Aplikasi Algoritma Runut Balik dalam Pembangkitan Elemen Awal Permainan Sudoku

Muhammad Farhan Kemal / 13513085[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13513085@std.stei.itb.ac.id

***Abstract*—Sudoku merupakan sebuah permainan menyusun angka dari satu hingga sembilan dalam *board* yang berukuran 9x9 kotak. Aturan permainan ini secara umum adalah penyusunan setiap angka dalam kolom 9x9 tersebut tidak boleh sama dalam setiap baris, kolom, dan dalam upa-*board* yang berukuran 3x3 kotak. Setiap *game* akan dimulai, *board* dalam kondisi beberapa kotak sudah berisi angka yang tidak menyalahi aturan Sudoku itu sendiri. Jumlah angka yang terisi dalam board tergantung dalam tingkat kesulitan game yang dipilih diawal game. Dalam sistem permainan Sudoku ini, setiap kotak dalam board telah memiliki nilai yang dibangkitkan pada saat game akan dimulai. Agar pembangkitan nilai pada setiap kotak memenuhi aturan permainan Sudoku ada banyak cara yang bisa diterapkan, salah satunya dengan menggunakan algoritma runut balik atau yang biasa disebut *backtrack*. Dalam algoritma backtrack, kita mencoba beberapa sekuens keputusan, sampai Anda menemukan sekuens yang "bekerja".**

***Index Terms*—Sudoku, *backtrack*, .**

I. PENDAHULUAN

Kehidupan manusia di bumi bersifat dinamis. Lingkungan tempat manusia hidup juga dinamis. Hal secara tidak langsung mengakibatkan cara berpikir manusia yang dinamis dan selalu berkembang sesuai dengan lingkungan dan kebutuhannya. Begitupun dengan cara berpikir manusia mengenai masalah komputasi dan pemecahan masalah. Telah banyak algoritma yang ditemukan manusia untuk memecahkan suatu masalah. Dimulai dari algoritma brute-force yang menyolusikan suatu masalah secara langsung tanpa menggunakan langkah yang efisien. Kemudian ditemukan algoritma-algoritma yang lebih mangkus dan sangkil sesuai dengan fungsi algoritma tersebut dan kebutuhan pengguna, contohnya algoritma *Greed* , algoritma *branch and bound* yang menggunakan strategi berbasis pencarian dan pohon ruang status, algoritma *divide and conquer* yang mengandalkan penyolusian atas-bawah, algoritma runut balik yang merupakan penyempurnaan dari *exhaustive search* yang merupakan algoritma *brute-force*, dan masih banyak lagi algoritma yang lain yang merupakan pengembangan dari algoritma yang lain ataupun algoritma lain yang ditemukan karena adanya kebutuhan dari user.

Umumnya suatu algoritma mampu menyolusikan banyak kondisi dan permasalahan. Mulai dari masalah yang sederhana hingga masalah yang butuh penyolusian dengan mengombinasikan banyak algoritma. Salah satu permasalahan yang akan dibahasa pada makalah ini adalah pembangkitan elemen awal pada permainan Sudoku.

Permainan Sudoku adalah permainan klasik yang menggunakan angka satu hingga Sembilan dan menempatkan angka-angka tersebut dalam sebuah *board* berukuran 9x9. Aturan dari permainan adalah tidak ada angka yang sama dalam suatu baris, kolom, dan upa-board. Dalam sebuah aplikasi Sudoku, setiap kotak telah dibangkitkan nilainya terlebih dahulu. Dalam pembangkitan nilai setiap kotak dalam *board* digunakan algoritma runut balik hingga didapatkan solusi yang memenuhi segala aturan dalam permainan Sudoku tersebut.

II. ALGORITMA RUNUT BALIK

Istilah *backtracking* pertama kali diperkenalkan oleh D.H. Lehmer pada tahun 1950 dan terdapat juga tokoh-tokoh seperti R.J.Walker, Golomb, Baumert yang menyjikan uraian umum tentang *backtracking*.

Algoritma *backtracking* adalah algoritma pencarian solusi yang berbasis *DFS (Depth First Search)*. Algoritma *backtracking* banyak diterapkan untuk program games:

1. Permainan *tic-tac-toe*
2. Menemukan jalan keluar dari sebuah *maze*
3. Permainan *Crossword puzzle*
4. Catur

Algoritma runut balik merupakan perbaikan dari algoritma *brute force*. Pada algoritma *brute force* semua kemungkinan solusi dieksplorasi satu per satu sedangkan pada algoritma runut balik hanya pilihan yang mengarah ke solusi yang dieksplorasi, pilihan yang tidak mengarah ke solusi tidak dipertimbangkan kembali.

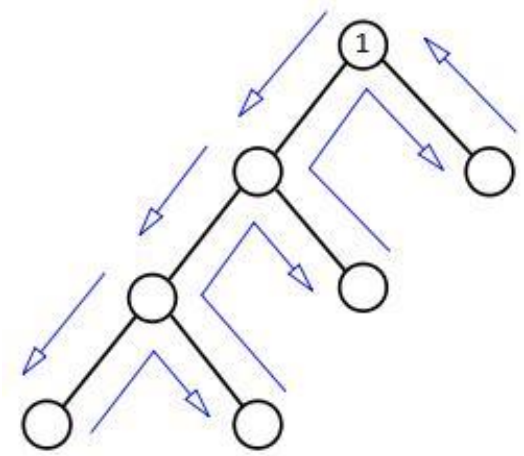
2.1. Properti Umum Algoritma Runut Balik

a. Solusi persoalan Solusi dinyatakan dalam bentuk vektor dengan tupel: $X = (x_1, x_2, x_3,...,x_n), x_i \in S_i$.
Mungkin terjadi $S_1 = S_2 = ... = S_n$ Contoh: $S_i = \{0,1\}$, $x_i = 0$ atau $x_i = 1$

b. Fungsi pembangkit Fungsi pembangkit nilai x_k Dinyatakan dalam predikat $T(k)$ dimana $T(k)$ membangkitkan nilai untuk x_k, yang merupakan komponen vektor solusi.

c. Fungsi pembatas Dinyatakan dalam predikat : $B(x_1, x_2,...,x_k)$ B bernilai benar jika $(x_1, x_2,...,x_k)$ mengarah ke solusi. Jika benar, maka pembangkitan nilai untuk $x_k + 1$ dilanjutkan, tetapi jika false, maka $(x_1, x_2,...,x_k)$ dibuang.

2.2. Prinsip Pencarian Solusi dengan Metode Runut Balik



Gambar 1. Prinspi algoritma runut balik

a. Solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai mengikuti aturan *depth-first order* (*DFS*).
b. Simpul-simpul yang sudah dilahirkan dinamakan **simpul hidup** (*live node*).
c. Simpul-simpul yang sedang diperluas dinamakan **simpul-E** (*expand node*).
d. Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang.
e. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi, maka simpul-E tersebut "dibunuh" sehingga menjadi **simpul mati** (*dead node*)
f. Fungsi yang digunakan untuk membunuh simpul-E adalah dengan menerapkan **fungsi pembatas**.

2.3. Kompleksitas Waktu Algoritma Runut Balik

Setiap simpul dalam pohon ruang status berasosiasi dengan sebuah pemangilan rekursif. Jika jumlah simpul dalam pohon ruang status adalah 2^n atau n!, maka untuk kasus terburuk, algoritma runut balik membutuhkan waktu dalam:

a. $O(p(n)\ 2^n)$ atau
b. $O(q(n)n!)$

Dengan p(n) dan q(n) adalah polinom derajat n yang menyatakan waktu komputasi simpul.

2.4. Skema Umum Algoritma Runut Balik

```
procedure RunutBalikR(input k:integer)

{Mencari semua solusi persoalan dengan
metode runut-balik; skema rekursif
Masukan: k, yaitu indeks komponen
vektor solusi, x[k]
Keluaran: solusi x = (x[1], x[2], …, x[n])}

Algoritma

for tiap x[k] yang belum dicoba sedemikian sehingga
([k]  T(k)) and B(x[1], x[2], ...
,x[k])= true do
        if (x[1], x[2], ... ,x[k]) adalah
        lintasan dari akar ke daun then
                CetakSolusi(x)
```

III. SUDOKU

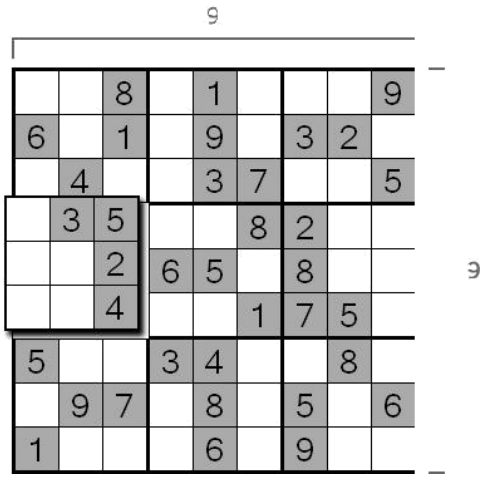
Permainan seperti sudoku sudah dikenal sejak tahun 1979 di majalah Dell Magazines dengan nama "Number Place". Permainan ini didesain oleh Howard Grans. Permainan ini mulai dikenal di Jepang pada tahun 1984, dimuat di majalah bulanan Nikoli, dengan nama "Suuji Wa Dokushin Kagiru". Selanjutnya puzzle ini dikenal sebagai Su-Doku (orang Jepang memang biasa menyingkat nama). Tahun 1986, Nikoli membuat aturan baru dalam permainan sudoku, yaitu :

1. Nomor yang disertakan dalam soal tidak boleh lebih dari 32 buah.
2. Soal harus simetris.

Tahun 2004, permainan ini mulai dikenalkan di Inggris, dan diterbitkan pertama kali di The Times, 12 November 2004, tetap menggunakan nama sudoku.

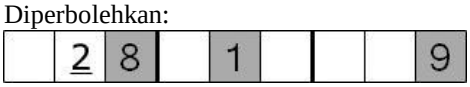
Sudoku dimainkan dengan cara mengisi kotak dalam *board* dengan angka dari satu hingga Sembilan dengan aturan permainan sebagai berikut:

1. Sudoku dimainkan dalam 9x9 kotak yang dibagi dalam 3x3 upa-board (sel) yang disebut "area".

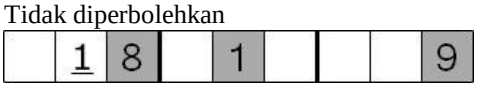


2. Sudoku dimulai dengan beberapa sel yang sudah terisi dengan angka
3. Angka hanya dapat muncul sekali dalam setiap baris:

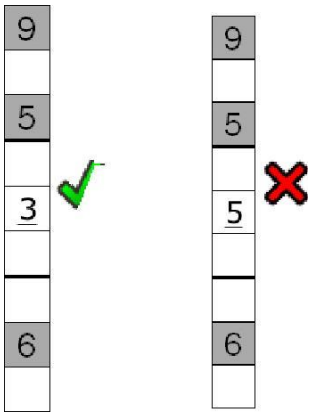
   - Diperbolehkan:

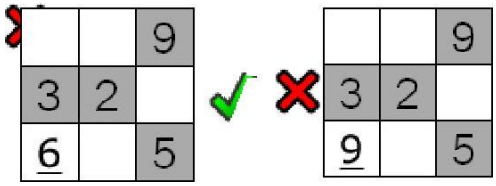
   - Tidak diperbolehkan

4. Angka hanya dapat muncul sekali dalam setiap kolom:



5. Angka hanya dapat muncul sekali dalam setiap area



Dalam permainan sudoku yang telah dibuat dalam bentuk aplikasi komputer, saat game dimulai setiap kotak telah dibangkitkan nilainya masing-masing.

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

Pembangkitan nilai awal pada sudoku berperan penting dalam penyelesaian suatu *game* sudoku. Sebab pada prinsipnya jika kombinasi awal angka dan posisi digenerate secara salah, maka terdapat paling tidak satu angka dan posisi yang tidak memenuhi aturan peraminan sudoku

IV. ANALISIS PEMBANGKITAN NILAI AWAL SUDOKU MENGGUNAKAN ALGORITMA RUNUT BALIK

Penerapan algoritma runut balik dalam pembangkitan nilai awal Sudoku adalah sebagai berikut.

1. Pada keadaan awal, seluruh kotak tidak memiliki nilai. Algoritma dimulai dari posisi 0,0 pada matriks Sudoku 9x9.
2. Pilih sebuah nilai x, dimana x merupakan anggota dari {1, 2, 3, 4, 5, 6, 7, 8, 9}.
3. Jika x memenuhi aturan Sudoku, maka isi kotak yang kosong tersebut dengan nilai x dan lanjutkan pemeriksaan ke kotak selanjutnya.
4. Jika nilai x tidak memenuhi aturan Sudoku, maka ubah nilai x menjadi nilai yang lain dalam himpunan keanggotaan x.
5. Jika seluruh kemungkinan x telah dicoba dan tidak ada yang memenuhi aturan permainan Sudoku, maka akan dilakukan *backtracking* ke kotak sebelumnya dan nilai kotak sebelumnya diubah menjadi kemungkinan nilai yang lain.
6. Lakukan langkah ke-3 untuk kotak-kotak yang lain.
7. Proses di atas akan dilakukan terus-menerus secara rekursih hingga seluruh kotak pada *board* Sudoku telah terisi seluruhnya dan memenuhi aturan Sudoku.
8. Untuk digunakan dalam sebuah game, nilai-nilai pada board tersebut dihapus secara *random* sebanyak y buah tergantung tingkat kesulitan game.

Langkah-langkah di atas akan direpresentasikan dalam *pseudocode.*

```
FUNCTION SUDOKU()

DEKLARASI

values : array [0..8] of
integer index , i : integer
num : integer
z , zindex : integer
maxbacktrack , backtrack , backtracked :
int NElemenAwal : integer
level : string
fungsi SesuaiAturan (input baris , kolom
, num : integer , output boolean)


ALGORITMA

1.  index <- 0;
2.  While (index < Nbaris * NKolom) do
3.        num <- 0
4.        for i <- 0 to 8 do
5.            values[i] <- i + 1
6.            i++
7.        endfor
8.        while (num=0 & values.length>0) do
9.            zindex <- rand(0, values.length)
10.           z <- values[zindex]
11.           values[zindex] = " "
12.           if (SesuaiAturan (z ,
              Grid[index] = true)) then
13.               num <- z
14.           else
15.               num <- 0
16.           endif
17.       endwhile
18.       if num = 0 then
19.          if index > 5 then
20.             maxbacktrack <- 5
21.          else
22.             maxbacktrack <- index
23.          endif
24.          backtrack <- random(1,
              maxbactrack)
25.          backtracked <- 0
26.          while (backtracked < bactrack)do
27.             selGrid[index - backtracked]<-0
28.             backtracked++
29.          endwhile
30.          index <- index – backtrack
31.       else
32.          selGrid[index] <- num
33.          index++
34.       endif
35. endwhile
```

Pada bagian values, didefinisikan variable yang digunakan dalam fungsi di atas, yaitu:

a. values , variable penampung
b. num , nilai setiap kotak yang mungkin
c. NElemenAwal, banyak elemen yang akan ditampilkan

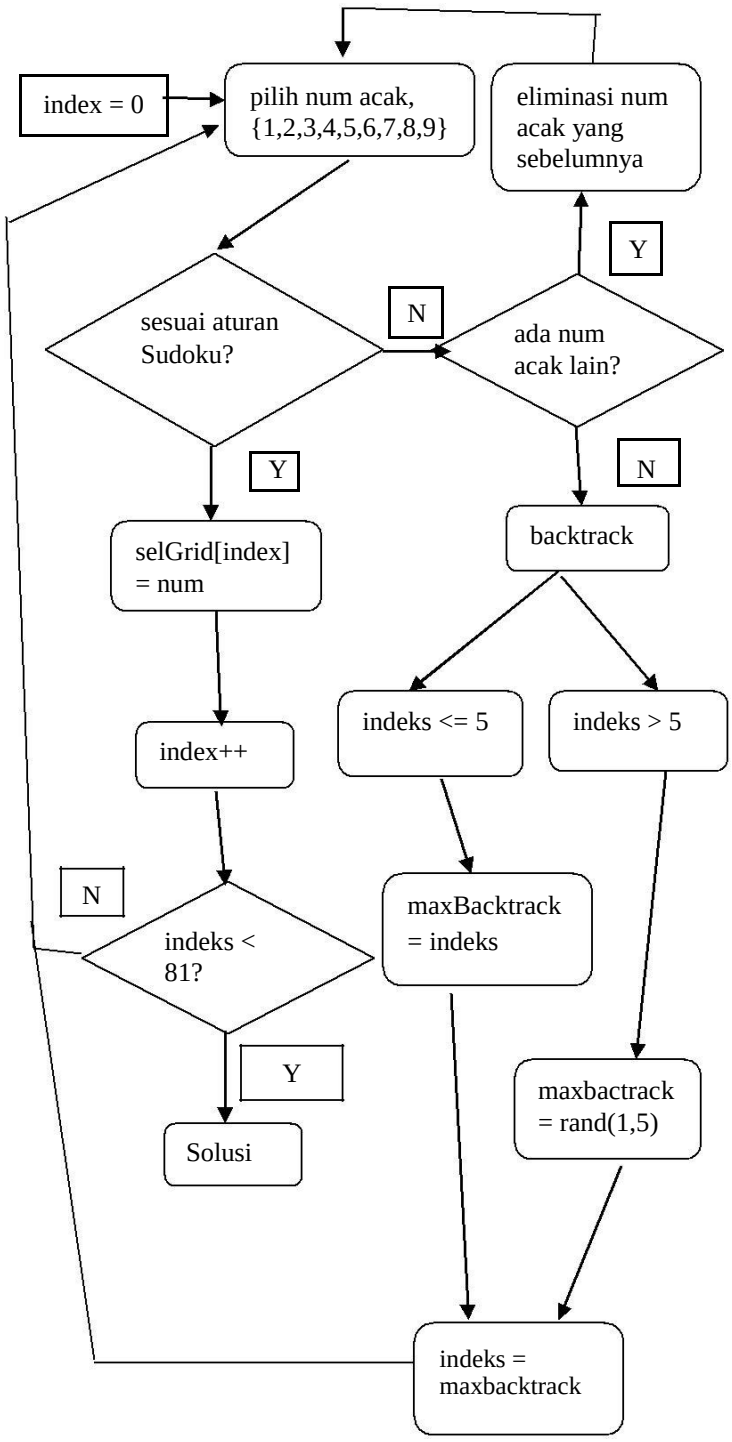
Pada baris 1 – 7, diinisialisasikan index dengan 0. selama indeks < 81, insialisasi num dengan 0. kemudian dilakukan pengisian array values = [1, 2, 3, 4, 5, 6, 7, 8, 9].

pada baris 8 – 11, fungsi akan memilih angka secara acak antara 0 sampai dengan banyaknya angka pada array values. angka acak akan dimasukkan ke dalam variable penampung zindex dan nilai dari elemen ke zindex dari values akan dimasukkann ke dalam z.

pada baris ke 12 – 17, fungsi akan meemriks apakah nilai z memenuhi aturan Sudoku atau tidak. jika z memenuhi aturan Sudoku maka akan dilanjutkan ke tahapan selanjutnya. namun jika tidak maka akan dilakukan kembali instruksi pada baris 9 – 17.

Jika ternyata banyaknya anka yang terdapat pada array values adalah nol, atau dengan kata lain tidak ada nilai yang memenuhi untuk indeks tersebut maka akan dilakukan backtrack dengan instruksi pada baris 18 – 35.

interpretasi code dalam flowchart.



V. KESIMPULAN

Dalam kehidupan sehari-hari, sangat banyak contoh pengaplikasian algoritma runut balik atau *backtracking.* Mulai dari masalah yang sangat rumit hingga masalah yang sangat sederhana. salah satunya dalam kasus pembangkitan elemen pada kotak-kotak Sudoku. Sebenarnya tidak hanya algoritma runut balik yang bisa digunakan dalam penyelesaian kasus ini, namun karena kompleksitas waktu dan ruang serta sistem pemecahan masalah yang lebih pas maka algoritma runut balik sangat efisien digunakan dalam kasus ini.

Dalam penyolusiannya, kasus ini juga dapat diselesaikan dengan menggunakan berbagai bahasa pemrograman sesuai dengan pseudocode yang telah dibuat pada makalah ini.

VI. DAFTAR PUSTAKA

1. Munir, Rinaldi, Diktat Kuliah IF2211 Strategi Algoritma. Program Studi Teknik Informatika ITB, 2015.
2. Nuralta, Novita, Penerapan Algoritma Backtrack, Perpustakaan UPI, 2013

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, 9 Mei 2016

Muhammad Farhan Kemal
13513085