Penerapan Algoritma DFS pada Permainan Sudoku dengan Backtracking

Krisna Dibyo Atmojo
13510075
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13510075@std.stei.itb.ac.id

Abstrak

Game Sudoku adalah game puzzle numerik yang paling populer di dunia. Game yang mengharuskan kita untuk mengisi angka pada kolom kosong matriks dengan peraturan tertentu. Makalah penulis ini akan membahas bagaimana menyelesaikan permainan sudoku dengan memanfaatkan algoritma *Depth First Search. Algoritma Deep First Search* atau biasa disebut DFS direpresentasikan dengan struktur pohon berakar. Algoritma DFS menggunakan pendekatan penelusuran solusi secara traversal pada setiap simpul pohon. Penulis akan mengimplementasikan Algoritma DFS dengan konsep BackTracking yang sebenarnya keduanya sama. Pada makalah ini penulis akan membahas 5 pokok bahasan yaitu Pendahuluan, Sudoku, Algoritma DFS dan backtracking, Implementasi, Analisis dan Kesimpulan.

Kata Kunci – Sudoku, DFS, Solusi,BackTracking

PENDAHULUAN

Permainan puzzle termasuk permainan yang membutuhkan nalar dan logika untuk menyelesaikan *goal* permainan tersebut. Permainan puzzle atau teka-teki sangat banyak jenisnya, salah satunya adalah teka-teki angka. Permainan Sudoku termasuk permainan teka-teki angka yang telah sangat populer di dunia. Sudoku banyak dimainkan oleh pelajar dan karyawan untuk mengisi waktu luang dengan mengasah otak. Di jepang sudoku banyak dimainkan oleh penumpang kereta saat menunggu kereta dan saat perjalanan. Rasa penasaran untuk menyelesaikan permasalahan sudoku membuat banyak orang yang akhirnya menjadi ketagihan dan fanatik kepada sudoku. Semakin tinggi level sudoku, semakin menantang persoalan yang harus diselesaikan. Walaupun sudoku merupakan permainan teka-teki angka, tidak berarti hanya orang yang memiliki kemampuan dan keahlian mengenai matematika saja yang dapat memainkannya, Sudoku dapat dimainkan siapa saja dari beragam umur, jenis kelamin, dan pekerjaan. Atas dasar ini, Penulis memiliki ketertarikan untuk melakukan eksplorasi pencarian solusi permainan sudoku dengan mengaplikasikan beberapa teori algoritma yang penulis pelajari pada matakuliah IF 3051 (Strategi Algoritma) yaitu Algoritma DFS dan *Backtracking*.

Tujuan penulis membuat makalah ini adalah untuk memenuhi persyaratan tugas matakuliah IF 3051 dan memberikan pengetahuan pada pembaca tentang Algoritma DFS serta aplikasinya untuk menyelesaikan permasalahan permainan tekateki sudoku.

SUDOKU

Sudoku adalah permainan teka-teki angka yang terdiri dari 81 kotak (9 x 9) yang berisi angka-angka antara 1 sampai 9 yang harus diisi penuh pada setiap kotaknya. Tujuan permainan ini adalah *refreshing* dengan mengasah otak. Pemain sudoku diharuskan untuk mengisi semua kotak kosong yang tersedia sedemikian setiap angkanya akan unik pada baris, kolom, dan daerahnya (akan dijelaskan). Contoh sudoku dapat dilihat pada gambar 1. Saat pertama memainkan sudoku, beberapa kotak sudah diisi angka yang merupakan *hints* untuk mengisi kotak kosong lain hingga penuh. Pada gambar di bawah, dapat kita lihat bahwa adanya perbedaan tebal garis kotak. Garis tebal kotak menandakan batas daerah unik yang harus diisi angka yang unik dari 1-9. Daerah unik terdiri dari 9 kotak (3x3). Sebagai contoh, Daerah unik pada posisi terkiri-teratas terdiri dari angka 5,3,6,9, dan 8, untuk itu 4 kotak kosong lain harus diisi dengan angka unik selain angka sudah ada pada daerah itu yaitu 1,2,4, dan 7. Setiap angka yang diisi pada suatu kotak harus unik secara horizontal, vertikal, dan daerahnya. Kita harus mencari semua nilai yang unik pada setiap kotak dengan strategi tertentu sehingga semua kotak kosong terisi oleh angka.

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 : Contoh Sudoku[sumber : Wikipedia]

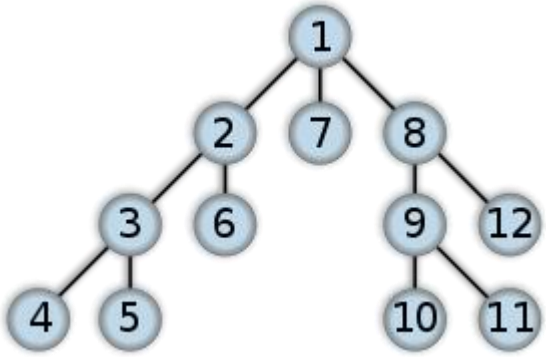
Hasil penyelesaian kasus sudoku pada gambar 1 ini dapat dilihat pada gambar 2 di bawah ini. Angka yang berwarna merah adalah angka kita isikan pada kotak kosong sudoku. Semua angka pada kotak sudoku memiliki nilai yang unik pada baris, kolom, dan daerahnya. Dengan demikian sudoku telah berhasil terselesaikan.

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

Gambar 2 : Contoh Solusi Sudoku [sumber : Wikipedia]

ALGORITMA DFS DAN BACKTRACKING

Pencarian solusi Algoritma DFS menggunakan pendekatan pencarian solusi penelusuran solusi pohon dinamis. Pada umumnya struktur pohon yang digunakan adalah pohon berakar. Pencarian Solusi dilakukan dengan mengunjungi semua simpul pada pohon. Setiap simpul diperiksa untuk memastikan apakah solusi telah dicapai. Pohon dibentuk dengan dinamis selama solusi berlangsung. Bila sebuah simpul yang dibentuk tidak mengarah ke solusi, maka pencarian solusi dilanjutkan dengan membentuk simpul berikutnya. Teknik penelesuruan simpul yang digunakan pada algoritma DFS adalah penelesuran "Mendalam". Pencarian solusi dengan DFS dicirikan dengan ekpansi simpul-simpul terdalam lebih dahulu. Mula-mula simpul akar dibangkitkan pertama kali, kemudian sebuah simpul pada aras kedua, lalu sebuah simpul di aras ketiga, dan seterusnya. Jadi, pencarian mengikuti sebuah lintasan tunggal mulai dari simpul akar terus menurun ke "Bawah" ke simpul-simpul pada aras di bawahnya. Sebagai contoh dapat dilihat pada gambar 3. Urutan penelesuran simpul dapat dilihat pada angka yang muncul pada simpul tersebut. Definisi Kedalaman (deept) atau tinggi (heigth) pohon berakar berdasarkan bahwa kedalaman simpul akar adalah 0 dan kedalaman simpul lainnya adalah 1 + kedalaman simpul orang tuanya. Model Pencarian solusi dengan melakukan penelesuran pada struktur pohon ada juga yang dilakukan dengan cara "Melebar" yang biasa disebut Algoritma BFS. Namun, Algoritma tersebut memiliki kelemahan yaitu lamanya waktu yang dibutuhkan untuk menuju simpul solusi. Untuk itu penulis lebih *prefer* untuk menggunakan algoritma pencarian solusi dengan DFS saja.



Gambar 3: Contoh pohon DFS[Sumber : Wikipedia]

Penelusuran semua simpul pada DFS jika dituliskan dengan bahasa pseudocode adalah sebagai berikut :

```
Procedure DFS (input V : integer)

{mengunjungi seluruh simpul graf
dengan algoritma pencarian DFS

Masukan  : V adalah simpul awal
kunjungan
keluaran : Semua simpul yang
dikunjungi ditulis layar

}

Deklarasi
    w : integer

Algoritma
    write(v)
    dikunjungi[v] <- true
    for tiap simpul w yang bertetangga
dengan simpul V do
        if not dikunjungi[w] then
            DFS(w)
```

```
    endif
  endfor
```

Pencarian solusi dengan metode DFS dilakukan dengan prosedur atau langkah-langkah sebagai berikut (Secara umu) :

1. Masukan simpul akar ke dalam antrian Q. Jika simpul akar adalah simpul tujuan (*goal* node), maka solusi telah ditemukan. Stop.
2. Jika Q kosong, tidak ada solusi. Stop
3. Ambil simpul v dari kepala (*head*) antrian. Jika kedalaman simpul v sama dengan batas kedalaman maksimum, kembali ke langkah 2.
4. Bangkitkan semua anak dari simpul v. Jika v tidak mempunyai anak lagi, kembali ke langkah 2. Tempatkan semua anak dari v di awal antrian Q.
5. Jika anak dari simpul v adalah simpul tujuan, berarti solusi telah ditemukan, kalau tidak, kembali lagi ke langkah 3.

Pada dasarnya penerapan pohon DFS dengan *backtracking* hanya memerlukan sedikit modifikasi saat terjadi *deadend* pada simpul. Jika simpul v tidak dapat menciptakan simpul baru namun simpul v tersebut bukan merupakan solusi dari persoalaan, maka simpul v dikatakan mencapai kondisi *deadend*. Simpul v akan melakukan runut balik pada orang tua simpul v terdekatnya untuk menciptakan simpul baru w yang setara kedalamannya dengan simpul v. Simpul w akan bangkitkan semua anaknya untuk melakukan penelusuran solusi.

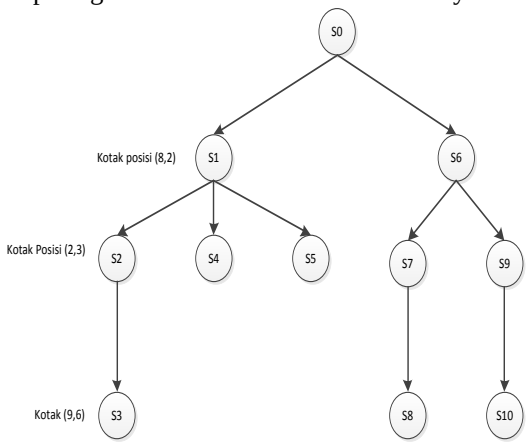
IMPLEMENTASI

Pertama-tama penulis akan memberikan indeks pada setiap kotak pada sudoku agar mudah untuk menunjuk posisi kotak tertentu. Posisi atau indeks kotak akan dinotasikan dengan (X,Y). Dengan X menyatakan posisi suatu kotak secara horizontal dan Y menyatakan posisi kotak secara vertikal. Pengindeksan kotak lengkap dapat dilihat pada gambar 4 di bawah ini :

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

Gambar 4: Indeks Sudoku

Angka yang ada di dalam matriks merupakan indeks posisi kotak, indeks dimulai dari ujung kiri-atas yaitu (1,1) sampai ujung kanan-bawah yaitu (9,9). Hal ini akan bermanfaat untuk proses implementasi selanjutnya. Penulis akan mendefinsikan isi simpul dan kedalaman pohon terkait dengan proses pencarian solusi. Simpul akan didefinisikan sebagai nilai yang pada suatu kotak di posisi tertentu. Ketinggian pohon akan didefinisikan sebagai posisi kotak kosong yang akan diisi. Jadi misalkan *problem sheet* sudoku terdiri dari 30 kotak kosong yang harus diisi dari 81 total kotak sudoku, maka pohon pencarian harus mencapai kedalaman 30 untuk mencapai solusi. Urutan posisi kotak yang akan dicari nilai simpulnya adalah dari paling kiri-atas ke kanan-bawah. Misalkan terdapat 3 kotak kosong yaitu (2,3),(8,2), dan (9,6), maka pada kedalaman 1 akan berisi simpul kotak posisi (8,2) , kedalaman 2 akan berisi simpul kotak posisi (2,3) , dan yang terakhir kedalaman 3 akan berisi simpul kotak posisi (9,6). Untuk mempermudah pemahaman mengenai definisi ini, pembaca dapat melihat penjelasan pada gambar 5 terkait contoh sebelumnya :



Gambar 5: Contoh teknik penelusuran 3 kotak kosong

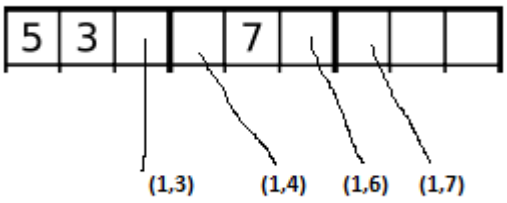
Pada gambar 5 diatas, kita dapat melihat contoh model penelusuran yang dilakukan pada contoh sebelumnya. Selanjutnya kita akan mendefinisikan fungsi kelayakan yaitu suatu nilai pada simpul dinyatakan layak jika pada saat kondisi dimana tidak ada nilai yang sama pada baris, kolom, dan daerahnya dengan nilai tersebut. Fungsi Solusi didefinisikan jika semua kotak kosong telah terisi dengan nilai yang sesuai (sesuai dengan fungsi kelayakan). Penulis akan memberikan contoh kasus sudoku dan menyelesaikan dengan menggunakan pohon status DFS dan *backtracking*. Misalkan diberikan kasus sudoku seperti gambar di bawah ini :

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 6 : Contoh kasus Sudoku

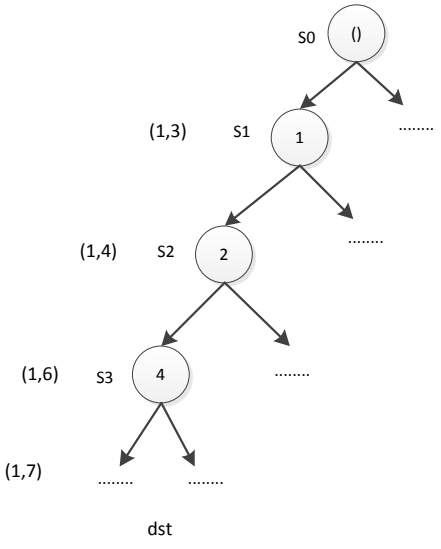
Langkah pertama adalah kita akan melakukan pencarian solusi pada kotak paling kiri-atas yang kosong, dalam hal

ini adalah kotak (1,3). Selanjutnya kotak (1,4) , (1,6), dan seterusnya.



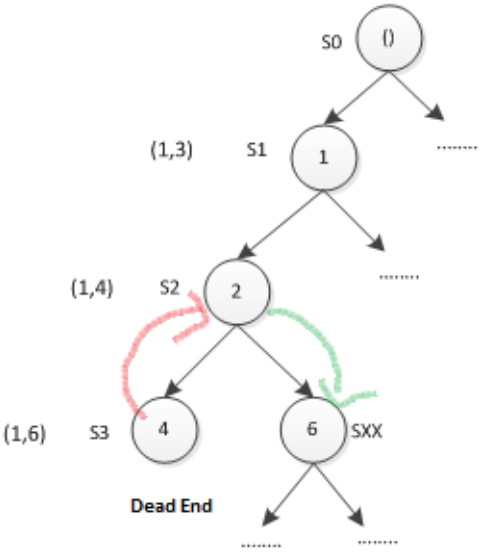
Gambar 7: Penjelasan

Pada kotak (1,3) yang kosong pada gambar 6 kotak tersebut dapat diisi dengan angka 1,2,4 (yang memenuhi fungsi kelayakan). Sesuai urutan angka kita akan mengambil angka 1 dahulu. Karena kita melakukannya dengan algoritma DFS, maka setelah kita memberi nilai pada suatu simpul kita akan melahirkan simpul pada level atau kedalaman selanjutnya yaitu level 2 untuk kotak (1,4). Kotak (1,4) dapat diisi oleh angka 2,6, dan 9 sesuai urutan kita akan mengambil angka 2 dahulu. Pada level-3, Kita akan mengisikan Kotak (1,6) dengan kandidat angka 4,6,8 (angka 2 tidak bisa diambil karna telah diambil oleh kotak <1,4>), Kita ambil angka 4. Dan seterusnya kita lakukan pada level / kotak selanjutnya sampai menemukan solusi. Pohon yang menggambarkan tentang proses yang kita lakukan sejauh ini dapat dilihat pada gambar dibawah ini :



Gambar 8: Pohon DFS

Penelusuran ini akan dilakukan sampai penelusuran mencapai level N, dimana N adalah jumlah kotak kosong pada sudoku. Jika pada suatu level simpul tidak bisa menghasilkan suatu simpul baru dengan kata lain tidak ada angka yang memenuhi fungsi kelayakan, maka kondisi tersebut dikatakan kondisi *deadend* atau simpul mati. Apabila terjadi *deadend*, simpul akan melakukan *backtrack* atau runut balik ke simpul *parent* nya dan akan membuat simpul lain, jika ada, dengan kandidat angka yang ada untuk ditelusuri, jika tidak ada simpul yang layak dihasilkan, maka *parent*-nya juga akan melakukan backtrack sampai ketemu *parent* yang dapat menghasilkan anak atau simpul baru. Untuk mengilustrasikannya, dapat dilihat pada gambar 9 di bawah ini :



Gambar 9: Contoh penanganan deadend

Misalkan terjadi *deadend* pada saat melakukan penelusuran anak-anak simpul 3 yang menimbulkan *backtrack* atau runut balik kembali pada *parent* dari simpul 3 yaitu simpul 2. Selanjutnya simpul 2 akan melahirkan simpul baru yang merupakan kandidat angka dari Level 3 (Kotak <1,6>). Sebelumnya kita mengetahui bahwa kandidat untuk kotak <1,6> dengan simpul parent S2 yaitu 4,6, dan 8. Nilai 4 pada S3 diasumsikan sudah mencapai *deadend* sehingga Simpul S2 akan melahirkan simpul baru SXX dengan nilai yaitu 6. Kemudian simpul SXX akan membangkitkan anak-anak barunya. Proses ini dilakukan terus-menerus sampai pohon mencapai kedalaman N dimana fungsi solusi terpenuhi.

ANALISIS

Penerapan Algoritma DFS dengan backtracking untuk memecahkan masalah sudoku telah dapat terimplementasikan. Berdasarkan analisis penulis, Algoritma dengan bactracking menghasilkan kompleksitas yang cukup mangkus. Bandingkan dengan penyelesaian masalah sudoku jika menggunakan Algoritma Brute Force atau greedy. Dengan algoritma brute force, kita akan melakukan penelusuran semua kemungkinan yang mungkin pada kotak kosong sudoku. Asumsikan kotak yang kosong pada permasalahan sudoku berjumlah N, maka kemungkinan terburuk untuk mencapai solusi dengan menggunakan algoritma bruteforce-nya adalah 9 ^ N. Tentu jumlah ini sangat besar dan tidak efektif. Sedangkan dengan menggunakan Algoritma DFS / backtracking kompleksitas yang dihasilkan dapat dipastikan akan lebih mangkus dari kompleksitas yang dihasilkan dengan menggunakan algoritma bruteforce.

Ada beberapa sumber yang beranggapan bahwa penelusuran pohon status dengan menggunakan algoritma

DFS sudah termasuk backtracking. Dengan kata lain semua penelusuran solusi dengan pohon status DFS telah mengimplementasikan bactracking. Tentu kompleksitas DFS dengan backtracking jauh lebih mangkus ketimbang DFS tanpa backtracking. Dengan backtracking hanya pencarian yang mengarah solusi saja yang selalu dipertimbangkan.

KESIMPULAN

Pencarian solusi permainan teka-teki Sudoku dapat diselesaikan dengan menggunakan algortima DFS dan *backtracking* . Pemilihan Algoritma DFS dan backtracking untuk memecahkan permainan sudoku dapat dikatakan cukup mangkus. Jika dibandingkan dengan algoritma klasik seperti brute force, algoritma DFS dan *backtracking* memiliki kompleksitas yang lebih baik, karna penelusuran solusi dilakukan lebih "cerdas" dibandingkan dengan algoritma brute force yang mencari semua kemungkinan yang ada. Misi atau goal permainan sudoku adalah menjawab semua kotak kosong yang ada dengan angka-angka yang unik sehingga solusi yang akan dihasilkan hanyalah satu solusi. Kelemahan dari Algoritma DFS atau backtracking adalah banyak langkah yang terbuang apabila penelusuran beberapa kali mencapai *deadend* . Untuk mendapatkan solusi yang lebih optimal maka dapat digunakan algoritma yang lebih kompleks seperti *Dynamic programming*.

REFERENCES

[1] Munir, Rinaldi, "Diktat kuliah IF3051 Strategi Algortma" , Program Studi Informatika ITB,2009

[2] Levitin, Anany, "Introduction to The Design Analysis of Algorithms", Villanova university, 2012

[3] http://yufi27.wordpress.com/2009/03/29/algoritma-dfs-bfs/

[4] http://www.sudokuessentials.com/how_to_play_sudoku.html

[5] http://www.su-doku.net/tech.php

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, 14 Desember 2012

Krisna Dibyo Atmojo
13510075