Aplikasi Algoritma *Backtracking* untuk Menentukan Rute sebagai Jawaban atas Persoalan *The Knight's Tour* pada Permainan Catur

Diky Restu Maulana - 13520017
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520017@std.stei.itb.ac.id

***Abstract*— Catur adalah permainan papan strategi dua orang yang dimainkan pada sebuah papan persegi berisi 64 kotak berwarna hitam dan putih. Kuda adalah salah satu buah catur yang memiliki keistimewaan, yaitu dapat melompati buah catur lain sehingga dapat berpindah dengan cepat dari satu petak ke petak lain. Jika ditempatkan pada petak ujung kiri atas, apakah kuda bisa melewati seluruh petak di atas papan berukuran nxn tepat sekali? Jika ya, bagaimana urutan kotak-kotak yang dilaluinya? Berapa ukuran minimal papan sehingga kuda bisa melakukan *tour* tersebut? Makalah ini akan menemukan jawabannya dengan memanfaatkan algoritma *backtracking*.**

Kata Kunci—catur, kuda, graf, backtracking.

I. PENDAHULUAN

Catur adalah permainan papan strategi dua orang yang dimainkan pada sebuah papan persegi berisi 64 buah kotak berwarna hitam putih. Catur diyakini berasal dari permainan India, *chaturanga* (yang menjadi asal nama catur), sekitar abad ke-7. *Chaturanga* juga diperkirakan merupakan nenek moyang dari permainan strategi serupa yang berasal dari Dunia Timur, seperti *xiangqi* (catur Cina), *janggi* (catur Korea), dan *shogi* (catur Jepang). Catur mencapai Eropa pada abad ke-9, saat terjadi penaklukan Hispania oleh Umayyah. Buah-buah catur tersebut diperkirakan mendapat bentuknya yang dikenal saat ini pada akhir abad ke-15 di Spanyol, sedangkan aturan catur modern distandardisasi pada abad ke-19.

Dahulu, catur merupakan pertandingan bergengsi yang hanya dimainkan oleh para bangsawan. Catur juga sering digunakan untuk melatih strategi perang para prajurit. Namun, kini catur sudah bisa dinikmati oleh siapa saja. Catur digemari oleh banyak orang di Indonesia, bahkan dunia. Meskipun dimainkan dalam keadaan duduk di kursi dan tidak berkeringat, catur dikategorikan sebagai salah satu cabang olahraga. Hal ini karena pemain catur perlu menggunakan otaknya untuk memikirkan seluruh kemungkinan langkah yang akan terjadi. Tentu perlu energi dan stamina yang besar jika ingin memenangkan sebuah pertandingan catur.

Saat ini, federasi catur seluruh dunia dinaungi oleh FIDE (*Fédération Internationale des Échecs*). FIDE menerbitkan peraturan resmi dalam bermain catur. Berbagai turnamen catur bergengsi pun bermunculan, seperti *Chess Olympiad*, *The Candidate*, dan *World Chess Championship*. Teori catur terus berkembang seiring berjalannya waktu. Mulai dari teori *opening* hingga teori *endgame*. Setiap pertandingan yang dimainkan selalu menghadirkan posisi baru sehingga bermunculan berbagai *problem* catur.

Hingga kini, catur masih menyimpan banyak misteri. Berbagai pertanyaan masih menarik untuk ditemukan jawabannya. Misalnya, ada berapa banyak kemungkinan langkah catur hingga langkah ke-n? Berapa jumlah langkah terbanyak yang mungkin tanpa terjadi saling memukul bidak di antara kedua pemain? Siapa yang akan memenangkan permainan antara 32 buah menteri dan 32 buah kuda? Masih banyak lagi yang lainnya.

Oleh karena itu, penulis tertarik untuk menjawab sebuah *problem* catur yang bernama *The Knight's Tour*, yaitu bisakah kuda menempati seluruh petak di atas papan berukuran nxn tepat sekali? Jika bisa, bagaimana urutan petak yang dilewatinya? Berapa ukuran minimal papan sehingga kuda bisa melakukan *tour* tersebut? Analisis untuk menjawab pertanyaan-pertanyaan itu dilakukan dengan memanfaatkan algoritma Runut Balik (*Backtracking)*.

Gambar 1. Potret Orang yang Sedang Bermain Catur

(Sumber: https://www.cnnindonesia.com/teknologi/20210322161953-192-620639/cuitan-kocak-netizen-usai-dewa-kipas-kalah-telak-dari-irene/)

II. LANDASAN TEORI

A. *Algoritma Backtracking*

Backtracking atau runut balik bisa dilihat sebagai sebuah fase dalam algoritma traversal depth first search (DFS) ataupun sebagai sebuah metode pemecahan masalah yang efektif, terstruktur, dan sistematis untuk sebuah persoalan.

Backtracking merupakan algoritma perbaikan dari *exhaustive search*. Dalam hal ini, *exhaustive search* mencari dan mengeksplorasi semua kemungkinan solusi dan mengevaluasinya satu per satu, sedangkan pada algoritma runut balik, pilihan yang dieksplorasi dan dievaluasi hanya pilihan yang mengarah ke solusi dan pilihan lainnya tidak dipertimbangkan lagi. Hal tersebut dilakukan dengan cara memangkas simpul-simpul yang tidak mengarah ke solusi.

Algoritma *backtracking* pertama kali diperkenalkan pada tahun 1950 oleh D. H. Lehmer. Kemudian, uraian umum tentang algoritma tersebut disajikan oleh R. J. Walker, Golomb, dan Baumert.

Algoritma *backtracking* memiliki tiga properti umum, yaitu:

1. Solusi persoalan

   Solusi dinyatakan sebagai sebuah vektor dengan *n-tuple*: $X = (x_1, x_2, ..., x_n)$, $x_i \in S_i$. Pada umumnya, $S_1 = S_2 = ... = S_n$.

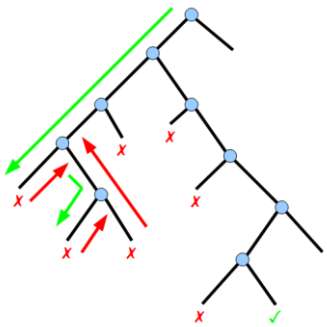
2. Fungsi pembangkit nilai x_k

   Dinyatakan sebagai predikat T(). Dalam hal ini, T(x[1], x[2], ..., x[k-1]) membangkitkan nilai untuk x_k yang merupakan sebuah komponen pada vektor solusi.

3. Fungsi pembatas (*bounding function*)

   Dinyatakan sebagai sebuah predikat $B(x_1, x_2, ..., x_k)$. B bernilai *true* jika $x_1, x_2, ..., x_k$ mengarah ke solusi atau tidak melanggar kendala (constraints). Jika *bounding function* bernilai *true*, pembangkitan untuk x_{k+1} dilanjutkan. Jika *false*, $(x_1, x_2, ..., x_k)$ dibuang.

Semua kemungkinan dari sebuah solusi persoalan merupakan ruang solusi (*solution space*) yang diorganisasikan ke dalam struktur pohon berakar. Setiap simpul pohon menyatakan status (*state*) persoalan, sedangkan sisi (cabang) dilabeli dengan nilai-nilai x_i. Lintasan dari akar ke daun menyatakan sebuah solusi yang mungkin. Seluruh lintasan dari akar ke daun membentuk ruang solusi. Pengorganisasian pohon ruang solusi dinamakan pohon ruang status (*state space tree*).

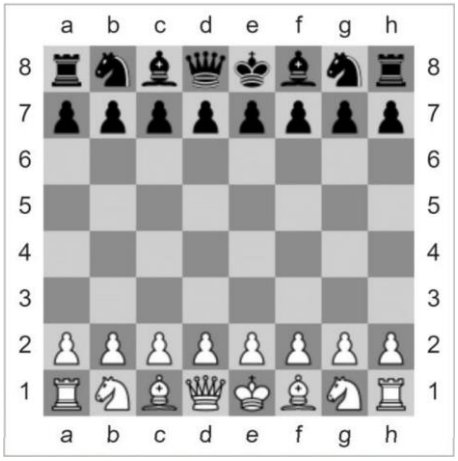


Gambar 2. Ilustrasi Pencarian Solusi dengan Algoritma *Backtracking*
(Sumber: https://www.w3.org/2011/Talks/01-14-steven-phenotype/)

Dalam algoritma *Backtracking*, solusi dicari dengan membangkitkan simpul-simpul status sehingga menghasilkan lintasan dari akar ke daun. Aturan pembangkitan simpul yang dipakai adalah mengikuti aturan DFS (*Depth First Search*). Simpul-simpul yang sudah dibangkitkan dinamakan simpul hidup (*live node*). Simpul hidup yang sedang diperluas dinamakan simpul-E (*expand node*). Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang. Jika lintasan yang sedang dibentuk tidak mengarah kepada solusi, maka simpul-E tersebut "dimatikan" sehingga menjadi simpul mati (*dead node*). Simpul-E dimatikan dengan cara menerapkan fungsi pembatas (*bounding function*). Ketika sebuah simpul dimatikan, secara implisit telah dilakukan pemangkasan (*pruning*) terhadap simpul anak-anaknya. Jika pembentukan lintasan berakhir di simpul mati, proses pencarian *backtrack* ke simpul pada tingkat di atasnya. Lalu, teruskan dengan membangkitkan simpul anak yang lainnya. Selanjutnya, simpul ini menjadi simpul-E yang baru. Pencarian dihentikan bila telah sampai kepada *goal node*.

B. *Catur*

Catur adalah permainan oleh dua orang, dilengkapi dengan buah catur sebanyak 16 buah berwarna hitam dan 16 buah lagi berwarna putih, masing-masing terdiri atas 8 bidak (pion), 2 benteng, 2 gajah, 2 kuda, 1 menteri, dan 1 raja. Catur dimainkan di atas papan persegi berukuran 8×8. Kolom ditandai dengan huruf dan baris ditandai dengan angka. Setiap petak dapat disebut sebagai koordinat baris dan kolomnya, misalnya petak A7.



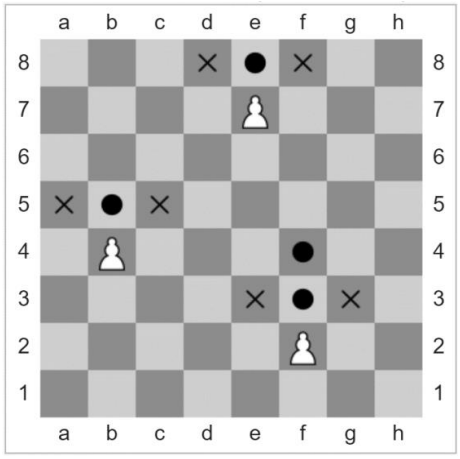
Gambar 3. Susunan Awal Bidak di Atas Papan Catur
(Sumber: https://id.wikipedia.org/wiki/Catur/)

Ada enam jenis bidak catur, yaitu:

1. Pion (*pawn*)

   Pion adalah buah paling lemah dalam catur. Pion hanya bisa berjalan selangkah ke depan, yang berarti ke arah barisan lawan, dan tidak menyerang buah catur lawan dalam arah ini. Dalam langkah pertama, pion dapat maju 2 kotak dan tidak ada yang menghambat

jalan ini. Untuk memakan, bidak harus mengambil arah diagonal sekali.

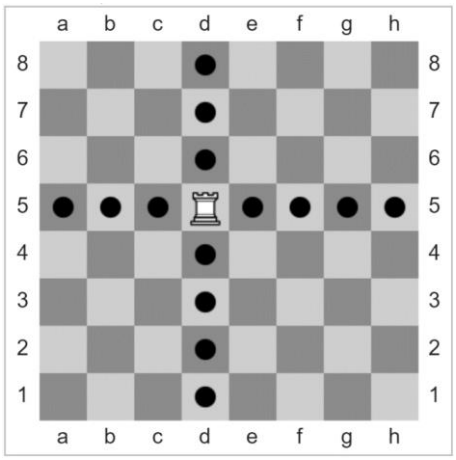


Gambar 4. Gerakan Pion

(Sumber: https://id.wikipedia.org/wiki/Catur/)

2. Benteng (*rook*)

Benteng adalah buah catur yang memiliki gerak lurus, baik ketika bergerak maupun ketika memukul buah catur lawan. Benteng memiliki gerakan istimewa, yaitu *rokade*. Setiap pemain catur memiliki dua benteng di setiap sudut permainan ketika memulai bermain. Pada awal permainan benteng tidak dapat bergerak karena terhalangi buah catur lainnya. Benteng baru dapat bergerak ketika medan permainan sudah terbuka. Benteng bisa melangkah lurus sepanjang baris dan lajur di papan kecuali bila ada buah catur lain yang menghalanginya. Benteng tidak dapat melompati buah catur lainnya kecuali saat melakukan *rokade*.

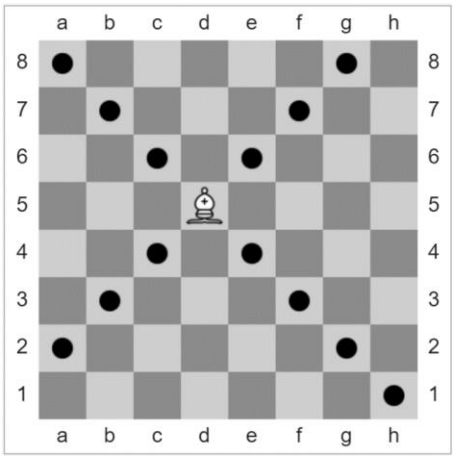


Gambar 5. Gerakan Benteng

(Sumber: https://id.wikipedia.org/wiki/Catur/)

3. Gajah (*bishop*)

Gajah adalah buah catur yang dapat bergerak sepanjang diagonal, tetapi tidak dapat melompati buah catur lain. Pada awal permainan, setiap pemain memiliki sepasang gajah yang terletak di warna kotak berbeda, yaitu gajah petak terang dan gelap. Gajah petak gelap akan selalu berada di petak terang, begitu pula gajah petak terang.

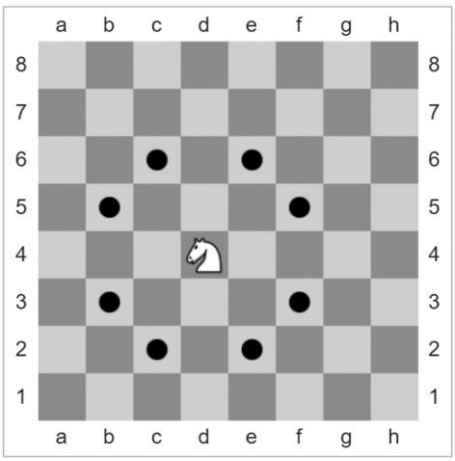


Gambar 6. Gerakan Gajah

(Sumber: https://id.wikipedia.org/wiki/Catur/)

4. Kuda (*knight*)

Kuda dapat bergerak ke segala arah sepanjang gerakannya seperti huruf L, yaitu memanjang dua petak dan melebar satu petak, atau memanjang satu petak dan melebar dua petak. Hanya kuda satu-satunya buah catur yang dapat melompati buah catur lain.

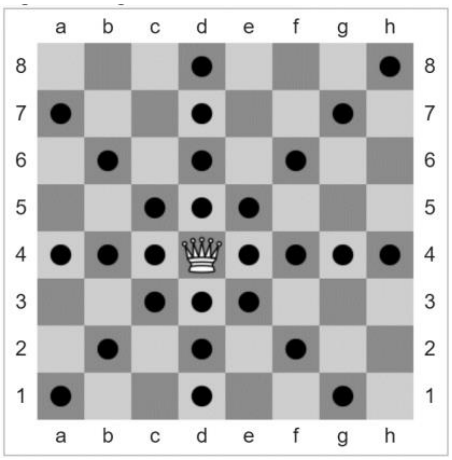


Gambar 7. Gerakan Kuda

(Sumber: https://id.wikipedia.org/wiki/Catur/)

5. Menteri (*queen*)

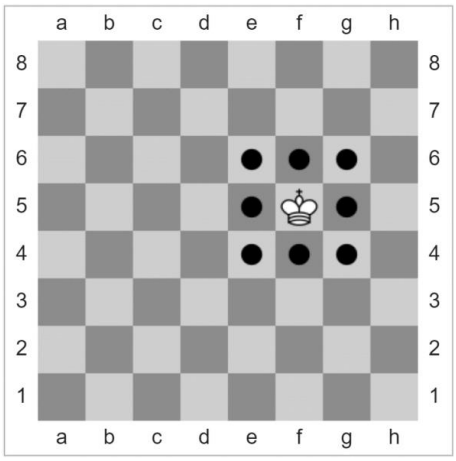
Menteri adalah buah catur yang paling kuat. Menteri memiliki gerakan kombinasi dari benteng dan gajah, sehingga dapat bergerak sepanjang petak ke segala arah, baik horizontal, vertikal, maupun diagonal, tetapi tidak dapat melompati buah catur lain.



Gambar 8. Gerakan Menteri
(Sumber: https://id.wikipedia.org/wiki/Catur/)

6. Raja (*king*)

   Raja adalah buah catur yang paling berharga. Permainan akan berakhir jika raja dalam posisi diserang dan tidak ada jalan untuk membebaskannya (*checkmate*). Raja dapat bergerak sejauh satu petak, baik horizontal, vertikal, maupun diagonal.



Gambar 9. Gerakan Raja
(Sumber: https://id.wikipedia.org/wiki/Catur/)

III. PEMBAHASAN

Penyelesaian masalah dengan menggunakan algoritma *backtracking* membutuhkan pendefinisian simpul dan sisi dari pohon ruang status (*state space tree*) yang akan dibentuk. Selain itu, property-properti dari algoritma *backtracking* juga perlu didefinisikan. Properti tersebut terdiri atas solusi persoalan, fungsi pembangkit, dan fungsi pembatas (*bounding function*).

A. Penentuan Simpul dan Sisi pada State Space Tree

Berdasarkan implementasi yang dibuat, suatu simpul atau status (*state*) dapat didefinisikan sebagai posisi kuda saat ini serta nilai yang dimiliki oleh seluruh petak lain. Setiap petak di atas papan diisi dengan nilai awal -1. Artinya petak tersebut belum pernah dikunjungi sama sekali oleh kuda. Nilai di atas petak yang sudah dikunjungi berkisar antara 0 sampai n^2-1 dengan n adalah ukuran sisi papan. Nilai tersebut menunjukkan urutan dikunjunginya suatu petak. Dalam persoalan *The Knight's Tour*, kuda tidak boleh melewati sebuah petak lebih dari satu kali. Akibatnya, nilai yang dimiliki oleh sebuah petak dijamin unik. Posisi terkini kuda dan nilai setiap petak dibuat dalam sebuah matriks (*board*) berukuran nxn dengan n adalah ukuran sisi papan. Setiap elemen pada matriks tersebut merupakan representasi dari sebuah petak pada papan catur.

Sebuah sisi dalam *state space tree* dapat didefinisikan sebagai gerakan yang mungkin dilakukan oleh kuda dari petak yang sedang ditempatinya saat ini. Dapat dilihat pada gambar 7, ada delapan kemungkinan langkah kuda. Namun, jika petak yang akan ditempati berada di luar papan, langkah tersebut akan diabaikan dan tidak dibangkitkan simpul untuk langkah tersebut.

B. Penentuan Solusi Persoalan

Persoalan *The Knight's Tour* mengharuskan kuda mengunjungi seluruh petak di atas papan tepat sekali. Artinya, setiap petak akan memiliki nilai yang unik dengan kemungkinan nilai berkisar di antara 0 sampai n^2-1 dengan n adalah ukuran sisi papan. Seluruh kemungkinan gerakan akan dicoba dan dibangkitkan sebuah node untuk gerakan tersebut hingga mencapai *goal node*.

Goal node pada implementasi ini adalah ketika seluruh petak telah dikunjungi. Dengan kata lain, tidak ada petak yang bernilai -1. Jadi, solusi persoalan pada implementasi ini adalah semua lintasan dari simpul akar ke *goal node*.

C. Penentuan Fungsi Pembangkit

Berdasarkan implementasi yang dibuat, fungsi pembangkit sangat bergantung kepada fungsi pembatas. Suatu simpul-E hanya akan dibangkitkan jika fungsi pembatas memperbolehkannya. Fungsi ini membangkitkan sebuah simpul yang merepresentasikan posisi kuda setelah melakukan gerakan yang valid.

Posisi kuda dimulai dari ujung kiri atas, maka gerakan yang diperiksa terlebih dahulu adalah gerakan yang menyebabkan kuda bergeser ke arah kanan atau bawah. Urutan prioritas pengecekan gerakan adalah sebagai berikut:

1. Dua petak ke kanan, satu petak ke bawah (2, 1)
2. Satu petak ke kanan, dua petak ke bawah (1, 2)
3. Satu petak ke kiri, dua petak ke bawah (-1, 2)
4. Dua petak ke kiri, satu petak ke bawah (-2, 1)
5. Dua petak ke kiri, satu petak ke atas (-2, -1)
6. Satu petak ke kiri, dua petak ke atas (-1, -2)
7. Satu petak ke kanan, dua petak ke atas (1, -2)
8. Dua petak ke kanan, satu petak ke atas (2, -1)

D. Penentuan Fungsi Pembatas

Berdasarkan implementasi yang dibuat, fungsi pembatas didefinisikan sebagai fungsi yang mengembalikan *true* jika sebuah gerakan diperbolehkan. Sebaliknya, fungsi ini akan mengembalikan *false*. Fungsi pembatas berfungsi untuk melakukan pengecekan terhadap sebuah gerakan yang mungkin dilakukan oleh kuda. Sebuah gerakan diperbolehkan jika petak selanjutnya belum pernah dikunjungi. Dengan kata lain, petak tersebut bernilai -1 pada matriks *board*. Sebuah gerakan juga akan ditolak oleh fungsi pembatas jika gerakan tersebut menyebabkan kuda keluar dari papan.

E. Langkah Penyelesaian

Setelah menentukan seluruh properti dari algoritma *backtracking*, langkah pencarian solusi bisa ditentukan. Langkah-langkah tersebut adalah sebagai berikut:

1. Posisi awal kuda dimulai dari petak ujung kiri atas papan, yaitu pada baris 0 dan kolom 0. Beri nilai 0 untuk petak ini sebagai tanda bahwa petak inilah yang pertama kali dilalui oleh kuda. Simpan urutan gerakan kuda untuk dilakukan *increment* pada pembangkitan simpul berikutnya.
2. Lakukan pengecekan dengan fungsi pembatas terhadap gerakan yang memungkinkan dari posisi kuda saat ini.
3. Jika gerakan valid, bangkitkan simpul untuk gerakan tersebut. Beri nilai urutan gerakan kuda sebelumnya ditambah 1 untuk petak saat ini.
4. Jika tidak valid, abaikan gerakan tersebut.
5. Jika tidak ada gerakan yang valid dari posisi terkini, lakukan *backtracking* ke petak sebelumnya dan ambil gerakan lain yang valid.
6. Untuk setiap simpul yang dibangkitkan, ulangi langkah kedua hingga kelima.
7. *Goal node* dicapai ketika nilai petak yang diberikan adalah n^2-1 dengan n adalah ukuran sisi papan. Artinya kuda telah berhasil melalui seluruh petak tepat sekali. Hentikan pencarian dan cetak matriks pada *goal node*.

F. Hasil Pengujian Terhadap Ukuran Papan yang Berbeda

Penulis telah membuat sebuah program yang menerima *input* berupa ukuran sisi papan dan memberikan *output* berupa matriks berukuran nxn berisi angka unik untuk setiap elemennya. Angka tersebut merupakan urutan suatu petak dilalui kuda. Angka yang lebih kecil menandakan petak tersebut dikunjungi terlebih dahulu.

Pengujian dilakukan dalam ukuran papan yang bervariasi antara 3 hingga 8 (1 dan 2 tidak mungkin karena gerakan kuda membutuhkan minimal tiga petak) untuk menentukan ukuran papan minimal sehingga kuda bisa melakukan perjalanan melewati seluruh petak di atas papan tepat sekali.

1. n = 3

Tabel 1. Hasil Pengujian untuk n = 3

Input	Output
`Enter the size of the board: 3`	`Solution does not exist` `Time taken: 0.000000 seconds`

2. n = 4

Tabel 2. Hasil Pengujian untuk n = 4

Input	Output
`Enter the size of the board: 4`	`Solution does not exist` `Time taken: 0.011001 seconds`

3. n = 5

Tabel 3. Hasil Pengujian untuk n = 5

Input	Output

`Enter the size of the board: 5`	` 0  5 14  9 20`
	`13  8 19  4 15`
	`18  1  6 21 10`
	` 7 12 23 16  3`
	`24 17  2 11 22`
	`Time taken: 0.070004 seconds`

4. n = 6

Tabel 4. Hasil Pengujian untuk n = 6

Input	Output

`Enter the size of the board: 6`	` 0 15  6 25 10 13`
	`33 24 11 14  5 26`
	`16  1 32  7 12  9`
	`31 34 23 20 27  4`
	`22 17  2 29  8 19`
	`35 30 21 18  3 28`
	`Time taken: 1.654061 seconds`

5. n = 7

Tabel 5. Hasil Pengujian untuk n = 7

Input	Output

`Enter the size of the board: 7`	` 0 37 30  7 18 35 14`
	`31 28 19 36 15  6 17`
	`38  1 32 29  8 13 34`
	`27 24 39 20 33 16  5`
	`40 21  2 25 44  9 12`
	`23 26 47 42 11  4 45`
	`48 41 22  3 46 43 10`
	`Time taken: 50.220800 seconds`

6. n = 8

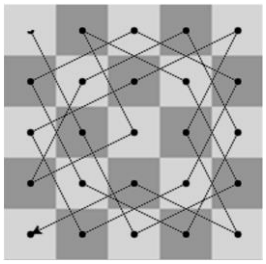
Tabel 6. Hasil Pengujian untuk n = 8

Input	Output

`Enter the size of the board: 8`	` 0 59 38 33 30 17  8 63`
	`37 34 31 60  9 62 29 16`
	`58  1 36 39 32 27 18  7`
	`35 48 41 26 61 10 15 28`
	`42 57  2 49 40 23  6 19`
	`47 50 45 54 25 20 11 14`
	`56 43 52  3 22 13 24  5`
	`51 46 55 44 53  4 21 12`
	`Time taken: 66.418714 seconds`

Supaya lebih jelas, di bawah ini terdapat rute yang digambarkan sebagai graf berarah di atas sebuah papan catur untuk seluruh ukuran papan yang telah diuji.

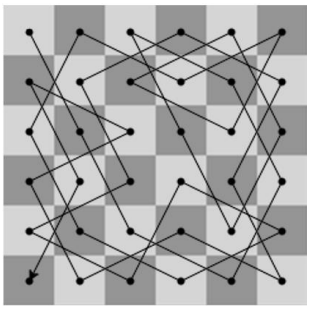
1. n = 5



Gambar 10. Rute Perjalanan Kuda untuk n = 5

(Sumber: dokumen penulis)

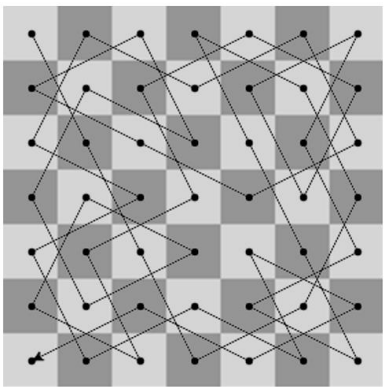
2. n = 6



Gambar 11. Rute Perjalanan Kuda untuk n = 6

(Sumber: dokumen penulis)

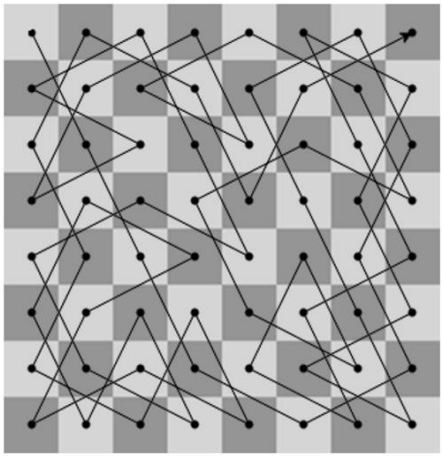
3. n = 7



Gambar 12. Rute Perjalanan Kuda untuk n = 7

(Sumber: dokumen penulis)

4. n = 8



Gambar 13. Rute Perjalanan Kuda untuk n = 5

(Sumber: dokumen penulis)

IV. KESIMPULAN

Algoritma *backtracking* dapat digunakan untuk menyelesaikan persoalan *The Knight's Tour* pada permainan catur. Pengujian yang telah dilakukan membuktikan bahwa perjalanan kuda mengunjungi seluruh petak di atas papan berukuran nxn tepat sekali dari posisi awal petak ujung kiri atas sangat mungkin terjadi. Bahkan, semakin besar ukuran papan, variasi rute pun akan semakin beragam. Dari pengujian ini juga didapatkan sebuah temuan bahwa ukuran papan minimum untuk menyelesaikan persoalan *The Knight's Tour* adalah sebesar 5x5.

Penulis berharap makalah ini dapat membantu siapapun yang sedang belajar catur dan dapat memuaskan rasa penasaran terhadap *problem* catur. Penulis menyadari masih banyak kekurangan di dalam makalah ini. Oleh karena itu, penulis berharap ide makalah ini bisa dikembangkan di masa depan.

TAUTAN REPOSITORI GITHUB

Berikut merupakan tautan repositori Github berisi *source code* program yang digunakan dalam pembahasan makalah ini.

https://github.com/dikyrest/Makalah_13520017.git

TAUTAN VIDEO YOUTUBE

Berikut merupakan tautan video YouTube berisi penjelasan terhadap isi makalah ini.

https://youtu.be/nLV4VkOo7pI

UCAPAN TERIMA KASIH

Penulis mengucapkan puji syukur ke hadirat Allah Swt. karena atas rahmat dan karunia-Nya, proyek makalah Strategi Algoritma yang berjudul "Aplikasi Algoritma *Backtracking* dalam Persoalan The Knight's Tour" dapat diselesaikan dengan baik dan tepat waktu.

Penulis juga mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi, M.T., Dr. Nur Ulfa Maulidevi, S.T., M.Sc., dan ibu Dr. Masayu Leylia Khodra, S.T, M.T. selaku dosen pengajar

mata kuliah IF2211 Strategi Algoritma yang telah membimbing penulis memahami materi yang digunakan untuk membuat makalah ini.

Terakhir, penulis juga mengucapkan terima kasih kepada kedua orang tua dan keluarga yang mendukung penulis dalam mengerjakan makalah ini.

REFERENSI

[1] H. A. Davidson. 1968. A short history of chess. New York: D. McKay Co.

[2] Brendan McKay. 1997. Knight's Tours on an 8 × 8 Chessboard. Canberra.

[3] "FIDE Laws of Chess, article 3.8.2". fide.com. World Chess Federation.

[4] Munir, Rinaldi. 2021. Graf (Bag. 1): Bahan Kuliah IF2120 Matematika Diskrit. Diakses pada tanggal 21 Mei 2022

[5] Munir, Rinaldi. 2022. Algoritma Runut-balik (*Backtracking*) (Bagian 1). Diakses pada tanggal 22 Mei 2022.

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, 22 Mei 2022

Diky Restu Maulana
1352001