**Permainan Sudoku Menggunakan Backtracking**

**Rizky Dewa Sakti**

Afiliasi Pengarang: Mahasiswa S1 Telkom University

Alamat : Jl. Kemasan No. 31, Krian, Sidoarjo

e-mail: [rizkydewasakti@gmail.com](mailto:rizkydewasakti@gmail.com)

### Abstrak

**Permainan sudoku juga dikenal *Number Place* atau *Nanpure* merupakan permainan teka-teki asal Jepang*.* Permainan Sudoku merupakan salah satu permainan teka-teki angka yang berbasis logika.**

**Tujuan dari permainan ini adalah untuk mengisikan angka – angka dari 1 sampai 9 kedalam papan 9x9 yang terdiri dari 9 boks 3x3 tanpa ada angka yang berulang di satu baris, satu kolom, dan satu boks. Permainan Sudoku mengharuskan pemainnya untuk berpikir secara logis sesuai dengan aturan-aturan permainan yang ada. Sehingga permainan ini sangat bermanfaat untuk melatih logika dan alur berpikir.**

### Dengan algoritma backtracking, kita tidak perlu memeriksa semua kemungkinan solusi yang ada, hanya pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan, akibatnya waktu pencarian dapat dihemat.

### Penelitian ini, bertujuan untuk membangun suatu penyelesaian dari suatu papan sudoku berukuraun 9x9 menggunakan algoritma backtracking dan menggunakan Python sebagai bahasa pemrogamannya.

### Sebenarnya ada banyak metode yang dapat digunakan untuk menyelesaikan suatu teka-teki sudoku selain dengan algoritma backtracking, contohnya dengan metode *Brute-Force.* Namun metode *Brute-Force* untuk menyelesaikan teka-teki sudoku kurang efektif,karena diperlukan sejumlah 981 pengecekan kebenaran jawaban untuk menyelesaikan sudoku. Kompleksitas waktu dari sudoku nxn sendiri merupakan NP-problem.

**Kata Kunci:** Sudoku, Algoritma backtracking, komplesitas waktu, solusi, Algoritma brute-force

1. **PENDAHULUAN**

Perkembangan teknologi informasi terjadi dalam berbagai bidang, salah satunya dalam bidang permainan. Selain sebagai hiburan, permainan juga dapat menjadi suatu hal yang menantang ataupun untuk mengasah kemampuan otak pemain, seperti permainan teka-teki yang saat ini sedang populer di

kalangan masyarakat.

Permainan sudoku juga dikenal Number Place atau Nanpure merupakan permainan teka-teki asal Jepang [1]. Sudoku adalah sebuah permainan teka-teki angka yang berbasis logika[2]. Nama "Sudoku" sendiri adalah singkatan bahasa Jepang dari "Suuji wa dokushin ni kagiru" artinya "angka-angkanya harus tetap tunggal" [3].

Pada umumnya, sebuah puzzle Sudoku terdiri dari 81 kotak yang disusun menjadi 9 baris, 9 kolom, dan 9 subbagian. Tujuan utama dari permainan ini adalah mengisi seluruh kotak tersebut dengan angka 1 sampai 9. Permainan Sudoku mengharuskan pemainnya untuk berpikir secara logis sesuai dengan aturan-aturan permainan yang ada. Salah satu algoritma sederhana dan paling umum yang sering digunakan untuk menyelesaikan permainan Sudoku adalah algoritma backtracking.

Algoritma backtracking merupakan perbaikan dari algoritma brute-force dan algoritma brute-force, di mana algoritma ini membangun pohon ruang status (statespace tree) untuk menemukan solusi[3]. Prinsip dari algoritma backtracking adalah jika terjadi kesalahan dalam pencarian solusi pada sebuah node, maka akan dilakukan backtrack ke node sebelumnya. Meski begitu, algoritma ini menjadi tidak efisien jika kandidat solusi yang ditemukan sama dengan kandidat solusi yang ditemukan pada algoritma brute-force.

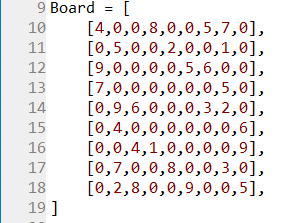
1. **ALGORITMA BACKTRACKING**

Algoritma backtrack pertama kali diperkenalkan oleh D.H. Lehmer pada tahun 1950[4]. Menurut Bakri[5], Algoritma backtracking adalah algoritma yang berbasis pada *Depth First Search (DFS)* untuk mencari solusi persoalan secara lebih mangkus.

Algoritma backtracking adalah sebuah algoritma yang berbasis depth-first search (DFS) dalam pencarian solusi pada pohon ruang status yang dibangun secara dinamis. Algoritma ini membangun solusi parsial (partial solution) dari sebuah kandidat solusi dan mengevaluasi solusi parsial tersebut pada suatu waktu. Jika solusi parsial yang dibangun tidak memenuhi syarat, maka kandidat solusi tersebut tidak akan dibangun lebih lanjut dan dilakukan backtrack ke kandidat solusi lain yang memenuhi syarat. Algoritma backtracking dilakukan secara berulangulang hingga menemukan sebuah solusi yang sesuai dengan syarat[6].

### PENJELASAN PROGRAM

Diberikan input soal sudoku berupa array asosiatif yang berisi 9x9 angka 1 sampai 9.



Gambar 1. **Input Soal Sudoku Bertipe Array Asosiatif.**

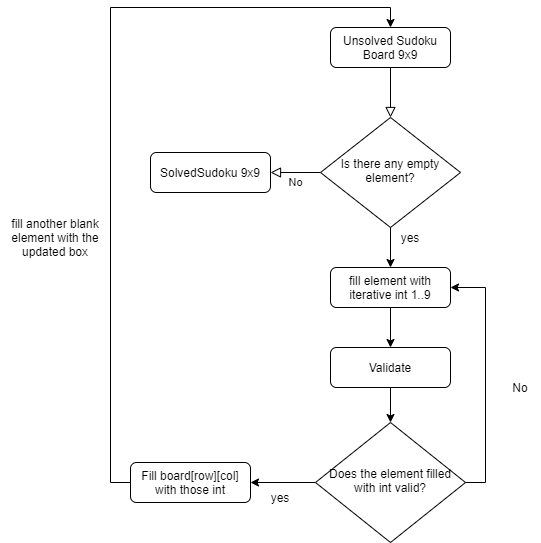
Berdasarkan dari gambar 1, angka 0 merupakan simbol dimana elemen sudoku belum dijawab dan perlu untuk ditemukan solusinya.

Setelah melalui input, proses selanjutnya adalah melakukan pengecekan untuk mendapatkan baris dan kolom dari setiap elemen yang bernilai 0 didalam array. Setelah itu, dilakukan pengisian elemen tersebut dengan angka 1 sampai 9 dan dicek kebeneran pengisian tersebut berdasarkan peraturan sudoku yaitu, setiap baris, kolom, dan subbox 3x3, tidak boleh ada angka yang sama atau duplikat.

Setelah itu, program telah memiliki hasil atau solusi dari persoalan sudoku. Output yang diberikan yaitu perbandingan sudoku sebelum diselesaikan dan sesudah diselesaikan. Untuk lebih jelasnya dapat dilihat di gambar 4.

1. **STRATEGI ALGORITMA**

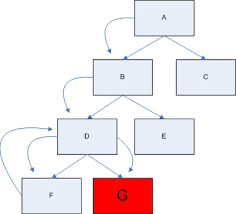
Batasan ruang lingkup yang penulis gunakan yaitu papan sudoku dengan ukuran 9x9 dan disetiap kotak pada papan tersebut dapat diisi dengan angka 1 s.d 9.



Gambar 2. **Pseudocode Sudoku Solver**

Berdasarkan gambar 2, tahap pertama dalam penerapan strategi algoritmanya adalah penulis menentukan inputan soal sudoku berupa array asosiatif seperti Gambar 1. Setelah itu membuat fungsi untuk mengoutputkan papan sudoku agar mempermudah pembacaan soal maupun solusi dari papan sudoku. Setelah itu penulis membuat fungsi dengan algoritma backtracking bernama sudoku\_backtrack(Board) yang mempunyai parameter bertipe array asosiatif sesuai dengan inputan soal sudoku. Fungsi ini akan bekerja secara rekursif. Namun untuk membangun fungsi ini, diperlukan 2 fungsi lagi yang bernama find\_empty yang berguna untuk menemukan kolom dan baris yang elemennya masih bernilai 0 dan mengembalikan data indeks baris dan kolomnya. Setelah itu terdapat fungsi untuk memvalidasi pengisian elemen sesuai dengan aturan sudoku. Proses yag dijalankan setelah ditemukan baris dan kolom yang elemennya masih bernilai 0, adalah melakukan proses pengisian dan validasi. Dan pada saat inilah algoritma backtracking mulai bekerja.

Terdapat 2 Skenario yang nantinya akan ada dalam proses *find\_empty*, pengisian, dan pengecekan. Skenario Pertama, jika telah ditemukan baris dan kolom yang elemenya bernilai angka 0, maka elemen – elemen pada baris tersebut akan mulai diisikan dengan angka 1 sampai 9 dan bergerak secara horizontal (mengisi elemen yang satu baris terlebih dahulu). Jika dilakukan pengecekan dan terbukti benar, maka akan berjalan menuju elemen yang terdapat pada baris berikutnya. Untuk memperjelas pemaparan diatas, silahkan lihat gambar 3.



Gambar 3. **Skenario Pohon Ruang Algoritma Backtracking**

Skenario kedua, apabila setelah dilakukan pengisian dan pengecekan terdapat kesalahan di tengah – tengah baris, maka elemen pada baris dan kolom tersebut akan di*reset* menjadi 0 dan Kembali ke node sebelumnya, diteruskan dengan mengganti nilai elemen yang berada pada iterasi sebelumnya dengan iterasi angka selanjutnya, dan lakukan pengecekan lagi mengarah ke elemen / node selanjutnya. Jika masih salah maka iterasi akan terus berjalan mundur sambil mereset elemen-elemen yang telah diganti. Dan jika benar maka data melanjutkan untuk mengisi dan mengecek elemen pada iterasi selanjutnya.

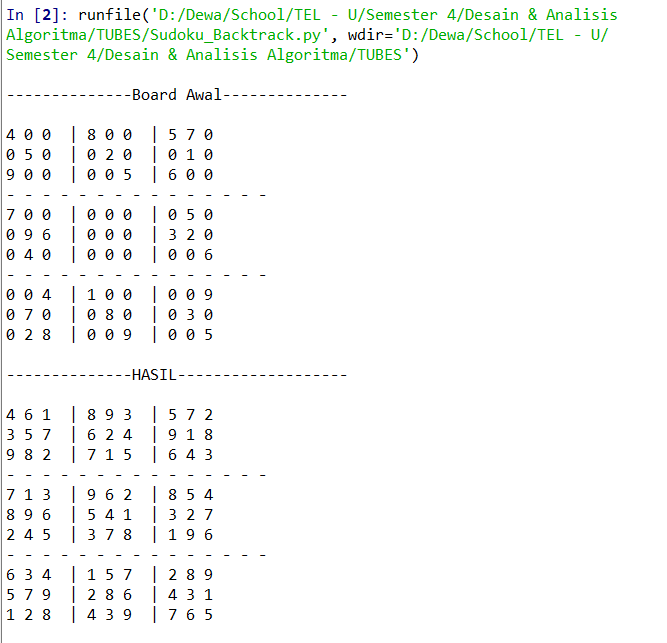
Kompleksitas Waktu dari algoritma backtracking ini adalah O(9(9\*9)), Untuk setiap elemen yang belum diisi, ada 9 kemungkinan pilihan. Kompleksitas waktu dari algoritma backtracking dan bruteforce memiliki nilai yang sama, namun yang membuat beda, pada algoritma backtracking akan ada beberapa pemangkasan awal sehingga waktu yang dieksekusi akan jauh lebih sedikit dibandingkan dengan algortima brute-force. Namun *upper bound* dari kedua algoritma tersebut masih sama.|  
Berikutnya, analisis efisiensi dari algoritma ini ialah O(9\*9) merepresentasikan setiap baris dan kolom yang dibutuhkan oleh matriks atau array 2 dimensi.

1. **FUNGSIONALITAS PROGRAM**

Terdapat 4 Fungsionalitas pada program ini yaitu:

1. Fungsi **find\_empty**(Board): Untuk mendapatkan Baris dan Kolom (yang kosong) dari soal Board Sudoku
2. Fungsi **valid**(Board, number, position): Untuk mengecek keabsahan suatu jawaban berdasarkan 3 peraturan sudoku diatas.
3. Fungsi **sudoku\_backtrack**(Board): Untuk menjawab elemen – elemen sudoku satu persatu dengan menggunakan fugsionalitas 1&2 dan dengan cara backtracking
4. Fungsi **Print\_Board**(Board): Untuk memberikan fungsi estetika pada output hasil.

**6. SCREEN SHOOT OUTPUT PROGRAM**



Gambar 4. **Screenshoot Output Program**

**7. REFERENSI**

[1] <https://www.blj.co.id/2015/06/06/mengenal-asal-permainan-teka-teki-angka-sudoku/>

[2] Syam, Umi Fadilah Wardati. 2007. “Perumusan Sudoku Dengan Solusi Unik Menggunakan Algoritma Pencabangan Dan Pembatasan (Branch And Bound)”. Dalam http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/.

[3] <https://www.blj.co.id/2015/06/06/mengenal-asal-permainan-teka-teki-angka-sudoku/>

[4] Teneng, Joko Purwadi, & Erick Kurniawan. 2010. “Penerapan Algoritma Backtracking Pada Permainan Math Maze”. Dalam http://ti.ukdw.ac.id/.

[5] Bakri, Addhal Huda. Analisis dan Implementasi AlgoritmaBacktracking pada Permainan Congklak. [Online]. Available : <http://repository.usu.ac.id>

[6] Widjaja, V. S., & Sudirman, D. Z. (2013, June). Implementasi Algoritma Backtracking Dengan Optimasi Menggunakan Teknik Hidden Single Pada Penyelesaian Permainan Sudoku. In Seminar Nasional Aplikasi Teknologi Informasi (SNATI) (Vol. 1, No. 1).