Analisis Kompleksitas Algoritma dalam Penyelesaian Sudoku menggunakan Algoritma *Minigrid Backtracking*

Dimpos Arie Ginarta Sitorus / 13513083
Program Sarjana Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
dimpossitorus@gmail.com

***Abstract*—Makalah ini akan membahas kompleksitas algoritma *Minigrid Backtracking* dalam penyelesaian permainan Sudoku pada level tertentu**

***Keywords*—kompleksitas algoritma, Sudoku, *minigrid backtracking, backtracking*.**

I. PENDAHULUAN

Permainan Sudoku adalah salah satu permainan teka-teki yang cukup terkenal di dunia. Permainan teka-teki adalah permainan yang melibatkan logika. Setiap teka-teki pasti mempunyai solusi sehingga dapat diselesaikan menggunakan komputer dengan memanfaatkan algoritma-algoritma tertentu. Begitu juga dengan Sudoku, pasti mempunyai solusi dan dapat dibuat algoritma penyelesaiannya

Oleh karena itu, penulis ingin menganalisis kompleksitas algoritma yang digunakan untuk menyelesaikan Sudoku. Dengan menganalisisi kompleksitas dapat diperkirakan waktu yang dibutuhkan untuk menyelesaikan satu teka-teki Sudoku dengan level tertentu

II. SUDOKU

A.Pengertian Sudoku

Sudoku adalah permainan teka-teki angka. Dimainkan dengan cara mengisi angka 1-9 pada kotak berukuran 9×9. Kotak 9×9 dibagi lagi ke dalam kotak 3×3. Tidak boleh ada angka yang sama pada setiap baris, kolom dan kotak 3×3. Sudoku berhasil diselesaikan jika seluruh 81 kotak-kotak terisi dan syarat-syarat di atas terpenuhi.

B. Sejarah Sudoku

Sudoku berawal dari abad ke 18 ketika Leonhard Euler, matematikawan yang berasal dari Swiss memperkenalkan '*Latin Squares',* dimana seitap baris dan kolom berisi simbol-simbol yang muncul hanya sekali. 'Latin Squares' pada awalnya berisikan huruf-huruf latin.

Kemudian pada abad ke 19 akhir, teka-teki angka yang mengikuti aturan awal 'Latin Squares' muncul di surat kabar *Le Siècle* di Perancis. Namun ini belum dapat dikatakan Sudoku karena pada teka-teki ini terdapat angka dua *digit* dan membutuhkan kemampuan aritmatika daripada logika. Surat kabar lain di Prancis, *La France*, memodifikasi teka-teki tersebut. Mereka membuat teka-teki tersebut sedemikian sehingga setiap baris, kolom dan diagonal (diagonal pada teka-teki ini seperti diagonal sekunder pada matriks) hanya terdapat angka 1 – 9 tanpa ada pengulangan. Teka-teki ini hampir mendekati model teka-teki Sudoku

Sudoku modern diciptakan oleh Howard Garns (2 Maret 1905 – 6 Oktober 1989), seorang pensiunan arsitek dan *freelancer* pembuat teka-teki. Dia menciptakan cikal bakal Sudoku dengan nama '*Number Place*'.

Kemudian *Number Place* ini dikenalkan di Jepang oleh Nikoli, sebuah perusahaan *publisher* khusus untuk permainan, terutama permainan teka-teki yang menggunakan logika, dengan nama *Sūji wa dokushin ni kagiru* (数字は独身に限る), jika diterjemahkan berarti : angkanya harus tunggal. Kemudian nama diatas disingkat menjadi Sudoku.

C. Tingkat kesulitan Sudoku

Sudoku dibagi ke dalam lima tingkat kesulitan yaitu *Beginner* (sangat mudah), *Easy* (mudah), *Medium* (sedang), *Hard* (sulit), *Expert* (sangat sulit). 1

Untuk tingkat kesulitan Beginner, terdapat 40 kotak sudah terisi dari 81 kotak yang ada pada Sudoku, sehingga kita hanya harus mengisi 41 kotak sisanya untuk menyelesaikan level beginner.

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

Gambar 1. Tingkat kesulitan *Beginner*
Sumber : http://www.sudoku.com

Untuk tingkat kesulitan *Easy*, terdapat 30 kotak sudah terisi dari 81 kotak yang ada pada Sudoku, sehingga kita hanya perlu mengisi 51 kotak kosong sisanya.

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

Gambar 2. Tingkat kesulitan *Easy*
Sumber : http://www.sudoku.com

Untuk tingkat kesulitan *Medium*, terdapat 25 kotak yang sudah terisi. Kita hanya perlu mengisi 56 kotak yang masih kosong untuk menyelesaikan teka-teki Sudoku.

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

Gambar 3. Tingkat kesulitan *Medium*
Sumber : http://www.sudoku.com

Untuk tingkat kesulitan *Hard*, terdapat hanya 20 kotak yang sudah terisi. Sehingga kita harus mengisi 61 kotak sisanya yang masih kosong untuk menyelesaikan teka-teki Sudoku

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

Gambar 4. Tingkat kesulitan *Hard*
Sumber : http://www.sudoku.com

Untuk tingkat kesulitan *Expert*, terdapat hanya 15 kotak yang sudah terisi, dan kita harus mengisi 66 kotak kosong sisanya untuk menyelesaikan puzzle Sudoku.

	4				8			7
								4
				1			7	
	8				3			5
7						1		
5								
							6	
	6		1					

Gambar 5. Tingkat kesulitan *Expert*
Sumber : http://www.sudoku.com

D. Tingkat kesulitan Sudoku

Sudoku dibagi ke dalam lima tingkat kesulitan yaitu *Beginner* (sangat mudah), *Easy* (mudah), *Medium* (sedang), *Hard* (sulit), *Expert* (sangat sulit). 1

Untuk tingkat kesulitan Beginner, terdapat 40 kotak sudah terisi dari 81 kotak yang ada pada Sudoku, sehingga kita hanya harus mengisi 41 kotak sisanya untuk menyelesaikan level beginner.

III. KOMPLEKSITAS ALGORITMA

A. Penjelasan Kompleksitas Algoritma

Hampir semua algoritma mempunyai alternatif-alternatif. Namun pasti terdapat algoritma yang terbaik diantara semua alternatif, yaitu algoritma yang mangkus dan sangkil. Algoritma yang mangkus adalah algoritma yang meminimumkan kebutuhan waktu dan kebutuhan ruang penyimpanan. Kompleksitas algoritma dapat menyatakan kemangkusan sebuah algoritma.

Kompleksitas algoritma dinilai dari kebutuhan waktu dan/atau kebutuhan ruang penyimpanan / memori sesuai dengan masukan yang diterima oleh algoritma.

B. Masukan pada Algoritma

Masukan adalah nilai/banyaknya data yang akan diproses oleh algoritma. Pada umumnya masukan dinyatakan dalam simbol '*n*'.

C. Kompleksitas Waktu (Tn)

Kompleksitas waktu diukur dari jumlah tahapan komputasi yang dibutuhkan untuk menjalankan algoritma sebagai fungsi dari *n*. Tahapan komputasi pada kompleksitas adalah tahapan - tahapan / operasi - operasi dasar seperti perbandingan elemen, penjumlahan, pengurangan, perkalian, dll.

Kompleksitas waktu dibedakan menjadi 3 macam, yaitu :

i. $T_{max}(n)$: Kompleksitas waktu untuk kasus terburuk (worst case), atau kebutuhan waktu maksimum
ii. $T_{min}(n)$: Kompleksitas waktu untuk kasus terbaik (best case), atau kebutuhan waktu minimum
iii. $T_{avg}(n)$: Kompleksitas waktu untuk kasus rata-rata (average case), atau kebutuhan waktu rata-rata

C. Kompleksitas Ruang (Sn)

Kompleksitas waktu diukur dari memori yang digunakan oleh struktur data yang terdapat di dalam algoritma sebagai fungsi dari *n.*

IV. ALGORITMA *MINIGRID BACKTRACKING*

Konsep algoritma minigrid backtracking adalah brute force attack. Dan ditambah dengan backtracking, yaitu bergerak mundur satu kotak jika ternyata tidak ditemukan solusi pada kotak yang sedang diproses

Sudoku terdiri atas kotak 9×9, dan terdapat kotak 3×3 yang disebut sebagai *minigrid*. Terdapat 9 *minigrid* pada sebuah Sudoku 9×9. Bentuk papan Sudoku yang diproses akan mengikuti ketentuan seperti gambar di bawah ini

C1	C2	C3						
C4	M1	C6		M2			M3	
C7	C8	C9						
	M4			M5			M6	
	M7			M8			M9	

Gambar 6. Papan Sudoku 9×9 dengan 9 *minigrid*

Algoritma ini melakukan pengisian mulai dari *minigrid* 1 (M1). Program akan mencek C1, apakah kosong. Jika kosong ,algoritma akan mengisikan angka 1, lalu kemudian melakukan cek apakah angka tersebut terdapat di kolom, baris dan *minigrid* dimana kotak itu berada. Jika ada, maka kemudian akan diganti dengan angka selanjutnya. Begitu seterusnya hingga angka masukan unik. Setelah ditemukan unik maka algoritma akan melanjut ke kotak berikutnya (C2) dan melakukan lagi

proses tadi. Jika pada waktu pengisian sebuah kotak (C_n) angka 1-9 tidak ada yang bisa memenuhi, maka kotak C*n* tersebut akan dikosongkan, kemudian mundur satu kotak ke kotak C_{n-1} dan kemudian isi kotak C_{n-1} ditambah satu, dan dilakukan lagi algoritma diatas

Jika ditulis ke dalam *pseudocode* maka akan seperti di bawah ini :

```
{Struktur data Sudoku dalam bentuk matriks}
while (not Accept) {accept : Sudoku penuh dan benar} do
      if ( IsEmpty (C1) ) then
            C1 ← 1
            while {not CekBaris(C1)
            and not CekKolom(C1)
            and not CekMinigrid(C1)
            and C1<=10) do
                  C1 ← C1+1
            If (C1=10) then
                  C1←0
                  Backtrack
      If (Position =/= C9) then
            Next (C1)
      else
            Next (M1)

      {Algoritma mengulang dari awal dengan input C2}
      {Jika sudah mencapai C9, algoritma akan melanjut ke minigrid berikutnya}

Keterangan fungsi/prosedur di dalam program
IsEmpty (N)     : menghasilkan true jika N kosong, false jika tidak
CekBaris (N)    : menghasilkan true jika elemen di N unik di dalam baris dimana N berada, false jika tidak
CekKolom (N)    : menghasilkan true jika elemen di N unik di dalam kolom dimana N berada, false jika tidak
Bac
Position : mengembalikan posisi kotak yang terakhir diproses
Next (N) : menghasilkan suksesor dari N, dalam hal ini Next (C1) adalah C2, Next (M1) adalah M2, dst
```

Algoritma *minigrid backtracking* yang digunakan tidak menggunakan pengecekan-pengecekan lain yang mempercepat proses penyelesaian Sudoku, hanya menggunakan prinsip *brute force attak* dan *backtracking*

V. ANALISIS KOMPLEKSITAS ALGORITMA *MINIGRID BACKTRACKING*

Operasi khas atau operasi dasar yang ada pada algoritma ini adalah perbandingan elemen dalam pencarian. Operasi khas ini terdapat pada fungsi CekBaris (N), CekKolom(N) dan CekMinigrid(N).

Karena ketiga fungsi diatas merupakan fungsi pencarian, maka terdapat tiga kemungkinan kompleksitas waktu, yaitu $T_{max}(n) = n$, $T_{min}(n) = 1$ atau $T_{avg}(n)=(n+1)/2$. $T_{max}(n)$ terjadi ketika elemen yang dicari berada pada ujung baris, kolom, minigrid atau elemen tersebut adalah unik. $T_{min}(n)$ terjadi ketika elemen N ditemukan di elemen awal pencarian. $T_{avg}(n)$ terjadi ketika elemen ditemukan di tengah-tengah, tidak di awal ataupun di akhir.

Algoritma ini akan mengikuti kompleksitas waktu $T_{max}(n)$, karena tujuan dari algoritma ini adalah mengisi kotak-kotak kosong dengan elemen unik, sehingga harus dilakukan pengecekan hingga ke ujung baris,kolom, dan, *minigrid* . Dan asumsi terjadi backtracking adalah sebanyak n/2 elemen Sudoku. Kemungkinan tidak terjadi backtracking adalah tidak mungkin. Mungkin juga terjadi *backtracking* hingga ke elemen awal C1 minigrid M1. Untuk level Beginner sampai medium, kemungkinan backtracking adalah sebanyak $\frac{n}{2}$ elemen, untuk level hard kemungkinan backtracking adalah $\frac{3n}{4}$,d an kemungkinan backtracking pada level expert adalah n elemen.

Untuk pengisian 1 kotak hingga sesuai, diambil rata-rata saja

$$Tn = (n+1)/2$$

Untuk satu kali pengulangan biasa :

$$Tn = n + n + n + 1$$
$$Tn = 3n + 1$$

Untuk pengisian 1 kotak hingga sesuai, diambil rata-rata saja

$$Tn = 3n + 1 \times (\frac{n+1}{2})$$
$$Tn = \frac{(3n^2 + 4n + 1)}{2}$$

Untuk *backtracking*, jumlah pengulangan adalah sebanyak $2 \times n_{backtrack}$

Untuk level *beginner* akan dilakukan pengulangan sebanyak 41 kali + 21 kali *backtracking*, yaitu sekitar 41+41 = 82 kali pengulangan, maka :

$$Tn = 82 \times \frac{(3n^2 + 4n + 1)}{2}$$
$$Tn = 41 \times (3n^2 + 4n + 1)$$

$$Tn = n \times (3n^2 + 4n + 1)$$
$$Tn = 3n^3 + 4n^2 + n$$
$$3n^3 + 4n^2 + n \leq 10n^3$$
$$C = 10 \; ; \; dan \; k = 1$$

Maka :

$$O(n) = n^3$$

Untuk level *Easy* akan dilakukan pengulangan sebanyak 51 kali + 26 kali *backtracking*, yaitu sekitar 102 kali pengulangan, maka :

$$Tn = \frac{102}{2} \times (3n^2 + 4n + 1)$$
$$Tn = 51 \times (3n^2 + 4n + 1)$$

$$Tn = n \times (3n^2 + 4n + 1)$$
$$Tn = 3n^3 + 4n^2 + n$$
$$3n^3 + 4n^2 + n \leq 10n^3$$
$$C = 10 \; ; \; dan \;\; k = 1$$
Maka :
$$O(n) = n^3$$

Untuk level *Medium* akan dilakukan pengulangan sebanyak 56 kali + 28 kali *backtracking*, yaitu sekitar 112 kali pengulangan, maka :

$$Tn = 112 \times \frac{(3n^2 + 4n + 1)}{2}$$
$$Tn = 56 \times (3n^2 + 4n + 1)$$
$$Tn = n \times (3n^2 + 4n + 1)$$
$$Tn = 3n^3 + 4n^2 + n$$
$$3n^3 + 4n^2 + n \leq 10n^3$$
$$C = 10 \; ; \; dan \;\; k = 1$$
Maka :
$$O(n) = n^3$$

Untuk level *Hard* akan dilakukan pengulangan sebanyak 61 kali. Untuk proses backtracking kemungkinan tidak akan hanya sebanyak $\frac{n}{2}$ input, namun kemungkinan akan menjadi $\frac{3 \times n}{4}$, yaitu sekitar 46 kali backtracking. Dan akan terjadi sekitar 61 + 92 = 151 kali pengulangan, maka :

$$Tn = 151 \times \frac{(3n^2 + 4n + 1)}{2}$$

$$Tn = \frac{4n}{3} \times (3n^2 + 4n + 1)$$
$$Tn = 4n^3 + \frac{16}{3}n^2 + \frac{4}{3}n$$
$$3n^3 + 4n^2 + n \leq 12n^3$$
$$C = 12 \; ; \; dan \;\; k \geq 1$$
Maka :
$$O(n) = n^3$$

Untuk level *Expert* akan dilakukan pengulangan sebanyak 66 kali. Untuk proses backtracking kemungkinan tidak akan hanya sebanyak $\frac{n}{2}$ input, namun kemungkinan akan menjadi n, yaitu sekitar 66 kali backtracking. Dan akan terjadi sekitar 66 + 132 = 198 kali pengulangan, maka :

$$Tn = 198 \times \frac{(3n^2 + 4n + 1)}{2}$$
$$Tn = 99 \times (3n^2 + 4n + 1)$$
$$Tn = \frac{7n}{2} \times (3n^2 + 4n + 1)$$
$$Tn = \frac{21}{2}n^3 + 14n^2 + \frac{7}{2}n$$
$$3n^3 + 4n^2 + n \leq 29n^3$$
$$C = 29 \; ; \; dan \;\; k \geq 1$$
Maka :
$$O(n) = n^3$$

VI. KESIMPULAN

Melalui pendekatan diatas, dapat disimpulkan bahwa algoritma *backtracking* bergerak mengikuti fungsi $O(n) = n^3$

Namun, untuk setiap level, Tn berbeda sesuai dengan banyaknya elemen yang diproses dan kemungkinan *backtracking* yang terjadi pada setiap proses. Hasil diatas hanya perkiraan, tidak 100% tepat, karena untuk setiap Sudoku letak setiap kotak yang berisi berbeda-beda, dan isi kotak yang berbeda-beda dari satu teka-teki Sudoku dengan yang lain dan dapat menyebabkan perbedaan *backtracking* yang terjadi.

REFERENCES

[1] Rosen, Kenneth H., "*Discrete Math and It's Applications*", 2nd ed., New York: McGraw-Hill, 2012, ch. 3.

[2] Boyer, Christian, "*Sudoku's French Ancestor", The Mathematical Intelliger*. *Vol 29*, 2007, pp. 37-44.

[3] http://www.sudoku.com/how-to-play/the-history-of-sudoku/ Tanggal akses : 7 Desember 2014

[4] Maji, Arnab Kumar, Rajat Kumar Pal, "*Sudoku Solver using Minigrid based Bactracking,"IEEE International Advance Computing Conference, 2014.*

[5] http://www.theguardian.com/media/2005/may/15/pressandpublishing.usnews Tanggal akses : 7 Desember 2014

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, 10 Desember 2014

Dimpos Arie Ginarta Sitorus / 13513083