SUDOKU SOLVER MENGGUNAKAN METODE BACKTRACK

Arief Budiman Eka Putra (10112636), **Fernando Winata (10112740)**

& **Sani Saefurochman (10112452)**

Universitas Komputer Indonesia

Teknik Informatika

Bandung, Indonesia

ariefbud688@gmail.com, winata.nando@gmail.com, darkevilxcries@gmail.com

ABSTRAK

Sudoku kadang kali membuat para pemainnya kebingungan dalam penyelesaiannya, apakah sudah benar atau belum, dan dirasa perlu waktu yang lama untuk menyelesaikannya.

Banyak metode yang digunakan untuk menyelesaikan sudoku seperti brute force, backtracking, dsb. Maka dibutuhkan suatu penelitian untuk menentukan metode mana yang kiranya mudah dan cepat dalam penggunaannya untuk menyelesaikan sudoku.

Kami memutuskan untuk mengetes salah satu metode yang biasa digunakan untuk menyelesaikan sudoku yaitu metode backtracking.

I. INTRODUCTION

Backtracking (Runut Balik) sendiri merupakan metode yang berbasis pada DFS (Depth First Search) dimana pencarian dilakukan pada leaf (tree)/di akhir baru kemudian naik keatas / ke node yang lebih tinggi, metode ini bertujuan untuk mencari solusi dari suatu masalah secara lebih efisien.

Backtracking sendiri merupakan perbaikan dari metode brute force,dimana hanya pencarian yang mengarah langsung ke solusi saja yang dipertimbangkan , berbeda dengan brute force diman semua kemungkina dicoba untuk mendapat solusi.

Dalam penelitian ini metode backtraking digunakan untuk mencari solusi/angka yang sekiranya belum terpakai dalam sudoku dengan grid 9 X 9, dimana sudoku sendiri merupakan sebuah matriks ordo 9 x 9, pencarian dilakukan dari matriks akhir.

II. PENJELASAN SINGKAT

A. SUDOKU

Sudoku merupakan permainan puzzle yang berbasiskan angka. Tujuan dari permainan ini adalah mengisi blok-blok dengan angka dimana angka-angka tersebut tidak boleh sama dalam satu kolom atau baris dalam blok tersebut.

Permainan seperti sudoku sudah dikenal sejak tahun 1979 di majalah *Dell Magazines* dengan nama "Number Place". Permainan ini didesain oleh Howard Grans. Permainan ini dikenal di Jepang pada tahun 1984, dimuat di Majalah Bulanan *Nikoli*, dengan nama "Suuji Wa Dokushin Kagiru". Selanjutnya puzzle ini dikenal sebagai Su-Doku. 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.

Lalu "sudoku craze" pun menjalar ke segala penjuru dunia, 222 tahun setelah "Latin Square". Wayne Gould, sendirian saja menyuplai ke 140 surat kabar dunia. Belum lagi master-master sudoku lain yang terus bermunculan. Koran-koran Inggris: Daily Mail, Telegraph, Times, Guardian, dan Independent, masing-masing menerbitkan kumpulan soal-soal sudoku dalam beberapa seri buku. Di Amerika Serikat buku-buku Sudoku masuk "top ten best seller". Di amazon.com, toko buku online terbesar, ada 314 judul buku Sudoku. Hanya

di Inggris, dua juta buku sudoku terjual dalam kurun waktu kurang dari setahun (Reuters, 22 Oktober 2005).

Internet pun semakin ramai. Situs-situs baru bermunculan, menyajikan soal sudoku maupun informasi lain, baik sejarah, cara-cara penyelesaian, bahkan software untuk komputer maupun handphone. Jika tahun 2005 kita ketikkan "sudoku" di Yahoo atau Google, hasil pencarian yang diberikan hanya puluhan. Pada tanggal 11 April 2006, Yahoo merespons hampir 36 juta, sedang Google memberi hasil pencarian lebih dari 93 juta untuk sudoku. Dan sekarang tentunya masih terus bertambah.

Ini fenomena tersendiri. Bukan computer game dengan tampilan grafis memikat. Hanya sebuah puzzle sederhana. Dalam kurun satu tahun bisa merambah seluruh benua, di sukai tua muda. Hampir semua koran nasional di setiap negara menyajikan Sudoku. Ada yang menganggapnya sebagai Rubik's Cube abad 21, mengacu pada popularitas permainan asal Rusia pada 1980-an.

Sejak saat itu permainan Sudoku menjadi mendunia dan menjadi booming di mana-mana.

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 berordo 9x9 yang belum terisi penuh penuh.

(Sumber: http://en.wikipedia.org/wiki/Sudoku)

Daya tarik Sudoku adalah kesederhanaannya. Pemain hanya mengisi setiap sel dengan angka 1 sampai 9 tanpa tertulis dua kali dalam lajur-lajur dan kotak yang sama. Tidak ada penjumlahan atau hitungan apa pun.

Selain menyenangkan, sudoku juga dapat bermanfaat bagi otak. Sudoku lebih menekankan pada permainan logika, maka dibutuhkan logika yang tepat untuk menyelesaikannya (tentunya tiap level kesulitannya, juga membutuhkan tingkat logika yang berbeda).

Dari sini, kita melatih otak kiri kita yang bekerja secara sistematis. Bukan hanya berhenti di situ, dengan bertambahnya level kesulitannya, kita juga dituntut kreatif mencari solusi penyelesaian materi. Dari sisi ini, kita dilatih juga menggunakan otak kanan yang bekerja secara kreatif dan solutif.

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 sudoku berordo 9x9 yang telah terisi penuh

(Sumber: http://en.wikipedia.org/wiki/Sudoku)

Permainan sudoku disamping untuk mengisi waktu luang juga bermanfaat dalam mengembangkan otak kita. Salah satu manfaat dari permainan Sudoku ini diharapkan anda dapat mengembangkan cara berpikir dan nalar yang benar sesuai dengan logika. Dasar untuk berpikir dan menalar yang baik perlu juga menjalankan prinsip analisis, disini sudoku juga melatih kita untuk mempertajam anilisis dalam menyelesaikan suatu masalah. Dengan demikian mempelajari

sudoku diharapkan kita mampu untuk mengambil keputusan yang tepat berdasarkan analisis dan pertimbangan – pertimbangan fungsional.

Beberapa manfaat lain dari permainan sudoku adalah sebagai berikut:

1. Mengembangkan cara berpikir dan nalar yang benar sesuai dengan logika.
2. Mempertajam anilisis kita dalam menyelesaikan suatu masalah.
3. Mengambil keputusan yang tepat berdasarkan analisis dan pertimbangan-pertimbangan fungsional.
4. Mengembangkan pola berfikir interdisipliner dalam menghadapi masalah.
5. Melatih ketajaman, ketelitian dan kesabaran dalam menyelesaikan masalah.
6. Mengembangkan diri secara mandiri sehingga dapat mengikuti perkembangan ilmu pengetahuan secara global
7. Melatih otak kiri kita yang bekerja secara sistematis.
8. Melatih otak kanan kita yang bekerja secara kreatif.
9. Untuk mengisi waktu luang dan sebagai sarana hiburan ataupun hobi yang bermanfaat dalam mempertajam otak.

Permainan sudoku dapat berupa blok/grid yang berbentuk ordo matriks 4x4; 6x6; 8x8; 9x9; 16x16; dan lain-lain, dengan syarat angka ordo blok dalam sudoku adalah bilangan yang hasilnya habis/berbentuk bilangan bulat apabila dibagi dengan angka-angka berikut: 2,3,5. Apabila hasilnya bukan merupakan bilangan bulat/pecahan, maka ordo tersebut tidak dapat dijadikan sebagai sudoku. Sebagai contoh, ordo 7x7 tidak dapat dijadikan sebagai blok sudoku, karena hasil pembagian 7 dengan bilangan 2, 3, atau 5 menghasilkan bilangan pecahan, sedangkan untuk ordo 9x9 dapat dijadikan sebagai sudoku karena hasil pembagian 9 dengan 3 menghasilkan bilangan bulat.

Dari uraian diatas dapat disimpulkan bahwa.

1. Semua bilangan genap kecuali angka 2 dapat dijadikan sudoku.
2. Semua bilangan ganjil kecuali kelipatan dari 5 (10, 15, 20, 25, 30, 35 …dst) tidak dapat dijadikan sudoku.
3. Angka 1, 2, 3 dan 5 tidak dapat dijadikan sudoku.
4. Angka 1, 2, 3 dan 5 tidak dapat dijadikan sudoku.

Dalam penelitian ini, kami membatasi ordo dalam program sudoku, yaitu berukuran 9x9.

B. ALGORITMA BACKTRACK

Istilah backtrack pertama kali diperkenalkan oleh D.H. Lehmer pada tahun 1950. Selanjutnya R.J. Walker, Golomb, dan Baumert menyajikan uraian umum tentang algoritma ini serta penerapannya pada berbagai persoalan. [2]

Algoritma backtrack merupakan algoritma yang berbasis pada DFS (Depth First Search). Algoritma ini merupakan perbaikan dari algoritma brute force. Algotitma backtracking ini tidak memeriksa semua kemungkinan solusi yang ada, hanya pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan, sehingga waktu pencarian akan lebih pendek/dihemat. Ada beberapa metode yang digunakan dalam algoritma Backtrack ini, yaitu algoritma secara rekursif dan algoritma backtrack secara iteratif.

Untuk menerapkan algoritma backtrack ini, diperlukan beberapa properti berikut ini.

1. Solusi persoalan
   $X = (x_1, x_2, x_3, ..., x_n)$
   Solusi persoalan dinyatakan dengan vektor.
2. Fungsi pembangkit
   Fungsi ini digunakan untuk membangkitkan komponen vektor solusi yang dinyatakan sebagai fungsi T(k). T(k) inilah yang membangkitkan nilai k.
3. Fungsi pembatas/fungsi kriteria
   Fungsi pembatas menentukan apakah (x_1, x_2, x_3, ..., x_n) mengarah kepada solusi atau tidak.

Algoritma backtrack mengorganisasi ruang solusi dengan struktur tree (pohon). Tiap simpul pohon menyatakan status persoalan, sedangkan sisi pohon menyatakan nilai-nilai x_i.

Pada prinsipnya, algoritma backtrack mencari solusi dengan membentuk lintasan dari akar ke daun. Aturan yang dipakai adalah mengikuti pencarian pada DFS. Simpul yang sudah dilahirkan dinamakan dengan "simpul hidup". Simpul hidup yang sedang diperluas dinamakan "simpul-E (expand-node)". Simpul dinomori sesuai dengan urutan pembangkitnya.

Setiap simpul-E diperluas, lintasan yang dibangun bertambah panjang. Jika lintasan yang dibentuk tidak mengarah kepada solusi maka simpul-E dihapus. Fungsi yang digunakan untuk menghapus simpul-E adalah fungsi pembatas/fungsi kriteria. Simpul yang sudah mati tidak akan diperluas lagi.

Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian diteruskan dengan membangkitkan simpul anak yang lainnya. Bila tidak ada simpul anak yang lainnya maka soludi dilanjutkan dengan melakukan runut-balik ke simpul hidup yang terdekat. Aaselanjutnya simpul ini menjadi simpul hidup yang baru. Lintasan baru dibangun kembali hingga lintasan tersebut membentuk solusi.

Pencarian dihentikan apabila tidak menemukan solusi atau tidak ada lagi simpul hidup untuk proses backtrack. [1]

III. BACKTRACKING SUDOKU

A. PENERAPAN METODE BACKTRACK DALAM APLIKASI SUDOKU

Berikut ini adalah penerapan algoritma runut-balik (backtrack) dalam penyelesaian sudoku puzzle adalah sebagai berikut:

1. Algoritma dimulai pada elemen kosong pertama pada matriks.
2. Periksa seluruh kemungkinan angka (1-9) yang dapat diisi oleh elemen kosong tersebut dengan memeriksa batasan.
3. Jika nilai tersebut memenuhi fungsi pembatas (*valid*), maka elemen kosong tersebut diisi dengan nilai *n*, dan lanjutkan pemeriksaan ke elemen kosong berikutnya.
4. Jika nilai *n* tersebut tidak memenuhi fungsi pembatas, maka uji dengan nilai *n* lain.
5. Jika seluruh nilai *n* telah diuji dan tidak ada nilai *n* yang memenuhi fungsi pembatas, maka lakukan backtracking ke elemen sebelumnya.
6. Elemen ini akan diuji lagi dengan nilai *n* baru berdasarkan fungsi pembatas.
7. Lakukan cara yang sama dengan poin nomor 3.
8. Proses diatas akan dilakukan terus-menerus secara rekursif hingga ditemukan suatu solusi atau tidak ditemukan suatu solusi.
9. Suatu solusi dinyatakan benar jika seluruh elemen kosong telah diisi dengan nilai *valid*.

B.DEKLARASI GLOBAL

```
const int INFINITY=32000;
struct trio
{
	int row;
	int col;
	int value;
};
```

Pendeklarasian di atas adalah variabel utama yang dibutuhkan untuk menentukan baris, kolom dan nilai yang akan digunakan di Sudoku.

C. INPUT POLA SUDOKU

```
for(int i=0;i<9;i++)
{
    for(int j=0;j<9;j++)
    {
	cin>>h;
	if(h!='0' && isdigit(h))
		grid.at(i).at(j)=int(h-
		'1');
    }
}
```

Sudoku yang digunakan dalam program ini yakni matriks berordo 9x9 yang merupakan bentuk standar dari Sudoku. Algoritma diatas digunakan untuk menentukan pola tersebut.

D. FUNGSI VALIDASI

```
bool isValid(const int &row,const int &col,
	const int &value)
{
	return (row>=0 && row<9 && col>=0 &&
		col<9 && value>=0 && value<9);
}
```

```
bool valid(const vector<vector<int> >
           &thegrid, const int &row, const
           int &col, const int &val)
{
    if(thegrid.at(row).at(col)==val)
          return true;
    if(thegrid.at(row).at(col)!=INFINITY)
          return false;
    for(int i=0;i<9;i++)
    if(thegrid.at(i).at(col)==val)
    return false;
    for(int j=0;j<9;j++)
    {
       if(thegrid.at(row).at(j)==val)
       return false;
    }
    int m,n;
    m=(row/3)*3;
    n=(col/3)*3;

    for(int i=m;i<m+3;i++)
      for(int j=n;j<n+3;j++)
        if(thegrid.at(i).at(j)==val)
        return false;

    return true;
}

bool cek_trio(const trio &sample)
{
    Return
    isValid(sample.row,sample.col,sample.va
    lue);
}
```

Fungsi diatas digunakan untuk membatasi program yang dibuat atau biasa disebut validasi.

E.*FUNGSI UTAMA (BACKTRACKING)*

Dalam penyelesaian menggunakan metode backtrack struktur algoritma dasar yang digunakan adalah sebagai berikut

```
void  process(vector<vector<int> > thegrid,
           int row, int col, int val)
{
    if(done || !isValid(row,col,val) ||
       !(valid(thegrid,row,col,val)))
       return;
    thegrid.at(row).at(col)=val;
    if(row==8 && col==8)
    {
       cout<<"Selesai: "<<endl;
       grid=thegrid;
       done=true;
    }
    bool flag=false;
    for(int i=0;i<9;i++)
    if(col<8)
       process(thegrid,row,col+1,i);
       else
          process(thegrid,row+1,0,i);
}

bool initProcess()
{
      for(int i=0;i<9;i++)
      for(int j=0;j<9;j++)
      {
            int temp=grid.at(i).at(j);
            grid.at(i).at(j)=INFINITY;
            if(!valid(grid,i,j,temp))
            {
                  error(2);
                  return false;
            }
            grid.at(i).at(j)=temp;
      }
      for(int i=0;i<9;i++)
            process(grid, 0, 0, i);
      return true;
}
```

Pada sintaks diatas digunakan metode backtrack secara rekursif.

F. FUNGSI CONTOH INPUT

Selain diinput oleh user, program sudoku ini pun dapat menampilkan contoh input beserta penyelesaiannya. Adapun syntax programnya adalah sebagai berikut.

```
void contoh_sudoku()
{
      cout<<"Contoh Input : ";
      string str=" 26 "
            "1 4 5 "
            "1 5 8 "
            "2 3 1 "
            "2 9 7 "
            "3 1 3 "
```

```
        "3 6 4 "
        "3 7 9 "
        "3 8 5 "
        "3 9 2 "
        "4 1 4 "
        "4 5 7 "
        "4 7 8 "
        "5 1 9 "
        "5 9 5 "
        "6 3 2 "
        "6 5 1 "
        "6 9 6 "
        "7 1 1 "
        "7 2 2 "
        "7 3 3 "
        "7 4 7 "
        "7 9 9 "
        "8 1 6 "
        "8 7 4 "
        "9 5 5 "
        "9 6 2 " ;
istringstream istr(str);
cout<<"Posisi yang diketahui : ";
int n;
istr>>n;
cout<<n<<endl;
trio sample;
for(int i=0;i<n;i++)
{

        istr>>sample.row>>sample.col>>sam
        ple.value;
        sample.row--;
        sample.col--;
        sample.value--;
        if(cek_trio(sample))
        {

                grid.at(sample.row).at(sam
                ple.col)=sample.value;
```

```
        }
        else
        i--;
}
cout<<"GRID input Sudoku :(B = Blank /
kosong)"<<endl;
tampil(grid);
```

IV. KESIMPULAN

Algoritma runut-balik (backtracking) merupakan algoritma yang cukup mangkus untuk menyelesaikan berbagai persoalan.

Hal ini disebabkan pada algoritma runut-balik, tidak perlu mencari semua kemungkinan solusi untuk diuji. Hanya kemungkinan solusi yang mengarah pada solusi yang dipertimbangkan saja yang perlu diuji, sehingga algoritma ini cukup mangkus untuk digunakan. Berbeda dengan algoritma brute-force yang mencari semua kemungkinan solusi untuk diuji, sehingga tidak mangkus.

Oleh karena itu, algoritma runut-balik banyak diterapkan dalam berbagai persoalan, terutama program game dan persoalan artificial intelligence.

Hasil analisis penggunaan algoritma runut-balik dalam menyelesaikan persoalan pengisian angka-angka Sudoku menunjukkan bahwa algoritma ini cukup mangkus untuk mendapatkan solusi persoalan tersebut

Sistem kerja algoritma runut-balik yang sistematis dan ciri khasnya yang hanya memeriksa kemungkinan solusi yang memang dapat dipertimbangkan untuk menjadi solusi akhir, diperkirakan dapat menjadi solusi yang efektif dan efisien.

DAFTAR PUSTAKA

[1] Rinaldi Munir, *Strategi Algoritmik*. Bandung, Indonesia, 2005.

[2] Ratih Dewi Pramudi Ismi, "Penggunaan Algoritma Backtracking Untuk Menentukan Keisomorfikan Graf".