PERBANDINGAN PENGGUNAAN ALGORITMA *BRUTEFORCE* DAN *BACKTRACKING* DALAM PENYELESAIAN SUDOKU

Rezza Mahyudin - 13505055

Departemen Teknik Informatika, Institute Teknologi Bandung

Jl. Ganesha 10, Bandung

E-mail : if15055@students.if.itb.ac.id

ABSTRACT

***Sudoku* merupakan salah satu permaina yang sangat populer. Pada permainan ini akan terdapat kotak besar yang disusun oleh kotak-kotak kecil. Kota kecil akan tersusun oleh n^2 *space* sebagai tempat untuk memposisikan angka. Kemudian kota-kota kecil terebut akan menyusun kota besar sehingga ukuran kota besar adalah n^2 x n^2. Objektif dari permainan ini adalah bagiamana kita bisa mengatur susunan angka-angka yang telah ditentukan kedalam suatu *space* pada kotak, dengan syarat bahwa angka pada kolom dan baris kotak besar tidak boleh sama demikian juga angka pada satu kotak kecil tidak boleh sama. Angka-angka yang dimaksud disini adalah bilangan 1-n^2. Permasalahan puzzle Sudoku sulit untuk dipecahkan karena masuk dalam permasalahan *NP-complete,* sehingga tidak bisa diselesaikan dalam waktu polinomial. Hingga saat ini banyak programmer yang mencari algoritma yang mangkus untuk menyelesaikan puzzle ini.**

Kata Kunci : *Sudoku, Brute Force, Backtracking, Algorithm*

1. PENDAHULUAN

Permainan Sudoku adalah bentuk permainan yang melatih logika yang dimiliki manusia dalam berpikir cepat namun teliti. Di dalam penyelesaian permainan sudoku ini, kita dituntut untuk memikirkan masak-masak langkah-langkah yang akan kita ambil. Hal ini dikarenakan apabila kita mengambil langkah tanpa pertimbangan terlebih dahulu atau kita mengambil langkah secara asal, maka nantinya hal tersebut justru akan menyulitkan kita ketika kita dituntut untuk mengambil langkah selanjutnya guna menyelesaikan permainan ini.

Pada umumnya, permainan sudoku diselesaikan secara manual oleh manusia, yaitu dengan cara mengeliminasi kemungkinan-kemungkinan pengisian sel. Akan tetapi, akhir-akhir ini penyelesaian sudoku menjadi sebuah kajian yang serius di kalangan programer-programer karena tingkat kerumitannya.

Tujuan penulisan makalah ini adalah :

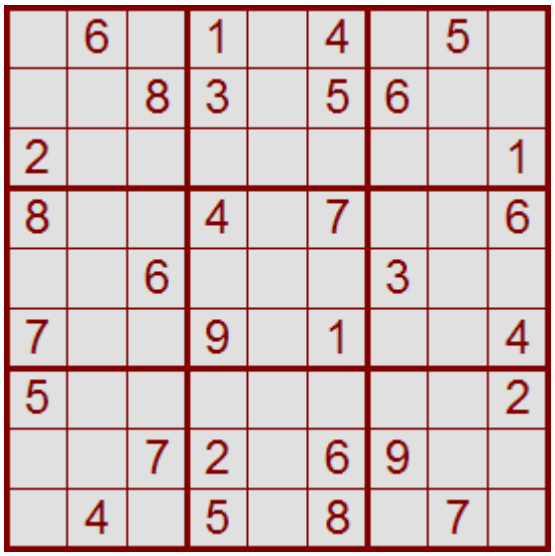
1. Menyelesaikan tugas yang diberikan oleh Dosen Mata Kuliah IF2251
2. Membandingkan algoritma *brute force* dengan algoritma *backtracking*

2. RUANG LINGKUP

Ruang lingkup pada makalah ini adalah pembahasan penggunaan algoritma *bruteforce* dan algoritma *backtracking* di dalam proses penyelesaian soal sudoku.

3. SUDOKU

Sudoku adalah sebuah permainan teka-teki penempatan angka yang berbasiskan pada logika manusia. Tujuan dari permainan ini adalah mengisi matriks yang berukuran 9x9 sehingga tiap kolom, tiap baris, dan tiap sub-matriks kecil yang berukuran 3x3 mengandung angka mulai dari 1 hingga 9 tepat satu buah. Sebagai penjelasan, lihat gambar sebagai berikut :



Gambar 1. Soal sudoku yang belum terselesaikan

Pada proses penyelesaian sudoku terdapat beberapa batasan yaitu adanya larangan untuk menempatkan angka yang sama pada satu baris, satu kolom, dan satu sub-matriks kecil yang sama. Untuk soal sudoku di atas, penyelesaiannya dapat dilihat pada gambar berikut :

Gambar 2. Soal sudoku yang telah diselesaikan

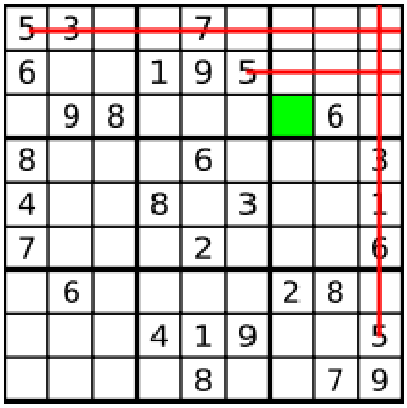
4. PENYELESAIAN SUDOKU SECARA MANUAL

Strategi di dalam menyelesaikan persoalan sudoku secara manual dapat dianggap sebagai gabungan dari 3 proses yang dikerjakan, yaitu memeriksa, menandai, dan menganalisis.

Proses memeriksa dilakukan pada awal dan saat permainan sudoku berlangsung. Proses ini sendiri terdiri dari 2 teknik yaitu *cross hatching* dan perhitungan.

Pada proses *cross hatching* dilakukan pemeriksaan tiap baris, mana yang dapat diisi oleh angka tertentu melalui proses eliminasi. Hal yang sama juga dilakukan pada tiap kolom. Untuk hasil yang lebih baik lagi, pemeriksaan dilakukan secara sekuensial terhadap frekuensi kemunculan angka tersebut.

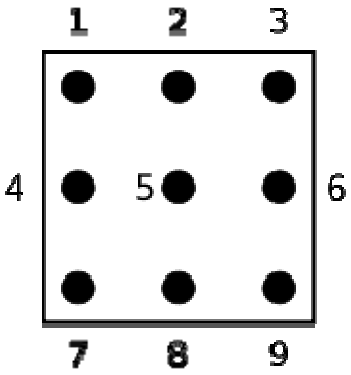
Proses perhitungan sendiri adalah sebuah proses dimana dilakukan perhitungan 1 sampai 9 pada baris, kolom, dan sub-matriks untuk menemukan angka yang “hilang”. Perhitungan yang dimulai dari angka terakhir yang ditemukan dapat lebih meningkatkan kecepatan proses ini. Pada beberapa soal yang memiliki tingkat kerumitan tinggi, akan lebih baik jika perhitungan dilakukan terhadap angka yang tidak mungkin ada di baris, kolom, atau sub-matriks tersebut.



Gambar 3. Proses memeriksa pada sudoku

Proses memeriksa dihentikan ketika tidak ditemukan lagi angka-angka lebih lanjut. Suatu metoda lanjutan dari proses ini adalah memberikan tanda berupa angka-angka yang mungkin pada sel-sel yang masih kosong. Pemberian tanda inipun terdiri dari 2 macam, yaitu pemberian tanda berupa notasi titik dan pemberian tanda berupa angka.

Pada pemberian tanda berupa notasi titik, lokasi titik yang terdapat pada sel yang kosong menunjukkan angka mulai dari 1 hingga 9.



Gambar 4. Pemberian tanda berupa titik

Sedangkan pada pemberian tanda berupa angka, angka-angka yang menjadi kandidat pada sel kosong tersebut dituliskan sebagai *subscript* pada sel kosong tersebut. Penggunaan dua warna yang berbeda sangatlah dianjurkan.

Gambar 5. Pemberian tanda berupa angka

Pendekatan utama pada proses selanjutnya adalah eliminasi kandidat dan skema "apa yang terjadi jika". Pada proses eliminasi kandidat, berlangsung eliminasi kandidat-kandidat angka pada sel kosong tersebut hingga hanya meninggalkan satu kandidat yang layak untuk sel tersebut. Pada skema "apa yang terjadi jika" biasanya terdapat dua pilihan angka dan dilakukan penebakan pada kedua angka tersebut.

5. PENYELESAIAN DENGAN BRUTE FORCE

Cara paling naif di dalam menyelesaikan persoalan sudoku dengan menggunakan algoritma *bruteforce* adalah dengan cara memeriksa semua kemungkinan pengisisan sel oleh angka 1 hingga 9 tanpa memperhatikan batasan-batasan yang ada di dalam permainan sudoku. Cara tersebut juga dinamakan teknik algoritma *bruteforce* murni. Pada teknik ini dilakukan pengecekan terhadap kemungkinan yang menyalahi aturan dari sudoku yaitu penempatan angka yang sama pada satu baris, kolom, ataupun sub-matriks. Sehingga apabila diperhitungkan, akan terdapat sekitar 9^{81} buah kemungkinan yang muncul, atau sekitar 1,9 x 10^{77} buah kemungkinan. Pada kasus terburuk, solusi permasalahan sudoku akan ditemukan pada proses pengecekan ke 9^{81}. Jelas sekali terlihat bahwa penyelesaian sudoku dengan menggunakan algoritma *bruteforce* murni bukanlah penyelesaian sudoku yang mangkus, sehingga kemudian ditemukan proses penyelesaian sudoku dengan algoritma yang lain, yaitu teknik *bruteforce constraint.*

Pada teknik ini, dilakukan proses eliminasi terhadap semua kemungkinan susunan angka yang menyalahi aturan dari permainan sudoku. Sehingga semua kemungkinan yang diperhitungkan hanyalah kemungkinan yang mengandung angka unik pada satu baris, kolom, dan sub-matriks yang ada.

Dengan menggunakan teknik *bruteforce constraint* ini, dapat dilakukan eliminasi kemungkinan hingga hanya tersisakan sekitar 6.670.903.752.021.072.936.960 kemungkinan, atau sekitar 6,67 x 10^{21} kemungkinan yang tidak menyalahi aturan permainan sudoku.

Hal ini menunjukkan bahawa penggunaan algoritma *bruteforce constraint* mampu menekan waktu dan jumlah pengecekan yang muncul, sehingga algoritma ini dianggap memiliki tingkat kemangkusan yang lebih bila dibandingkan dengan algoritma *bruteforce* murni.

6. PENYELESAIAN DENGAN BACKTRACKING

Algoritma Runut-balik (*backtracking*) berbasis pada *DFS* untuk mencari solusi permasalahan secara lebih mangkus. Runut-balik, yang merupakan perbaikan dari algoritma *brute-force*, secara sistematis mencari solusi permasalahan di antara semua kemungkinan solusi yang ada. Hanya pencarian yang mengarah pada solusi saja yang selalu dipertimbangkan. Akibatnya waktu pencarian solusi dapat menjadi lebih hemat.

Skema umum algoritma runut balik adalah :

- Meninjau pencarian solusi pada pohon ruang status yang dibangun secara dinamis
- Solusi dicari dengan membentuk lintasan dari akar ke daun menggunakan DFS.
- Tiap kali simpul-E (simpul yang sedang diperluas) diperluas, lintasan yang dibangun bertambah panjang. Bila ternyata tidak mengarah ke solusi, maka simpul tersebut dibunuh (simpul mati).

Penggunaan algoritma runut-balik di dalam proses penyelesaian soal sudoku sendiri dapat dilihat sebagai berikut,

Setiap elemen diisi dengan nilai pertama yang memungkinkan mulai dari angka 1 hingga angka maksimal yaitu n*n. Jika angka pada elemen tersebut telah dimasukkan, maka kita akan bergerak ke elemen berikutnya(di baris yang sama kolom selanjutnya) dengan pengisian nilai yang juga memiliki aturan yang sama dengan sebelumnya. Yaitu harus mengisi elemen tersebut dengan nilai yang memungkinkan dimulai dengan angka 1 hingga n*n.

Apabila suatu elemen tidak bisa diisi dengan semua angka (1 hingga n*n) atau deadlock, maka program akan melakukan runut balik ke elemen sebelumnya. Kemudian elemen tersebut diisi dengan nilai yang memungkinkan selain angka yang ada tersebut. Angka yang dimasukkan adalah angka yang lebih besar dari angka sebelumnya. Jika telah menemukan angka yang memngkinkan, maka angka tersebut akan dimasukkan dan maju ke elemen selanjutnya. Algoritma tersebut dilakukan hingga mencapai elemen terakhir atau kondisi tidak memungkinkan lagi untuk meneruskan algoritma (tidak ada solusi).

Berdasarkan analisa yang dilakukan, dapat ditarik kesimpulan bahwa hasil yang didapat dari pencarian solusi Sudoku dengan menggunakan Algoritma Runut balik telah mampu memberikan jawaban pertama yang benar jika soal di jamin mempunyai solusi. Jika dibandingkan dengan solusi brute force yang mencoba semua kemungkinan kombinasi yang ada, maka algoritma runut balik ini jauh lebih mangkus dan sangkil. Sayangnya untuk ukuran Sudoku yang berukuran besar, dibutuhkan waktu yang sangat lama untuk pencarian solusinya.

7. KESIMPULAN

Algoritma runut-balik merupakan salah satu teknik pencarian solusi yang terbilang cukup mangkus. Hal ini disebabkan oleh sifat dari algoritma runut-balik yang hanya mempertimbangkan pencarian yang mengarah pada solusi saja. Bila dibandingkan dengan algoritma *bruteforce*, yang mempertimbangkan semua kemungkinan yang ada, waktu yang dibutuhkan oleh algoritma runut-balik lebih hemat. Lebih lagi untuk mencari solusi penyelesaian permainan Sudoku dengan nilai n yang besar.

REFERENSI

[1] Rinaldi Munir, Diktat Kuliah Matematika Diskrit. Departemen Teknik Informatika ITB : 2005

[2] Sudoku. (2007), http://en.wikipedia.org/wiki/Sudoku. *Tanggal akses: 20 Mei 2007 pukul 16:00.*

[3] Liem, Inggriani. 2003**.** *Diktat Kuliah IF1281 Algoritma dan Pemrograman.* Bandung : ITB, 2003.