Penyelesaian Sudoku dengan Algoritma Brute Force

Gregorius Moses Marevson – 13520052
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520052@std.stei.itb.ac.id

***Abstract*—Sudoku adalah permainan berbasis logika yang bertujuan untuk mengisi semua bagian yang kosong dengan angka (1-9). Makalah ini berisi pencarian solusi permainan Sudoku dengan menggunakan algoritma Brute Force. Algoritma Brute Force adalah algoritma sederhana dimana akan dicoba untuk mencari seluruh kemungkinan angka. Penerapan algoritma ini tentu akan memakan komplesitas waktu yang besar sebagai hasil dari pemilihan algoritma yang sangat sederhana.**

Kata Kunci—Sudoku; Brute Force; algoritma; solusi; permainan; kompleksitas waktu;

I. PENDAHULUAN

Sudoku adalah permainan asah otak yang berasal dari Jepang. Pemain akan diberikan kotak berukuran 9 x 9 dimana ada bagian yang sudah berisi dan bagian yang masih kosong. Tujuan permainan ini adalah untuk mengisi bagian yang masih kosong dengan mengikuti aturan tertentu.

Sudah banyak algoritma yang berhasil ditemukan untuk menyelesaikan Sudoku. Penulis ingin mencoba mengimplementasikan algoritma Brute Force dalam pencarian solusi permainan Sudoku.

Implementasi ini bisa menunjukkan kalau algoritma Brute Force hampir bisa dipergunakan untuk mencari solusi dari segala masalah, tetapi atas imbalan kompleksitas waktu yang tinggi.

II. DASAR TEORI

A. Permainan Sudoku

Sudoku adalah permainan logika yang berasal dari Jepang. Permainan ini menyajikan sekumpulan kotak berukuran 9 x 9. Kotak – kotak ini ada yang sudah berisi dan ada yang masih kosong. Tujuan dari permainan ini adalah untuk mengisi bagian yang masih kosong dengan suatu angka sembari mengikuti serangkaian aturan.

Aturan dalam permainan Sudoku sangatlah sederhana, yaitu tidak boleh ada angka yang sama dalam satu baris, satu kolom, dan satu kotak kecil berukuran (3 x 3). Kotak kecil ini biasanya ditandai dengan garis lebih tebal sebagaimana bisa dilihat pada gambar berikut.

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

Gambar 1 Permainan Sudoku

Permainan ini memiliki berbagai variasi seperti Logi-5 (Sudoku 5 x 5 dengan pentomino sebagai pembatas), Mini Sudoku (Sudoku 6 x 6 dengan batasan kotak 3 x 2), Killer Sudoku (Gabungan Sudoku dan Kakuro), Alphabetical Sudoku (Sudoku dengan huruf), dan masih banyak lagi.

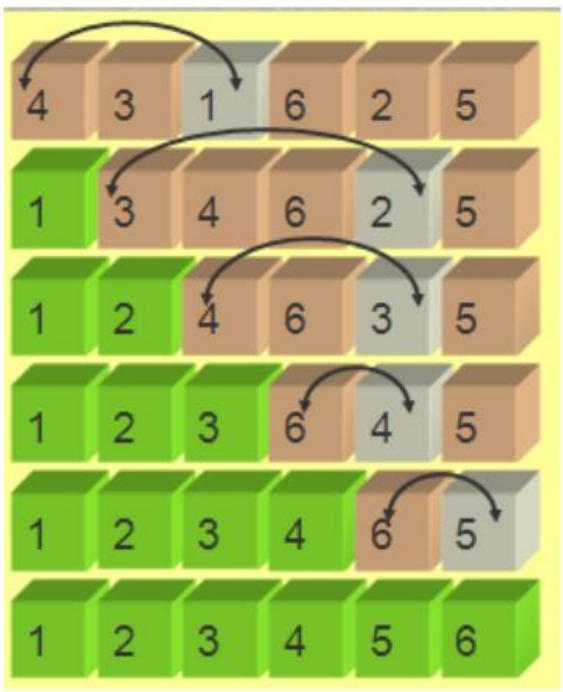
Perkembangan permainain Sudoku saat ini cukup pesat, sehingga permainan ini sudah sering diperlombakan.

B. Algoritma Brute Force

Algoritma Brute Force adalah algoritma *straight-forward* dimana dalam implementasinya hanya dibutuhkan kemampuan komputasi computer. Algoritma ini banyak dipergunakan sebagai algoritma pengganti apabila masih belum yakin terhadap penerapan algortima yang lebih kompleks. Algoritma ini umumnya bisa menyelesaikan hampir semua solusi selama daya komputasinya masih memadai.

Penerapan algoritma ini biasanya dilakukan dengan mengecek seluruh kemungkinan secara berurutan. Permasalahnya muncul dalam penerapan algoritma ini untuk menyelesaikan Sudoku. Dalam Sudoku terdapat sekitar 10^{21} kombinasi angka yang mungkin. Penerapan algoritma ini secara naif pasti akan memakan daya komputasi yang besar.

Salah satu contoh penerapan algoritma Brute Force adalah dalam algoritma Bubble sort dan Selection Sort.



Gambar 2 Selection Sort

Seperti terlihat pada ilustrasi di atas, langkah kerja dari Selection Sort adalah mencari nilai terkecil dari pool array (dalam kasus ini juga menggunakan Brute Force), lalu memindahkannya ke posisi terawal array. Algoritma ini akan memakan banyak waktu dengan kompleksitas waktu $O(n^2)$. Pencarian solusi dengan algoritma ini selalu menghasilkan hasil yang tepat tapi akan memakan banyak sekali sumber daya.

Adapun keuntungan dan kelemahan dari pendekatan Brute Force adalah sebagai berikut :

Keuntungan :

1. Karena seluruh kemungkinan solusi diiterasi, maka solusi paling optimal pasti bisa ditemukan
2. Pendekatan Brute Force bisa diaplikasikan untuk persolan komputasi secara luas
3. Pendekatan Brute Force sangat membantu untuk menyelesaikan persoalan komputasi skala kecil
4. Pendekatan Brute Force sangat sederhana sehingga tidak diperlukan prior knowledge dalam pengimplementasiannya

Kelemahan :

1. Algoritma Brute Force sangatlah tidak efisien, apalagi untuk pool data yang besar
2. Mencari solusi yang tepat menggunakan algoritma ini cenderung lambat
3. Algoritma ini hanya mengandalkan kemampuan komputasi computer tanpa mempedulikan desain rancangan algoritma yang baik
4. Algoritma ini tidak konstruktif maupun kreatif apabila dibandingkan dengan paradigma desain yang lain

III. Implementasi dan Pengujian

Implementasi algoritma Brute Force dalam pencarian solusi Sudoku dilakukan dengan langkah - langkah sebagai berikut:

1. Temukan posisi baris, kolom yang masih kosong
2. Iterasi setiap angka dari 1 – 9
3. Untuk setiap iterasi, apabila angka tersebut bisa ditempatkan secara legal pada posisi tersebut, lanjutkan iterasi ke posisi berikutnya.
4. Apabila iterasi sudah selesai sampai angka ke 9 dan tidak ada nomor yang legal, hapus angka pada posisi itu, lalu mundur satu langkah ke posisi sebelumnya.
5. Pencarian dilakukan sampai ditemukan satu buah solusi atau tidak ditemukannya satu solusi legal.

Dalam pengimplementasian algoritma ini dibutuhkan suatu fungsi pembantu yang bertugas untuk mengecek apakah suatu angka valid pada posisi tertentu.

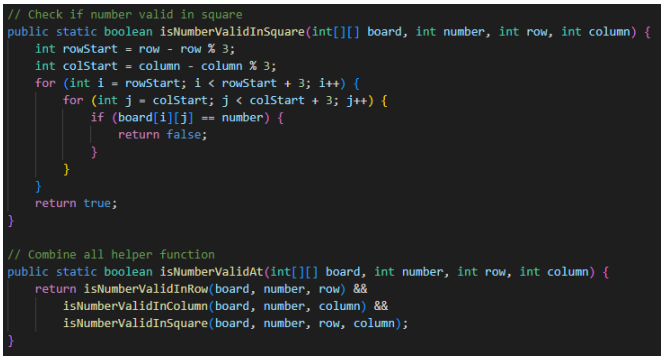
A. Fungsi Pembantu

Fungsi pembantu yang diciptakan akan membantu untuk melakukan validasi nomor di posisi tertentu. Validasi dilakukan pada 3 arah yaitu menelusuri baris, menelurusi kolom, dan menelurusi satu kotak. Implementasi fungsi pembantu ini cukup mudah karena hanya dilakukan iterasi secara berurut, jika ditemukan nomor yang sama akan direturn nilai false yang berarti nomor tidak valid.

Berikut dilampirkan snippet atas fungsi pembantu yang telah dibuat:

```
// Check if number is valid in row
public static boolean isNumberValidInRow(int[][] board, int number, int row) {
    for (int i = 0; i < 9; i++) {
        if (board[row][i] == number) {
            return false;
        }
    }
    return true;
}

// Check if number is valid in column
public static boolean isNumberValidInColumn(int[][] board, int number, int column) {
    for (int i = 0; i < 9; i++) {
        if (board[i][column] == number) {
            return false;
        }
    }
    return true;
}
```



```java
// Check if number valid in square
public static boolean isNumberValidInSquare(int[][] board, int number, int row, int column) {
    int rowStart = row - row % 3;
    int colStart = column - column % 3;
    for (int i = rowStart; i < rowStart + 3; i++) {
        for (int j = colStart; j < colStart + 3; j++) {
            if (board[i][j] == number) {
                return false;
            }
        }
    }
    return true;
}

// Combine all helper function
public static boolean isNumberValidAt(int[][] board, int number, int row, int column) {
    return isNumberValidInRow(board, number, row) &&
        isNumberValidInColumn(board, number, column) &&
        isNumberValidInSquare(board, number, row, column);
}
```

Gambar 3 Implementasi Helper Function

Pada implementasi terlampir, pengecekan secara menjulur pada kolom ataupun baris proses kerjanya hampir sama. Dilakukan iterasi sepanjang baris atau kolom yang diinput. Apabila ditemukan angka yang sama akan langsung dilakukan return nilai false. Sebaliknya, apabila angka yang diinput tidak ditemukan pada baris atau kolom akan direturn nilai true.

Pengimplemtasian validasi pada kotak memiliki sedikit perbedaan dibandingkan dengan fungsi pembantu sebelumnya. Fungsi ini diinisiasi dengan pencarian lokasi kotak kecil tempat angka yang diinput berada. Karena lokasi kotak kecil selalu berada dalam rentang 3 kotak, maka lokasi ujung kiri atas kotak kecil bisa dihitung menggunakan rumus berikut :

$$\text{indexAwal} = \text{posisiAngka} - \text{posisiAngka} \bmod 3$$

Rumus ini berlaku dengan asumsi bahwa index dimulai dari angka 0 hingga angka 8.

Dilakukan iterasi terhadap kotak kecil 3 x 3 dimulai dari ujung kiri atas, sampai kiri bawah. Apabila angka tersebut ditemukan berada di dalam kotak kecil maka akan direturn nilai false, sedangkan apabila iterasi sudah selesai dan tidak ditemukan maka akan direturn nilai true.

Validasi angka yang dilakukan bisa disatukan menjadi satu fungsi yang manggabungkan hasil dari ketiga fungsi pembantu sebelumnya. Seperti terlihat dalam snippet fungsi validasi ini akan mengeluarkan nilai true apabila semua fungsi lain mengeluarkan nilai true. Fungsi ini akan mengeluarkan nilai false apabila terdapat 1 fungsi yang mengeluarkan nilai false.

Kompleksitas algoritma untuk fungsi pembantu ini cukup rendah yaitu sebesar 3 * 9 = 27 pengecekan. Pengimplementasian fungsi pembantu ini juga menggunakan algoritma Brute Force karena sangat straight-forward.

B. Pseudocode

Berikut adalah pseudocode sebagai acuan dalam pengimplemntasian Sudoku Solver :

```
function solve(input board : matrix of integer) -> bool
ALGORITMA
    for row <- 0 to row = 9 - 1
        for column <- 0 to column = 9 - 1
            if board[row][column] = 0 then
                for number <- 1 to number = 9
                    if validate(number) then
                        board[row][column] <- number
                        if solve(board) then
                            return true
                        else
                            return false
```

Gambar 4 Pseudocode

Dari pseudocode bisa dilihat cara kerja program adalah sebagai berikut :

1. Dilakukan pencarian posisi yang masih kosong (dalam pseudocode ditandai dengan board yang bernilai 0
2. Setelah ditemukan posisi board yang kosong, akan dilakukan Brute Force dari angka 1 hingga 9
3. Untuk setiap iterasi dilakukan validasi
4. Bila angka tersebut valid, isi board dengan angka tersebut
5. Lakukan kembali function solve pada board yang sama (setelah dilakukan pengisian)
6. Apabila tidak ada angka yang valid, program akan mereturn nilai false yang bisa dipakai untuk melakukan backtrack pada program rekursif atau untuk menginformasikan kalau puzzle tersebut tidak mempunyai solusi.

Pseudocode tersebut masih mempunyai beberapa kelemahan, seperti :

1. Tidak adanya validasi apabila sudoku yang disediakan layak untuk diselesaikan atau tidak. Kasus ini terjadi apabila board yang dimasukkan sebagai input adalah board yang sejak awal sudah terselesaikan (tidak ada posisi kosong) tetapi atas nilai yang salah.
2. Apabila iterasi sudah dilakukan sampai akhir tidak adanya return nilai. Hal ini bisa menyebabkan error setelah memasuki bagian rekursif

Kelemahan pada poin 1 bisa diabaikan dengan asumsi bahwa board yang dimasukkan sejak awal sejak awal punya setidaknya satu posisi kosong. Fungsi ini bertanggung jawab untuk mencari solusi pada posisi kosong, bukan untuk melakukan pengecekan apakah board benar atau tidak.

Kelemahan pada poin 2 bisa diatasi dengan menambahkan return value setelah *for loop* berakhir. Return value ini akan menjadi pengakhir fungsi apabila tidak ditemukan solusi.

C. Implementasi

Penulis mengimplementasikan program ini dalam Bahasa Java dengan snippet solver sebagai berikut :

```
public static boolean solve(int[][] board) {
    for (int row = 0; row < 9; row++) {
        for (int column = 0; column < 9; column++) {
            if (board[row][column] == 0) {
                for (int number = 1; number <= 9; number++) {
                    if (isNumberValidAt(board, number, row, column)) {
                        board[row][column] = number; // Fill if empty

                        if (solve(board)) {
                            return true;
                        } else {
                            board[row][column] = 0; // Emptying if fail
                        }

                    }
                }
                return false;
            }
        }
    }
    return true; // because the board is already full
}
```

Gambar 5 Implementasi Solver

Implementasi program dilakukan berdasarkan pseudocode yang telah dilampirkan. Cara kerja program sama seperti yang sudah tertulis pada pseudocode. Implementasi ini disertai tambahan return nilai false setelah for loop sebagai solusi atas permasalahan yang muncul pada pseudocode, dan tambahan return nilai true apabila tidak ada nilai kosong pada board. Implementasi program ini masih mempunyai kelemahan karena tidak adanya pengecekan apakah solusi akhir sudah tepat atau tidak. Hanya ada pengecekan angka apa sajakah yang mungkin memenuhi board tanpa menyalahi aturan Sudoku.

Kompleksitas waktu algoritma ini bisa dibilang cukup mahal. Worst case dihasilkan dari input board yang seluruhnya kosong. Hal ini disebabkan oleh pengiterasian seluruh nilai yang mungkin mulai dari 1 dan rekursifme program. Sebaliknya program ini memiliki kompleksitas ruang yang rendah karena hanya mengolah satu board.

Implementasi Main Program dengan tambahan procedure drawBoard adalah sebagai berikut :

```
public static void main(String[] args) {

    int[][] board = {
        {7,6,2,0,0,9,0,1,0},
        {3,0,4,0,0,0,0,0,2},
        {8,0,0,0,5,6,4,0,7},
        {0,9,7,1,0,0,0,0,0},
        {0,0,6,0,0,8,7,0,3},
        {5,8,0,0,6,0,2,0,1},
        {0,7,0,9,0,0,3,0,0},
        {9,0,1,4,2,0,0,8,6},
        {2,4,5,0,0,3,0,0,0}
    };

    if (solve(board)) {
        System.out.println(x: "This Sudoku is solvable with solution as follow :");
    } else {
        System.out.println(x: "This Sudoku is unsolvable");
    }

    drawBoard(board);

}
```

```
public static void drawBoard(int[][] board) {
    String line;
    for (int i = 0; i < 9; i++) {
        line = "";
        if (i % 3 == 0) {
            System.out.println(x: "+---+---+---+");
        }
        for (int j = 0; j < 9; j++) {
            if (j % 3 == 0) {
                line += "|";
            }
            line += board[i][j];
        }
        System.out.println(line + "|");
    }
    System.out.println(x: "+---+---+---+");
}
```

Gambar 6 Implementasi main dan drawBoard

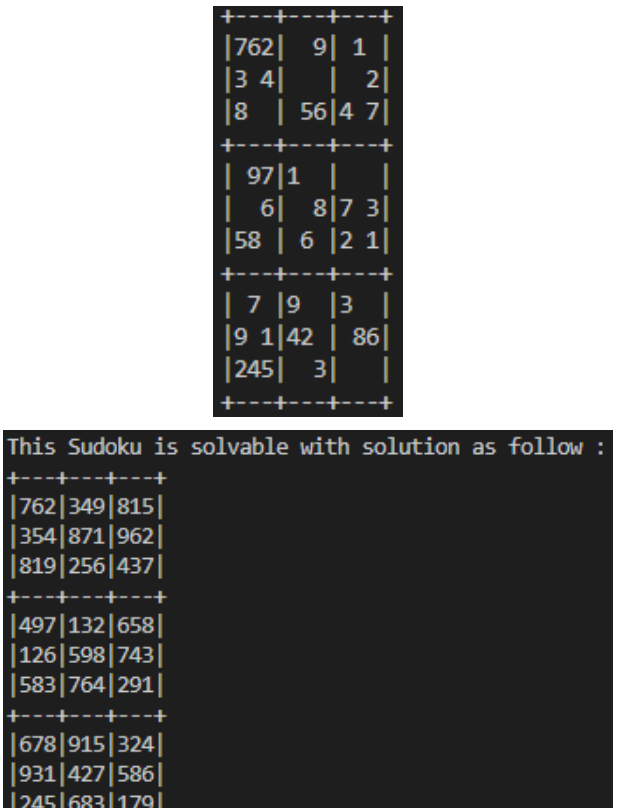
Program main adalah program yang akan dijalankan setelah program Java di compile. Seperti terlihat pada implementasi program ini dimulai dengan pengisian board secara manual. Kemudian akan dijalankan program solve yang akan mengubah seluruh board apabila berhasil atau akan kembali mengembalikan board ke kondisi awal apabila gagal.

Procedure drawBoard akan menggambarkan board ke console. Hal ini dilakukan untuk mengecek hasil dari program yang dibuat.

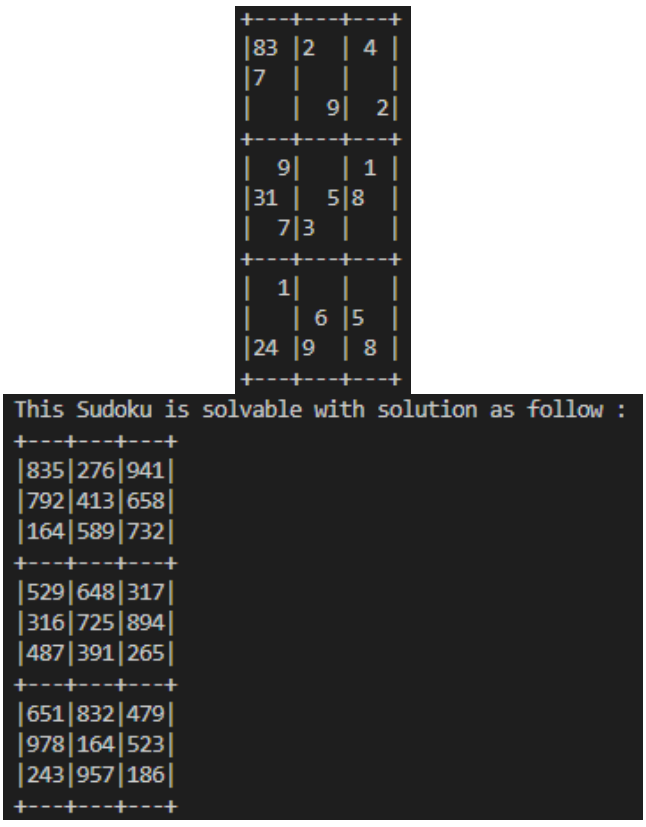
D. Pengujian

Program ini diuji dengan beberapa test case. Test case diambil dari sudoku online dengan beberapa level yaitu easy, medium, hard, expert dan evil. Yang menjadi pembeda dari tiap level adalah jumlah dari kotak yang masih kosong. Semakin banyak kotak yang kosong semakin sulit puzzle ini terselesaikan.

Berikut adalah beberapa dari test case yang diujikan :



Gambar 7 Test Case Easy



Gambar 8 Test Case Evil

Pada gambar dapat dilihat bahwa test case easy dan evil bisa diselesaikan dengan baik. Hal ini menunjukkan kalau program ini berjalan dengan semestinya.

IV. KESIMPULAN

Melaui pembuatan program ini bisa disimpulkan :

- Algoritma Brute Force adalah algoritma yang serba bisa biarpun mengorbankan komplesitas waktu (juga kompleksitas ruang untuk kasus lain)
- Permainan Sudoku bisa diselesaikan dengan algoritma Brute Force secara rekursif dengan kompleksitas waktu yang tinggi dan kompleksitas ruang yang rendah

Adapun saran dalam pengerjaan makalah ini adalah :

- Program yang dibuat bisa dilengkapi lagi dengan fungsi checker, dimana akan dilakukan pengecekan untuk semua angka pada board
- Program yang dibuat bisa ditambahi dengan metode input yang lebih efisien seperti melalui file lain ataupun input berupa string

VIDEO LINK AT YOUTUBE

Karena kesibukan penulis, penulis tidak sempat mengerjakan bonus video.

UCAPAN SYUKUR

Penulis berterimakasih atas berkat Tuhan YME karena penulis bisa menyelesaikan tugas makalah ini dengan tepat waktu. Penulis juga berterimakasih atas seluruh pengajaran baik yang diberikan oleh seluruh dosen Strategi Algoritma. Pengajaran yang beliau sampaikan telah menjadi dasar ilmu dalam penyelesaian tugas ini. Akhir kata, penulis menyampaikan terimakasih sebesar-sebesarnya atas segala dukungan yang telah penulis terima.

REFERENSI

Adapun referensi yang digunakan dalam penulisan makalah ini adalah sebagai berikut:

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf – Diakses pada 23 Mei 2022

[2] https://www.technologyreview.com/2012/01/06/188520/mathematicians-solve-minimum-sudoku-problem/ - Diakses pada 23 Mei 2022

[3] https://sudoku.com/ - Diakses pada 23 Mei 2022

[4] https://www.theguardian.com/media/2005/may/15/pressandpublishing.usnews#:~:text=The%20Sudoku%20story%20began%20in,in%20each%20row%20or%20column. – Diakses pada 23 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

Gregorius Moses Marevson 13520052