

LAPORAN TUGAS BESAR II
Simulasi Penyebaran Virus Penyakit dengan
Memanfaatkan Algoritma BFS untuk Penelusuran pada Graf

DIAJUKAN UNTUK MEMENUHI TUGAS MATA KULIAH
IF 2211 - Strategi Algoritma
PADA SEMESTER II TAHUN 2019/2020

Disusun Oleh:

Byan Sakura Kireyna Aji	13518066
Arthur Edgar Yunanto	13518090
Moch. Nafkhan Alzamzami	13518132

DOSEN : Dr. Ir. Rinaldi Munir, MT.



SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2020

DAFTAR ISI

DAFTAR ISI	1
BAB DESKRIPSI TUGAS	3
BAB II DASAR TEORI	6
Graf	6
Breadth First Search (Pencarian Melebar)	6
BAB III ANALISIS PEMECAHAN MASALAH	8
BAB IV IMPLEMENTASI & PENGUJIAN	10
Implementasi	10
Program.cs	10
City.cs	10
CityGraph.cs	13
Form1.cs	16
Pengujian	20
Contoh hasil input dengan hari ke-0:	21
Contoh hasil input hari ke-10:	21
Contoh input hari ke-23:	23
BAB V KESIMPULAN & SARAN	25
Kesimpulan	25
Saran	25
DAFTAR PUSTAKA	26

BAB I

DESKRIPSI TUGAS

Pada tugas kali ini kami merancang sebuah program yang dapat mendeteksi persebaran suatu virus dengan menyiapkan beberapa informasi yang berguna sebagai berikut.

- Virus berasal dari suatu daerah, dan virus dapat menyebar ke daerah lain.
- Populasi masyarakat di suatu daerah didefinisikan dengan konstan $P(A)$ dengan A adalah nama suatu daerah.
- Waktu pertama kali virus menginfeksi suatu kota didefinisikan dengan $T(A)$ dan total hari sejak infeksi pertama muncul di suatu daerah didefinisikan dengan $t(A)$.
- Populasi masyarakat yang terkena virus didefinisikan dengan fungsi $I(A, t(A))$. Fungsi I independen untuk setiap kota. Fungsi $I(A, t(A))$ akan menggunakan fungsi logistik, dengan :

$$I(A, t(A)) = \frac{P(A)}{1 + (P(A) - 1)e^{-\gamma t(A)}}, \gamma = 0.25$$

- Peluang orang melakukan perjalanan dari daerah A ke daerah B didefinisikan dengan $Tr(A,B)$ dengan A adalah kota asal dan B adalah kota tujuan.
- Fungsi penyebaran virus dari suatu daerah A ke daerah B selalu sama dan dirumuskan dengan fungsi $S(A,B)$. Jika luaran fungsi S lebih besar dari 1 maka daerah A berhasil menularkan virus ke daerah tujuan.

$$S(A,B) = I(P(A),t(A)) \times Tr(A, B)$$

- Jika virus dari daerah awal berhasil tersebar ke daerah tujuan, maka jumlah orang terkena virus penyakit di daerah tujuan akan dirumuskan dengan fungsi I yang sudah dijelaskan sebelumnya.
- Jika dari daerah A berhasil menyebarkan virus ke daerah B , maka akan dicek apakah daerah B mampu menyebarkan virus ke daerah lain yang terhubung dan belum terkena virus penyakit.

Pada awalnya, hanya ada satu daerah saja yang memiliki virus penyakit yang berpotensi menyebar ke daerah lain. Tugas Anda adalah membantu WHO untuk membuat simulasi penyebaran virus penyakit, dengan membuat program yang menerima query berupa hari (T) dan menunjukkan daerah mana saja yang sudah tersebar virus penyakit pada hari ke T beserta jalur penyebarannya dengan menggunakan algoritma BFS. Peta yang menghubungkan daerah-daerah di dunia diberikan dalam berkas eksternal untuk dibaca oleh program dengan format sebagai berikut. Baris pertama pada berkas adalah N yang merupakan banyaknya jalur yang akan diberikan. Sebanyak N baris berikutnya berisi dua buah string (A, B) dan sebuah float ($Tr(A,B)$). Gambar 1 akan menunjukkan gambar berkas eksternal yang berisi peta yang menghubungkan daerah-daerah di dunia.

```

8
A B 0.02
A C 0.005
B A 0.005
B D 0.005
C A 0.1
C B 0.1
D A 0.05
D C 0.1

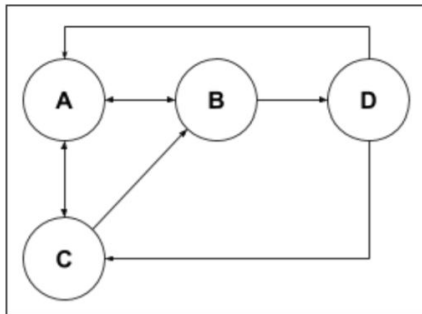
```

Populasi masyarakat di suatu daerah di seluruh dunia akan diberikan dalam berkas eksternal untuk dibaca oleh program dengan format sebagai berikut. Baris pertama pada berkas adalah N yang Tugas II IF2211 Strategi Algoritma Halaman 3 merupakan banyaknya daerah yang terdapat di dunia ini dan sebuah string (A) yang merupakan sumber dari virus penyakit. Sebanyak N baris berikutnya berisi sebuah string (A) dan sebuah integer (P(A)). Gambar 2 akan menunjukkan gambar berkas eksternal yang berisi populasi masyarakat di suatu daerah di seluruh dunia.

```

4 A
A 1000
B 5000
C 1000
D 1000

```



Program yang dibuat harus memenuhi spesifikasi sebagai berikut.

1. Aplikasi bisa menerima peta daerah-daerah di seluruh dunia berikut populasinya seperti contoh pada Gambar 1 dan Gambar 2.
2. Dari berkas eksternal yang telah diterima, aplikasi dapat menampilkan visualisasi graf peta daerah-daerah di seluruh dunia. Proses visualisasi ini boleh memanfaatkan pustaka atau kakas yang tersedia. Sebagai referensi, salah satu kakas yang tersedia untuk melakukan visualisasi adalah MSAGL : <https://github.com/microsoft/automatic-graph-layout> Tugas II IF2211 Strategi Algoritma Halaman 6

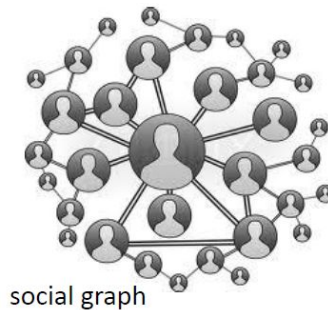
3. Aplikasi bisa menerima pertanyaan (query) berupa hari. Pertanyaan akan diterima melalui aplikasi berbasis GUI yang dibangun sesuai kreativitas masing-masing.
4. Dari graph yang sudah dibentuk, aplikasi harus dapat menyusun jalur infeksi. Aplikasi juga harus dapat menunjukkan langkah-langkah proses penentuan daerah-daerah yang terinfeksi dengan memanfaatkan algoritma BFS. Mahasiswa tidak diperkenankan untuk melihat atau menyalin library lain yang mungkin tersedia bebas terkait dengan pemanfaatan BFS.
5. Aplikasi harus dapat membedakan simpul untuk kota yang sudah terinfeksi dan yang belum terinfeksi berikut juga dengan jalurnya.
6. Data Uji akan diberikan oleh asisten.

BAB II

DASAR TEORI

1. Graf

Traversal Graf merupakan cara untuk mengunjungi seluruh simpul yang ada pada Graf dengan cara yang sistematis. Ada beberapa teorema yang dapat digunakan dalam traversal graf



<http://www.oreilly.de/catalog/9780596518172/toc.html>

ini. Ada yang berdasarkan tanpa informasi, yaitu Pencarian Melebar(Breadth First Search) dan Pencarian mendalam(Depth First Search) dengan menggunakan asumsi bahwa seluruh graf terhubung. Selain itu, ada cara dengan berdasarkan informasi seperti *Best First Search*.

Selain itu, dalam melakukan pencarian solusi ada dua pendekatan yang bisa dilakukan yaitu Graf Statis dan Graf Dinamis. Graf Statis adalah graf yang sudah terbentuk sebelum proses pencarian dilakukan. Sedangkan Graf dinamis adalah graf yang terbentuk saat proses pencarian dilakukan. Pada graf dinamis, graf tidak tersedia sebelum pencarian melainkan baru terbentuk selama solusi dicari.

2. Breadth First Search (Pencarian Melebar)

Breadth First Search adalah suatu algoritma yang bergerak secara traversal dalam pencarian suatu nilai pada pohon pencarian atau data graph. Pencarian dimulai dari akar pohon atau node awal yang ditujukan sebagai *search key* dan akan melakukan kunjungan ke semua node tetangga yang ada pada kedalaman tertentu.

Input dari algoritma ini adalah suatu graph dengan akar permulaan dari graph tersebut dan akan mengeluarkan output berupa *goal state*.

```

1  procedure BFS(G, start_v) is
2      let Q be a queue
3      label start_v as discovered
4      Q.enqueue(start_v)
5      while Q is not empty do
6          v := Q.dequeue()
7          if v is the goal then
8              return v
9          for all edges from v to w in G.adjacentEdges(v) do
10             if w is not labeled as discovered then
11                 label w as discovered
12                 w.parent := v
13                 Q.enqueue(w)

```

Berbeda dengan Depth First Search, BFS menggunakan antrian *First In First Out* dibandingkan DFS yang menggunakan *stack* dan algoritma BFS melakukan pengecekan apakah vertex sudah dikunjungi sebelum meletakkannya pada antrian daripada algoritma DFS yang menunda pengecekan hingga vertex harus dikeluarkan dari antrian.

Algoritma ini memenuhi kaidah completeness selama kedalaman memiliki batasan tertentu karena meskipun secara analisis input dari pencarian BFS seharusnya terbatas, dalam realita terkadang graph yang terbentuk dapat mencapai *infinity*. Meski demikian, *goal state* akan tetap dapat dicapai dengan BFS karena pencariannya yang tidak menyusuri graf hingga akhir seperti pada DFS.

BAB III

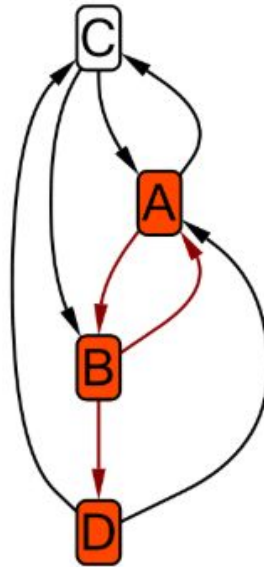
ANALISIS PEMECAHAN MASALAH

Dalam pemecahan masalah kali ini, kami menggunakan algoritma BFS untuk melakukan pengecekan satu-persatu terhadap City yang ada dalam antrian. Algoritma ini akan membuat suatu antrian yang berisi kumpulan City yang harus dikunjungi berdasarkan kedekatan hubungan dengan City lainnya.

Algoritma BFS ini dituangkan ke dalam script kode prosedur Query yang meminta masukkan berupa root dan hari target. Prosedur ini akan melakukan reset kepada semua City dan memulai startDay pada hari ke 0 dan membuat queue yang berisi City.

Selama isi antrian masih lebih dari 0, prosedur akan melakukan update City yang terkena infeksi dengan melakukan pengecekan terlebih dahulu apakah terjadi penularan di City tersebut. Prosedur akan diulangi hingga semua City selesai dicek. Pengecekan akan dilanjutkan dengan hari selanjutnya sebagai hari awal.

City yang terinfeksi akan digambarkan dengan node berwarna merah, berbeda dengan City yang belum terinfeksi yang digambarkan dengan node berwarna putih. Panah penyebaran akan diwarnai merah gelap jika terjadi penyebaran pada arah tersebut.



Penyebaran dicek melalui fungsi S dengan parameter City awal dan City tujuan yang mengembalikan nilai hasil perkalian jumlah penduduk yang terinfeksi (dilambangkan dalam fungsi I dengan parameter populasi dan lama hari terkena virus) pada hari tertentu dengan kemungkinan seseorang menyebrang dari City awal ke City tujuan (dilambangkan oleh Tr dengan parameter City awal dan City tujuan).

$$S(A,B) = I(P(A),t(A)) \times Tr(A, B)$$

Populasi penduduk yang terinfeksi dipaparkan dalam rumus berikut yang bergantung pada populasi total dan lama hari City tersebut terpapar oleh infeksi.

$$I(A, t(A)) = \frac{P(A)}{1 + (P(A) - 1)e^{-\gamma t(A)}}, \gamma = 0.25$$

Pengecekan apakah infeksi telah menyebar akan dilakukan dengan fungsi S sebagai parameter dimana jika fungsi ini mengembalikan nilai yang lebih dari satu, prosedur akan mengupdate City target sebagai terinfeksi.

$$S = \frac{P(A).Tr(A,B)}{1+(P(A)-1)e^{-\frac{t}{4}}} > 1$$

$$P(A).Tr(A, B) > 1 + (P(A) - 1) e^{-\frac{t}{4}}$$

$$e^{-\frac{t}{4}} < \frac{P(A).Tr(A,B)-1}{(P(A)-1)}$$

$$-\frac{t}{4} < \ln\left(\frac{P(A).Tr(A,B)-1}{(P(A)-1)}\right)$$

$$t > -4\ln\left(\frac{P(A).Tr(A,B)-1}{(P(A)-1)}\right)$$

Algoritma breadth-first search ini mengecek masing-masing City melalui jalur antar City masing-masing. Maka kompleksitas algoritma breadth-first search ini adalah $O(n)$, dengan n adalah banyak City.

BAB IV

IMPLEMENTASI & PENGUJIAN

1. Implementasi

Program.cs

Program.cs adalah implementasi dari gabungan GUI dengan file-file lain yang merancang rumus dan menunjukkan interface dari kumpulan program tersebut. Program ini akan menunjukkan hasil visualisasi graf dalam bentuk GUI yang juga menerima input dari user dan menjalankan form aplikasi.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Tubes_Stima_2
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            var form = new Form1
            {
                Text = "Anaroc v1.0"
            };
            Application.Run(form);
        }
    }
}
```

City.cs

City.cs mengimplementasikan rumus-rumus yang menentukan terjadi atau tidaknya persebaran virus bergantung kepada populasi, populasi terjangkit, hari penyebaran, dan probabilitas perpindahan penduduk. Pada program City.cs ini diimplementasikan juga suatu kelas City yang dikonstruksikan oleh nama City dan jumlah populasi penduduk.

Dalam program ini terdapat beberapa fungsi yang bisa mengembalikan nama City, jumlah populasi City, hari awal virus menyebar dan hari target, dan kemungkinan penyebaran

virus. Selain itu terdapat juga metode yang dapat melakukan reset pada City dan mengembalikan keadaan City pada hari ke-0 sebelum ada infeksi.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Tubes_Stima_2
{
    public class City
    {
        private readonly int population;
        public int startDay; // hari pertama terkena virus
        private readonly string name;

        public City(string name, int population)
        {
            this.population = population;
            this.name = name;
        }

        public City(City city) : this(city.name, city.population) { }

        public string GetName()
        {
            return this.name;
        }

        public void Reset()
        {
            startDay = int.MaxValue;
        }

        public int P()
        {
            return this.population;
        }

        //Hari Terkena Virus
        public int T()
        {
            return this.startDay;
        }

        //Lama hari setelah terkena virus
    }
}
```

```
public int t(int targetDay)
{
    return targetDay - startDay;
}

public double I(int targetDay)
{
    double p = (double)P();
    return p / (1 + (p - 1) * Math.Exp(-0.25 * t(targetDay)));
}
}
```

CityGraph.cs

Pada kode CityGraph.cs terdapat konstruksi graph yang akan divisualisasi. Node pada graph ini akan menunjukkan nama City sedangkan edge akan menghubungkan kedua City.

Pembentukan graph diawali dengan memasukkan City-City sebagai node dengan memasukkan list dari City yang dibaca dari input pengguna. Hubungan antar City kemudian akan digambarkan dengan memasang node yang sudah terbentuk. Apabila kedua City yang ingin dibentuk belum ada dalam Graph, City tersebut akan ditambahkan terlebih dahulu.

Disini juga diimplementasikan penanda apabila suatu City terjangkit infeksi, yaitu node dari City itu akan berwarna merah. Hal ini dirancang dalam fungsi UpdateInfected dengan parameter City. pada fungsi UpdateInfected dengan parameter dua City, warna yang akan diubah adalah panah yang menghubungkan suatu City asal dengan City tujuan.

```
using Microsoft.Msagl.Drawing;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Tubes_Stima_2
{
    public class CityGraph
    {
        public readonly Dictionary<City, List<City>> graph;
        private readonly Dictionary<KeyValuePair<City, City>, double> tr;
        public readonly Graph MSAGLGraph;
        public CityGraph()
        {
            graph = new Dictionary<City, List<City>>();
            tr = new Dictionary<KeyValuePair<City, City>, double>();
            MSAGLGraph = new Graph();
        }

        public void AddEdge(City a, City b, double p)
        {
            if (!graph.ContainsKey(a))
            {
                graph.Add(a, new List<City>());
            }
            if (!graph.ContainsKey(b))
            {
                graph.Add(b, new List<City>());
            }
        }
    }
}
```

```

        MSAGLGraph.AddEdge(a.GetName(), b.GetName());
        graph[a].Add(b);
        tr.Add(new KeyValuePair<City, City>(a, b), p);
    }

    public void Query(City root, int targetDay)
    {
        foreach (City City in graph.Keys)
        {
            City.Reset();
        }
        root.startDay = 0;
        var q = new Queue<City>();
        q.Enqueue(root);

        while (q.Count > 0)
        {
            City now = q.Dequeue();
            UpdateInfected(now);
            foreach (City next in graph[now])
            {
                if (S(now, next, targetDay) > 1)
                {
                    UpdateInfected(now, next);
                    int nextDay = DayPenyebaran(now, next) +
now.startDay;

                    if (next.startDay > nextDay)
                    {
                        next.startDay = nextDay;
                        q.Enqueue(next);
                    }
                }
            }
        }

        private void UpdateInfected(City City)
        {
            MSAGLGraph.FindNode(City.GetName()).Attr.FillColor =
Color.OrangeRed;
        }

        private void UpdateInfected(City from, City to)
        {
            Edge e = MSAGLGraph.Edges.Where(edge => edge.Source ==
from.GetName() && edge.Target == to.GetName()).FirstOrDefault();
            if (e != null)

```

```

        {
            e.Attr.Color = Color.DarkRed;
        }
    }

    public double Tr(City from, City to)
    {
        return tr[new KeyValuePair<City, City>(from, to)];
    }

    public double S(City from, City to, int targetDay)
    {
        return from.I(targetDay) * Tr(from, to);
    }

    public int DayPenyebaran(City from, City to)
    {
        double d = -4*Math.Log((from.P()*Tr(from, to)-1)/(from.P()-1));
        int next = (int)Math.Ceiling(d);
        if (next == d)
        {
            ++next;
        }
        return next;
    }
}

```

Form1.cs

Form1.cs menggabungkan antara GUI dengan main program. Program ini akan mengatur layout dari GUI yang telah disusun dan menentukan aksi apa saja yang dilakukan oleh program apabila pengguna memasukkan input tertentu pada GUI.

Pada GUI terdapat textbox yang meminta inputan hari dimana hasil inputan itu akan dijadikan sebagai targetDay. Ada juga input berupa populasi tiap kota dan peta persebaran dimana user dapat memilih mengambil data dari file external atau memasukkan input secara manual. Apabila data diambil dari file external, GUI akan membuka dialog baru yang menunjukkan penyimpanan drive komputer bagi user untuk memilih file mana yang akan diupload.

Program ini juga mengatur apa yang dijalankan apabila button GO ditekan. Setelah button GO ditekan, program akan melakukan parsing input hari di textbox menjadi integer untuk dimasukkan dalam targetDay dan input berupa list nama City dan peta keterhubungan antar City menjadi graf.

Selanjutnya program menghubungkan antara implementasi-implementasi lainnya dengan GUI yang terancang.

```
using Microsoft.Msagl.Drawing;
using Microsoft.Msagl.GraphViewerGdi;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Tubes_Stima_2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```



```

    }

    private void BG_Click(object sender, EventArgs e)
    {
        //
    }

    private void label1_Click(object sender, EventArgs e)
    {
        //
    }

    private void label1_Click_1(object sender, EventArgs e)
    {
        //
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        //
    }

    private void richTextBox1_TextChanged(object sender, EventArgs e)
    {
        //
    }

    private void label2_Click(object sender, EventArgs e)
    {
        //
    }

    private void button1_Click(object sender, EventArgs e)
    {
        try
        {
            int targetDay = int.Parse(dayTextBox.Text);
            if (targetDay < 0)
            {
                throw new Exception("Query hari tidak boleh negatif!");
            }
            string[] g = listKotaTextBox.Text.Split('\n');
            Dictionary<string, City> map = new Dictionary<string,
City>();

            int i = 0;
            string[] l = g[i++].Split(' ');
            string root = l[1];

```

```

        CityGraph graph = new CityGraph();
        int n = int.Parse(l[0]);
        while (n-- > 0)
        {
            string[] line = g[i++].Split(' ');
            map[line[0]] = new City(line[0], int.Parse(line[1]));
            graph.MSAGLGraph.AddNode(new Node(line[0]));
        }
        g = petaTxtBox.Text.Split('\n');
        i = 0;
        n = int.Parse(g[i++]);
        while (n-- > 0)
        {
            string[] line = g[i++].Split(' ');
            graph.AddEdge(map[line[0]], map[line[1]],
double.Parse(line[2]));
        }
        graph.Query(map[root], targetDay);
        SetGraph(graph);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Input tidak valid!\n" + ex.Message);
    }
}

private void SetGraph(CityGraph g)
{
    GViewer viewer = new GViewer();
    viewer.Graph = g.MSAGLGraph;
    panel1.SuspendLayout();
    viewer.Dock = DockStyle.Fill;
    panel1.Controls.Clear();
    panel1.Controls.Add(viewer);
    panel1.ResumeLayout();
}

private void label3_Click(object sender, EventArgs e)
{
}

private void button2_Click(object sender, EventArgs e)
{
    // Peta hubungan file
    openFilePetaDialog.ShowDialog();
}

```

```

private void button3_Click(object sender, EventArgs e)
{
    // List kota file
    openFileListKotaDialog.ShowDialog();
}

private void richTextBox1_TextChanged_1(object sender, EventArgs e)
{
}

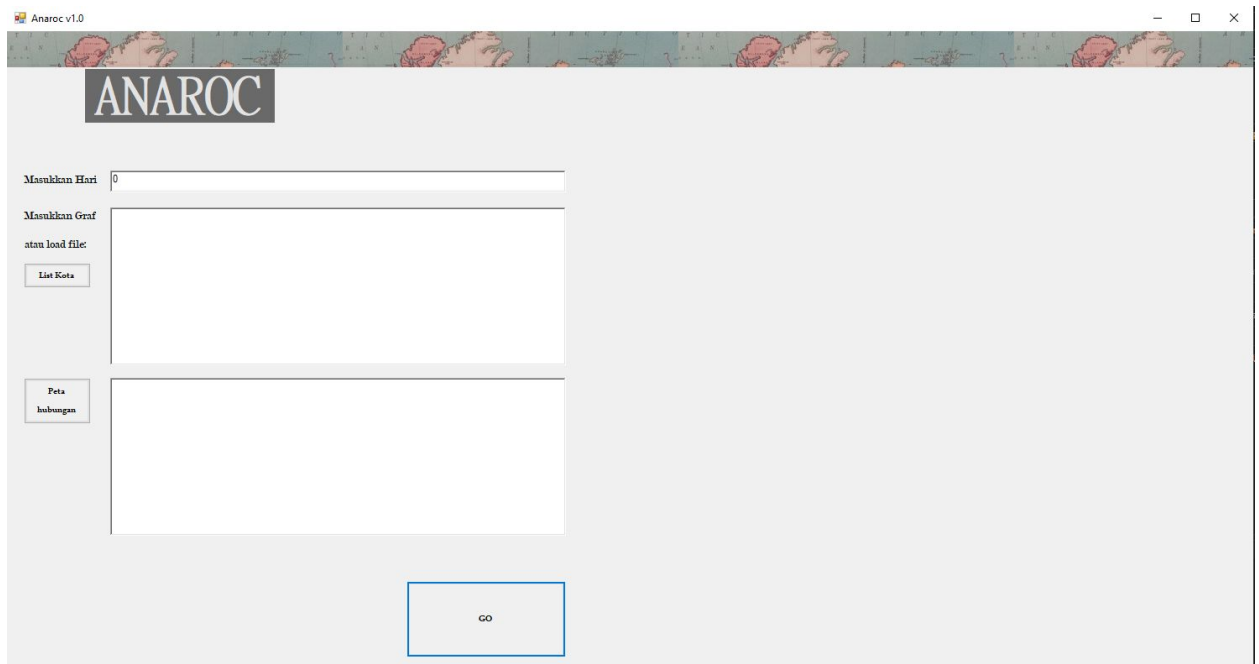
private void openFileDialog1_FileOk(object sender, CancelEventArgs
e)
{
    try
    {
        listKotaTxtBox.Text =
File.ReadAllText(openFileListKotaDialog.FileName, Encoding.UTF8);
    }
    catch
    {
        MessageBox.Show("Gagal membaca file!");
    }
}

private void openFilePetaDialog_FileOk(object sender,
CancelEventArgs e)
{
    try
    {
        petaTxtBox.Text =
File.ReadAllText(openFilePetaDialog.FileName, Encoding.UTF8);
    }
    catch
    {
        MessageBox.Show("Gagal membaca file!");
    }
}
}

```

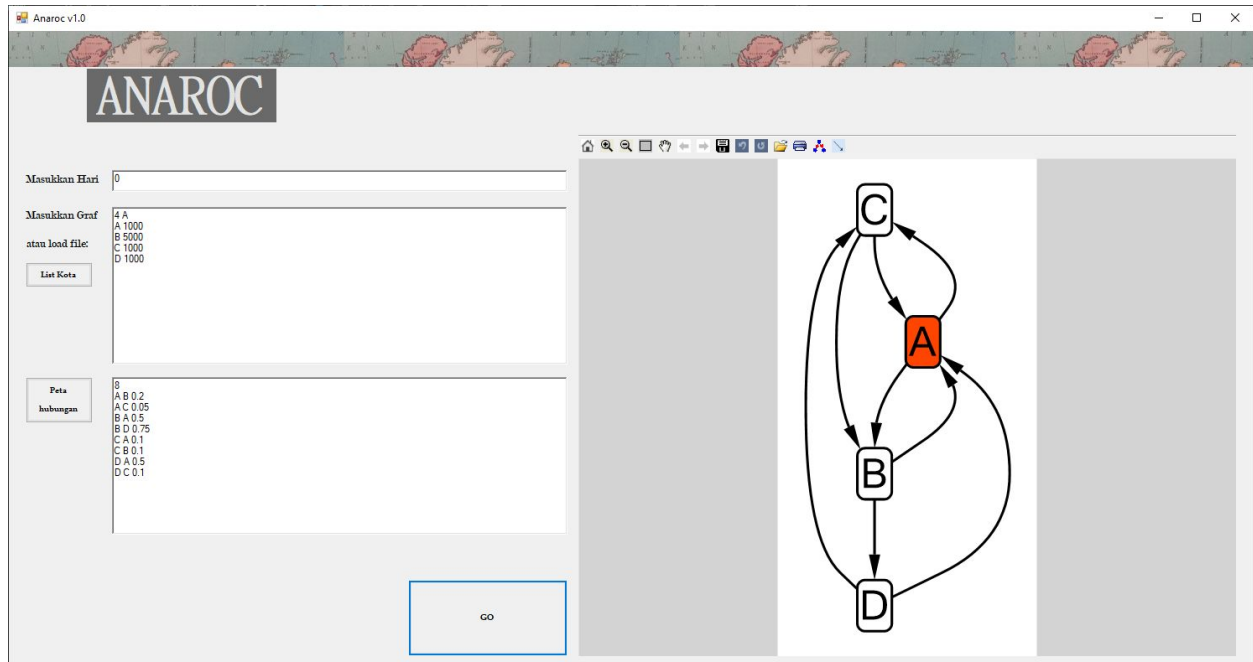
2. Pengujian

Tampilan awal aplikasi:

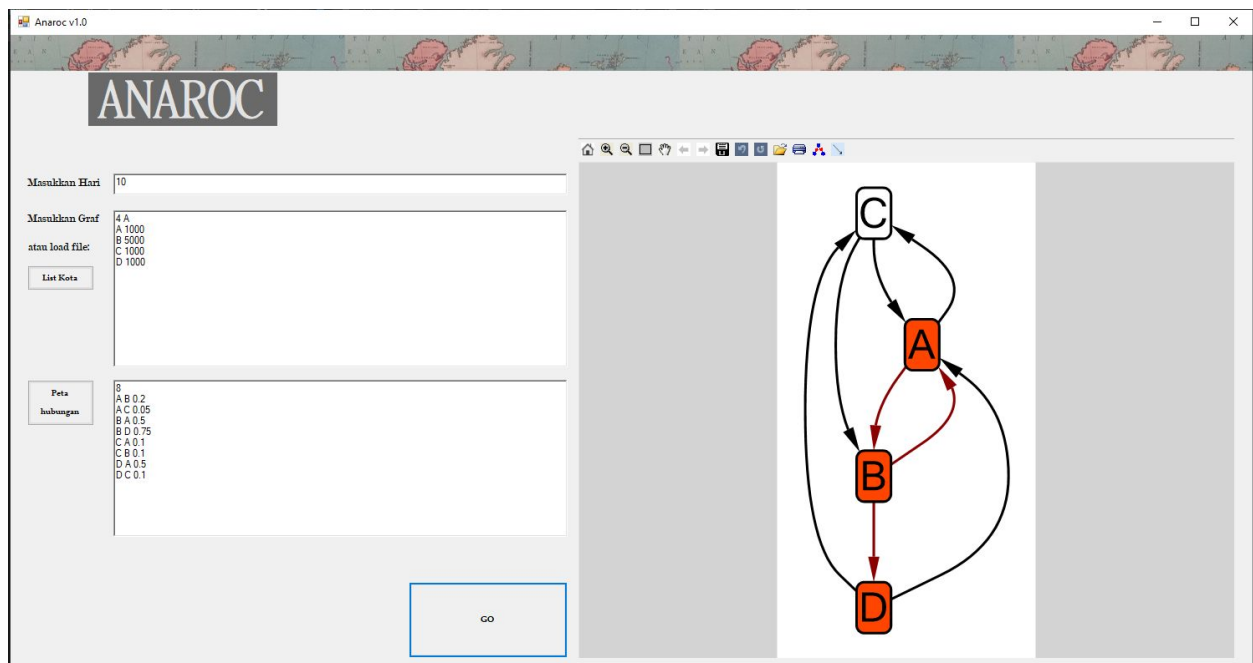


The screenshot shows the initial interface of the ANAROC application. At the top, there is a header with the text "ANAROC" in a stylized font. Below the header, the interface is divided into several sections. On the left, there are two input fields: "Masukkan Hari" with a value of "0" and "Masukkan Graf atau load file:". Below the "Masukkan Graf" field is a button labeled "List Kota". Below the "Masukkan Graf" field is another button labeled "Peta hubungan". At the bottom center, there is a button labeled "GO". The background of the application window has a decorative border with a pattern of red flowers.

Di awal tampilan ini, kita harus memasukkan list kota dan peta hubungan sebagai sebuah input untuk membangun graf tersebut. Kita dapat menulis secara manual pada textbox yang tersedia atau kita juga bisa membaca input dari file dengan mengklik button list kota dan peta hubungan. Lalu kita input hari ke berapa yang ingin kita amati, lalu kita klik tombol GO untuk menampilkan grafnya.

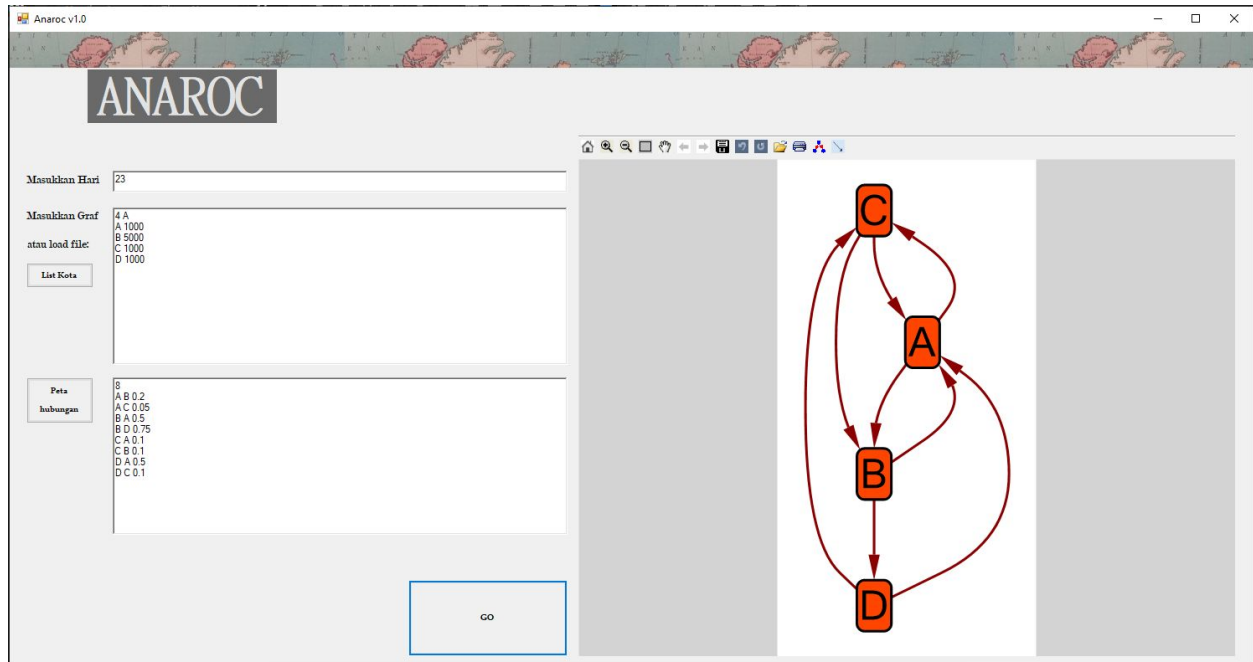
Contoh hasil input dengan hari ke-0:

Pada hari ke-0, hanya kota A saja yg terkena virus, karena virus dimulai dari kota A.

Contoh hasil input hari ke-10:

Pada hari ke-10, virus menyebar dari awalnya dari kota A ke kota B dan D. Berdasarkan strategi algoritma Breadth-First Search, tahapan penyebarannya adalah sebagai berikut:

i	q	Asal	Tujuan	P(Asal)	Tr	T(Asal)	T(Tujuan)	t(Asal)	I(Asal, t(Asal))	S(Asal, Tujuan)	Tersebar?	t tersebar	T'(tujuan)	q
0	[]	-	-	-	-	-	-	-	-	-	-	-	-	[A->B, A->C]
1	[A->B, A->C]	A	B	1000	0.2	0	Infinite	10	12.04776985	2.409553969	YA	7	7	[A->C, B->A, B->D]
2	[A->C, B->A, B->D]	A	C	1000	0.05	0	Infinite	10	12.04776985	0.6023884924	TIDAK	0	Infinite	[B->A, B->D]
3	[B->A, B->D]	B	A	5000	0.5	7	0	3	2.116527184	1.058263592	YA	3	10	[B->D]
4	[B->D]	B	D	5000	0.75	7	Infinite	3	2.116527184	1.587395388	YA	2	9	[D->A, D->C]
5	[D->A, D->C]	D	A	1000	0.5	9	0	1	1.283660824	0.6418304122	TIDAK	0	0	[D->C]
6	[D->C]	D	C	1000	0.1	9	Infinite	1	1.283660824	0.1283660824	TIDAK	0	Infinite	[]

Contoh input hari ke-23:

Pada hari ke-23, virus berhasil menyebar ke semua kota dan jalur dengan tahapan penyebaran sebagai berikut:

i	q	Asal	Tujuan	P(Asal)	Tr	T(Asal)	T(Tujuan)	t(Asal)	I(Asal, t(Asal))	S(Asal, Tujuan)	Tersebar?	t tersebar	T'(tujuan)	q
0	[]	-	-	-	-	-	-	-	-	-	-	-	-	[A->B, A->C]
1	[A->B, A->C]	A	B	1000	0.2	0	Infinite	23	239.2575	47.85149	YA	7	7	[A->C, B->A, B->D]
2	[A->C, B->A, B->D]	A	C	1000	0.05	0	Infinite	23	239.2575	11.96287	YA	13	13	[B->A, B->D, C->A, C->B]
3	[B->A, B->D, C->A, C->B]	B	A	5000	0.5	7	0	16	54.01909	27.00954	YA	3	10	[B->D, C->A, C->B]

4	[B->D , C->A, C->B]	B	D	5000	0.75	7	Infinite	16	54.0 1909	40.5 1431	YA	2	9	[C->A , C->B, D->A, D->C]
5	[C->A , C->B, D->A, D->C]	C	A	1000	0.1	13	0	10	12.0 4777	1.20 4777	YA	10	23	[C->B , D->A, D->C]
6	[C->B , D->A, D->C]	C	B	1000	0.1	13	7	10	12.0 4777	1.20 4777	YA	10	23	[D->A , D->C]
7	[D->A , D->C]	D	A	1000	0.5	9	0	14	32.0 8503	16.0 4251	YA	3	12	[D->C]
8	[D->C]	D	C	1000	0.1	9	13	14	32.0 8503	3.20 8503	YA	10	19	[]

BAB V

KESIMPULAN & SARAN

1. Kesimpulan

Tugas yang diberikan sangat menyenangkan dan mengikuti hal yang sedang terjadi dan *viral* di dunia nyata sehingga terasa realistis.

Dengan penerapan algoritma-algoritma yang dipelajari selama kuliah, kita dapat menyelesaikan masalah-masalah yang ingin diatasi. Contohnya pada tugas ini, kita dapat memperkirakan kecepatan penyebaran virus dari kota ke kota dengan data statistik tertentu.

Selain itu kami juga dapat mempelajari penggunaan bahasa pemrograman baru yaitu C# dengan adanya tugas besar ini, selain itu kami juga menjadi terbiasa dengan untuk menerapkan GUI untuk program-program kecil agar dapat digunakan dengan nyaman dan mudah.

2. Saran

Menurut kami, tugas ini memberikan pengalaman baru dalam menggunakan bahasa pemrograman baru yaitu C# sehingga kedepannya kami harap dapat menggunakan bahasa pemrograman yang lain yang dapat membantu kami untuk terus berkembang.

Selain itu, saran dari kami adalah cukup menjaga kualitas-kualitas tugas yang diberikan, tidak perlu yang terlalu susah atau mudah tapi memberikan dampak yang nyata untuk perkembangan mahasiswa.

DAFTAR PUSTAKA

<http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Tugas-Besar-2-IF2211-Strategi-Algoritma-2020.pdf> diakses pada 27 Februari 2020

<https://www.educative.io/edpresso/what-is-breadth-first-search> diakses pada 27 Februari 2020

<https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/> diakses pada 27 Februari 2020

<https://www.hackerearth.com/de/practice/algorithms/graphs/breadth-first-search/tutorial/> diakses pada 27 Februari 2020