Penerapan Algoritma Breadth First Search pada Eliminasi Gauss-Jordan

Penerapan BFS untuk mencari solusi dari sistem persamaan linear dengan jumlah operasi paling sedikit

Kristianto Karim - 13514075
Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514075@stei.std.itb.ac.id

Abstraksi—Makalah ini akan membahas mengenai penerapan algoritma pencarian secara melebar atau yang dikenal dengan Breadth First Search Algorithm pada penyelesaian masalah sistem persamaan linear multivariable yang direpresentasikan dalam bentuk matriks dengan operasi barisan elementer. Operasi barisan elementer adalah operasi untuk mereduksi matriks menjadi matriks eselon tereduksi.

Kata kunci—bfs;pencarian;operasi bilangan elementer;obe; gauss;gauss-jordan;eliminasi;sistem persamaan linear;persamaan linear

I. PENDAHULUAN

Sistem persamaan linear banyak digunakan dalam banyak hal terutama yang berhubungan dengan pencarian suatu nilai dengan beberapa syarat persamaan.

Akibat banyaknya penggunaan sistem persamaan linear maka terdapat juga berbagai cara untuk menyelesaikan persoalan sistem persamaan linear ini, salah satunya yaitu dengan Metode Operasi Barisan Elementer. Metode ini menerapkan eliminasi Gauss-Jordan atau eliminasi Gauss untuk mendapatkan matriks solusi. Masalahnya yaitu sering terjadi operasi – operasi yang redundan pada saat proses eliminasi sehingga penyelesaian menjadi tidak efektif. Oleh karena itu, digunakanlah metode pencarian solusi BFS untuk mendapatkan proses reduksi yang paling efektif.

II. TEORI DASAR

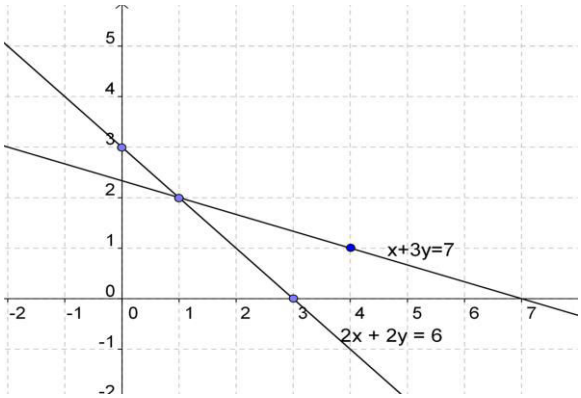
A. *Persamaan Linear*

Persamaan linear adalah sebuah persamaan aljabar, yang tiap sukunya mengandung konstanta, atau perkalian konstanta dengan variabel tunggal. Persamaan ini dikatakan linear sebab hubungan matematis ini dapat digambarkan sebagai garis lurus dalam Sistem koordinat Kartesius.

Bentuk umum persamaan linear adalah

$$y = mx \pm c$$

Jika digambarkan pada koordinat kartesius maka akan tampak seperti gambar dibawah :



Gambar 1 : Persamaan Linear pada Koordinat Kartesius

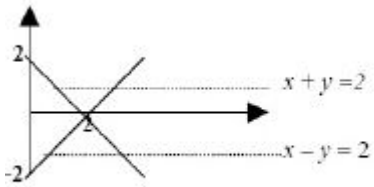
Sumber : https://mumathica.files.wordpress.com/2013/01/grafik2.jpg

B. *Sistem Persamaan Linear*

Sistem persamaan linear adalah suatu sistem yang terdiri dari beberapa persamaan linear.

Sistem persamaan linear dapat memiliki solusi unik, solusi banyak ataupun tidak memiliki solusi.

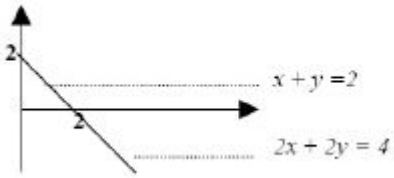
Solusi unik yaitu dimana pada saat solusi dari sistem persamaan linear hanya memiliki 1 solusi saja (berpotongan pada 1 titik)



Gambar 2 : Solusi unik

Sumber : http://image.slidesharecdn.com/bab1-120724031520-phpapp01/95/bab-1-2-728.jpg?cb=1343099756

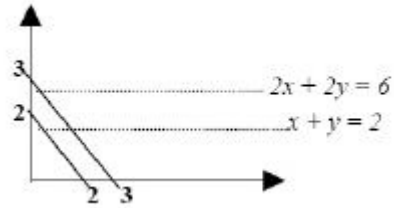
Solusi banyak dimana solusinya sangat banyak karena berpotongan pada bnayak titik.



Gambar 3 : Solusi banyak

Sumber : http://image.slidesharecdn.com/bab1-120724031520-phpapp01/95/bab-1-2-728.jpg?cb=1343099756

Tidak ada solusi yaitu dimana sistem persamaan linear tidak memiliki titik yang berpotongan pada saat digambarkan pada koordinat kartesius



Gambar 4 : Tidak ada solusi

Sumber : http://image.slidesharecdn.com/bab1-120724031520-phpapp01/95/bab-1-2-728.jpg?cb=1343099756

C. Operasi Barisan Elementer

Operasi barisan elementer merupakan salah satu cara penyelesaian sistem persamaan elementer dengan menerapakan eliminasi Gauss ataupun eliminasi Gauss-Jordan.

- Eliminasi Gauss

  Eliminasi Gauss adalah suatu cara mengoperasikan nilai-nilai di dalam matriks sehingga menjadi matriks yang lebih sederhana . Caranya adalah dengan melakukan operasi baris sehingga matriks tersebut menjadi matriks yang Eselon-baris.

  Contoh :

$$x + 2y + z = 6$$

$$x + 3y + 2z = 9$$

$$2x + y + 2z = 12$$

Jadikan sistem persamaan linear tersebut menjadi bentuk matriks :

$$\begin{bmatrix} 1 & 2 & 1 & 6 \\ 1 & 3 & 2 & 9 \\ 2 & 1 & 2 & 12 \end{bmatrix}$$

Kemudian terapkan eliminasi Gauss :

$$\begin{bmatrix} 1 & 2 & 1 & 6 \\ 1 & 3 & 2 & 9 \\ 2 & 1 & 2 & 12 \end{bmatrix}$$ B2-B1

$$\begin{bmatrix} 1 & 2 & 1 & 6 \\ 0 & 1 & 1 & 3 \\ 2 & 1 & 2 & 12 \end{bmatrix}$$ B3-B1*2

$$\begin{bmatrix} 1 & 2 & 1 & 6 \\ 0 & 1 & 1 & 3 \\ 0 & -3 & 0 & 0 \end{bmatrix}$$ B3+B2*3

$$\begin{bmatrix} 1 & 2 & 1 & 6 \\ 0 & 1 & 1 & 3 \\ 0 & 0 & 3 & 9 \end{bmatrix}$$ B3/3

$$\begin{bmatrix} 1 & 2 & 1 & 6 \\ 0 & 1 & 1 & 3 \\ 0 & 0 & 1 & 3 \end{bmatrix}$$

Sehingga didapatkan suatu matriks eselon yang jika kita terapkan sulir balik, akan menghasilkan solusi dari persamaan linear tersebut.

- Eliminasi Gauss-Jordan

  Eliminasi Gauss-Jordan adalah pengembangan dari eliminasi Gauss yang hasilnya lebih sederhana. Caranya adalah dengan meneruskan operasi baris dari eliminasi Gauss sehingga menghasilkan matriks yang Eselon-baris tereduksi.

  Contoh :

$$x + 2y + 3z = 3$$

$$2x + 3y + 2z = 3$$

$$2x + y + 2z = 5$$

Jadikan ke dalam bentuk matriks

$$\begin{bmatrix} 1 & 2 & 3 & 3 \\ 2 & 3 & 2 & 3 \\ 2 & 1 & 2 & 5 \end{bmatrix}$$

Kemudian terapkan eliminasi Gauss-Jordan :

$$\begin{bmatrix} 1 & 2 & 3 & 3 \\ 2 & 3 & 2 & 3 \\ 2 & 1 & 2 & 5 \end{bmatrix}$$ B2-B1*2

$$\begin{bmatrix} 1 & 2 & 3 & 3 \\ 0 & -1 & -4 & -3 \\ 2 & 1 & 2 & 5 \end{bmatrix}$$ B3-B1*2

$$\begin{bmatrix} 1 & 2 & 3 & 3 \\ 0 & -1 & -4 & -3 \\ 0 & -3 & -4 & -1 \end{bmatrix}$$ B3-B2*3

$$\begin{bmatrix} 1 & 2 & 3 & 3 \\ 0 & -1 & -4 & -3 \\ 0 & 0 & 8 & 8 \end{bmatrix}$$ B3/8 dan B2*-1

$$\begin{bmatrix} 1 & 2 & 3 & 3 \\ 0 & 1 & 4 & 3 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$ B2 – B3*4

$$\begin{bmatrix} 1 & 2 & 3 & 3 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$ B1-B3*3

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$ B1-B2*2

$$\begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Sehingga didapatkan sebuah matriks eselon tereduksi yang adalah solusi dari sistem persamaan linier.

D. Algoritma Pencarian

Algoritma pencarian terdapat beberapa yaitu :

1. Breadth-First Search
2. Depth-First Search
3. Iterative Deepening Depth-First Search
4. A*
5. Best First Search
6. Uniform Cost Search

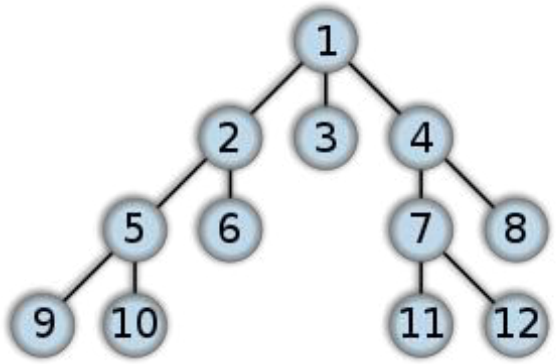
Yang dapat digunakan untuk mencari jarak terpendek dari simpul akar ke simpul solusi yaitu Breadth-Firsth Search dan Iterative Deepening Depth-First Search. Pada makalah ini kita akan menggunakan metode Breadth-First Search untuk mencari jumlah operasi minimal yang diperlukan agar ditemukan solusi.

E. Breadth-First Search

Breadth first search sebuah algoritma pencarian graf yang dimulai dari node pangkal dan menjelajahi semua node yang berdekatan.dan untuk setiap node yang berdekatan, bfs menjelajahi node-node yang tidak terlihat sebelumnya (unexplored) dan seterusnya.

Algoritmanya yaitu

1. Masukan node pangkal ke queue
2. Ambil node dari queue kemudian masukan semua node anak dari node yang diambil ke dalam queue
3. Ulangi langkah 2 sampai ditemukan node solusi atau queue kosong



Gambar 5 : Urutan node yang dikunjungi dengan BFS

Sumber : Wikipedia.org

III. IMPLEMENTASI

Implementasi penerapan BFS pada eliminasi gauss-jordan membutuhkan suatu struktur data simpul pohon yang menyimpan alamat dari orang tua dari simpul tersebut agar dapat dilakukan backtrack dan masuk ke simpul unexplored selanjutnya dan yang menyimpan status matriks.

Pohon status dibangun pada waktu pencarian. Pencarian selesai jika pohon status tidak dapat di bangun lagi yaitu pada saat matriks tidak dapat dioperasikan lagi.

Algoritma secara umumnya adalah sebagai berikut :

1. Masukan matriks soal ke dalam antrian
2. Ambil 1 isi antrian paling depan (kita sebut dengan A)
3. lakukan berbagai operasi barisan pada A, (misalkan eliminasi baris 3 dengan baris 1)
4. masukan matriks baru ke dalam antrian
5. ulangi 3 - 4 sampai semua kemungkinan operasi pada A dilakukan

6. ulangi 2-6 sampai ditemukan matriks solusi

Berikut implementasi dalam bahasa pemograman java :

```
package bfsinobe;
import static java.lang.System.arraycopy;
import java.util.LinkedList;
/**
 * @author Kris
 */
class Antrian extends LinkedList<Simpul>
{
   public Antrian()
   {

   }
   public void AddQueue(Simpul x)
   {
      this.addLast(x);
   }
   public Simpul DelQueue()
   {
      return this.poll();
   }
}
class Simpul
{
   private final double[][] matriks;
   private final Simpul ortu;
   private final int row,col;

   public Simpul(double[][] mat, Simpul ot, int r, int c) {
      row = r; col = c;
      matriks = new double[row][col];
      for (int i =0 ; i < row ; i++)
      {
         System.arraycopy(mat[i], 0, matriks[i], 0, col);
      }
      ortu = ot;
   }
   public int getCol() {
      return col;
   }
   public int getRow() {
      return row;
   }
   public double[][] getMatriks() {
      return matriks;
   }
   public Simpul getOrtu() {
      return ortu;
   }
}

public class BFSinOBE {

   private Antrian antrian = new Antrian();
```

```
   boolean isSolusi(Simpul sim)
   {
      if (sim == null)
      {
         return false;
      }
      int [] countzero = new int[sim.getRow()];
      for (int i = 0; i < sim.getRow() ; i++)
      {
         int j =0;
         while((j < sim.getCol())
&&(sim.getMatriks()[i][j] == 0))
         {
            countzero[i]++;
         }
      }
      int conx = 0;
      for (int i = 0 ; i < sim.getRow() ; i++)
      {
         if (countzero[i] >= i)
         {
            conx++;
         }
      }
      boolean cek1 = false;
      if (conx  == sim.getRow())
      {
         cek1 = true;
      }
      countzero = new int[sim.getRow()];
      for (int i = 0; i < sim.getRow() ; i++)
      {
         int j = sim.getCol() -1;
         while((j >= 0) &&(sim.getMatriks()[i][j] == 0))
         {
            countzero[i]++;
         }
      }
      conx = 0;
      for (int i = 0 ; i < sim.getRow() ; i++)
```

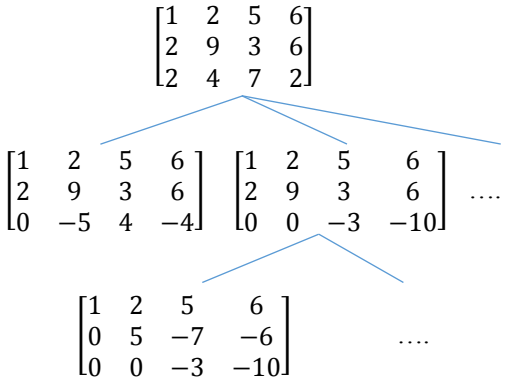
```
        {
            if (countzero[i] >= sim.getRow()-1 - i)
            {
                conx++;
            }
        }
        boolean cek2 = false;
        if (conx  == sim.getRow())
        {
            cek2 = true;
        }
        return (cek1 && cek2);
    }

    boolean cekKosong(double[] arr, int c)
    {
        int x = 0;
        for (int i = 0 ; i < c ; i++)
        {
            if (arr[i] == 0)
            {
                x++;
            }
        }
        if (x == c)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    LinkedList<Simpul> BFSinOBE(double[][] soal, int r,
int c)
    {
        Simpul awal = new Simpul(soal,null,r,c);
        antrian.AddQueue(awal);
        Simpul proses;
        do
        {
            proses = antrian.DelQueue();
            double[][] newMat = new double[r][c];
            for (int i = 0 ; i < r ; i++)
            {
                System.arraycopy(proses.getMatriks()[i], 0,
newMat[i], 0, c);
            }
            int x = 0;
            Simpul baru =null;
            while((x < proses.getRow()) &&
(!isSolusi(baru)))
            {
                if
(!cekKosong(proses.getMatriks()[x],proses.getCol()))
                {
                    int j = 0;
                    while (proses.getMatriks()[x][j] != 0)
                    {
                        j++;
                    }
                    int y = x;
                    y++;
                    while((proses.getMatriks()[y][j] != 0) && (x
!= y))
                    {
                        if (x != proses.getRow())
                        {
                            y++;
                        }
                        else
                        {
                            y = 0;
                        }
                    }
                    if (proses.getMatriks()[x][j] != 1)
                    {
                        for (int k = j ; k < proses.getCol() ; k++)
                        {
                            newMat[x][k] = newMat[x][k] /
newMat[x][j];
                        }
                    }
                    if (y != x)
                    {
                        int h = 0;
                        while (h < proses.getCol())
                        {
                            newMat[y][h] = newMat[y][h] -
(newMat[x][j] / newMat[y][j]);
                            h++;
                        }
                    }
                    baru = new Simpul(newMat,proses,r,c);
                    antrian.AddQueue(baru);
                }
                if (!isSolusi(baru))
                {
                    x++;
                }
            }
        }
        while((!antrian.isEmpty()) && (!isSolusi(proses)));
        LinkedList<Simpul> path = new
LinkedList<Simpul>();
        do
        {
            path.addFirst(proses);
            proses = proses.getOrtu();
        }
        while(proses.getOrtu() != null);
        return path;
    }
```

IV. Penjelasan

Program akan menelusuri pohon yang dimasukan ke dalam antrian sambil membentuk cabang pohon yang baru, dimana cabang pohon baru ini adalah matriks yang di-OBE.

Contoh :



Pohon dibentuk dan ditelusuri terus-menerus sampai ditemukan matriks yang tidak dapat dieliminasi lagi (solusi). Solusi pertama yang didapatkan adalah solusi yang paling sedikit jumlah operasinya karena BFS memeriksa semua kemungkinan pada pohon status matriks untuk setiap tingkat pohon. Cara ini kelihatan seperti IDS, tetapi pada IDS pemeriksaan dilakukan dari simpul akar sampai simpul tingkat ke-n, jika tidak ditemukan maka antrian akan dikosongkan dan akan dilakukan pencarian dari simpul pertama sampai simpul tingkat ke-n+1 habis. Pada metode ini, untuk mengefisienkan pencarian, antrian tidak dikosongkan dan pencarian diteruskan, bukan diulang dari awal.

V. Kesimpulan

Breadth first search dapat diterapkan pada penyelesaian permasalahan sistem persamaan linear dengan operasi barisan elementer untuk mendapatkan solusi optimal dengan jarak simpul solusi dengan simpul awal yang paling sedikit (solusi dengan jumlah operasi paling banyak).

VI. Daftar Pustaka

[1] Munir, Rinaldi. “Strategi Algoritma”, Informatika, Bandung : 2010

[2] https://rosettacode.org/wiki/Reduced_row_echelon_form [diakses pada tanggal 20 april 2016]

[3] http://www.tutorialspoint.com/data_structures_algorithms/breadth_first_traversal.htm [diakses pada 20 april 2016]

[4] Cormen, Thomas H. “Introduction to Algorithm 3rd Edition”, 2009

[5] Knuth, Donald E. (1997), The Art of Computer Programming Vol 1. 3rd ed., Boston: Addison-Wesley

VII. 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, 26 April 2016

Kristianto Karim
13514075