Implementasi dan Analisis Algoritma Backtracking Dalam Penyelesaian Permainan Sudoku Berbasis Bahasa C

Mochammad Syahrindra Akbar Suharno[1], Raihan Nur Hanif[2], Juan Marten Daniel Lolowang[3], Jie Catur Nugraha[4]

[1, 2, 3, 4] Jurusan Teknik Informatika, Universitas Widyatama Jl. Cikutra, No. 204 A, Bandung, Jawa Barat, 40125

Email: [1]raihan.hanif@widyatama.ac.id, [2]mochammad.syahrindra@widyatama.ac.id, [3]juan.lolowang@widyatama.ac.id, [4] jie.catur@widyatama.ac.id

Abstrak

Permainan teka-teki logika populer bernama Sudoku yang pertama kali diciptakan oleh Howard Garnes pada tahun 1979 dan diberi nama oleh Maki Kaji dari Jepang pada tahun 1984, memiliki nama yang merupakan singkatan dari kata-kata bahasa Jepang yang berarti "angka-angkanya harus tetap tunggal". Sudoku memiliki berbagai tingkat kesulitan, namun sulit untuk menentukan tingkat kesulitan suatu sudoku hanya dengan melihat jumlah sel yang kosong. Penelitian ini bertujuan untuk mengevaluasi efektivitas algoritma backtraking dalam mengklasifikasikan tingkat kesulitan sudoku yang diimplementasikan dalam program bahasa C. Algoritma ini akan mencoba menebak setiap kombinasi angka untuk menemukan solusi yang valid, dan jika tebakan tersebut tidak berhasil, akan kembali ke tebakan sebelumnya dan mencoba angka yang berbeda. Setelah melakukan penelitian ini, dapat disimpulkan bahwa algoritma backtraking efektif dalam mengklasifikasikan tingkat kesulitan sudoku.

Keywords: Sudoku, Penelitian, Algoritma Backtraking, Program Bahasa C, Tingkat Kesulitan Sudoku.

Abstrak

A popular logic puzzle game called Sudoku, which was first created by Howard Garnes in 1979 and named by Maki Kaji from Japan in 1984, has a name that is an acronym in Japanese meaning "the numbers must remain single". Sudoku has various levels of difficulty, but it is difficult to determine the difficulty level of a Sudoku puzzle just by looking at the number of empty cells. This study aims to evaluate the effectiveness of the backtracking algorithm in classifying the difficulty level of Sudoku puzzles implemented in C language. The algorithm will try to guess every combination of numbers to find a valid solution, and if the guess fails, it will go back to the previous guess and try a different number. After conducting this study, it can be concluded that the backtracking algorithm is effective in classifying the difficulty level of Sudoku puzzles.

Kata kunci: sudoku, penelitian, algoritma backtraking, program bahasa c, tingkat kesulitan sudoku.

1. Pendahuluan

Sudoku adalah sebuah permainan teka-teki logika yang sangat populer. Sudoku sudah dikenal di kalangan mulai dari anak-anak hingga orang dewasa [1]. (Hadinata, 2011).

Teka-teki sudoku pertama kali diciptakan oleh Howard Garnes, pada tahun 1979 [2]. (Job & Paul, 2017,). Pada tahun 1984 Maki kaji dari jepang merilis di majalah teka-tekinya yang bernama Nikoli dan memberi nama sudoku pada permainan teka-tekinya tersebut yang berarti “Angka tunggal” [3] (Lin & Wu, 2011, #). Nama sudoku adalah singkatan bahasa jepang dari “Suji wa dokushio ni kagiru” yang berarti “angka-angkanya harus tetap tunggal”.

2. Tinjauan Pustaka

2.1 Definisi Sudoku

Sebuah (standar) sudoku adalah permainan puzzle yang mengandung 9×9 kotak persegi yang dibagi menjadi sembilan 3×3 kotak persegi, dengan total terdapat 81 sel. Beberapa dari 81 sel dapat diisi angka dari {1, 2, 3, 4, 5, 6, 7, 8, 9} [4]. (Robertson et al., 1997, #)

Pertama-tama, beberapa sel akan diberi angka awal yang dapat digunakan sebagai pentunjuk dalam menyelesaikan sudoku [5] (Santos-Garcia & Palomino, 2007, #).

Tujuan Permainan adalah untuk mengisi seluruh kotak persegi menggunakan

sembilan digit antara 1 sampai 9 sehingga setiap baris, kolom dan kotak atau blok berisi setiap angka tepat satu kali [2] (lihat contoh di gambar 1).

Gambar 1. Contoh teka-teki sudoku dan solusinya. Sumber wikipedia [6]

Sudoku yang berbeda tentu akan memiliki tingkat kesulitan yang berbeda pula. Namun, tingkat kesulitan sebuah sudoku tidak mudah untuk diklasifikasikan kesulitannya hanya dengan melikat jumlah sel yang kosong. Kesalahpahaman yang sering terjadi tentang sudoku adalah bahwa jumlah petunjuk menentukan betapa sulitnya sudoku itu. Walaupun hal itu benar untuk gambaran yang lebih besar, sudoku yang memiliki 17 petunjuk jauh lebih sulit dari sudoku yang memiliki 30 petunjuk [7] (Berggren & Nilsson, 2012)].

Tingkat kesulitan sebuah sudoku tidak hanya sulit untuk ditentukan, tetapi juga karena tidak ada kesepakatan khusus yang diterima secara umum [8] (Mantere & Koljonen, 2007, #).

2.2 Algoritma Bakctraking (Runut-balik)

Algoritma *Bakctraking* merupakan salah satu algoritma yang diperlukan dalam memecahkan sebuah msalah yang berkaitan dengan pemenuhan terbatas, misalnya pada penyelesaian permainan puzzle [9] (Salsabila, 2020).

Backtracking adalah teknik pencarian *brute force* yang merupakan teknik umum dalam kecerdasan buatan, yang dimana teknik ini akan mencoba segala kemungkinan sampai solusi tercabai [10] (N & Khazam, 2010, #). Backtraking adalah metode sistematis untuk mengulangi semua kombinasi yang mungkin [11] (Khachiyan et al., 2005, #).

Dalam penyelesian permainan sudoku, algoritma ini akan mencoba angka yang berbeda pada sebuah sel, dan jika gagal, maka akan kembali ke tebakan sebelumnya dan mencoba angka atau kombinasi yang berbeda [7](Berggren & Nilsson, 2012).

3. Metodologi Penelitian

3.1 Implementasi

Program peyelesaian sudoku akan diimplementasikan menggunakan bahasa C yang dijelaskan dibawah ini:

```
int solve(int puzzle[][9]) {
 int row;
 int column;

 if(!find_empty_cell(puzzle, &row,
&column)) return 1; // seluruh sel telah
terisi

 for (int guess = 1; guess < 10; guess++) {
// opsinya adalah 1-9
if (valid(puzzle, row, column,
guess)) { // mencoba mengisi
dengan  angka
```

```
puzzle[row][column] = guess; //
jika tebakan valid maka angka akan diinput ke array sudoku

 if(solve(puzzle)) return 1; //berulang jika berhasil berhenti
puzzle[row][column] = 0; // ulang dan mencoba kembali
}
}
 return 0;
}
```

Pertama-tama sudoku akan disimpan dengan array dua dimensi, lalu algoritma *backtraking* akan mencoba atau menebak setiap kombinasi angka dan jika gagal maka akan kembali ke tebakan sebelumnya dan menebak angka yang berbeda.

Pada penelitian ini pengujian sudoku program sudoku dilakukan dengan 30 sudoku yang memiliki tingkat kesulitan yang berbeda, yaitu, mudah, sedang, dan sulit. Tingkat kesulitan dan set sudoku ditentukan dan diambil dari sebuah website permainan sudoku [12]. (Easybrain, n.d.)

Pengujian ini dilakukan pada perangkat laptop yang memiliki spesifikasi AMD Athon silver 3050U @ 2.3GHz, RAM 12GB DDR4 Memory dan SDD 256 GB. Pengujiann ini juga dilakukan pada software yang Bernama Dev C/c++ dan menggunakan Bahasa Pemrograman C.

3.2 Analisis

Program akan dianalisis berdasarkan *time complexity*, *space complexity*, dan *runtime* program.

Time complexity atau kompleksitas waktu mendeskripsikan jumlah waktu yang komputer perlukan untuk menjalankan sebuah algoritma. Pada umumnya, *time complexity* diperkirakan dengan menghitung jumllah operasi dasar yang dilakukan oleh sebuah algoritma, dengan asumsi bahwa setiap operasi dasar membutuhkan sejumlah waktu yang sama untuk dijalankan [13] (*Time Complexity*, n.d.). *Time complexity* biasanya dinyatakan menggunakan notasi *Big-O*[1], misalnya O(n) yang dimana n adalah ukuran dalam satuan bit yang diperlukan untuk mewakili input algoritmanya.

Space complexity atau kompleksitas ruang adalah seberapa banyak ruang memori yang diperlukan untuk menyelesaikan suatu algoritma hingga algoritma itu tereksekusi sepenuhya [14] (*Space Complexity*, n.d.)]. Sama seperti *time complexity*, *space complexity* juga biasanya dinyatakan dengan menggunakan notasi *Big-O*. namun, n disini merepresentasikan karakteristik input yang memengaruhi kompleksitas ruang.

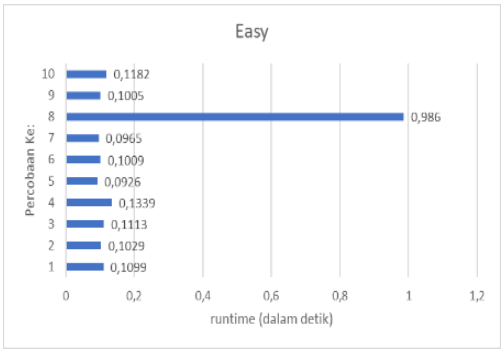
Runtime adalah fase berjalannya sebuah program yang dimulai sejak program dibuka hingga program berhenti [15] (*Runtime (Program Lifecycle Phase)*, n.d.).

4. Hasil dan Pembahasan

Time complexity pada algoritma ini adalah $O(9^{(n\times n)})$ dikarenakan terdapat 9 kemungkinan pada setiap sel. Dimana n adalah banyaknya kolom. Dikarenakan sudoku disimpan dalam array dua dimensi maka *space complexity*-nya adalah $O(n \times n)$.

Percobaan pertama adalah dengan menyelesaikan sudoku dengan tingkat kesulitan mudah. Pada penelitian ini tingkat kesulitan mudah adalah sudoku yang memiliki 38 petunjuk. Hasil *runtime* dari percobaan algortima *backtraking* pada 10 sudoku dengan tingkat kesulitan mudah adalah sebagai berikut:

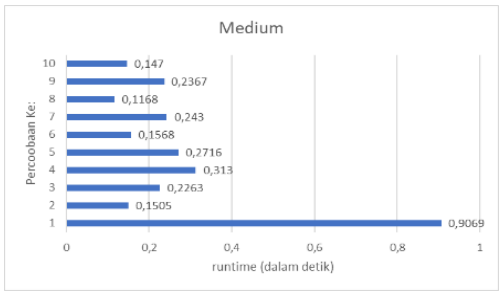
[1] Big-O digunakan dalam bidang ilmu komputer untuk memperkirakan batas atas atau kemungkinan terburuk runtime dari sebuah algoritma [16] (Miranda & Lewis, 2014, #).



Gambar 2. Grafik pengujian pada 10 sudoku dengan tingkat kesulitan mudah.

Waktu tercepat didapat pada percobaan ke 5 dengan waktu *runtime* 0,0926 detik dan waktu terlambat didapat pada saat percobaan ke 8 dengan waktu *runtime* 0,986 detik. Berdasarkan percobaan pada 10 sudoku dengan tingakat kesulitan mudah, didapat rata-rata *runtime* 0,19527 detik.

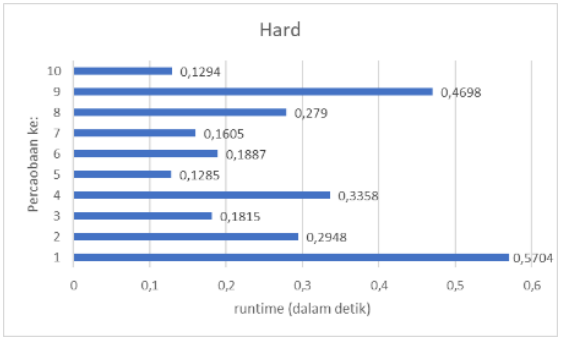
Percobaan kedua adalah dengan menyelesaikan 10 sudoku dengan tingkat kesulitan sedang. Pada penelitian kali ini sudoku dengan tingkat kesulitas sedang adalah sudoku yang memiliki 30 petunjuk. Hasil *runtime* dari percobaan algortima *backtraking* pada 10 sudoku dengan tingkat kesulitan mudah adalah sebagai berikut:



Gambar 3. Grafik pengujian pada 10 sudoku dengan tingkat kesulitan sedang

Waktu tercepat didapat pada percobaan ke-8 dengan waktu *runtime* 0,1168 detik dan waktu terlambat didapat pada saat percobaan ke-satu dengan waktu *runtime* 0,9069 detik. Berdasarkan percobaan pada 10 sudoku dengan tingakat kesulitan mudah, didapat rata-rata *runtime 0,27686* detik.

Percobaan ketiga adalah dengan menyelesaikan 10 sudoku dengan tingkat kesulitan sulit. Pada penelitian kali ini sudoku dengan tingkat kesulitas sedang adalah sudoku yang memiliki 25petunjuk. Hasil *runtime* dari percobaan algortima *backtraking* pada 10 sudoku dengan tingkat kesulitan mudah adalah sebagai berikut:



Gambar 4. Grafik pengujian pada 10 sudoku dengan tingkat kesulitan sulit.

Waktu tercepat didapat pada percobaan ke 5 dengan waktu *runtime* 0,1285 detik dan waktu terlambat didapat pada saat percobaan ke 8 dengan waktu *runtime* 0,5704 detik. Berdasarkan percobaan pada 10 sudoku dengan tingakat kesulitan mudah, didapat rata-rata *runtime 0,27384* detik.

Tabel 4.1 Hasil ekperimental

Banyaknya sudoku	10	10	10
Banyaknya petunjuk	38	30	25
Waktu penyelesaian	0,19527	*0,27686*	*0,27384*

Setelah melakukan percobaan pada 30 ` sudoku dengan tingkat kesulitan yang berbeda-beda, didapat bahwa rata-rata penyelesaian sudoku adalah 0,2486 s.

5. Kesimpulan

Algoritma backtracking memiliki time complexity $O(9^{(n \times n)})$ dan space complexity $O(n \times n)$ pada penyelesaian sudoku, dengan n adalah banyaknya kolom. Telah dilakukan tiga percobaan dengan tingkat kesulitan yang berbeda,

yaitu mudah, sedang, dan sulit, dengan masing-masing memiliki jumlah petunjuk yang berbeda. Hasil dari setiap percobaan menunjukkan bahwa waktu runtime rata-rata algoritma backtracking semakin lama semakin meningkat seiring dengan tingkat kesulitan sudoku yang semakin tinggi dan didapatkan bahwa rata-rata penyelesaian sudoku adalah 0,2486 s.

6. Daftar Pustaka

[1] Hadinata, J. (2011). Problem Solving Sudoku Menggunakan Algoritma Genetika. *SISFOTENIKA*, *1*(1), 48-58. https://media.neliti.com/media/publications/226228-problem-solving-sudoku-menggunakan-algor-8b2ec5a0.pdf

[2] Job, D., & Paul, V. (2017). Recursive Backtracking for Solving 9*9 Sudoku Puzzle. *Bonfring International Journal of Data Mining*, *6*(1).

[3] Lin, H.-H., & Wu, I.-C. (2011, December 1). An Efficient Approach to Solving the Minimum Sudoku Problem. *ICGA Journal*, *34*(4), 191-208. 10.3233/ICG-2011-34403

[4] Robertson, N., Sanders, D., Seymour, P., & Thomas, R. (1997, May). The Four-Colour Theorem. *Journal Of Combinatorial Theory*, *70*(1), 2-44. https://reader.elsevier.com/reader/sd/pii/S0095895697917500?token=7FFA5A2261BAB1744D8A98F6CB8FF15EA026AD03313B08BA27AF7D528746AA845BA059C35AE13A3CA35D1F9047877E54&originRegion=eu-west-1&originCreation=20230107154742

[5] Santos-Garcia, G., & Palomino, M. (2007, July 28). Solving Sudoku Puzzles with Rewriting Rules. *Electronic Notes in Theoretical Computer Science, 176(4), 79-93. https://doi.org/10.1016/j.entcs.2007.06.009*

[6] Stellmach, T. (n.d.). *File:Sudoku Puzzle by L2G-20050714 standardized layout.svg*. Wikimedia Commons. Retrieved January 8, 2023, from https://en.wikipedia.org/wiki/File:Sudoku_Puzzle_by_L2G-20050714_standardized_layout.svg

[7] Berggren, P., & Nilsson, D. (2012). *A study of Sudoku Solving Algorithms*. Bachelor's Thesis Bachelor's Thesis, Royal Institute of Techology.

[8] Mantere, T., & Koljonen, J. (2007). Solving, rating and generating Sudoku puzzles with GA. *Publications of the finish Artificial Intelligence Society*. 10.1109/CEC.2007.4424632

[9] Salsabila, P. N. (2020). *Penerapan Algoritma Backtracking dalam Menyelesaikan Sudoku with 4 Given Digits*.

[10] N, F., & Khazam, A. (2010, November). A Kernelization algorithm for d-Hitting Set. *Journal of Computer and System Sciences*, *76*(7), 524-531. https://doi.org/10.1016/j.jcss.2009.09.002

[11] Khachiyan, L., Boros, E., Elbassioni, K., & Gurvich, V. (2005). A New Algorithm for the Hypergraph Transversal Problem. *11th Annual International Conference,*, *3595*, 767-776.

[12] Easybrain. (n.d.). Sudoku. https://sudoku.com/

[13]*Time complexity*. (n.d.). Wikipedia. Retrieved January 8, 2023, from https://en.wikipedia.org/wiki/Time_complexity#Table_of_common_time_complexities

[14] *Space complexity*. (n.d.). Wikipedia. Retrieved January 8, 2023, from https://en.wikipedia.org/wiki/Space_complexity

[15] *Runtime (program lifecycle phase)*. (n.d.). Wikipedia. Retrieved January 8, 2023, from https://en.wikipedia.org/wiki/Runtime_(program_lifecycle_phase)

[16] Miranda, P., & Lewis, C. (2014). What Makes Big-O Analysis Difficult: Understanding How Students Understand Runtime Analysis. *Journal of Computing Sciences in College*.