Perbandingan Efektivitas Algoritma Brute Force dengan Algoritma Backtracking dalam Penyelesaian Persoalan Knight Tour

Fathan Mubina -13518064
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail : fathmubina12@gmail.com

***Abstract*—Permainan Catur adalah sebuah permainan papan yang mengandalkan strategi. Dalam permainan ini pemain berusaha mengalahkan lawan dengan membuat lawan dalam kondisi skakmat dengan menggunakan bidak-bidak secara bergantian. Bidak dalam permainan ini terdapat enak jenis yaitu raja, menteri, gajah, kuda, benteng, dan pion. Masing-masing bidak memiliki pola pergerakan tertentu. Kuda memiliki pergerakan yang unik yaitu membentuk huruf L. Pergerakan kuda untuk melewati semua kotak pada papan catur tepat satu kali yang inilah yang disebut Persoalan *Knight's Tour*.**

Persoalan *Knight's Tour* juga merupakan salah satu persoalan lintasan hamilton dalam teori graf. Dalam dalam persoalan *Knight's Tour* ini, ada beberapa Algoritma yang dapat digunakan untuk menyelesaikannya diantaranya adalah Algoritma Backtracking dan BruteForce. Dalam makalah ini akan dibahas perbandingan efektivitas Algoritma Brute Force dengan Algoritma Backtracking dalam penyelesaian persoalan *Knight's Tour*.

***Keywords*—Catur, kuda, Brute Force, *Knight's Tour*, *Backtracking*, Kontribusi, Algoritma.**

I. PENDAHULUAN

Permainan catur adalah permainan papan bergenre strategi yang dimainkan oleh dua orang. Catur biasanya dimainkan di atas papan kotak-kotak yang terdiri dari 64 kotak, yang disusun dalam ukuran 8x8 kotak, yang terbagi menjadi dua warna yaitu warna hitam dan putih secara berselingan.

Gambar 1.1. Permainan Catur
(sumber : sejarahlengkap.com)

Sejarah mencatat Catur mulai dimainkan pada abad ke-7 di India. Di India catur melambangkan tentang alam semesta yang terbagi menjadi ke empat unsur yaitu api, udara, tanah dan air. Karena itu lah dinamakan Caturanga memiliki makna 'empat unsur yang terpisah,'. Kemudian pedagang Islam dari India membawa permainan ini menuju ke Persia. Disana catur disebut dengan nama shatranj di Sassanid. Kemudian catur juga mulai dikenal di seluruh penjuru dunia. Disini catur juga mengalami perkembangan dari segi permainan sehingga lebih menarik dan lebih seru hingga menjadikan catur sebagai permainan rekreasi paling favorit di Persia. Karena keseruan ini kemudian catur menyebar lagi hingga ke daratan Arab dan akhirnya ke seluruh dunia hingga saat ini.

Cara bermainnya adalah dengan menggerakkan buah catur. Pada mulanya, setiap pemain memiliki 16 buah catur, yaitu satu raja (*king*), satu menteri (*queen* atau ratu), dua benteng (*rook*), dua kuda (*knight* atau kesatrian) , dua gajah (*bishop* atau uskup), dan delapan pion. Setiap jenis buah catur memiliki pergrakan tertentu, dengan yang paling kuat adalah ratu dan yang paling lemah adalah pion.

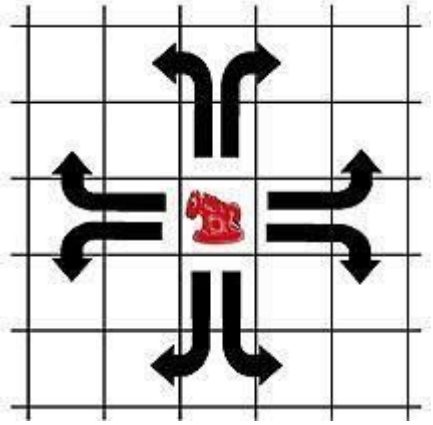
Tujuan dari permainan ini adalah melakukan skakmat pada raja lawan dengan menempatkan buah tersebut pada posisi di mana ancaman untuk ditangkap atau dimakan tidak dapat dihindari. Untuk mencapai tujuan ini, buah-buah milik pemain harus digunakan untuk menyerang dan menangkap buah

lawan, sambil buah-buah tersebut tetap menjaga satu sama lain. Selama permainan berlangsung, jalan permainan biasanya melibatkan makan memakan buah dengan buah lawan, serta penemuan dan perekayasaan peluang untuk melakukan pertukaran secara menguntungkan atau untuk mendapatkan posisi yang lebih baik. Selain skakmat, seorang pemain memenangkan permainan jika pemain lawan menyerah, atau kehabisan waktu dalam permainan catur cepat atau berwaktu. Ada juga beberapa cara yang membuat permainan berakhir dalam keadaan remis atau seri.

Perkembangan permainan catur dari abad ke abad memunculkan banyak perkembangan dan variasi permainan catur dan muncul pula berbagai persoalan seperti persoalan *N-Queen* dan persoalan *Knight Tour*. Persoalan *Knight's Tour* Pergerakan kuda untuk melewati semua kotak pada papan catur tepat satu kali. Terdapat beberapa cara yang dapat digunakan untuk menyelesaikan permasalahan ini diantaranya adalah dengan algoritma Divide and Conquer, algoritma Backtracking, aturan Warnsdorff, dan metode De Moivre.

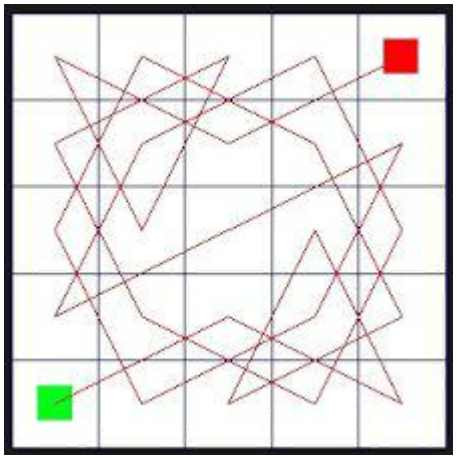
II. Persoalan Knight's Tour

Persoalan Knight's Tour merupakan sebuah persoalan menggunakan kuda dengan cara melewati seluruh kotak yang ada tepat satu kali pada sebuah papan. Permainan selesai ketika seluruh kotak pada papan catur sudah dilewati oleh kuda dengan gerakan seperti dalam permainan catur yaitu membentuk huruf L.



Gambar 3.1. Gerak buah catur kuda

(sumber:mcmahel.blogspot.com)



Gambar 3.2. Lintasan dari Knight's Tour

(sumber: ibmathsresources.com)

Ada dua jenis permainan The Knight's Tour yaitu Closed Tour dan Open Tour. Closed Tour yaitu kondisi dimana posisi awal buah catur kuda juga harus menjadi posisi akhir buah catur kuda setelah menempati semua petak 1 kali. Yang kedua yaitu Open Tour yaitu kondisi dimana posisi awal buah catur kuda dan posisi akhir buah catur kuda tidak harus sama.

Persoalan The Knight's Tour sama seperti menyelesaikan persoalan lintasan Hamilton. Kotak pada papan catur dapat diibaratkan sebagai simpul, sedangkan pilihan jalannya kuda dapat diibaratkan sebagai sisi. Persoalan lintasan Hamilton adalah bagaimana melewati semua simpul tepat satu kali, sama seperti permainan The Knight's Tour yaitu menempati setiap petak tepat satu kali.

III. Brute Force

Algoritma Brute Force merupakan pendekatan yang lempang (straightforward) untuk memecahkan suatu persoalan. Algoritma ini lebih mempertimbangkan solusi dari persoalan tanpa mempertimbangkan efisiensi dan meminimumkan jumlah operasi. Algoritma ini didasarkan pada pernyataan pada persoalan (problem statement) dan definisi konsep yang dilibatkan. Algoritma brute force memecahkan persoalan dengan sangat sederhana, langsung, dan jelas.

Algoritma Brute Force merupakan sebuah metode pemecahan masalah logis yang memiliki kemampuan untuk memperoleh pemecahan masalah dengan baik. Hampir semua masalah yang dipecahkan dengan menggunakan metode algoritma brute force ini. Algoritma ini sederhana dan mudah dipahami. Namun, meskipun hampir semua masalah dapat diselesaikan dengan Brute Force, algoritma brute force jarang menghasilkan algoritma yang mangkus. ALgoritma ini dalam beberapa kasus merupakan algoritma yang lambat. Hal ini disebabkan karena algoritma brute force mencoba seluruh kemungkinan solusi untuk mendapatkan solusi yang diinginkan.

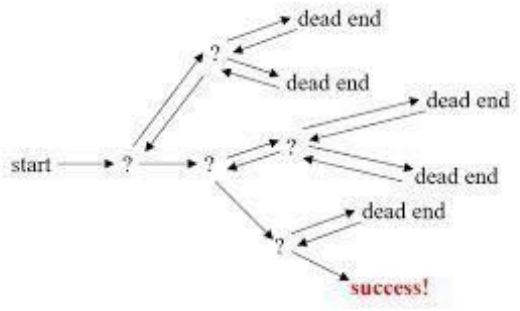
IV. Backtracking

Algoritma bactracking pertama kali diperkenalkan oleh D. H. Lehmer pada tahun 1950. Algoritma backtracking merupakan pengembangan dari algoritma brute force yang algoritma tersebut mencoba semua kemungkinan bahan yang sebenarnya tidak perlu. berbagai permasalahan seperti tic-tac-toe, mazeMath, Knight's Tour, 8 Queen, dan permasalahan artificial inteligence.

Algoritma backtracking bekerja dengan membentuk lintasan dari akar ke daun. Aturan yang digunakan adalah aturan DFS (Depth First Search). Simpul-simpul yang telah dihasilkan

dinamakan simpul hidup (live node), sedangkan simpul yang diperluas dinamakan simpul E (expand node). Simpul yang sudah tidak dipakai lagi dinamakan simpul mati (dead node).

Algoritma backtracking akan mencari solusi parsial dari sebuah simpul tertentu dan kemudian simpul tersebut diperluas. Algoritma backtracking akan membuang simpul jika lintasan yang terbentuk tidak mengarah kepada solusi dan menjadi simpul mati. Untuk membuang simpul tersebut digunakan fungsi pembatas (bounding function). Jika pada saat pencarian berakhir pada simpul mati, maka pencarian akan dilakukan pada simpul anak lainnya. Akan tetapi, jika sudah tidak ada lagi simpul anak yang belum dicari, maka akan dilakukan backtracking ke simpul sebelumnya. Pencarian akan berhenti dilakukan ketika sudah didapatkan solusi atau jika sudah tidak bisa melakukan backtracking lagi yaitu tidak ada simpul hidup (live node) lagi.



Algoritma backtracking dibuat untuk membuang semua kemungkinan yang tidak perlu pada algoritma brute force. Algoritma Backtracking banyak digunakan dalam bidang pembuatan kecerdasan buatan seperti dalam game catur, tic-tac-toe, teka teki silang, sudoku, dan lain sebagainya.

Langkah-langkah backtracking pada persoalan The Knight'sTour yaitu:

I. Awalnya kuda ditempatkan, lalu mendata semua langkah-langkah yang mungkin dilalui oleh kuda dari penempatan kuda pertama kali.

II. Mengambil salah satu langkah yang merupakan simpul hidup (live node) lalu langkah tersebut diperluas lagi. dan menempatkan kuda pada petak yang telah dipilih.

III. Mengulangi langkah pertama sampai kedua untuk petak yang sedang ditempati.

IV. Jika belum ditemukan solusi maka kembali ke langkah sebelumnya (backtracking) dengan menjadikan simpul mati pada simpul sebelumnya.

V. Pencarian dihentikan ketika solusi telah ditemukan atau tidak ada lagi langkah yang memungkinkan semua menjadi simpul mati (dead node).

V. Perbandingan Algoritma Brute Force dengan Algoritma Backtracking

Berikut adalah Algoritma Brute Force yang digunakan untuk menyelesaikan persoalan Knight's Tour yang di implementasikan dalam bahasa java.

```
import java.text.DecimalFormat;

public class BruteForceKT {
    int[][] solution;
    int path = 0;
    static int N = 5;
    static int totalMoves=0;

    public static void main(String[] args) {
        BruteForceKT i = new BruteForceKT(N);
        i.solve();
        System.out.println("Total moves = "+ totalMoves);
    }

    public BruteForceKT(int N) {
        solution = new int[N][N];
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                solution[i][j] = 0;
            }
        }
    }

    public void solve() {
        if (findPath(0, 0, 0, solution.length)) {
            print();
        } else {
            System.out.println("NO PATH FOUND");
        }
    }

    public boolean findPath(int row, int column, int index,
int N) {
        totalMoves++;
        if (solution[row][column] != 0) {
            return false;
        }
        solution[row][column] = path++;
        if (index == N * N - 1) {
            return true;
        }
        if (canMove(row + 2, column + 1, N)
                && findPath(row + 2, column + 1, index + 1,
N)) {
            return true;
        }
        if (canMove(row + 1, column + 2, N)
                && findPath(row + 1, column + 2, index + 1,
N)) {
            return true;
        }
```

```
        if (canMove(row - 1, column + 2, N)
                && findPath(row - 1, column + 2, index + 1,
N)) {
            return true;
        }
        if (canMove(row - 2, column + 1, N)
                && findPath(row - 2, column + 1, index + 1,
N)) {
            return true;
        }
        if (canMove(row - 2, column - 1, N)
                && findPath(row - 2, column - 1, index + 1,
N)) {
            return true;
        }
        if (canMove(row - 1, column - 2, N)
                && findPath(row - 1, column - 2, index + 1,
N)) {
            return true;
        }
        if (canMove(row + 1, column - 2, N)
                && findPath(row + 1, column - 2, index + 1,
N)) {
            return true;
        }
        if (canMove(row + 2, column - 1, N)
                && findPath(row + 2, column - 1, index + 1,
N)) {
            return true;
        }
        solution[row][column] = 0;
        path--;
        return false;

    }

    public boolean canMove(int row, int col, int N) {
        if (row >= 0 && col >= 0 && row < N && col < N) {
            return true;
        }
        return false;
    }

    public void print() {
        DecimalFormat twodigits = new DecimalFormat("00");
        for (int i = 0; i < solution.length; i++) {
            for (int j = 0; j < solution.length; j++) {
                System.out.print("   " +
twodigits.format(solution[i][j]));
            }
            System.out.println();
        }
    }

}
```

Dan berikut adalah Algoritma Backtracking yang digunakan untuk menyelesaikan persoalan Knight's Tour yang di implementasikan dalam bahasa java.

```
import java.text.DecimalFormat;

class BacktrackingKT {

    static final int N = 5;

    static int totalMoves=0;



    public static void main(String[] args) {

        startKnightTour();

        System.out.println("Total moves = "+ totalMoves);

    }


     public static boolean isValid(int i, int j, int
sol[][]) {

      if (i>=1 && i<=N && j>=1 && j<=N) {

        if(sol[i][j]==-1)

          return true;

      }

      return false;

    }


     public static boolean knightTour(int sol[][], int i,
int j, int stepCount, int xMove[], int yMove[]) {

        totalMoves++;

        if (stepCount == N*N)

        return true;


      for(int k=0; k<8; k++) {

        int nextI = i+xMove[k];

        int nextJ = j+yMove[k];


        if(isValid(nextI, nextJ, sol)) {

          sol[nextI][nextJ] = stepCount;

            if (knightTour(sol, nextI, nextJ, stepCount+1,
xMove, yMove))

              return true;

          sol[nextI][nextJ] = -1; // backtracking

        }

      }
```

```
        return false;
      }

      public static boolean startKnightTour() {
        int[][] sol = new int[N+1][N+1];

        for(int i=1; i<=N; i++) {
          for(int j=1; j<=N; j++) {
            sol[i][j] = -1;
          }
        }

        int xMove[] = {2, 1, -1, -2, -2, -1, 1, 2};
        int yMove[] = {1, 2, 2, 1, -1, -2, -2, -1};

        sol[1][1] = 0; // placing knight at cell(1, 1)

        if (knightTour(sol, 1, 1, 1, xMove, yMove)) {
          DecimalFormat twodigits = new DecimalFormat("00");
          for(int i=1; i<=N; i++) {
            for(int j=1; j<=N; j++) {
                              System.out.print("       " +
twodigits.format(sol[i][j]));
            }
            System.out.println();
          }
          return true;
        }
        return false;
      }
    }
```

Dalam makalah ini akan memperlihatkan perbandingan efektivitas dari Algoritma Brute Force dengan Algoritma Backtracking dari banyaknya langkah (operasi) yang dilalui oleh bidak kuda.

A. Perbandingan hasil solusi dari Knight's Tour pada papan catur berukuran 5 x 5

Pada Penyelesaian dengan Algoritma Brute Force didapat solusi :

24 05 14 09 22
13 08 23 04 15
18 01 06 21 10
07 12 19 16 03
00 17 02 11 20

Total moves = 31951

Pada Penyelesaian dengan Algoritma Backtracking didapat solusi :

00 05 14 09 20
13 08 19 04 15
18 01 06 21 10
07 12 23 16 03
24 17 02 11 22

Total moves = 8840

B. Perbandingan hasil solusi dari Knight's Tour pada papan catur berukuran 6 x 6

Pada Penyelesaian dengan Algoritma Brute Force didapat solusi :

00 15 06 25 10 13
33 24 11 14 05 26
16 01 32 07 12 09
31 34 23 20 27 04
22 17 02 29 08 19
35 30 21 18 03 28

Total moves = 986273

Pada Penyelesaian dengan Algoritma Backtracking didapat solusi :

00 15 06 25 10 13
33 24 11 14 05 26
16 01 32 07 12 09
31 34 23 20 27 04
22 17 02 29 08 19
35 30 21 18 03 28

Total moves = 248169

C. Perbandingan hasil solusi dari Knight's Tour pada papan catur berukuran 8 x 8

Pada Penyelesaian dengan Algoritma Brute Force didapat solusi :

00 59 38 33 30 17 08 63

37 34 31 60 09 62 29 16

58 01 36 39 32 27 18 07

35 48 41 26 61 10 15 28

42 57 02 49 40 23 06 19

47 50 45 54 25 20 11 14

56 43 52 03 22 13 24 05

51 46 55 44 53 04 21 12

Total moves = 39965558

Pada Penyelesaian dengan Algoritma Backtracking didapat solusi :

00 59 38 33 30 17 08 63

37 34 31 60 09 62 29 16

58 01 36 39 32 27 18 07

35 48 41 26 61 10 15 28

42 57 02 49 40 23 06 19

47 50 45 54 25 20 11 14

56 43 52 03 22 13 24 05

51 46 55 44 53 04 21 12

Total moves = 8250733

VI. KESIMPULAN

Dari beberapa percobaan diatas terlihat bahwa Algoritma Backtracking lebih efektif dan efisien dari pada Algoritma Brute Force dalam penyelesaian persoalan *Knight's Tour*. Hal ini karena Algoritma Backtracking bekerja lebih cerdas dari pada Algoritma Brute Force yaitu ia memotong simpul simpul yang tidak lagi memungkinkan mengarah menuju solusi tidak seperti Algoritma Brute Force yang mencoba seluruh kemungkinan solusi secara naif.

VIDEO LINK AT YOUTUBE

https://youtu.be/SWwcvCfL_Fo

REFERENCES

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Brute-Force-(2016).pdf diakses pada 4 Mei 2020

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Algoritma-Runut-balik-(2020).pdf diakses pada 4 Mei 2020

[3] https://asalmula-permainancatur.blogspot.co.id diakses pada 4 Mei 2020

[4] https://gaebler.us/share/Knight_tour.html iakses pada 4 Mei 2020

[5] http://www.markkeen.com/knight/ diakses pada 4 Mei 2020

[6] https://sejarahlengkap.com/olahraga/sejarah-catur diakses pada 4 Mei 2020

[7] https://www.transtutors.com/questions/knight-s-tour-brute-force-approaches-in-exercise-6-24-we-developed-a-solution-to-th-2311877.html diakses pada 4 Mei 2020

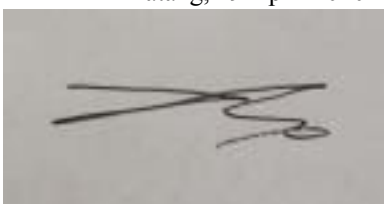
[8] https://runestone.academy/runestone/books/published/pythonds/Graphs/TheKnightsTourProblem.html diakses pada 4 Mei 2020

[9] Mubina, Fathan. *Penerapan Teori Lintasan Hamilton untuk menyelesaikan Persoalan Knight's Tourpada Permainan Catur dengan Backtracking*. 2018. diakses pada 4 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.

Batang, 29 April 2020



Fathan Mubina - 13518064