Penerapan Algoritma *Backtracking* dan *Warnsdorff Rule* pada Permainan *Knight's Tour*

Bintang Fajarianto 13519138
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519138@std.stei.itb.ac.id

***Abstract*—*Knight's Tour* merupakan variasi dari permainan catur yang hanya dimainkan hanya dengan menggunakan satu jenis bidak, yaitu *knight* atau sering kita sebut dengan kuda. Dengan pergerakannya yang menyerupai "*letter* L", permainan ini mengharuskan pemain untuk melewati seluruh kotak dengan menggunakan kuda dalam papan permainan tepat sekali. Jika kuda diharuskan kembali ke titik awal, maka permainan disebut dengan *Closed Knight's Tour*. Jika tidak, maka permainan disebut dengan *Open Knight's Tour*. Dalam makalah ini, akan dibahas penerapan dari algoritma *Backtracking* serta *Warnsdorff Rule* pada permainan *Knight's Tour*.**

***Keywords*—*Knight's tour*, *backtracking*, *warnsdorff rule*.**

I. PENDAHULUAN

Algoritma *backtracking* atau algoritma runut balik merupakan suatu metode pemecahan masalah yang mangkus, sistematis, dan terstruktur. Algoritma ini juga merupakan perbaikan dari algoritma *exhaustive search*. Pada algoritma *exhaustive search*, semua kemungkinan solusi dieksplorasi dan dievaluasi satu per satu. Sementara itu, pada algoritma *backtracking*, hanya pilihan yang mengarah ke solusi yang akan dieksplorasi, pilihan yang tidak mengarah ke solusi tidak akan dipertimbangkan kembali dengan cara dipangkas (*pruning*).

The Knight's Tour merupakan salah satu permainan yang solusinya dapat diselesaikan dengan algoritma *backtracking*. *Knight's Tour* merupakan variasi dari permainan catur yang hanya dimainkan dengan menggunakan satu jenis bidak, yaitu *knight* atau kuda. Peraturan dari permainan ini culup sederhana, pemain hanya perlu memindahkan kuda hingga seluruh petak pada papan permainan telah lewati tepat satu kali.

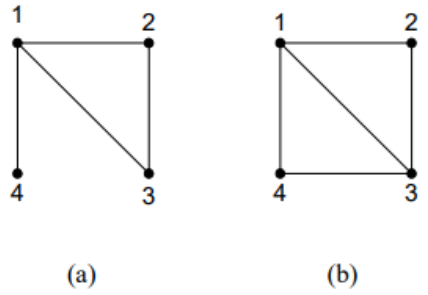
Jika permainan berhasil diselesaikan dengan posisi kuda dapat kembali ke petak asal, maka akan terbentuk Sirkuit Hamilton dan permainan disebut *Closed Knight's Tour*. Jika permainan berhasil diselesaikan dan kuda tidak dapat kembali ke petak asal, maka akan terbentuk Lintasan Hamilton dan permainan disebut *Open Knight's Tour*.

II. LANDASAN TEORI

A. Sirkuit dan Lintasan Hamilton

Sirkuit Hamilton merupakan sirkuit yang melalui tiap simpul di dalam suatu graf tepat satu kali, kecuali simpul asal (sekaligus simpul akhir) yang dilalui dua kali.

Lintasan Hamilton merupakan lintasan yang melalui tiap simpul di dalam graf tepat satu kali.



Gambar 2.1 (a) Lintasan Hamilton (ex: 3,2,1,4), (b) Sirkuit Hamilton (ex: 1,2,3,4,1) [1]

B. *Depth First Search*

Depth First Search atau sering kita sebut dengan DFS merupakan salah satu algoritma pencarian yang dilakukan pada suatu graf. Algoritma DFS mengeksplorasi graf berdasarkan kedalamannya dimulai dari simpul akar (*root*) menuju simpul anaknya dan pencarian terus berulang hingga simpul yang dieksplorasi tidak memiliki anak. Jika hal tersebut terjadi, maka penelusuran akan kembali kepada simpul sebelumnya dan eksplorasi akan dilakukan pada anak simpul lain yang belum dieksplorasi sebelumnya.

Penelusuran pada algoritma DFS dapat diimplementasikan dengan fungsi rekursif dan juga ADT *stack*. Berikut ini merupakan langkah-langkah dari algoritma DFS:

1. Kunjungi simpul v sebagai simpul akar (*root*)
2. Kunjungi simpul w yang merupakan anak dari simpul v.
3. Ulangi DFS mulai dari simpul w (rekursif).
4. Apabila telah mencapai simpul u (simpul daun) yang tidak memiliki simpul anak, maka penelusuran akan

kembali (*backtrack*) ke simpul sebelumnya yang mempunyai simpul anak yang belum dieksplorasi.
5. Penelusuran berakhir apabila tidak ada lagi simpul yang belum dikunjungi dari simpul yang telah dikunjungi.

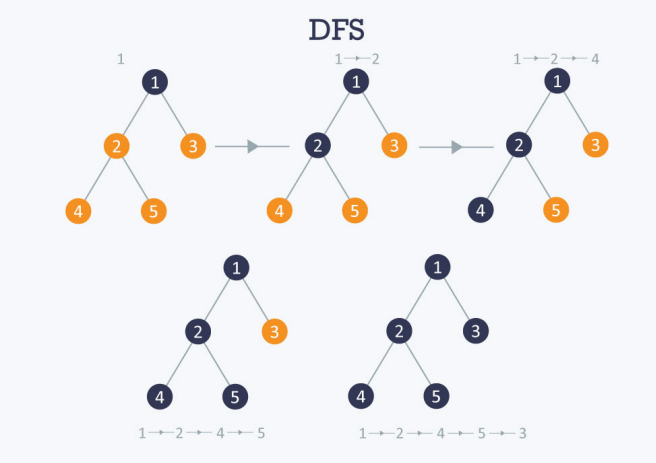
Berikut ini merupakan *pseudocode* dari algoritma DFS serta visualisasi dari penelusuran graf dengan algoritma DFS.

```
procedure DFS(input v:integer)
{Mengunjungi seluruh simpul graf dengan algoritma pencarian DFS

Masukan: v adalah simpul awal kunjungan
Keluaran: semua simpulyang dikunjungi ditulis ke layar
}
Deklarasi
   w : integer

Algoritma:
   write(v)
   dikunjungi[v]←true
   for w←1 to n do
      if A[v,w]=1 then {simpul v dan simpul w bertetangga }
         if not dikunjungi[w] then
            DFS(w)
         endif
      endif
   endfor
```

Gambar 2.2 Pseudocode algoritma DFS [2]



Gambar 2.3 Visualisasi penelusuran graf dengan algoritma DFS [5]

C. *Backtracking*

Backtracking atau algoritma runut-balik merupakan suatu metode pemecahan masalah yang mangkus, sistematis, dan terstruktur. Algoritma ini merupakan pengembangan dari algoritma *Depth First Search* (DFS) dengan memangkas (*pruning*) pilihan atau simpul yang tidak mengarah pada solusi.

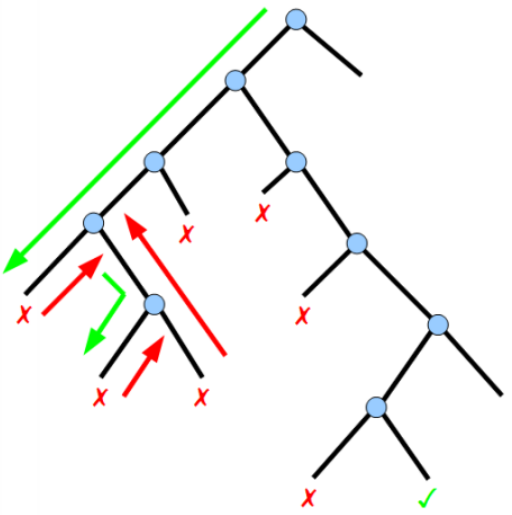
Penelusuran pada algoritma *backtracking* diimplementasikan dengan fungsi rekursif. Berikut ini prinsip-prinsip dari algoritma *backtracking* dalam mencari solusi:

1. Solusi dicari dengan membangkitkan simpul-simpul status sehingga menghasilkan lintasan dari simpul akar (*root*) menuju simpul daun.
2. Aturan pembangkitan simpul yang dipakai mengikuti aturan DFS.
3. Simpul yang sudah dibangkitkan dinamakan simpul hidup (*live node*). Sementara itu, simpul yang sedang diperluas dinamakan simpul-E (*Expand node*).
4. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi, maka simpul-E tersebut dimatikan sehingga menjadi simpul mati (*dead node*).
5. Fungsi yang digunakan untuk mematikan simpul-E menerapkan fungsi pembatas (*bounding function*).
6. Jika pembentukan lintasan berakhir pada simpul mati, maka proses penelusuran akan kembali (*backtrack*) ke simpul sebelumnya.
7. Berikutnya akan dibangkitkan simpul anak lainnya yang belum dibangkitkan sebelumnya.
8. Pencarian dihentikan bila kita telah sampai pada *goal node*.

Berikut ini merupakan *pseudocode* dari algoritma *backtracking* serta visualisasi dari penelusuran dengan algoritma *backtracking*.

```
procedure RunutBalikR(input k : integer)
{Mencari semua solusi persoalan dengan metode runut-balik; skema rekursif
Masukan: k, yaitu indeks komponen vektor solusi, x[k]. Diasumsikan x[1], x[2], ..., x[k – 1] sudah
ditentukan nilainya.
Luaran: semua solusi x = (x[1], x[2], ..., x[n])
}
Algoritma:
  for setiap x[k] ∈ T(x[1], x[2], ..., x[k – 1]) do
     if B(x[1], x[2], ... ,x[k]) = true then
        if (x[1], x[2], ... , x[k]) adalah lintasan dari akar ke simpul solusi then
           write(x[1], x[2], ..., x[k])    { cetak solusi }
        endif
        if k < n then
           RunutBalikR(k+1)    { tentukan nilai untuk x[k+1]}
        endif
     endif
  endfor
                                                    Pemanggilan pertama kali: RunutBalikR(1)
```

Gambar 2.4 Pseudocode algoritma *backtracking* [3]



Gambar 2.5 Visualisasi penelusuran dengan algoritma *backtracking* [3]

D. *Warnsdorff Rule*

Warnsdorff Rule merupakan sebuah metode yang disampaikan oleh H. C. Warnsdorff pada awal abad 19 untuk menyelesaikan permasalahan *Knight's Tour*. *Warnsdorff Rule* bertujuan untuk menghindari solusi jalan buntu yang

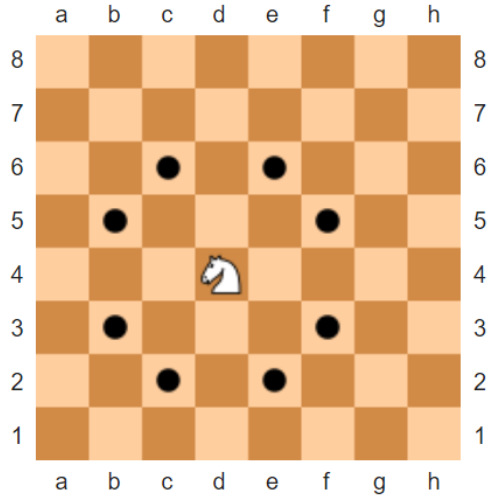
mengakibatkan kuda tidak bisa melangkah lebih jauh tanpa sampai ke petak akhir. Oleh karena itu, pilihan petak yang akan dipilih harus diperiksa dan dipertimbangkan terlebih dahulu sebelum memutuskan petak mana yang akan dipilih. Dalam aturan ini, petak yang akan dipilih merupakan petak yang memiliki pilihan langkah berikutnya dengan jumlah paling sedikit atau dengan kata lain petak yang cenderung terisolasi akan dikunjungi terlebih dahulu.

E. Knights's Tour

Permainan catur memiliki banyak variasi permainan, salah satunya adalah *Knight's Tour* atau dikenal juga dengan sebutan Tur Kuda. Dalam permainan ini, hanya satu pemain yang dapat memainkannya dan hanya satu jenis bidak yang digunakan, yaitu *knight* atau kuda.

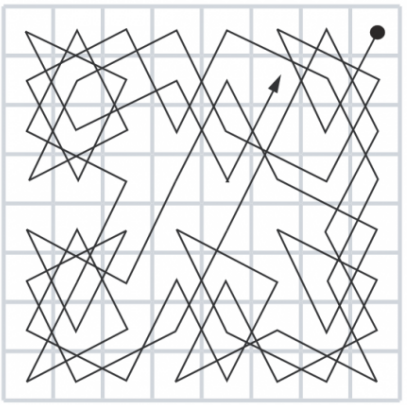
Gambar 2.6 *Knight* atau kuda (www.shutterstock.com)

Dalam permainan catur, kuda memiliki pola pergerakan yang menyerupai "*letter L*". Pada permainan Knight's Tour, pemain diminta untuk melewati seluruh petak pada papan permainan **tepat sekali**, baik pada papan berukuran 8x8 (ukuran *chess board*) maupun papan dengan ukuran variasi.



Gambar 2.7 Pola pergerakan *Knight* atau kuda (www.wikipedia.com)

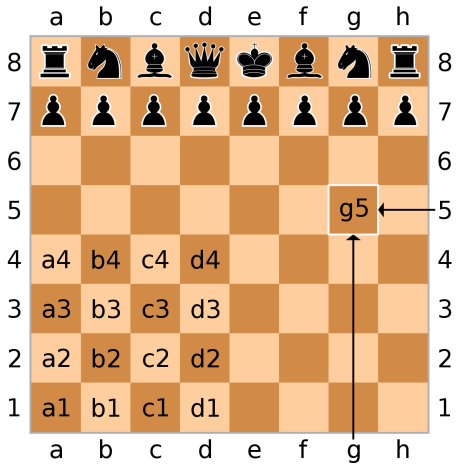
Pada permainan ini, terdapat dua kondisi ketika berhasil menyelesaikan perjalanan kuda pada papan permainan. Pertama, jika permainan berhasil diselesaikan dengan posisi kuda dapat kembali ke petak asal, maka permainan disebut *Closed Knight's Tour*. Kedua, jika permainan berhasil diselesaikan dan kuda tidak dapat kembali ke petak asal, maka permainan disebut *Open Knight's Tour*.



Gambar 2.8 *Closed Knight's Tour* (www.lastwordonnothing.com)

III. IMPLEMENTASI

Dalam mencari solusi dari permainan *Knight's Tour* ini, penulis mengimplementasikan program menggunakan bahasa python karena cukup mudah dalam merepresentasikan matriks papan catur dan pengolahannya. Berikut ini merupakan repository dari source code program yang penulis buat: https://github.com/frnzoshi/Knights-Tour-Problem



Gambar 4.1 *Chess board index* (www.wikipedia.com)

Dalam program yang dibuat, pertama-tama akan diminta inputan index posisi awal kuda (ex: A8) serta tipe permainan (close/open)

A. Open Knight's Tour

Misalkan, pada permainan *Knight's Tour* ini, posisi pertama kuda berada pada B8. Maka, petak yang dapat menjadi langkah berikutnya (valid) adalah [D7, C6, A6]. Berdasarkan *Warnsdorff Rule*, akan dihitung banyak petak valid yang dapat dijadikan langkah berikutnya dari masing-masing kemungkinan petak yang kita dapat sebelumnya.

- D7

  Petak valid yang dimiliki D7 untuk dijadikan langkah berikutnya: [F8, F6, E5, C5, B6] berjumlah 5 petak.

- C6

  Petak valid yang dimiliki C6 untuk dijadikan langkah berikutnya: [D8, E7, E5, D4, B4, A5, A7] berjumlah 7 petak.

- A6

  Petak valid yang dimiliki A6 untuk dijadikan langkah berikutnya: [C7, C5, B4] berjumlah 4 petak.

Oleh karena itu, karena A6 memiliki jumlah petak valid terkecil untuk dijadikan langkah berikutnya, maka kuda akan berpindah dari B8 menuju A6. Proses itu berulang hingga seluruh petak berhasil dilewati kuda tepat satu kali. Pada tipe permainan ini (*Open Knight's Tour*), *backtracking* tidak akan dilakukan karena algoritma telah disempurnakan dengan *Warnsdorff Rule*.

B. Closed Knight's Tour

Misalkan, pada permainan *Knight's Tour* ini, posisi pertama kuda berada pada B8. Maka, proses yang dilakukan pada *Open Knight's Tour* akan dilakukan terlebih dahulu karena tidak ada proses *backtracking* pada tahap ini. Apabila solusi tidak bersifat *Closed Knight's Tour* (kuda dapat kembali ke petak asal), maka akan dilakukan backtracking sehingga batasan bahwa petak yang dapat dijadikan langkah berikutnya dari petak awal seminimalnya ada satu petak yang tidak dilewati untuk dijadikan petak akhir.

IV. UJI COBA

Pada bagian ini, penulis akan tampilkan hasil pengetesan program untuk beberapa kasus yang akan dituliskan di bawah ini. Akan diuji 6 kasus dengan rincian sebagai berikut:

1. Lokasi awal: A8 - Open
2. Lokasi awal: A5 - Closed
3. Lokasi awal: C2 - Open
4. Lokasi awal: C2 - Closed
5. Lokasi awal: D4 - Open
6. Lokasi awal: D4 - Closed

A. *Case* 1

Pada kasus ini, akan dicari solusi *Knight's Tour* dari posisi awal A8 dengan tipe *Open Knight's Tour*.

```
Papan Catur

    A   B   C   D   E   F   G   H
8  A8  B8  C8  D8  E8  F8  G8  H8
7  A7  B7  C7  D7  E7  F7  G7  H7
6  A6  B6  C6  D6  E6  F6  G6  H6
5  A5  B5  C5  D5  E5  F5  G5  H5
4  A4  B4  C4  D4  E4  F4  G4  H4
3  A3  B3  C3  D3  E3  F3  G3  H3
2  A2  B2  C2  D2  E2  F2  G2  H2
1  A1  B1  C1  D1  E1  F1  G1  H1

Masukkan posisi awal: A8
Masukkan tipe (close/open): open
```

Gambar 5.1 Inputan *case* 1

```
Hasil Akhir

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

Solusi ditemukan,
waktu yang diperlukan 0.024601200000000212 detik
```

Gambar 5.2 Solusi *case* 1

B. *Case* 2

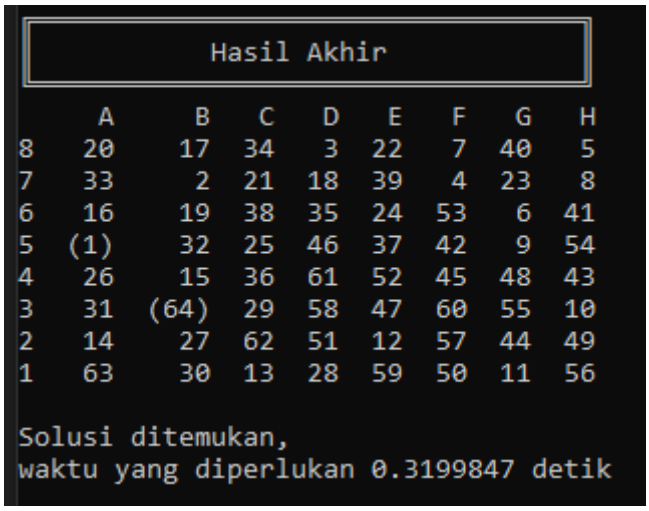
Pada kasus ini, akan dicari solusi *Knight's Tour* dari posisi awal A5 dengan tipe *Closed Knight's Tour*.

```
Papan Catur

    A   B   C   D   E   F   G   H
8  A8  B8  C8  D8  E8  F8  G8  H8
7  A7  B7  C7  D7  E7  F7  G7  H7
6  A6  B6  C6  D6  E6  F6  G6  H6
5  A5  B5  C5  D5  E5  F5  G5  H5
4  A4  B4  C4  D4  E4  F4  G4  H4
3  A3  B3  C3  D3  E3  F3  G3  H3
2  A2  B2  C2  D2  E2  F2  G2  H2
1  A1  B1  C1  D1  E1  F1  G1  H1

Masukkan posisi awal: A5
Masukkan tipe (close/open): close
```

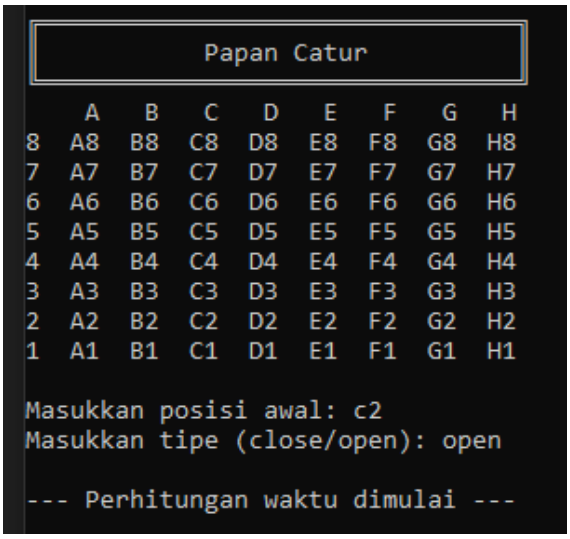
Gambar 5.3 Inputan *case* 2



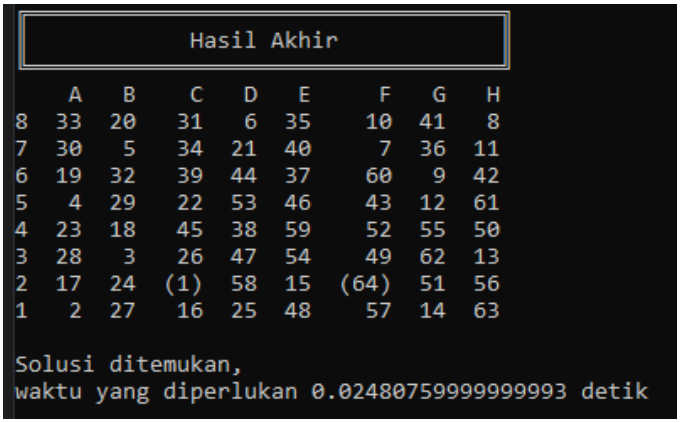
Gambar 5.4 Solusi *case* 2

C. *Case* 3

Pada kasus ini, akan dicari solusi *Knight's Tour* dari posisi awal C2 dengan tipe *Open Knight's Tour*.



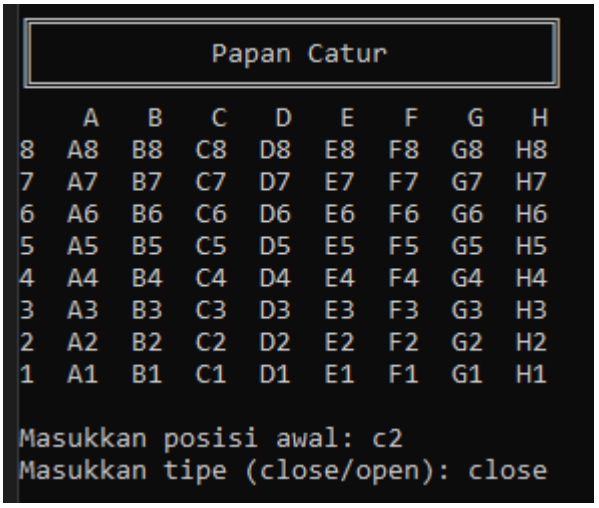
Gambar 5.5 Inputan *case* 3



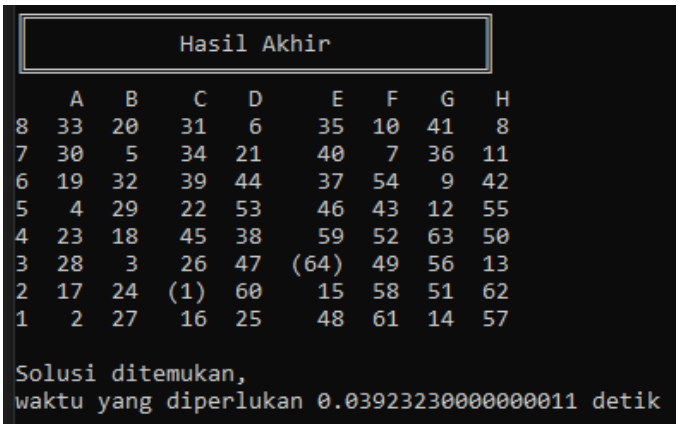
Gambar 5.6 Solusi *case* 3

D. *Case* 4

Pada kasus ini, akan dicari solusi *Knight's Tour* dari posisi awal C2 dengan tipe *Closed Knight's Tour*.



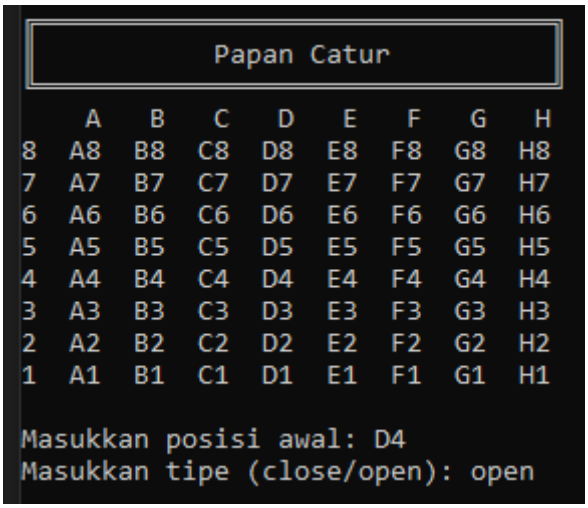
Gambar 5.7 Inputan *case* 4



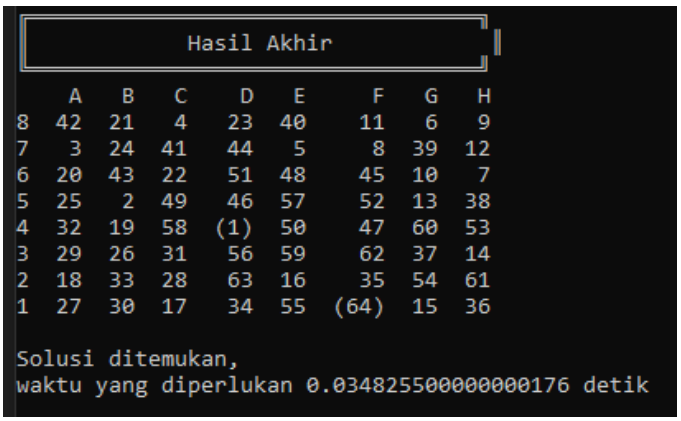
Gambar 5.8 Solusi *case* 4

E. *Case* 5

Pada kasus ini, akan dicari solusi *Knight's Tour* dari posisi awal D4 dengan tipe *Open Knight's Tour*.



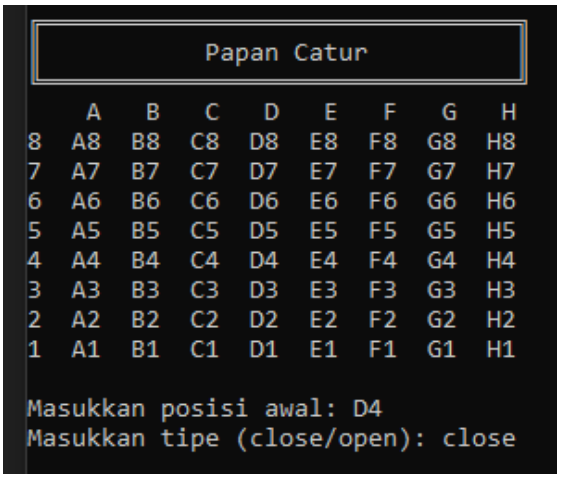
Gambar 5.9 Inputan *case* 5



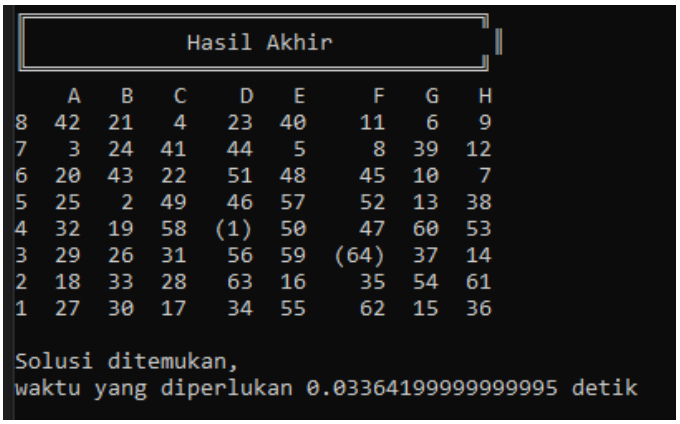
Gambar 5.10 Solusi *case* 5

F. *Case* 6

Pada kasus ini, akan dicari solusi *Knight's Tour* dari posisi awal D4 dengan tipe *Closed Knight's Tour*.



Gambar 5.11 Inputan *case* 6



Gambar 5.12 Solusi *case* 6

V. KESIMPULAN

Dari berbagai kasus uji coba yang diujikan pada Bab IV, terlihat bahwa tipe permainan *Closed Knight's Tour* cenderung memakan waktu lebih lama karena memiliki *worst case* yang lebih buruk (harus *backtracking*) dibandingkan tipe permainan *Open Knight's Tour*. Hal ini terjadi karena pada tipe permainan *Open Knight's Tour* telah disempurnakan dengan *Warnsdorffs Rule*.

UCAPAN TERIMA KASIH

Segala puji serta syukur penulis panjatkan kepada Allah SWT karena atas limpahan nikmat serta ridho-Nya, penulis dapat menyelesaikan tugas makalah ini dengan baik dan tepat waktu. Selain itu, saya ucapkan terima kasih banyak kepada para dosen, khususnya dosen mata kuliah IF2211 Strategi Algoritma yang telah memberikan pengajaran sehingga penulis mampu memahami beragam ilmu yang diterapkan dalam pembuatan makalah ini.

REFERENSI

[1] Munir, Rinaldi, 2020, "Matematika Diskrit: Graf bagian 3", Bandung: Informatika ITB.

[2] Munir, Rinaldi, 2021, "Strategi Algoritma: BFS/DFS bagian 1", Bandung: Informatika ITB.

[3] Munir, Rinaldi, 2021, "Strategi Algoritma: Algoritma Backtracking bagian 1", Bandung: Informatika ITB.

[4] https://www.futilitycloset.com/2014/11/10/warnsdorffs-rule/
diakses pada tanggal 11 Mei 2021

[5] https://www.hackerearth.com/practice/algorithms/graphs/depth-first-search/tutorial/
diakses pada tanggal 11 Mei 2021

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.

Depok, 11 Mei 2021

Bintang Fajarianto - 13519138