LAPORAN PRAKTIKUM ANALISIS ALGORITMA

UNDIRECTED GRAPH

Oleh

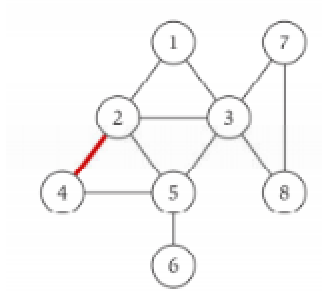
Shalvina Zahwa Aulia (140810180052)

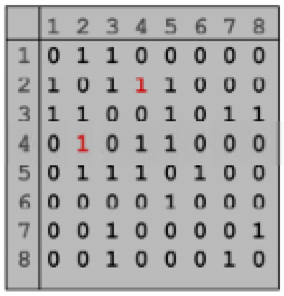
PROGRAM STUDI S-1 TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN MARET

2020



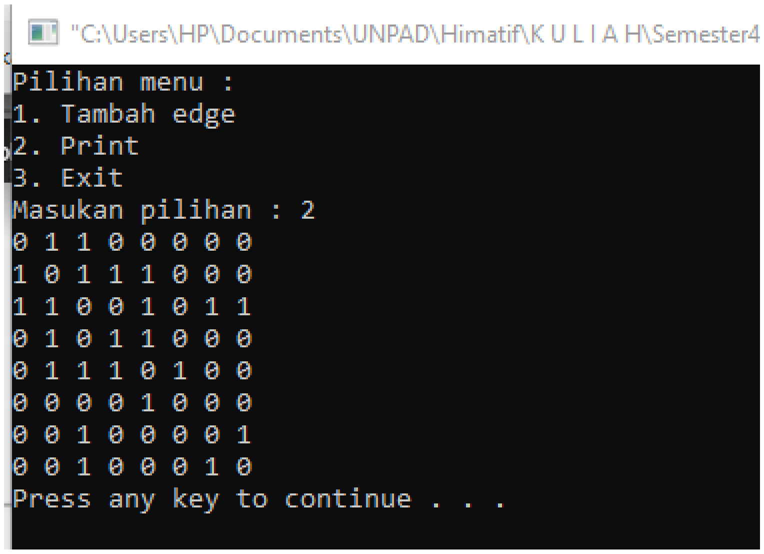


	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	1	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

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

   Jawaban :

```
/*
Nama    : Shalvina Zahwa Aulia
NPM     : 140810180052
Tugas 6 Adjacency Matrix
*/
#include <iostream>
#include<windows.h>
using namespace std;

int vertex[20][20];
int count = 0;
void printMatrix(int v)
{
    int i, j;
    for (i = 0; i < v; i++)
    {
        for (j = 0; j < v; j++)
        {
            cout << vertex[i][j] << " ";
        }
        cout << endl;
    }
}
void add_edge(int a, int b)
{
    vertex[a][b] = 1;
    vertex[b][a] = 1;
}
main(int argc, char *argv[])
{
    int v;
    cout << "Masukkan jumlah matrix : ";cin >> v;

    int pilihan,a,b;
    while(true){
        cout << "Pilihan menu : " << endl;
        cout << "1. Tambah edge " << endl;
        cout << "2. Print " << endl;
        cout << "3. Exit " << endl;
        cout << "Masukan pilihan : "; cin >> pilihan;
        switch (pilihan)
        {
            case 1:
                cout << "Masukkan node A : "; cin >> a;
                cout << "Masukkan node B : "; cin >> b;
                add_edge(a,b);
                cout << "Edge ditambahkan\n";
                system("Pause");
                system("CLS");
                break;
            case 2:
                printMatrix(v);
            system("Pause");
            system("CLS");
            break;
        case 3:
            return 0;
            break;
        default:
            break;
      }
   }
}
```



"C:\Users\HP\Documents\UNPAD\Himatif\K U L I A H\Semester4

```
Pilihan menu :
1. Tambah edge
2. Print
3. Exit
Masukan pilihan : 2
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
Press any key to continue . . .
```

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

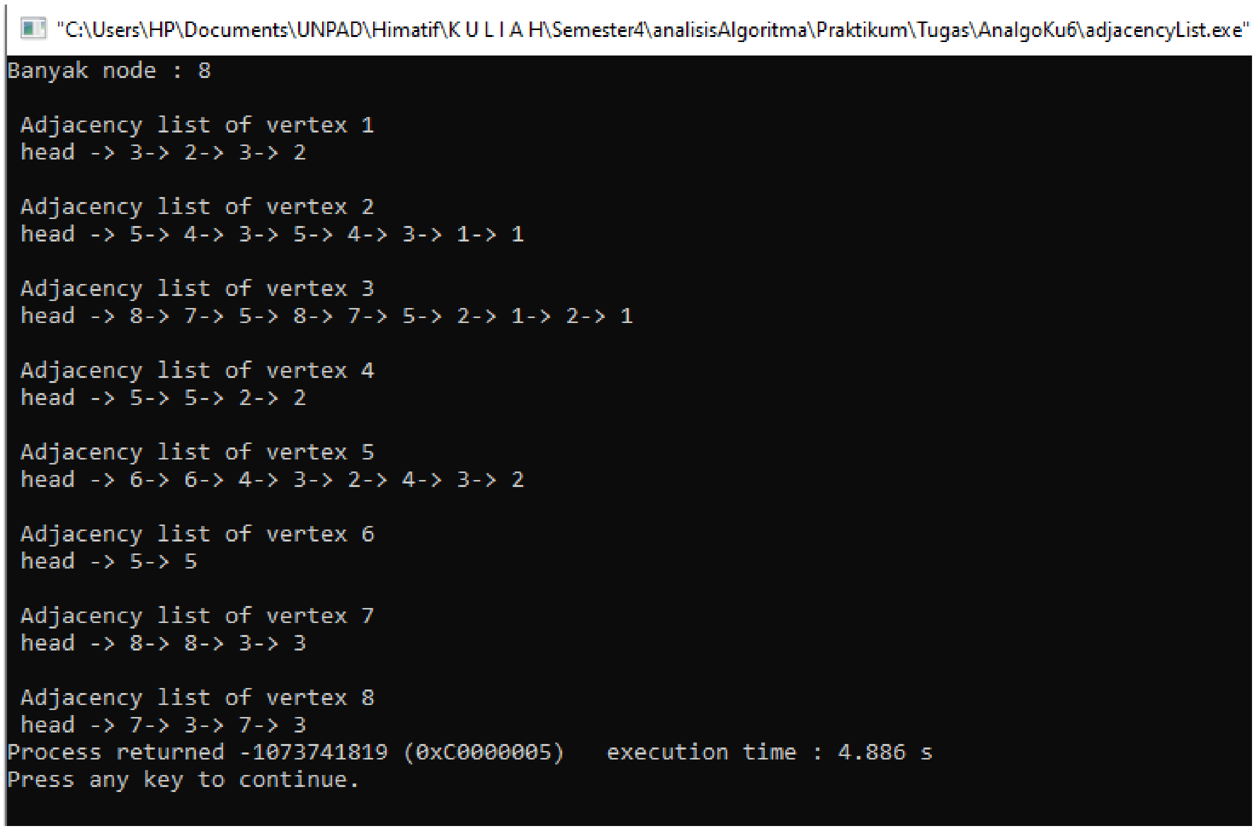
Jawaban :

```
/*
Nama    : Shalvina Zahwa Aulia
NPM     : 140810180052
Tugas 6 Adjacency List
```

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

struct AdjListNode{
    int dest;
    struct AdjListNode* next;
};

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

/*
* Class Graph
*/
class Graph{
    private:
        int V;
        struct AdjList* array;
    public:
        Graph(int V){
            this->V = V;
            array = new AdjList [V];
            for (int i = 1; i <= V; ++i)
                array[i].head = NULL;
        }
        /*
```

```
* Creating New Adjacency List Node
*/
AdjListNode* newAdjListNode(int dest){
    AdjListNode* newNode = new AdjListNode;
    newNode->dest = dest;
    newNode->next = NULL;
    return newNode;
}
/*
* Adding Edge to Graph
*/
void addEdge(int src, int dest){
    AdjListNode* newNode = newAdjListNode(dest);
    newNode->next = array[src].head;
    array[src].head = newNode;
    newNode = newAdjListNode(src);
    newNode->next = array[dest].head;
    array[dest].head = newNode;
}
/*
* Print the 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->dest;
            pCrawl = pCrawl->next;
        }
```

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

int main(){
    int n;
    cout << "Banyak node : "; cin >> n;
    Graph gh(n);
        gh.addEdge(1, 2);
        gh.addEdge(1, 3);
        gh.addEdge(2, 1);
        gh.addEdge(2, 3);
        gh.addEdge(2, 4);
        gh.addEdge(2, 5);
        gh.addEdge(3, 1);
        gh.addEdge(3, 2);
        gh.addEdge(3, 5);
        gh.addEdge(3, 7);
        gh.addEdge(3, 8);
        gh.addEdge(4, 2);
        gh.addEdge(4, 5);
        gh.addEdge(5, 2);
        gh.addEdge(5, 3);
        gh.addEdge(5, 4);
        gh.addEdge(5, 6);
        gh.addEdge(6, 5);
        gh.addEdge(7, 3);
        gh.addEdge(7, 8);
        gh.addEdge(8, 3);
        gh.addEdge(8, 7);
        gh.printGraph();

    return 0;
}
```



"C:\Users\HP\Documents\UNPAD\Himatif\K U L I A H\Semester4\analisisAlgoritma\Praktikum\Tugas\AnalgoKu6\adjacencyList.exe"

```
Banyak node : 8

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

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

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

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

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

 Adjacency list of vertex 6
 head -> 5-> 5

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

 Adjacency list of vertex 8
 head -> 7-> 3-> 7-> 3
Process returned -1073741819 (0xC0000005)   execution time : 4.886 s
Press any key to continue.
```

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!

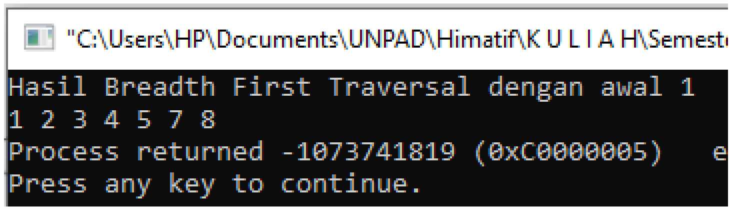
Jawaban :

```
/*
Nama    : Shalvina Zahwa Aulia
NPM     : 140810180052
Tugas 6 BFS
*/
```

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

using namespace std;

class Graph{
        int V;
        list<int> *adj;
public:
        Graph(int V);
        void addEdge(int v, int w);
        void BFS(int s);
};

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

void Graph::addEdge(int v, int w){
        adj[v].push_back(w);
}

void Graph::BFS(int s){
        bool *visited = new bool[V];
        for(int i = 0; i < V; i++)
                visited[i] = false;
        list<int> queue;
        visited[s] = true;
        queue.push_back(s);
        list<int>::iterator i;
        while(!queue.empty()){
                s = queue.front();
                cout << s << " ";
                queue.pop_front();
                for (i = adj[s].begin(); i != adj[s].end(); ++i){
                        if (!visited[*i]){
                                visited[*i] = true;
                                queue.push_back(*i);
                        }
                }
        }
}

int main(){
        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 << "Hasil Breadth First Traversal dengan awal 1\n";
        g.BFS(1);
```

return 0;

}



```
"C:\Users\HP\Documents\UNPAD\Himatif\K U L I A H\Semest
Hasil Breadth First Traversal dengan awal 1
1 2 3 4 5 7 8
Process returned -1073741819 (0xC0000005)   e
Press any key to continue.
```

- Dalam worst case, BFS harus mempertimbangkan semua jalur (dari kiri ke kanan) untuk semua node yang mungkin, maka

  T(n) = O(|V| + |E|)

  n = v+e

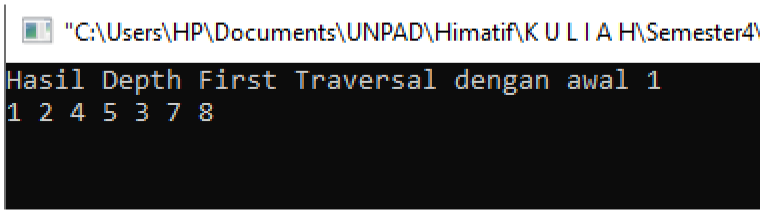
  Big-O =O(n)

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!

   Jawaban :

```
/*
Nama    : Shalvina Zahwa Aulia
NPM     : 140810180052
Tugas 6 DFS
*/
#include<iostream>
#include<list>
using namespace std;
class Graph
{
   int V;
   list<int> *adj;
   void DFSUtil(int v, bool visited[]);
```

```
public:
    Graph(int V);
    void addEdge(int v, int w);
    void DFS(int v);
};

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

void Graph::addEdge(int v, int w)
{
    adj[v].push_back(w);
}

void Graph::DFSUtil(int v, bool visited[])
{
    visited[v] = true;
    cout << v << " ";
    list<int>::iterator i;
    for (i = adj[v].begin(); i != adj[v].end(); ++i)
        if (!visited[*i])
            DFSUtil(*i, visited);
}
void Graph::DFS(int v)
{
    bool *visited = new bool[V];
    for (int i = 0; i < V; i++)
        visited[i] = false;
    DFSUtil(v, visited);
}

int main()
{
    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 << "Hasil Depth First Traversal dengan awal 1\n";
    g.DFS(1);

    return 0;
}
```



"C:\Users\HP\Documents\UNPAD\Himatif\K U L I A H\Semester4\

```
Hasil Depth First Traversal dengan awal 1
1 2 4 5 3 7 8
```

- Kompleksitas waktu DFS adalah O(bm) karena hanya perlu menyimpan satu lintasan tunggal dari aka sampai daun, ditambah simpul saudara kandungnya yang belum dikembangkan
- Kompleksitas total :

  (V+E) = n

  T(n) = O(n)