bidak kuda mulai bergerak dari petak 1 dan berakhir di petak 64. Algoritma *brute force* yang digunakan pada program ini hanya bisa memecahkan persoalan *open knight's tour*, tetapi tidak dengan *closed knight's tour*.

3. KESIMPULAN

Algoritma *brute force* memiliki banyak aplikasi dalam kehidupan. Salah satu contohnya adalah menyelesaikan persoalan *knight's tour*. Meskipun algoritma ini mampu menyelesaikan persoalan *knight's tour*, tetapi waktu yang diperlukan oleh algoritma ini untuk menyelesaikan persoalan ini tergolong lama atau dengan kata lain, algoritma ini kurang mangkus. Karena itulah, algoritma ini hanya cocok untuk menyelesaikan *open knight's tour*, tetapi tidak dengan *closed knight's tour*.

Jika algoritma *brute force* digunakan untuk menyelesaikan persoalan *closed knight's tour*, tentu saja dibutuhkan waktu yang sangat lama untuk menyelesaikan persoalan ini. Mencari jalur yang tepat agar bisa kembali ke posisi awal tentunya membutuhkan waktu yang lama. Penggunaan algoritma seperti algoritma *breadth first search,* algoritma *depth first search*, algoritma *divide and conquer*, dan lain-lain akan lebih cocok untuk menyelesaikan persoalan *knight's tour* dikarenakan algoritma-algoritma tersebut mangkus sehingga memiliki waktu eksekusi yang lebih singkat.

REFERENSI

[1] http://dedyjlsn.wordpress.com/2008/03/24/algoritma-brute-force/
tanggal akses : 08 Desember 2011
waktu akses : 14:00

[2] http://en.wikipedia.org/wiki/Knight%27s_tour
tanggal akses : 08 Desember 2011
waktu akses : 14:05

[3] http://pascal-central.com/images/knight-fig2.jpg
tanggal akses : 08 Desember 2011
waktu akses : 14:10

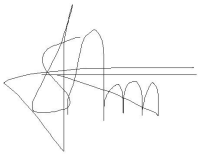
[4] http://pascal-central.com/images/knight-fig1.jpg
tanggal akses : 08 Desember 2011
waktu akses : 14:15

[5] http://4.bp.blogspot.com/_EngB9VOPD1U/Suz_216KVpI/AAAAAAAAACI/gUvqn4XziPA/s320/shatar_mongolian_chess_knight_horse-move.jpg
tanggal akses : 08 Desember 2011
waktu akses : 14:20

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, 08 Desember 2011



Sahat Nicholas Simangunsong - 13509095