Jurnal Informatika Universitas Pamulang **ISSN: 2541-1004**
Penerbit: Program Studi Teknik Informatika Universitas Pamulang **e-ISSN: 2622-4615**
Vol. 6, No. 3, September 2021 (506-511) **10.32493/informatika.v6i3.10686**

Aplikasi Pemecahan Soal Sudoku dengan Metode Backtracking

Chyquitha Danuputri[1], Nico Santosa[2]

[1,2]Teknik Informatika, Teknologi dan Desain, Universitas Bunda Mulia
Jl. Lodan Raya No. 2 Ancol, Indonesia
e-mail: [1]chyquitha@gmail.com, [2]nico.santosa1@gmail.com

Submitted Date: May 20th, 2021 Reviewed Date: July 25th, 2021
Revised Date: October 01st, 2021 Accepted Date: October 12th, 2021

Abstract

In solving a puzzle requires dexterity, intelligence, and time depending on the difficulty of the puzzle to be solved, one of the popular puzzles is sudoku. This puzzle game often takes a long time to complete, especially at high difficulty levels. To solve puzzle problems more quickly and efficiently, a backtracking algorithm can be applied, which is a systematic logical sequence used to find a solution to a problem where there are several possible solutions. This research is meant to test the backtracking ability in solving sudoku problems at extreme difficulty levels and to test the speed of the backtracking algorithm in solving sudoku problems at extreme difficulty levels. In the application test, 20 questions were used and from testing the 20 questions, the accuracy rate of solving sudoku questions was 100%, while for the length of time for solving sudoku questions, the average length of time was 0.0880295 seconds. The application made is a desktop-based application made by Python programming language and PyGame library to create a user interface.

Keywords: Backtracking; Python; Pygame; Sudoku

Abstrak

Dalam menyelesaikan sebuah teka-teki dibutuhkan ketangkasan, kecerdasan, dan waktu tergantung pada tignkat kesulitan teka-teki yang akan diselesaikan, salah satu teka-teki yang populer adalah sudoku. Permainana teka-teki ini seringkali membutuhkan waktu yang cukup lama untuk menyelesaikanya terutama pada tingkat kesulitan yang tinggi. Untuk menyelesaikan permasalahan teka-teki yang lebih cepat dan efisian dapat diterapkan algoritma backtracking, yaitu urutan logis secara sistematis yang digunakan untuk mencari solusi dari sebuah permasalahan di mana terdapat beberapa kemungkinan solusi yang ada. Dalam penelitian ini dibangun sebuah aplikasi yang berfungsi untuk mencari jawaban dari sebuah soal sudoku, tujuan dibuat aplikasi ini adalah untuk menguji kemampuan backtracking dalam memecahkan soal sudoku pada tingkatan kesulitan extreme dan menguji kecepatan algoritma backtracking dalam menyelesaikan soal sudoku pada tingkat kesulitan extreme. Pada pengujian aplikasi digunakan soal sebanyak 20 buah dan dari pengujian ke-20 soal tersebut didapatkan tingkat akurasi pemecahan soal sudoku sebesar 100% sedangkan untuk lama waktu pemecahan soal sudoku mendapatkan rata-rata lama waktu sebesar 0.0880295 detik. Aplikasi yang dibuat merupakan aplikasi berbasis desktop dengan bahasa pemrograman Python dan library PyGame untuk membuat tampilan antarmuka pengguna.

Kata Kunci: Backtracking; Python; Pygame; Sudoku

1. Pendahuluan

Algoritma *backtracking* (runut balik) adalah sebuah urutan logis secara sistematis yang digunakan untuk mencari solusi dari sebuah permasalahan di mana terdapat beberapa kemungkinan solusi yang ada (Rifqo & Apridiansyah, 2017). Algoritma *backtracking* (runut balik) dikemukakan oleh seorang matematikawan D.H. Lehmer pada tahun 1950 yang kemudian dikembangkan oleh RJ Walker, Golomb, dan Baumert yang kemudian memberikan uraian mengenai runut balik serta berbagai penerapannya di dalam persoalan (Azanuddin et al., 2017). Salah satu pengaplikasian metode *backtracking* adalah dalam

menyelesaikan permasalahan dalam teka-teki (Herimanto et al., 2020).

Salah teka-teki yang cukup terkenal adalah *sudoku*. *Sudoku* merupakan sebuah permainan teka-teki angka yang berasal dari Jepang (Putrilani et al., 2016). *Sudoku* dimainkan di atas papan yang terbuat dari 9 buah kotak di mana masing-amasing kotak berukuran 3×3 (*subgrid*). Pada awal permainan beberapa kotak sudah terisi diawal dengan angka acak yang memili rentang dari 1 sampai 9 dan kotak-kotak yang terisi diawal tersebut memiliki fungsi sebagai petunjuk awal untuk menyelesaikan teka-teki sudoki dan tugas pemain adalah melengkapi kotak lain yang masih kosong atau belum terisi sehingga akhirnya keseluruhan papan *sudoku* terisi angka (Rahayu et al., 2017) Adapun peraturan permainan *sudoku* sangatlah sederhana, yaitu:

1. Angka pada setiap baris, kolom, darn *grid* haruslah unik sehingga tidak ada angka yang berulang.
2. Angka yang diisikan haruslah bernilai dengan rentang 1 sampai 9 dan angka tersebut harus diisikan pada setiap kotak sehingga tidak ada kotak yang kosong.

Walaupun dalam peraturan permainan ini hanya memiliki 2 aturan, tetapi dalam menyelesaikan sebuah teka-teki *sudoku* dalam tingkatan yang sulit membutuhkan logika dan ketelitian yang baik. Sehingga tidak jarang dalam mengerjakan teka-teki *sudoku* dengan tingkatan yang sulit dapat menghabiskan waktu yang cukup lama.

Sehingga berdasarkan pemaparan di atas maka peneliti bermaksud untuk membuat aplikasi yang dapat memecahkan permasalahan soal *sudoku* pada tingkat *extreme* dengan menerapkan metode *backtracking* dan menguji lama waktu aplikasi ini dalam memecahkan soal sudoku. Aplikasi yang dibuat merupakan aplikasi berbasis *desktop* dengan bahasa pemrograman *Python* dan *library PyGame* untuk membuat tampilan antarmuka pengguna.

Penelitian ini menggunakan soal sebanyak 20 yang diambil dari *website* sudoku.ws di mana pada *wesbite* ini peneliti mengambil contoh soal dengan tingkat paling tinggi yaitu *extreme*, pada *website* ini disediakan juga pemecahan masalah atas soal *sudoku* yang diberikan sehingga nantinya dapat digunakan sebagai validasi atas hasil aplikasi dengan kunci jawaban yang disediakan.

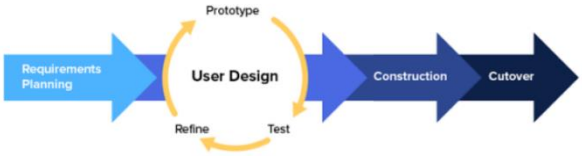
Peneliti berharap agar penelitian ini nantinya dapat menjadi landasan untuk penelitian serupa serta membantu menyelesaikan permasalahan soal sudoku dengan tingkat kesulitan yang tinggi.

2. Metode Penelitian

2.1. Studi Pustaka

Dalam menjalankan penelitian ini peneliti menerapkan metode studi pustaka yaitu sebuah metode yang melakukan beberapa kegiatan seperti pengumpulan data pustaka, membaca dan mencatat bahan yang memiliki hubungan penelitian yang dilakukan (Supriyadi, 2017). Oleh karena itu dalam penelitian ini, maka diperlukan pengumpulan data yang peneliti lakukan adalah dengan membaca serta mengeksplorasi beberapa buku, jurnal dan *website* serta sumber data yang dapat mendukung dalam melaksanakan penelitian ini.

2.2. Rapid Application Development (RAD)



Gambar 1. Siklus Metode Rapid Application Development (RAD)

Model pengembangan perangkat lunak yang peneliti digunakan adalah *Rapid Application Development (RAD)*). *Rapid Application Development (RAD)* adalah model pembangunan perangkat lunak yang masih termasuk dalam teknik *incremental* (bertingkat) (Sagala, 2018). *RAD* menekankan pada siklus pembangunan dengan waktu pendek, singkat, dan cepat. *Rapid Application Development (RAD)*. Tahapan dari model *Rapid Application Development (RAD)* terdiri dari 4 tahap (Hendini, 2016), yaitu:

1. Analisis persyaratan (*requirement)*
   Tahap ini merupakan tahapan awal yang bertujuan untuk melakukan identifikasi *requirement* dari aplikasi yang akan dibuat.
2. Analisis pemodelan (*modeling)*
   Pada tahapan ini dilakukan analisis terhadap sistem secara keseluruhan yang akan dibuat.
3. Desain pemodelan (*modeling)*
   Pada tahap ini dilakukan perancangan sistem berdasarkan tahap sebelumnya. Tahap analisis dan desain mengalami perulangan sehingga nantinya

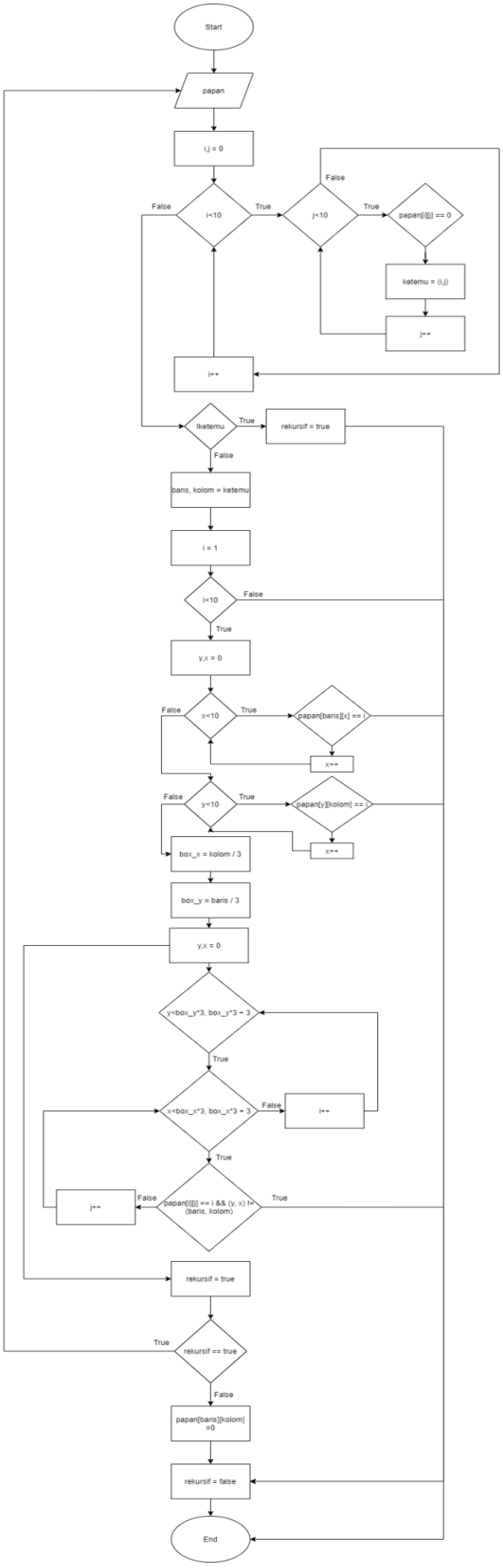
diperoleh hasil rancangan sistem yang benar-benar memenuhi kebutuhan.

4. Konstruksi (*construction*)

   Pada tahap ini dilakukan implementasi pemodelan yang telah dibuat ke dalam bentuk bahasa pemrograman serta melakukan pengujian terhadap perangkat lunak yang telah dibuat. Hasil akhir dari fase ini dapat berupa *hardware* atau *software.*

2.3. Algoritma *Backtracking*

Algoritma *backtracking* (runut balik) merupakan metode yang digunakan untuk mecnari solusi dari pemecahan masalah yang berbasis pada pencarian ruang status(Rifqo & Apridiansyah, 2017). Algoritma ini bekerja secara rekursif dalam mencari solusi dari sebuah permasalahan di mana terdapat beberapa kemungkinan solusi. Algoritma ini berbasis pada algoritma *Depth-First Search* (*DFS*) untuk mencari solusi persoalan yang lebih efektif. Alogritma *Backtracking* (runut balik) sendiri adalah bentuk tipikal dari algoritma rekursif dan berbasis pada algoritma *Depth-First Search* (*DFS*) (Utama et al., 2016). Algoritma *backtracking* (runut balik) ini akan mencoba beberapa kemungkinan yang ada sampai menemukan salah satu kemungkinan yang dapat dijalankan hingga mencapai keputusan. Pada metode ini kemungkinan yang menemukan jalan buntu atau tidak mencapai keputusan maka tidak perlu dilanjutkan(*pruning*) sehingga waktu pencarian solusi dapat lebih singkat. Algoritma *backtracking* sering diterapkan untuk melakukan pemecahan masalah pada *games* dan pada bidang *artificial intelligence*. Alur algoritma *backtracking* yang diimplementasikan di dalam aplikasi yang dibuat ditunjukkan pada Gambar 2.

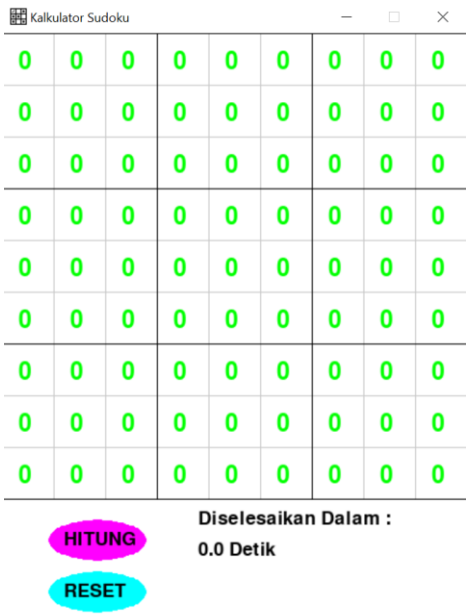


Gambar 2. Flowchart Aplikasi

1. Alur dimulai dengan memasukan *input* berupa papan kemudian menginisialisasi variabel i dan j dengan nilai 0, variabel ini nantinya akan digunakan untuk perulangan.
2. Kemudian melakukan pencarian kotak yang bernilai 0 (kosong) dengan mengecek baris dan kolom dengan melakukan perulangan di mana variabel i yang dibuat tadi digunakan untuk perulangan baris sedangkan variabel j digunakan untuk perulangan kolom. Kemudian dilakukan pengkondisian jika papan pada kolom dan baris bernilai 0 maka simpan kolom dan baris itu ke dalam variabel ketemu dan nilai j akan ditambah 1 tiap perulangan dan jika perulangan kolom(variabel j) selesai tambahkan nilai i sebesar 1 dan lakukan perulangan kembali sampai nilai i mencapai 10.
3. Lakukan pengkondisian di mana jika *not* ketemu maka nilai variabel rekursif *true* dan program selesai(atau kembali ke *stack* sebelumnya) jika tidak maka lanjut ke tahap selanjutnya.
4. Buat variabel baris dan kolom dengan nilai variabel ketemu dan buat variabel i dengan nilai 1.
5. Selanjutnya mencari nilai untuk dimasukan kedalam kotak. Pertama dilakukan pengkondisian jika i kurang dari 10(mencari nilai 1-9) jika bernilai *true* maka lanjut ke tahap selanjutnya jika tidak maka program selesai(atau kembali ke stack sebelumnya).
6. Sebelum memasukan angka dilakukan pengecekan sesuai dengan ketentuan permainan *sudoku* yaitu mengecek baris, kolom, dan *subgrid* untuk memastikan angka yang dimasukan tidak ada yang duplikasi. Pertama buat variabel x dan y dengan nilai 0 kemudian lakukan pengkondisian untuk baris jika terdapat nilai yang sama dalam satu baris papan begitu juga dengan kolom. Jika salah satu pengkondisian bernilai *true* maka program selesai(atau kembali ke *stack* sebelumnya) Kemudian untuk mengecek *subgrid* dibuat variabel pembantu box_x yang bernilai kolom/3, box_y yang bernilai baris/3, dan variabel y serta x yang bernilai 0 kemudian dilakukan pengulangan di mana selama y berada di antara nilai box_y*3 dan box_y*3+3, lalu dilakukan pengulangan lagi di dalamnya di mana selama x berada diantara nilai box_x*3 dan box_x*3+3 kemudian dilakukan pengkondisian jika nilai papan baris ke-idan kolom ke-j dan (i,j) tidak sama dengan (baris,kolom). Jika bernilai *true* maka program selesai (atau kembali ke *stack* sebelumnya). Jika tidak maka variabel rekursif bernilai *true.*
7. Kemudian dilakukan pengkondisian lagi jika variabel rekursif bernilai *true* maka akan melakukan rekursif (dan otomatis membuat *stack* di *memory*). Jika tidak maka lanjut ke tahap selanjutnya.
8. Buat nilai variabel papan baris ke-baris dan kolom ke-kolom sama dengan 0.
9. Dan buat nilai variabel rekursif sama dengan *false.*
10. Program selesai (atau kembali ke *stack* sebelumnya).

2.4. Desain Antarmuka Pengguna

Antarmuka pengguna merupakan suatu bahasan penting dalam interaksi manusia dengan komputer sehingga pengguna dapat mencapai dan melaksanakan fungsi dari sebuah sistem (Purnomo et al., 2018).Desain antarmuka pengguna dibuat seminimalis mungkin untuk memudahkan dalam pengoperasian aplikasi sehingga tidak membuat pengguna kebingungan selain itu juga membuat aplikasi menjadi lebih ringan.



Gambar 3. *Interface* Awal Program

Gambar 3 merupakan *Interface* awal aplikasi, terdapat 2 buah tombol yaitu HITUNG dan *RESET*. Untuk mengisi kotak digunakan *mouse*, klik kiri atau *scroll* atas untuk menambahkan angka, klik kanan untuk mengurangi angka atau *scroll* bawah dan klik tengah *scroll* untuk mengembalikan kotak ke angka 0.

Gambar 4. Tampilan soal yang dimasukkan ke aplikasi

Gambar 4 menunjukkan tampilan ketika aplikasi sudah dimasukan soal, di mana soal merupakan angka berwarna merah dan angka hijau akan menjadi jawaban dari soal.

Gambar 5 menunjukkan tampilan akhir dari aplikasi di mana semua kotak dengan angka hijau 0 berubah menjadi jawaban, di pojok bawah kanan aplikasi juga ditunjukkan lamanya aplikasi dalam menyelesaikan soal *sudoku*. Untuk mengisi kembali soal *sudoku* dapat menekan tombol *RESET* untuk mengembalikan semua kotak ke angka 0.

Gambar 5. Tampilan akhir setelah menyelesaikan sudoku

3. Hasil dan Pembahasan

Dalam menguji aplikasi yang dikembangkan dilakukan pengujian sebanyak 20 kali dengan tingkat kesulitan *Extreme*. Adapun soal yang diambil berasal dari *website* sudoku.ws di mana didalam *website* tersebut menyediakan soal beserta dengan jawabanya.

Table 1. Hasil Uji

Keterangan	Hasil
Rata-rata lama pemecahan soal	0.0880295 detik
Akurasi pemecahan soal	100%

4. Kesimpulan

Kesimpulan diambil berdasarkan analisa, perancangan dan hasil pengujian aplikasi yang dikembangkan maka dapat ditarik kesimpulan:

1. Aplikasi *sudoku* dapat memecahkan soal *sudoku* pada tingkat yang sulit dalam hal ini berdasarkan *website* sumber soal yang diambil yaitu pada tingkat *extreme* yang juga merupakan tingkat tersulit pada *website* tersebut. Adapun akurasi dari pemecahan soal yang diuji adalah 100%.
2. Aplikasi *sudoku* dapat memecahkan soal dalam waktu yang cukup cepat yaitu dengan rata-rata lama pemecahan masalah 0.0880295 detik dari 20 soal *sudoku* yang diuji pada aplikasi.

5. Saran

Berdasarkan penelitian yang telah dilakukan, terdapat beberapa saran yang dapat digunakan untuk mengembangkan penelitian selanjutnya:

1. Membuat tampilan *UI* yang lebih menarik dan lebih modern.
2. Mengembangkan aplikasi ke perangkat berbasis *mobile* sehingga dapat di akses lebih banyak orang.

Referensi

Azanuddin, Iskandar, Z., & Purwadi. (2017). Algoritma Backtracking Sebagai Solusi Game Word Search Puzzle Berbasis Java Mobile. *Saintikom*, *16*(3), 295–304.

Hendini, A. (2016). Pemodelan Uml Sistem Informasi Monitoring Penjualan Dan Stok Barang (Studi Kasus: Distro Zhezha Pontianak. Jurnal Khatulistiwa Informatika, Vol. Iv, No. 2 Desember 2016 Pemodelan. *Crop Science*, *IV*(2), 107–116.

Herimanto, Sitorus, P., & Zamzami, E. M. (2020). An Implementation of Backtracking Algorithm for Solving A Sudoku-Puzzle Based on Android. *Journal of Physics: Conference Series*. https://doi.org/10.1088/1742-6596/1566/1/012038

Purnomo, Anang, & Ardiansyah. (2018). Pengembangan User Experience (Ux) Dan User Interface (Ui) Aplikasi Ibeauty Berbasis Android. *JSTIE (Jurnal Sarjana Teknik Informatika) (E-Journal)*, *6*(3), 18–27. https://doi.org/10.12928/jstie.v6i3.15251

Putrilani, K. A., Renariah, R., & Sutjiati, N. (2016). Efektivitas Media Permainan Sudoku Dalam Menghafal Huruf Kana (Menggunakan Metode Eksperimen Quasi Terhadap Siswa Japanese Club SMP Laboratorium Percontohan UPI). *JAPANEDU: Jurnal Pendidikan Dan Pengajaran Bahasa Jepang*, *1*(3), 35–43. https://doi.org/10.17509/japanedu.v1i3.5840

Rahayu, D. S., Suryapratama, A., Amongsaufa, A. Z., & Koloay, B. I. K. (2017). Evaluasi Algoritma Runut Balik Dan Simulated Annealing Pada Permainan Sudoku. *Jurnal Teknik Informatika Dan Sistem Informasi*. https://doi.org/10.28932/jutisi.v3i1.592

Rifqo, M. H., & Apridiansyah, Y. (2017). Implementasi Algoritma Backtracking Dalam Sistem Informasi Perpustakaan Untuk Pencarian Judul Buku (Studi Kasus Unit Pelayanan Terpadu Perpustakaan Universitas Muhammadiyah Bengkulu). *Pseudocode*, *4*(1), 90–96. https://doi.org/10.33369/pseudocode.4.1.90-96

Sagala, J. R. (2018). Model Rapid Application Development (Rad) Dalam Pengembangan Sistem Informasi Penjadwalan Belajar Mengajar. *Jurnal Mantik Penusa*, *2*(1), 87–90.

Supriyadi, S. (2017). Community of Practitioners: Solusi Alternatif Berbagi Pengetahuan antar Pustakawan. *Lentera Pustaka: Jurnal Kajian Ilmu Perpustakaan, Informasi Dan Kearsipan*, *2*(2), 83–93. https://doi.org/10.14710/lenpust.v2i2.13476

Utama, F., Kridalaksana, A. H., & Astuti, I. F. (2016). Implementasi Backtracking Algorithm Untuk Penyelesaian Permainan Su Doku Pola 9x9. *Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer*, *11*(1), 29–36. https://doi.org/10.30872/jim.v11i1.200