LAPORAN PRAKTIKUM 6
Analisis Algoritma

Disusun oleh :

Asep Budiyana Muharam
140810180029

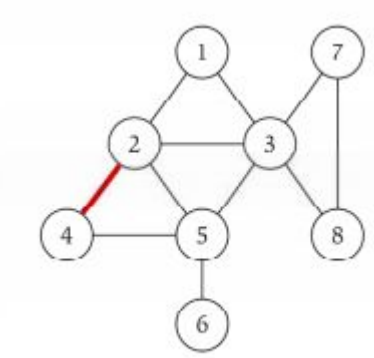
PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PADJADJARAN

2020

Asep Budiyana M
140810180029
Tugas 5

Tugas Anda

1. Dengan menggunakan undirected graph dan adjacency matrix berikut, buatlah coding programnya 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:

adj_matrix.cpp:

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

struct Graph {
    int vertex;
    bool** edge;
};
Graph G;

void makeGraph(Graph& G, int vertex)
{
    G.vertex = vertex;
    G.edge = new bool*[vertex];
    for (int i=0; i<vertex; i++) {
        G.edge[i] = new bool[vertex];
        for (int j=0; j<vertex; j++)
            G.edge[i][j] = false;
    }
}

void addEdge(Graph& G, int i, int j)
{
    G.edge[i][j] = true;
    G.edge[j][i] = true;
}

void traversal(Graph G)
```

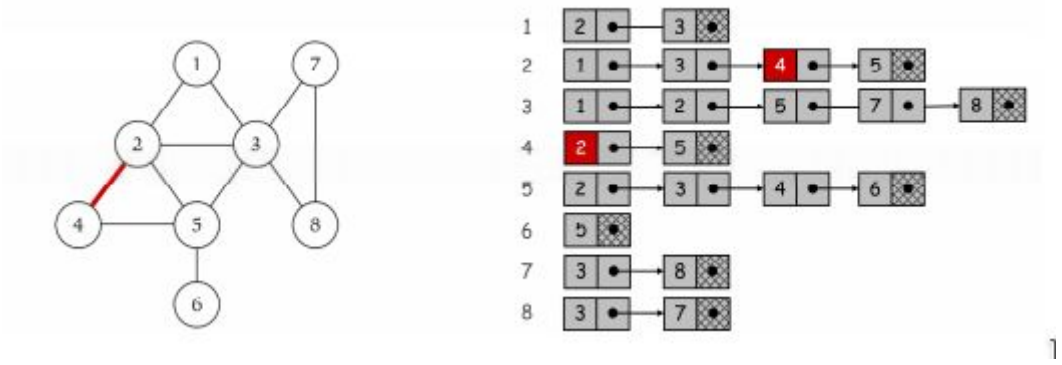
Asep Budiyana M
140810180029
Tugas 5

```
{
    cout<<"# : ";
    for (int i=0; i<G.vertex; i++) {
        cout<<i<<" ";
    }
    cout<<endl<<"---";
    for (int i=0; i<G.vertex; i++) {
        cout<<"--";
    }
    cout<<endl;
    for (int i=0; i<G.vertex; i++) {
        cout<<i<<" : ";
        for (int j=0; j<G.vertex; j++)
            cout<<G.edge[i][j]<< " ";
        cout<<endl;
    }
}

int main()
{
    makeGraph(G, 8);

    addEdge(G, 0, 1);
    addEdge(G, 0, 2);

    addEdge(G, 1, 0);
    addEdge(G, 1, 2);
    addEdge(G, 1, 3);
    addEdge(G, 1, 4);

    addEdge(G, 2, 0);
    addEdge(G, 2, 1);
    addEdge(G, 2, 4);
    addEdge(G, 2, 6);
    addEdge(G, 2, 7);

    addEdge(G, 3, 1);
    addEdge(G, 3, 4);

    addEdge(G, 4, 1);
    addEdge(G, 4, 2);
    addEdge(G, 4, 3);
    addEdge(G, 4, 5);

    addEdge(G, 5, 4);
```

Asep Budiyana M
140810180029
Tugas 5

```
    addEdge(G, 6, 2);
    addEdge(G, 6, 7);

    addEdge(G, 7, 2);
    addEdge(G, 7, 6);

    traversal(G);
}
```

2. Dengan menggunakan undirected graph dan representasi adjacency list, buatlah coding programnya menggunakan bahasa C++.



Jawab:
adj_list.cpp:

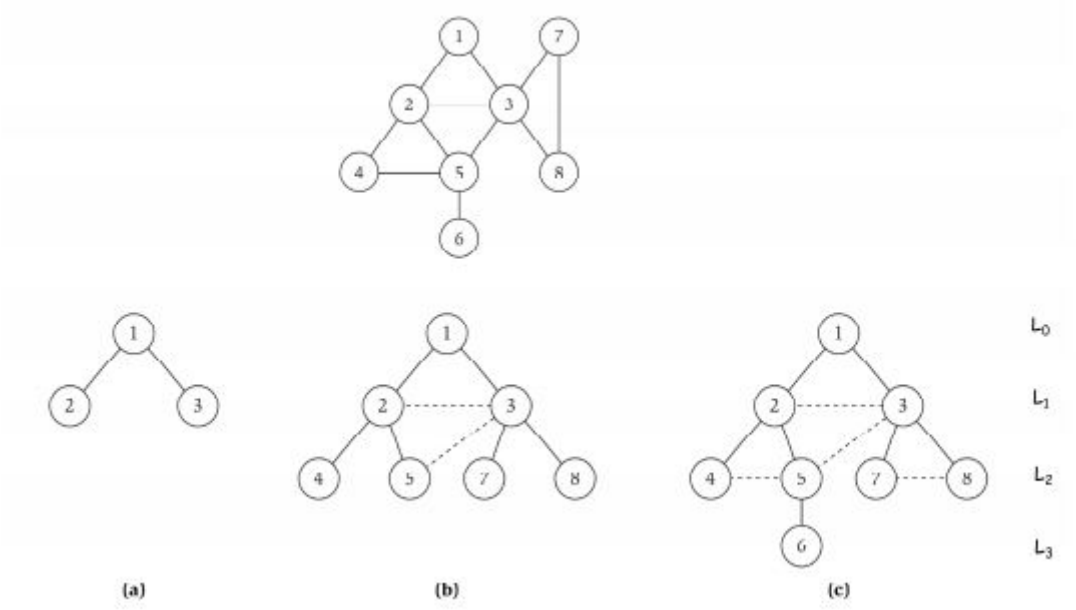
```
#include <iostream>
#include <list>
using namespace std;

struct Graph {
    int vertex;
    list<int>* edge;
};
Graph G;

void makeGraph(Graph& G, int vertex)
{
    G.vertex = vertex;
    G.edge = new list<int>[vertex];
}

void addEdge(Graph& G, int i, int j)
{
```

Asep Budiyana M
140810180029
Tugas 5

```
    G.edge[i].push_back(j);
}

void traversal(Graph G)
{
    for (int i=0; i<G.vertex; ++i)
    {
        cout<<"\n vertex "<<i<<"\n head";
        for (auto x : G.edge[i])
            cout<<" -> "<<x;
        cout<<endl;
    }
}

int main()
{
    makeGraph(G, 8);
    addEdge(G, 0, 1);
    addEdge(G, 0, 2);
    addEdge(G, 1, 0);
    addEdge(G, 1, 2);
    addEdge(G, 1, 3);
    addEdge(G, 1, 4);

    addEdge(G, 2, 0);
    addEdge(G, 2, 1);
    addEdge(G, 2, 4);
    addEdge(G, 2, 6);
    addEdge(G, 2, 7);

    addEdge(G, 3, 1);
    addEdge(G, 3, 4);
    addEdge(G, 4, 1);
    addEdge(G, 4, 2);
    addEdge(G, 4, 3);
    addEdge(G, 4, 5);

    addEdge(G, 5, 4);
    addEdge(G, 6, 2);
    addEdge(G, 6, 7);
    addEdge(G, 7, 2);
    addEdge(G, 7, 6);

    traversal(G);
}
```

Asep Budiyana M
140810180029
Tugas 5

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:

fungsi BFS dengan adjacency list : BFS.cpp:

```
void BFS(Graph G, int s)
{
    bool *visited = new bool[G.vertex];
    for (int i=0; i<G.vertex; i++)
        visited[i] = false;
    list<int> queue;
    visited[s] = true;
    queue.push_back(s);

    while (!queue.empty()) {
        s = queue.front();
        cout<<s<<" ";
        queue.pop_front();
        for (list<int>::iterator i=G.edge[s].begin(); i != G.edge[s].end(); ++i) {
            if (!visited[*i]) {
                visited[*i] = true;
                queue.push_back(*i);
            }
        }
    }
}
```

Asep Budiyana M
140810180029
Tugas 5

uji coba program:

```
BFS starting from vertex 0
0 1 2 3 4 6 7 5
```

Kompleksitas waktu asimptotik:

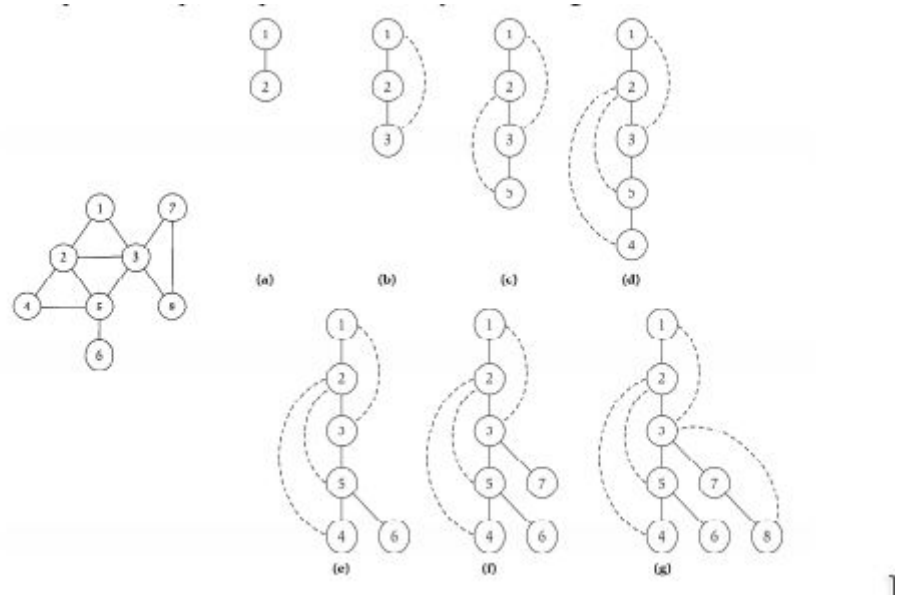
V : Jumlah Vertex
E : Jumlah Edge

- menandai setiap vertex belum dikunjungi : O(V)
- menandai vertex awal telah dikunjungi lalu masukan ke queue : O(1)
- keluarkan vertex dari queue kemudian cetak : O(V)
- kunjungi setiap vertex yang belum dikunjungi kemudian masukan ke queue : O(E)

maka :

T(n) = O(V) + O(1) + O(V) + O(E)
= O(mak(V,1)) + O(V) + O(E)
= O(V) + O (V) + O(E)
= O(maks(V,V)) + O(E)
= O(V) + O(E)
= O(V+E)

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!



Asep Budiyana M
140810180029
Tugas 5

Jawab :

fungsi DFS dengan adjacency list : DFS.cpp:

```
void DFSUtil(int v, bool visited[])
{
    visited[v] = true;
    cout<<v<<" ";

    for (list<int>::iterator i = G.edge[v].begin(); i != G.edge[v].end(); ++i)
        if(!visited[*i])
            DFSUtil(*i, visited);
}

void DFS(Graph G, int s)
{
    bool *visited = new bool[G.vertex];
    for (int i=0; i<G.vertex; i++)
        visited[i] = false;

    for (int i=0; i<G.vertex; i++)
        if (visited[i] == false)
            DFSUtil(i, visited);
}
```

uji coba program:

```
DFS starting from vertex 0
0 1 2 4 3 5 6 7
```

Kompleksitas waktu asimptotik:

V : Jumlah Vertex
E : Jumlah Edge

- menandai vertex awal telah dikunjungi kemudian cetak : O(1)
- rekursif untuk semua vertex : T(E/1)
- tandi semua vertex belum dikunjungi : O(V)
- rekursif untuk mencetak DFS : T(V/1)

maka :

T(n) = O(1) + T(E/1) + O(V) + T(V/1)
= O(1) + O(E) + O(V) + O(V)
= O(mak(1,E)) + O(V) + O(V)
= O(E) + O (V) + O(V)
= O(maks(V,V)) + O(E)
= O(V) + O(E)
= O(V+E)