Penerapan Algoritma Runut-Balik (*Backtracking*) pada Permainan *Puzzle* Sudoku

Yoseph Alexander Siregar - 13520141
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520141@std.stei.itb.ac.id / yoseph141002@gmail.com

***Abstract*— Permainan Sudoku merupakan salah bentuk permainan *puzzle* yang sudah banyak dimainkan di kalangan masyarakat. Dalam permainan ini, pemain diberikan *grid* atau kisi-kisi berukuran 9 x 9 yang harus diisi dengan angka 1 – 9 dengan aturan tertentu. Algoritma *backtraing* merupakan salah satu algotima yang banyak digunakan untuk memecahkan suatu masalah atau persoalan. Algoritma *Backtracking* ini akan digunakan untuk menyelesaikan permainan Sudoku.**

Keywords—Sudoku, Backtracking, Puzzle

I. Pendahuluan

Setiap persoalan memiliki banyak alternatif untuk menyelesaikannya. Dalam pengaplikasiannya dalam dunia pemrograman, algoritma memiliki posisi yang sangat penting. Dengan algoritma, suatu persoalan dapat diselesaikan secara terstruktur dan sistematis. Karena suatu persoalan memiliki banyak alternatif dalam penyelesaiannya, begitu juga dengan algoritma yang memiliki banyak jenis yang masing-masing memiliki keunggulannya masing-masing.

Algoritma runut-balik atau lebih dikenal dengan Algoritma *Backtracking* merupakan salah satu dari sekian banyak algoritma dasar yang umum digunakan untuk memecahkan suatu persoalan.Dengan algoritma ini, suatu persoalan dapat diselesaikan secara efektif dan efisien, pilihan yang setelah dievaluasi tidak mengarah pada hasil yang diinginkan tidak akan diperiksa lebih lanjut. Algoritma ini banyak diterapkan dalam banyak permainan terutama permainan *puzzle* yang memiliki banyak kemungkinan pergerakan atau langkah.

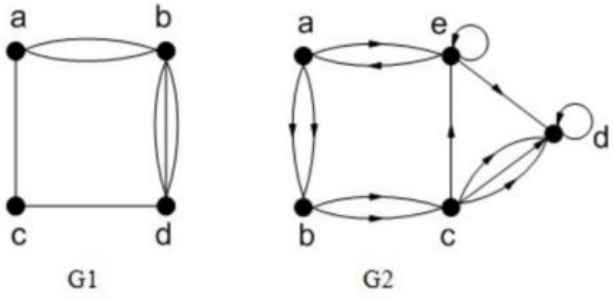
Sudoku merupakan permainan *puzzle* asal Jepang yang cukup populer. Sudoku dapat dimainkan kapan saja dan dimana saja. Permainan ini sangat mudah dimainkan karena tidak membutuhkan media atau peralatan yang banyak. Permainan ini hanya membutuhkan konsentrasi dan ketenangan. Dalam permainan ini diberikan suatu *grid* berbentuk 9 x 9 yang didalamnya dibagi menjadi 9 *grid* berukuran 3 x 3. Permainan dimulai dengan *grid* yang awal sudah terisi sebagian dan harus diisi dengan angka 1 – 9 pada bagian yang kosong dengan aturan yang ada.

II. Landasan Teori

A. Graph

Graf merupakan sebuah struktur diskrit yang terdiri atas simpul dan sisi. Simpul dan sisi tersebut merepresentasikan keterhubungan keduanya.

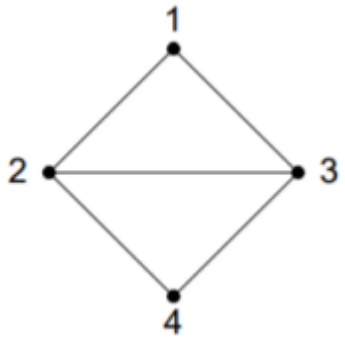
Graf dapat dibedakan berdasarkan 2 kategori, yaitu ada atau tidaknya arah pada graf tersebut.



Gambar 2.1 Graf G1 tak-bearah dan G2 berarah

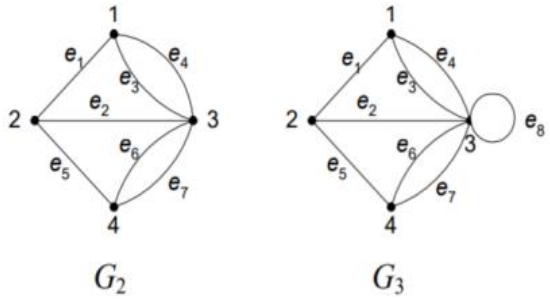
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

Lalu yang berikutnya adalah berdasarkan apakah graf memiliki gelang / sisi ganda atau tidak



Gambar 2.2 Graf sederhana

Sumber :
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf



Gambar 2.3 Graf tak-sederhana

Sumber :
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

B. Algoritma Backtracking

Algoritma *Backtracking* dapat dimaknai menjadi 2 hal, yaitu :

a. Metode penyelesaian persoalan (optimasi maupun non-optimasi) yang mangkus, terstruktur, dan sistematis.

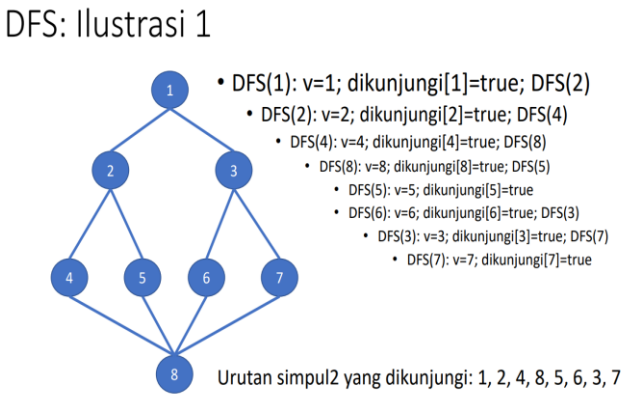
b. Sebuah fase pada proses traversal di algoritma DFS (*Depth-First-Search)*

Lalu seperti apakah algoritma DFS ? Algoritma DFS merupakan algoritma yang menyelesaikan suatu persoalan dengan merepresentasikannya dalam bentuk graf dan melakukan proses traversal dimana algoritma DFS akan mengunjungi tiap simpul yang ada pada graf secara sistematik. Pada algoritma DFS, penelusuran simpul dilakukan secara mendalam (*Depth).*

Dalam proses penyelesaian persoalan, representasi persoalan dalam graf memiliki dua pendakatan, yaitu graf statis dan graf dinamis. Graf statis merupakan graf yang sudah tersedia sebelum proses pencarian dilakukan sedangkan graf dinamis merupakan graf yang akan terbentuk saat proses pencarian dilakukan.

Pada algoritma DFS, pencarian dimulasi dari simpul akar *root* dengan algoritma sebagai berikut :

1. Kunjungi simpul *root*
2. Kunjungi salah satu simpul yang bertetangga dengan simpul *root.*
3. Ulangi proses DFS pada simpul tetangga tersebut.
4. Ketika mencapai satu simpul yang seluruh simpul tetangganya sudah dikunjungi, proses pencarian dirunut-balik (backtrack) ke simpul sebelumnya yang sudah dikunjungi yang memiliki simpul tetangga yang belum dikunjungi
5. Apabila sudah tidak lagi terdapat simpul yang dapat dikunjungi atau sudah ditemukan penyelesaian persoalan pada salah satu solusi maka proses pencarian sudah selesai

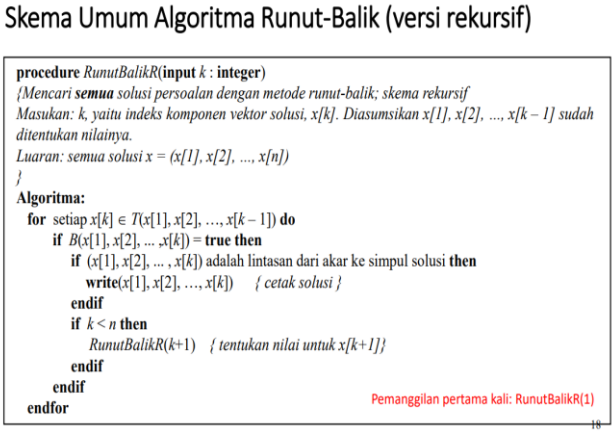


Gambar 2.4 Ilustrasi Penyelesaian dengan algoritma DFS

Sumber :
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf

Pada langkah ke-4, terjadi fase runut-balik dimana proses pencarian akan mundur ke simpul yang sudah dikunjungi sebelumnya. Fase inilah yang menerapkan proses *backtracking.* Algoritma *backtracking* dapat dimakanai sebagai bentuk yang lebih baik dari *exhaustive search*. Karena pada *exhaustive search*, tiap kemungkinan solusi akan dieksplorasi seluruhnya sedangkan pada *backtracking* akan dipangkas seluruh kemungkinan yang tidak mengarah pada solusi (*pruning*) sehingga proses pencarian akan lebih cepat dan efisien.

Algoritma *backtracking* pada umumnya diimplementasikan dengan menggunakan konsep rekursif. Berikut merupakan skema *pseudocode* untuk algoritma *Backtracking* versi rekursif



Gambar 2.5 Skema *backtracking* versi rekursif

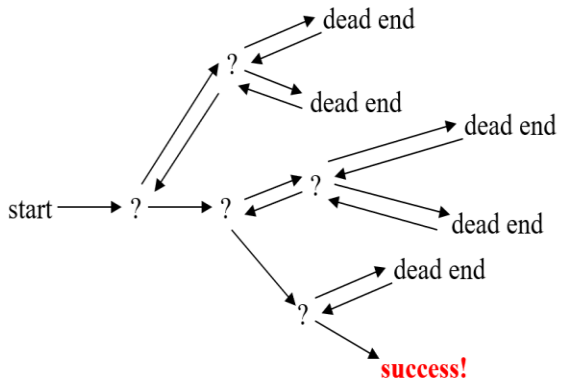
Sumber :
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf

Algoritma *backtracking* memiliki properti-properti umum dalam algoritmanya, yaitu solusi persoalan, fungsi pembangkit, dan fungsi pembatas

1. Solusi persoalan, direpresentasikan dengan *vector* seperti X = {x_1, x_2, x_3, ... ,x_n)
2. Fungsi pembangkit, untuk nilai x_i yang direpresentasikan dengan predikat T() yang akan membangkitkan nilai untuk x_i tersebut.
3. Fungsi pembatas, direpresentasikan dengan B() yang akan bernilai *true* apabila mengarah ke solusi

Pada algoritma *backtracking*, seperti DFS, pencarian juga dimulai dari simpul akar *root* dengan algoritma sebagai berikut :

1. Membangkitkan simpul-simpul status sehingga menghasilkan lintasan dari akar ke daun dengan aturan seperti pada algoritma DFS dan dinamakan simpul hidup
2. Simpul hidup yang sedang diperluas dinamakan simpul-E (Expand-node).
3. Jika hasil perluasan tidak menuju ke solusi, maka simpul-E tersebut "dimatikan" dan menjadi simpul mati dengan menerapkan fungsi pembatas
4. Jika perluasan lintasan berakhir dengan simpul mati, maka proses terjadi proses *backtrack* ke simpul pada sebelumnya dan dilanjutkan membangkitkan simpul anaknya
5. Selanjutnya simpul ini menjadi simpul-E yang baru.
6. Pencarian dihentikan bila telah sampai pada solusi yang diinginkan



Gambar 2.6 Ilustrasi algoritma *backtracking*

Sumber :
https://www.cis.upenn.edu/~matuszek/cit594-2009/Lectures/35-backtracking.ppt

C. Sudoku

Sudoku merupakan permainan *puzzle* asal Jepang yang cukup terkenal di kalangan pecinta *puzzle*. Permainan sudoku dikenal sebagai permainan yang mengasah otak karena konsentrasi dan kesabaran untuk menyelesaikannya.

			2	6		7		1
6	8			7			9	
1	9				4	5		
8	2		1				4	
		4	6		2	9		
	5				3		2	8
		9	3				7	4
	4			5			3	6
7		3		1	8			

Gambar 2.7 Permainan Sudoku

Sumber :
https://sandiway.arizona.edu/sudoku/examples.html

Pada permainan sudoku, diberikan *grid* berukuran 9 x 9 yang didalamnya berisi sembilan *grid* kecil berukuran 3 x 3. Pada grid tersebut, sudah diisikan angka 1 – 9 pada beberapa kotak dan kotak lainnya dibiarkan kosong. Tujuan dari permainan ini adalah mengisi angka 1 – 9 pada kotak kosong lainnya dengan sebuah aturan.

Permainan sudoku biasanya dibagi menjadi beberapa tingkatan berdasarkan kesulitannya. Kesulitannya didasarkan pada penempatan angka-angka awal yang diberikan.

Aturan yang ada pada permainan sudoku cukup sederhana tetapi juga dapat membuat permainan menjadi terasa sulit. Aturan yang ada permainan sudoku adalah sebagai berikut :

1. Tiap baris hanya boleh memiliki masing-masing satu dari angka satu hingga sembilan.
2. Tiap kolom hanya boleh memiliki masing-masing satu dari angka satu hingga sembilan.
3. Tiap *grid* kecil berukuran 3 x 3 hanya boleh memiliki masing-masing satu dari angka satu hingga sembilan.

4	3	5	2	6	9	7	8	1
6	8	2	5	7	1	4	9	3
1	9	7	8	3	4	5	6	2
8	2	6	1	9	5	3	4	7
3	7	4	6	8	2	9	1	5
9	5	1	7	4	3	6	2	8
5	1	9	3	2	6	8	7	4
2	4	8	9	5	7	1	3	6
7	6	3	4	1	8	2	5	9

Gambar 2.8 Solusi Permainan Sudoku

Sumber :
https://sandiway.arizona.edu/sudoku/examples.html

III. IMPLEMENTASI

Dalam menyelesaikan permainan sudoku menggunakan algoritma *backtracking*, aturan dari permainan sudoku akan menjadi fungsi pembatas yang akan mematikan ekspansi simpul yang tidak mengarah menuju solusi dari permainan sudoku.

Penggunaan algoritma *backtracking* dalam penyelesaian masalah sudoku adalah sebagai berikut :

1. Pada sel kosong, masukkan angka mulai dari 1 hingga 9 satu per satu.
2. Sebelum dimasukkan, diperiksa terlebih dahulu apakah angka tersebut dapat dimasukkan dengan mengecek sesuai dengan aturan sudoku
3. Hal ini terus dilakukan pada tiap sel kosong dan dicek apakah didapatkan solusi yang diinginkan
4. Apabila tidak, dilakukan runut-balik dengan mengganti angka sebelumnya menjadi angka berikutnya
5. Apabila angka dari 1 hingga 9 tidak menghasilkan solusi yang diinginkan, dilakukan runut balik lagi ke kotak sebelumnya.
6. Apabila setelah dilakukan runut balik hingga kotak pertama dan tidak didapatkan solusi setelah dicoba tiap angka dari 1 hingga 9, maka *puzzle* sudoku tersebut tidak memiliki solusi.

Dalam pengimplementasian program, *grid* awal akan direpresentasikan dengan matriks dan kotak kosong akan direpresentasikan dengan angka 0.

Dalam pengimplementasian program, terdapat beberapa fungsi dan prosedur yang dibuat untuk menyelesaikan permainan sudoku ini, yaitu :

1. Fungsi untuk mencari kotak yang kosong pada *grid* yang diberikan
2. Prosedur untuk memeriksa apakah angka yang diberikan sudah ada pada baris yang diberikan
3. Prosedur untuk memeriksa apakah angka yang diberikan sudah ada pada kolom yang diberikan
4. Prosedur untuk memeriksa apakah angka yang diberikan sudah ada pada *grid* ukuran 3 x 3 yang diberikan
5. Prosedur untuk memeriksa apakah posisi kotak kosong bisa dimasukkan dengan memanfaatkan tiga prosedur sebelumnya
6. Fungsi utama untuk menyelesaikan persoalan permainan *puzzle* yang diberikan

Fungsi utama yang ada diimplementasikan menggunakan konsep rekursif. Pada fungsi utama, pertama akan dicari kotak yang kosong. Pada kotak kosong tersebut akan dimasukkan angka 1 hingga 9 dengan iterasi dengan dilakukan pengecekan terlebih dahulu apakah angka tersebut dapat dimasukkan. Lalu setelah dimasukkan, diterapkan konsep rekursif dengan memanggil kembali fungsi utama terhadap *grid* baru setelah angka dimasukkan.

```
def solver(grid):
    l =[0, 0] # dummy empty box

    if(not findEmpty(grid, l)):
        return True

    row = l[0]
    col = l[1]

    for num in range(1, 10):

        if(isSafe(grid,row, col, num)):

            grid[row][col]= num

            if(solver(grid)):
                return True

            grid[row][col] = 0

    return False
```

Gambar 3.1 *Snippet* Fungsi Utama

Sumber : Arsip Pribadi

IV. Uji Coba

Pada bagian ini, akan ditampilkan hasil uji coba penyelesaian beberapa permainan sudoku dengan menggunakan algoritma backtracking.

Persoalan sudoku yang digunakan dalam uji coba ini dan hasil akhirnya diambil dari permainan sudoku yang didapat dari https://sandiway.arizona.edu/sudoku/examples.html

```
grid =[[0, 0, 0, 2, 6, 0, 7, 0, 1],
       [6, 8, 0, 0, 7, 0, 0, 9, 0],
       [1, 9, 0, 0, 0, 4, 5, 0, 0],
       [8, 2, 0, 1, 0, 0, 0, 4, 0],
       [0, 0, 4, 6, 0, 2, 9, 0, 0],
       [0, 5, 0, 0, 0, 3, 0, 2, 8],
       [0, 0, 9, 3, 0, 0, 0, 7, 4],
       [0, 4, 0, 0, 5, 0, 0, 3, 6],
       [7, 0, 3, 0, 1, 8, 0, 0, 0]]
```

Gambar 4.1 Representasi sudoku pada Gambar 2.7

Sumber : Arsip Pribadi

```
Sudoku berhasil diselesaikan :

4 3 5 2 6 9 7 8 1
6 8 2 5 7 1 4 9 3
1 9 7 8 3 4 5 6 2
8 2 6 1 9 5 3 4 7
3 7 4 6 8 2 9 1 5
9 5 1 7 4 3 6 2 8
5 1 9 3 2 6 8 7 4
2 4 8 9 5 7 1 3 6
7 6 3 4 1 8 2 5 9
```

Gambar 4.2 Hasil Persoalan Sudoku sesuai dengan Gambar 2.8

Sumber : Arsip Pribadi

```
grid =[[1, 0, 0, 4, 8, 9, 0, 0, 6],
       [7, 3, 0, 0, 0, 0, 0, 4, 0],
       [0, 0, 0, 0, 0, 1, 2, 9, 5],
       [0, 0, 7, 1, 2, 0, 6, 0, 0],
       [5, 0, 0, 7, 0, 3, 0, 0, 8],
       [0, 0, 6, 0, 9, 5, 7, 0, 0],
       [9, 1, 4, 6, 0, 0, 0, 0, 0],
       [0, 2, 0, 0, 0, 0, 0, 3, 7],
       [8, 0, 0, 5, 1, 2, 0, 0, 4]]
```

Gambar 4.3 Representasi persoalan Sudoku

Sumber : Arsip Pribadi

```
Sudoku berhasil diselesaikan :

1 5 2 4 8 9 3 7 6
7 3 9 2 5 6 8 4 1
4 6 8 3 7 1 2 9 5
3 8 7 1 2 4 6 5 9
5 9 1 7 6 3 4 2 8
2 4 6 8 9 5 7 1 3
9 1 4 6 3 7 5 8 2
6 2 5 9 4 8 1 3 7
8 7 3 5 1 2 9 6 4
```

Gambar 4.4 Hasil Persoalan Sudoku

Sumber : Arsip Pribadi

```
grid =[[0, 2, 0, 6, 0, 8, 0, 0, 0],
       [5, 8, 0, 0, 0, 9, 7, 0, 0],
       [0, 0, 0, 0, 4, 0, 0, 0, 0],
       [3, 7, 0, 0, 0, 0, 5, 0, 0],
       [6, 0, 0, 0, 0, 0, 0, 0, 4],
       [0, 0, 8, 0, 0, 0, 0, 1, 3],
       [0, 0, 0, 0, 2, 0, 0, 0, 0],
       [0, 0, 9, 8, 0, 0, 0, 3, 6],
       [0, 0, 0, 3, 0, 6, 0, 9, 0]]
```

Gambar 4.5 Representasi persoalan Sudoku

Sumber : Arsip Pribadi

```
Sudoku berhasil diselesaikan :

1 2 3 6 7 8 9 4 5
5 8 4 2 3 9 7 6 1
9 6 7 1 4 5 3 2 8
3 7 2 4 6 1 5 8 9
6 9 1 5 8 3 2 7 4
4 5 8 7 9 2 6 1 3
8 3 6 9 2 4 1 5 7
2 1 9 8 5 7 4 3 6
7 4 5 3 1 6 8 9 2
```

Gambar 4.6 Hasil Persoalan Sudoku

Sumber : Arsip Pribadi

V. Kesimpulan

Berdasarkan hasil uji coba yang dilakukan di atas, algoritma *backtracking* merupakan algoritma yang dapat menyelesaikan permainan Sudoku dengan baik.

LINK VIDEO

https://drive.google.com/file/d/1Eb8G-Ld7tSuccjZlFLZiJRLUJgZWknNy/view?usp=sharing

UCAPAN TERIMA KASIH

Penulis pertama-tama mengucapkan puji syukur dan terima kasih kepada Tuhan Yang Maha Esa atas segala berkah dan bimbingannya sehingga makalah ini dapat terselesaikan dengan baik. Tidak lupa juga penulis ingin mengucapkan terima kasih kepada seluruh asisten dan dosen mata kuliah Strategi Algoritma, terutama kepada Dr. Ir. Rinaldi Munir, M.T selaku dosen pengampu Kelas K03, yang sudah mengajarkan dan mengarahkan penulis selama pembelajaran dalam satu semester.

REFERENSI

[1] Munir, Rinaldi. 2021. Graf (Bag. 1). https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf2020-Bagian1.pdf . Diakes pada 20 Mei 2022

[2] Munir, Rinaldi. 2021. Breadth First Search(BFS) dan Depth First Search(DFS)(Bagian 1). https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf. Diakes pada 20 Mei 2022

[3] Munir, Rinaldi. 2021. Algoritma runut – balik (backtracking)(Bagian 1) . https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf. Diakes pada 20 Mei 2022

[4] GeeksforGeeks. 2022. Backtracking Algorithms. https://www.geeksforgeeks.org/backtracking-algorithms/. Diakses pada 20 Mei 2022

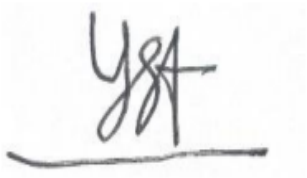
[5] Sudoku. 2022. https://sudoku.com/. Diakses pada 20 Mei 2022

[6] SandiwayArizona.2022.Examples Puzzles and Solutions. https://sandiway.arizona.edu/sudoku/examples.html. Diakses pada 20 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, 20 Mei 2022



13520141 Yoseph Alexander Siregar