LAPORAN PRAKTIKUM 6
Analisis algoritma

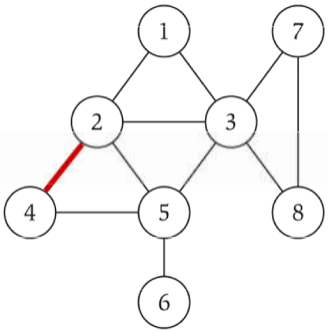
Sarah Navianti Dwi Sutisna
140810180021
Kelas A

PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PADJADJARAN
2020

Studi Kasus

Tugas Anda

1. Dengan menggunakan *undirected graph* dan *adjacency matrix* berikut, buatlah koding programmnya menggunakan bahasa C++.



	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	1	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

Jawab :
SourceCodes :

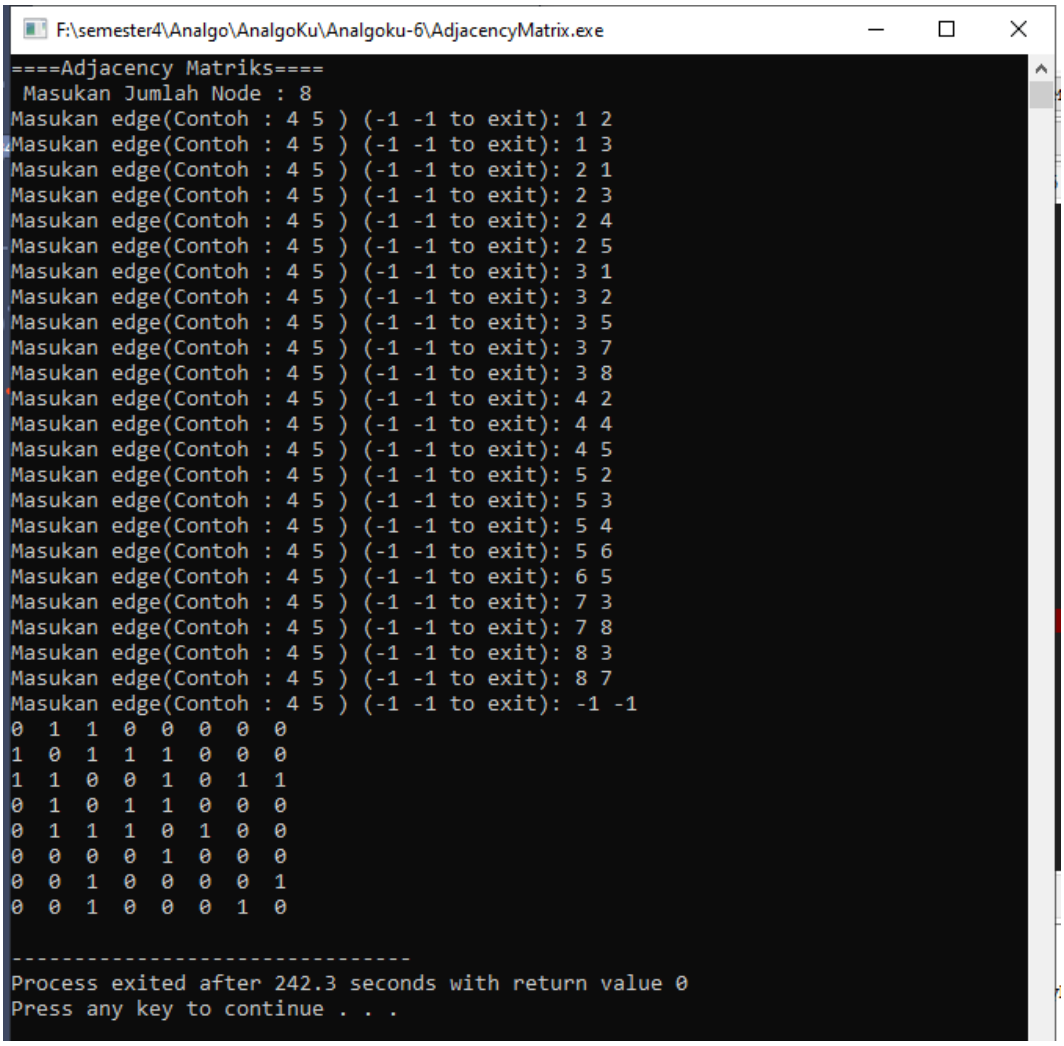
```
/*
Nama        : Sarah Navianti
NPM         : 140810180021
Kelas       : A
Deskripsi   : Adjacency Matrix
*/

#include <iostream>
#include <cstdlib>

using namespace std;
#define MAX 20

 //Class untuk Adjacency Matrix

class AdjacencyMatrix
{
    private:
        int n;
        int **adj;
        bool *visited;

    public:
        AdjacencyMatrix(int n)
```

```
        {
            this->n = n;
            visited = new bool [n];
            adj = new int* [n];
            for (int i = 0; i < n; i++)
            {
                adj[i] = new int [n];
                for(int j = 0; j < n; j++)
                {
                    adj[i][j] = 0;
                }
            }
        }

         //Menambahkan edge ke graf

        void AddEdge(int origin, int destin)
        {
            if( origin > n || destin > n || origin < 0 || destin < 0)
            {
                cout<<"Edge Tidak Valid!\n";
            }
            else
            {
                adj[origin - 1][destin - 1] = 1;
            }
        }
        // Mencetak graf

        void display()
        {
            int a,b;
            for(a = 0;a < n;a++)
            {
                for(b = 0; b < n; b++)
                    cout<<adj[a][b]<<"  ";
                cout<<endl;
            }
        }
};

int main(){

    int nodes, max_edges, origin, destin;
```

Sarah Navianti-140810180021-Laprak6

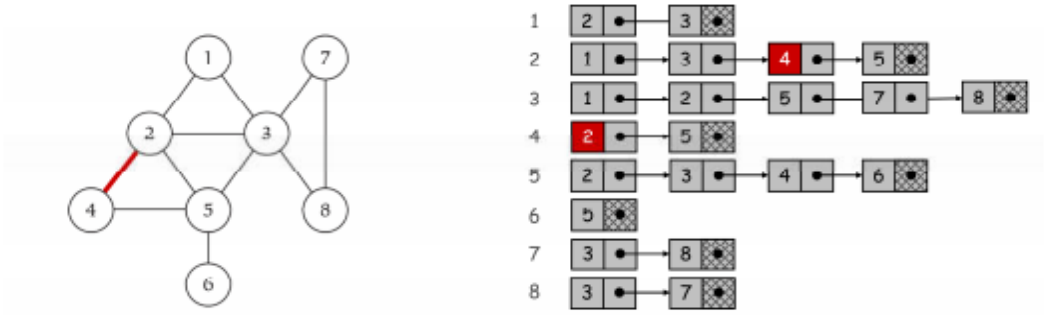
```
    cout <<"====Adjacency Matriks====";
    cout <<"\t\n Masukan Jumlah Node : ";
    cin  >>nodes;
    AdjacencyMatrix am(nodes);
    max_edges = nodes * (nodes - 1);
    for (int i = 0; i < max_edges; i++)
    {
        cout<<"Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): ";
        cin>>origin>>destin;
        if((origin == -1) && (destin == -1))
            break;
        am.AddEdge(origin, destin);
    }
    am.display();
    return 0;
}
```

ScreenShoot



```
F:\semester4\Analgo\AnalgoKu\Analgoku-6\AdjacencyMatrix.exe
====Adjacency Matriks====
 Masukan Jumlah Node : 8
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 1 2
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 1 3
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 2 1
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 2 3
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 2 4
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 2 5
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 3 1
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 3 2
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 3 5
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 3 7
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 3 8
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 4 2
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 4 4
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 4 5
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 5 2
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 5 3
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 5 4
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 5 6
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 6 5
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 7 3
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 7 8
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 8 3
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): 8 7
Masukan edge(Contoh : 4 5 ) (-1 -1 to exit): -1 -1
0  1  1  0  0  0  0  0
1  0  1  1  1  0  0  0
1  1  0  0  1  0  1  1
0  1  0  1  1  0  0  0
0  1  1  1  0  1  0  0
0  0  0  0  1  0  0  0
0  0  1  0  0  0  0  1
0  0  1  0  0  0  1  0

--------------------------------
Process exited after 242.3 seconds with return value 0
Press any key to continue . . .
```

2. Dengan menggunakan *undirected graph* dan representasi *adjacency list*, buatlah koding programmnya menggunakan bahasa C++.



Jawab :

```
/*
Nama        : Sarah Navianti
NPM         : 140810180021
Kelas       : A
Deskripsi   : Adjacency List
*/

#include <iostream>
#include <cstdlib>
using namespace std;


//Adjacency List Node
struct AdjListNode
{
    int destin;
    struct AdjListNode* next;
};

//Adjacency List
struct AdjList
{
    struct AdjListNode *head;
};

class Graph
{
```

```
    private:
        int V;
        struct AdjList* array;
    public:
        Graph(int V)
        {
            this->V = V;
            array = new AdjList [V];
            for (int i = 0; i < V; ++i)
                array[i].head = NULL;
        }

        // Membuat Baru Adjacency List Node
        AdjListNode* newAdjListNode(int destin)
        {
            AdjListNode* newNode = new AdjListNode;
            newNode->destin = destin;
            newNode->next   = NULL;
            return newNode;
        }

        //Menambahkan Edge ke Graph
        void AddEdge(int src, int destin)
        {
            AdjListNode* newNode = newAdjListNode(destin);
            newNode->next = array[src].head;
            array[src].head = newNode;
            newNode = newAdjListNode(src);
            newNode->next = array[destin].head;
            array[destin].head = newNode;
        }

        //Mencetak graph
        void printGraph()
        {
            int v;
            for (v = 1; v <= V; ++v)
            {
                AdjListNode* pCrawl = array[v].head;
                cout<<"\n Adjacency list of vertex "<<v<<"\n head ";
                while (pCrawl)
                {
                    cout<<"-> "<<pCrawl->destin;
                    pCrawl = pCrawl->next;
                }
```

```
                cout<<endl;
            }
        }
};

int main(){
    int pilih,a,b,n;
    cout << "Masukan Banyak node : "; cin >> n;
    Graph gh(n);
    for(; ;){
        cout << "\nMenu Adjacency List\n";
        cout << "1. Tambah edge\n";
        cout << "2. Print Edge\n";
        cout << "3. Keluar\n\n";
        cout << "Pilihan : "; cin >> pilih;

        switch (pilih){
            case 1:
                cout << "\n edge(a,b)\n";
                cout << "Masukan nilai a : "; cin >> a;
                cout << "Masukan nilai b : "; cin >> b;
                gh.AddEdge(a,b);
                continue;
            case 2:
                gh.printGraph();
                continue;
            case 3:
                return 0;
                break;
            default:
                continue;
        }
    }

    return 0;
}
```

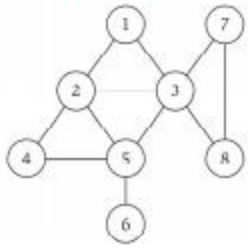
Sarah Navianti-140810180021-Laprak6

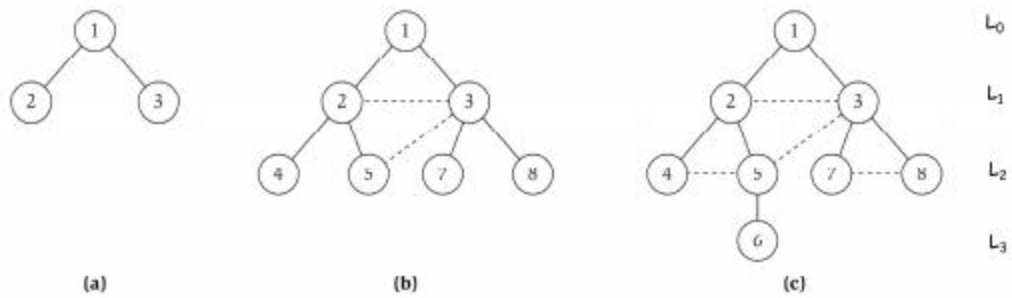
Screen Shoot :

```
F:\semester4\Analgo\AnalgoKu\Analgoku-6\AdjacencyList.exe
Menu
1. Tambah edge
2. Print Edge
3. Keluar

Pilihan : 2

 Adjacency list of vertex 1
 head -> 3-> 2

 Adjacency list of vertex 2
 head -> 3-> 5-> 4-> 1

 Adjacency list of vertex 3
 head -> 5-> 8-> 7-> 2-> 1

 Adjacency list of vertex 4
 head -> 5-> 2

 Adjacency list of vertex 5
 head -> 6-> 3-> 4-> 2

 Adjacency list of vertex 6
 head -> 5

 Adjacency list of vertex 7
 head -> 8-> 3

 Adjacency list of vertex 8
 head -> 7-> 3
--------------------------------
Process exited after 140.7 seconds with return value 3221225477
Press any key to continue . . .
```

Sarah Navianti-140810180021-Laprak6

3. Buatlah program Breadth First Search dari algoritma BFS yang telah diberikan. Kemudian uji coba program Anda dengan menginputkan *undirected graph* sehingga menghasilkan tree BFS. Hitung dan berikan secara asimptotik berapa kompleksitas waktunya dalam Big-Θ!





Jawab :

```
/*
Nama        : Sarah Navianti
NPM         : 140810180021
Kelas       : A
Deskripsi   : BFS
*/

#include<iostream>
#include <list>

using namespace std;

// adjacency list representation
class Graph
{
    int A; // No vertex

    // Pointer Array yang mengandung adjacency
    // lists
    list<int> *adj;
public:
    Graph(int A); // Constructor
```

```
    // menambahkan edge ke graph
    void addEdge(int a, int b);

    // print BFS traversal
    void BFS(int s);
};

Graph::Graph(int A)
{
    this->A = A;
    adj = new list<int>[A];
}

void Graph::addEdge(int a, int b)
{
    adj[a].push_back(b); // menambahkan b ke a’s pada list.
}

void Graph::BFS(int s)
{
    // Mark all the vertices as not visited
    bool *visited = new bool[A];
    for(int i = 0; i < A; i++)
        visited[i] = false;

    // Create a queue for BFS
    list<int> queue;

    // Mark the current node as visited and enqueue it
    visited[s] = true;
    queue.push_back(s);

    // 'i' will be used to get all adjacent
    // vertices of a vertex
    list<int>::iterator i;

    while(!queue.empty())
    {
        // Dequeue a vertex from queue and print it
        s = queue.front();
        cout << s << " ";
        queue.pop_front();

        // Get all adjacent vertices of the dequeued
```

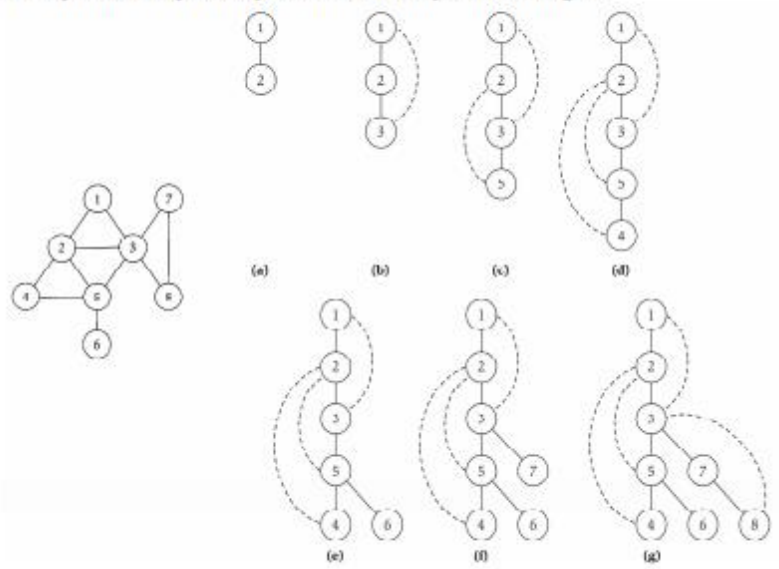
```
        // vertex s. If a adjacent has not been visited,
        // then mark it visited and enqueue it
        for (i = adj[s].begin(); i != adj[s].end(); ++i)
        {
            if (!visited[*i])
            {
                visited[*i] = true;
                queue.push_back(*i);
            }
        }
    }
}

int main()
{
    // Membuat graf di diagram
    Graph g(8);
    g.addEdge(1, 2);
    g.addEdge(1, 3);
    g.addEdge(2, 4);
    g.addEdge(2, 5);
    g.addEdge(2, 3);
    g.addEdge(3, 7);
    g.addEdge(3, 8);
    g.addEdge(4, 5);
    g.addEdge(5, 3);
    g.addEdge(5, 6);
    g.addEdge(7, 8);

    cout << "====Breadth First Traversal===== "
        << "\n(Dimulai dari Vertex 1) \n";
    g.BFS(1);

    return 0;
}
```

Sarah Navianti-140810180021-Laprak6

F:\semester4\Analgo\AnalgoKu\Analgoku-6\BFS.exe

```
====Breadth First Traversal=====
(Dimulai dari Vertex 1)
1 2 3 4 5 7 8
```

4. Buatlah program Depth First Search dari algoritma DFS yang telah diberikan. Kemudian uji coba program Anda dengan menginputkan *undirected graph* sehingga menghasilkan tree DFS. Hitung dan berikan secara asimptotik berapa kompleksitas waktunya dalam Big-Θ!



Jawab :

```
/*
Nama        : Sarah Navianti
NPM         : 140810180021
Kelas       : A
Deskripsi   : DFS
*/

#include<iostream>
#include<list>
```

Sarah Navianti-140810180021-Laprak6

```
using namespace std;

// Graph class merepresentasikan graf berarah menggunakan representasi
 adjacency list
class Graph
{
    int A; // No. simpul

    // Pointer ke array yang memiliki adjacency lists
    list<int> *adj;

    // Fungsi rekursif yang digunakan DFS
    void DFSUtil(int a, bool visited[]);
public:
    Graph(int A); // Constructor

    // fungsi untuk menambah tepian ke graf
    void addEdge(int a, int b);

    // DFS traversal dari simpul yang terjangkau dari a
    void DFS(int a);
};

Graph::Graph(int A)
{
    this->A = A;
    adj = new list<int>[A];
}

void Graph::addEdge(int a, int b)
{
    adj[a].push_back(b); // Menambah b ke list a.
}

void Graph::DFSUtil(int a, bool visited[])
{
    // Menandakan node bersangkutan sudah dikunjungi lalu cetak
    visited[a] = true;
    cout << a << " ";

    // Ulang simpul berdekatan ke node ini
    list<int>::iterator i;
    for (i = adj[a].begin(); i != adj[a].end(); ++i)
        if (!visited[*i])
            DFSUtil(*i, visited);
```

```
}

// DFS traversal dari simpul terjangkau dari v.
// Menggunakan rekursif DFSUtil()
void Graph::DFS(int a)
{
    // Menandakan semua simpul belum dikunjungi
    bool *visited = new bool[A];
    for (int i = 0; i < A; i++)
        visited[i] = false;

    // Memanggil fungsi rekursif pembantu untuk mencetak DFS traversal

    DFSUtil(a, visited);
}

int main()
{
    // Membuat graf di diagram
    Graph g(8);
    g.addEdge(1, 2);
    g.addEdge(1, 3);
    g.addEdge(2, 5);
    g.addEdge(2, 4);
    g.addEdge(5, 6);
    g.addEdge(3, 7);
    g.addEdge(3, 8);
    g.addEdge(7, 8);

    cout << "=====Depth First Traversal====="
            "\n(dimulai dari node 1) \n";
    g.DFS(1);

    return 0;
}
```

```
F:\semester4\Analgo\AnalgoKu\Analgoku-6\DFS.exe
=====Depth First Traversal====
(dimulai dari node 1)
1 2 5 6 4 3 7 8
--------------------------------
Process exited after 6.524 seconds with return value 3221225477
Press any key to continue . . .
```

Sarah Navianti-140810180021-Laprak6