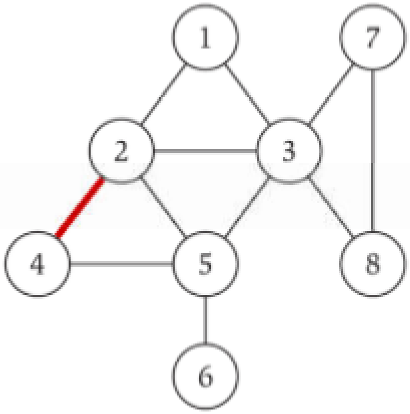
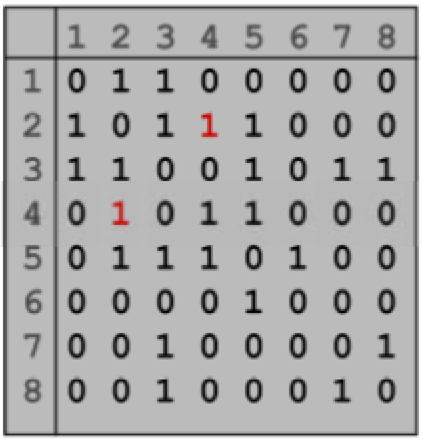
Nama : Prayudha Adhitia Libramawan

NPM : 140810180008

Tugas 6

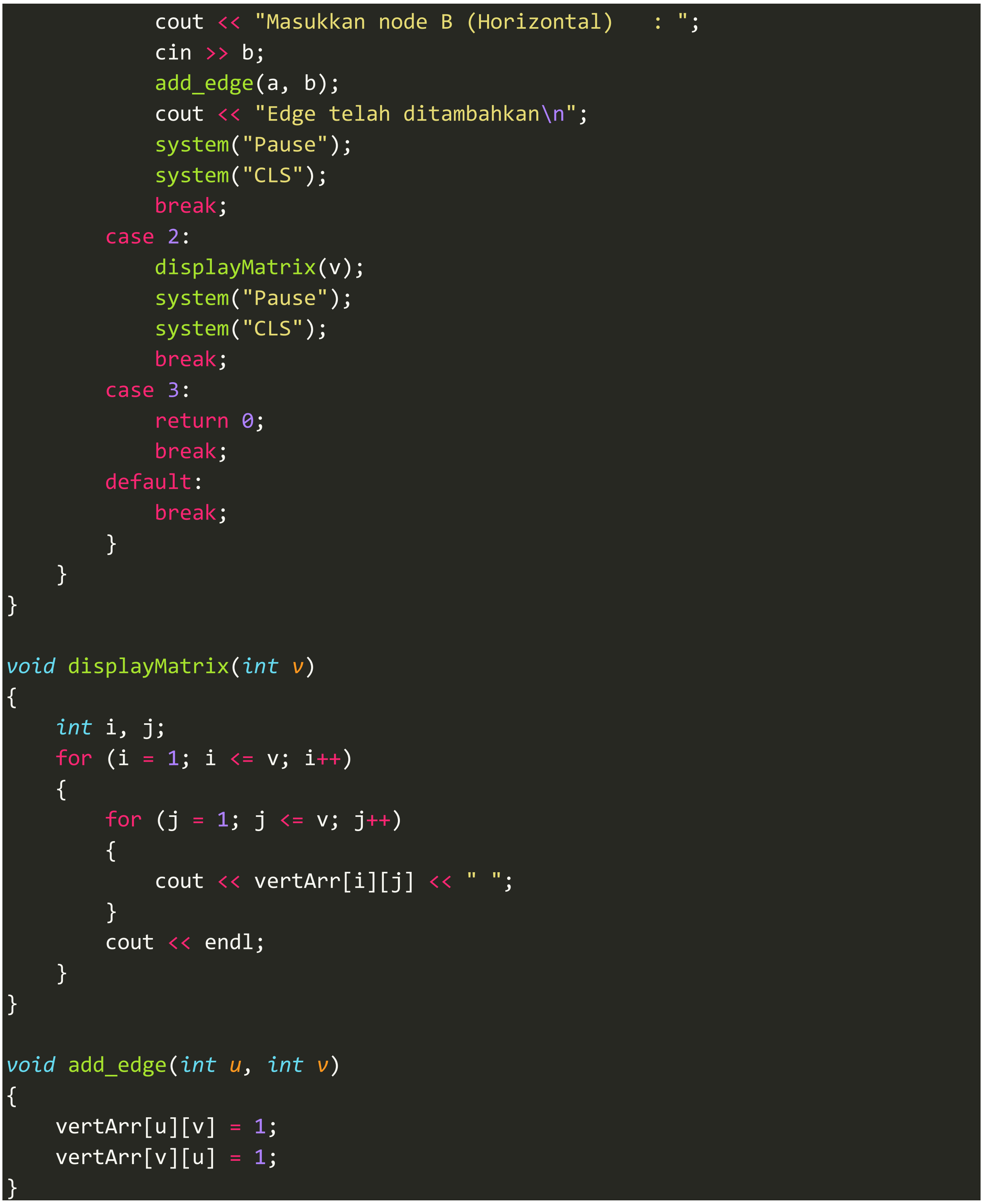
1. Dengan menggunakan undirected graph dan adjacency matrix berikut, buatlah koding programmnya menggunakan bahasa C++.





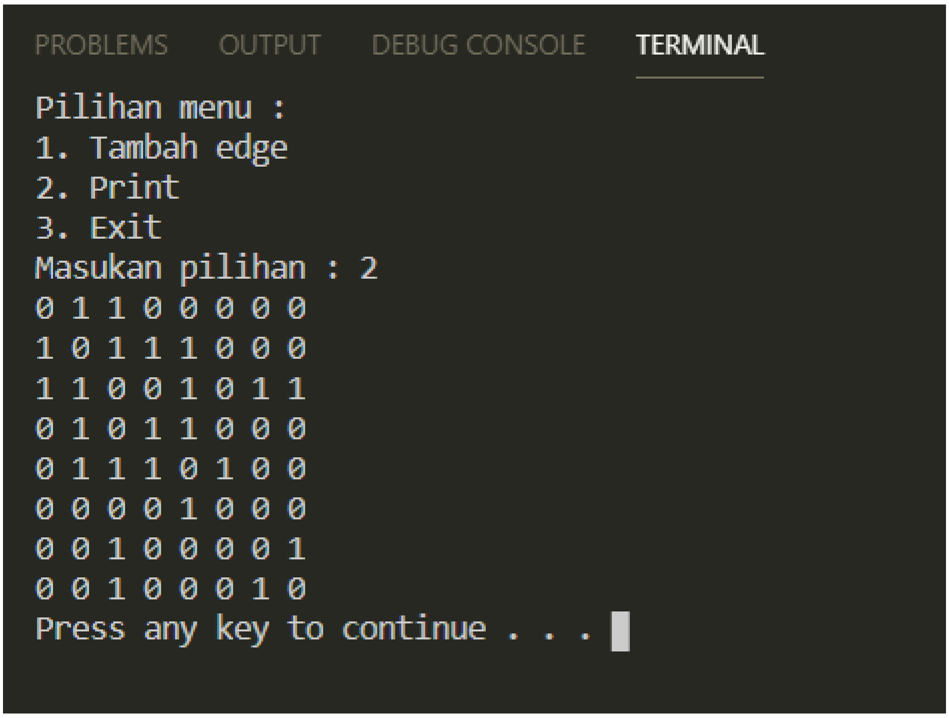
	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	1	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

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

int vertArr[20][20];
int count = 0;
void displayMatrix(int v);
void add_edge(int u, int v);

int 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 (vertical)      : ";
            cin >> a;
```



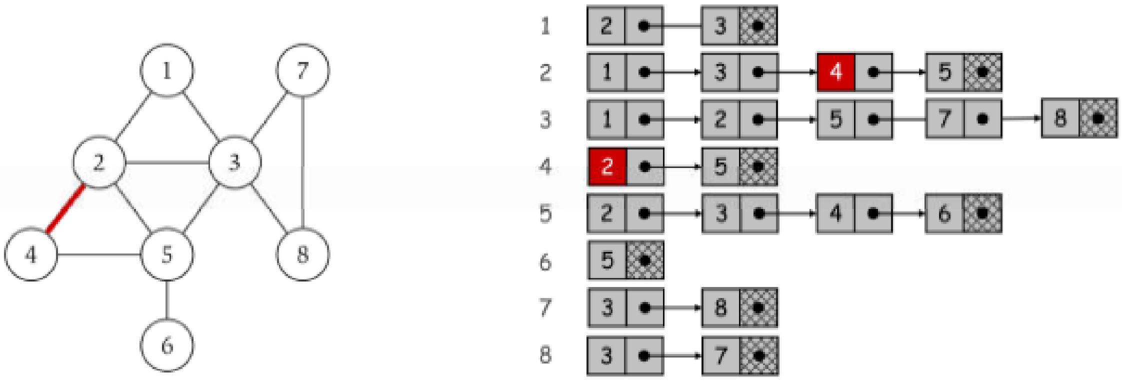
```
            cout << "Masukkan node B (Horizontal)   : ";
            cin >> b;
            add_edge(a, b);
            cout << "Edge telah ditambahkan\n";
            system("Pause");
            system("CLS");
            break;
        case 2:
            displayMatrix(v);
            system("Pause");
            system("CLS");
            break;
        case 3:
            return 0;
            break;
        default:
            break;
        }
    }
}

void displayMatrix(int v)
{
    int i, j;
    for (i = 1; i <= v; i++)
    {
        for (j = 1; j <= v; j++)
        {
            cout << vertArr[i][j] << " ";
        }
        cout << endl;
    }
}

void add_edge(int u, int v)
{
    vertArr[u][v] = 1;
    vertArr[v][u] = 1;
}
```

Screenshot



```
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 . . .
```

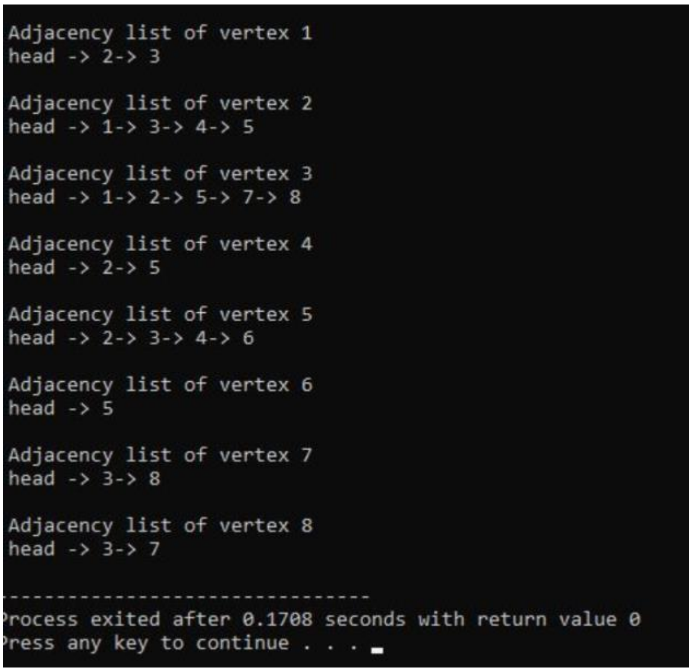
2. Dengan menggunakan undirected graph dan representasi adjacency list, buatlah koding programmnya menggunakan bahasa C++.



```
/**
 * Author    : Prayudha Adhitia Libramawan
 * NPM       : 140810180008
 * Deskripsi: adjacency list
 * Tahun     : 2020
 */

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

struct AdjListNode
```

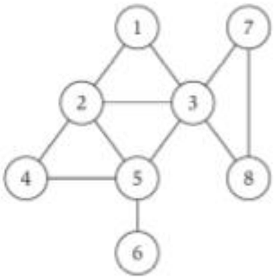
```
{
    int dest;
    struct AdjListNode *next;
};

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 = 1; i <= V; ++i)
            array[i].head = NULL;
    }

    AdjListNode *newAdjListNode(int dest)
    {
        AdjListNode *newNode = new AdjListNode;
        newNode->dest = dest;
        newNode->next = NULL;
        return newNode;
    }

    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;
    }

    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()
{
    Graph g(8);
    g.addEdge(7, 8);
    g.addEdge(5, 6);
    g.addEdge(3, 8);
    g.addEdge(3, 7);
    g.addEdge(4, 5);
    g.addEdge(5, 3);
    g.addEdge(2, 5);
    g.addEdge(2, 4);
    g.addEdge(2, 3);
    g.addEdge(1, 3);
    g.addEdge(1, 2);

    g.printGraph();
}
```

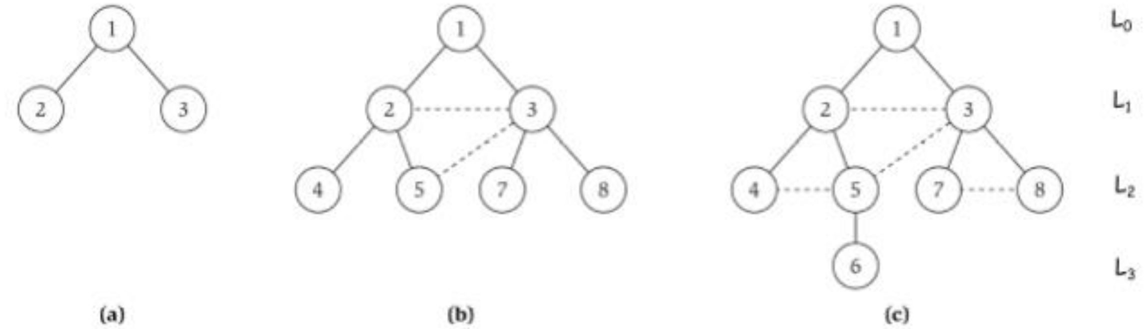
Screenshot



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

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

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

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

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

Adjacency list of vertex 6
head -> 5

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

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

--------------------------------
rocess exited after 0.1708 seconds with return value 0
ress 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!



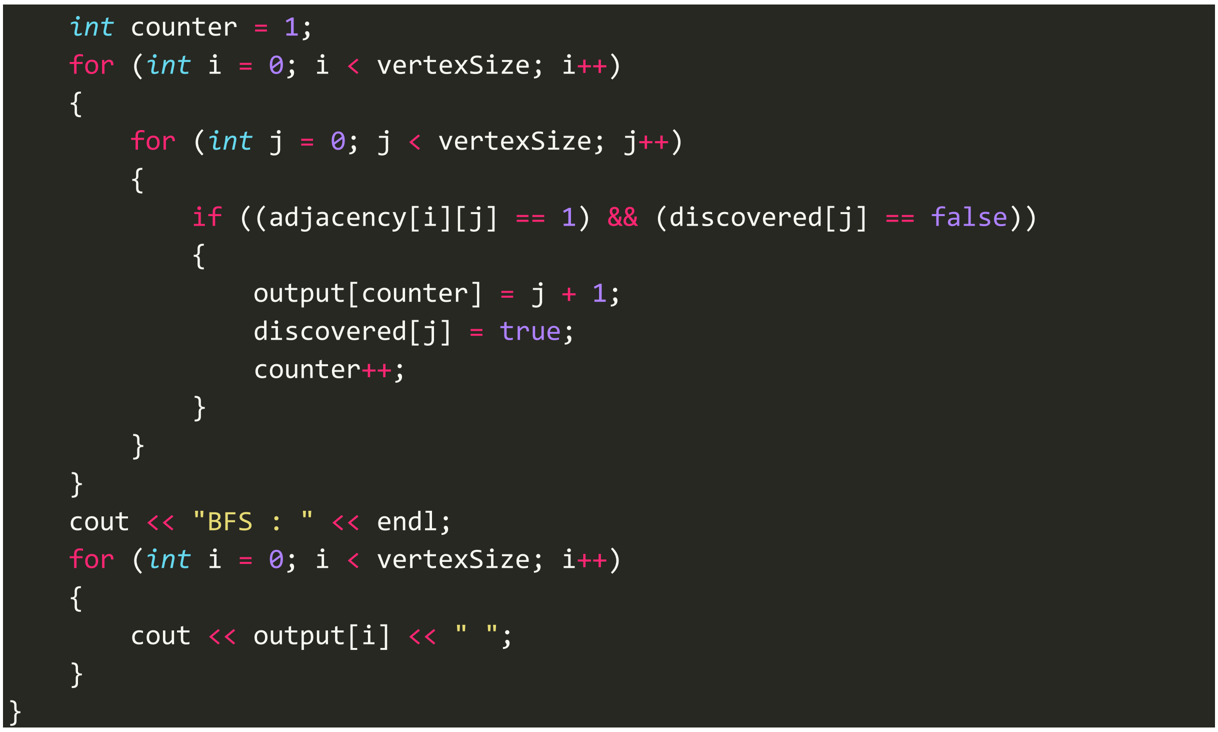


(a)

(b)

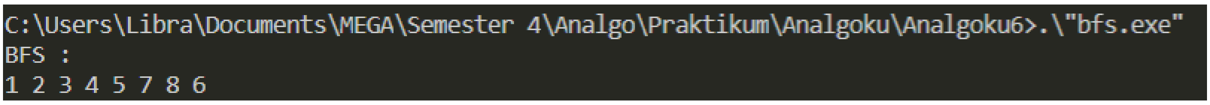
(c)

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

int main()
{
    int vertexSize = 8;
    int adjacency[8][8] = {
        {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, 0, 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}};
    bool discovered[vertexSize];
    for (int i = 0; i < vertexSize; i++)
    {
        discovered[i] = false;
    }
    int output[vertexSize];

    //inisialisasi start
    discovered[0] = true;
    output[0] = 1;
```



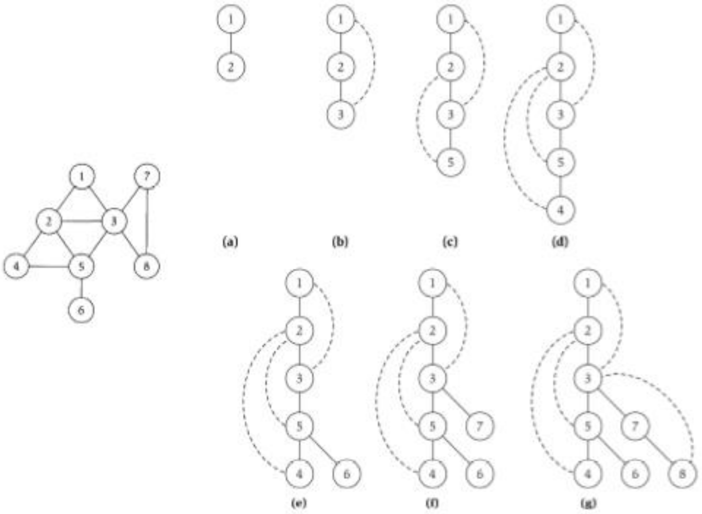
```
    int counter = 1;
    for (int i = 0; i < vertexSize; i++)
    {
        for (int j = 0; j < vertexSize; j++)
        {
            if ((adjacency[i][j] == 1) && (discovered[j] == false))
            {
                output[counter] = j + 1;
                discovered[j] = true;
                counter++;
            }
        }
    }
    cout << "BFS : " << endl;
    for (int i = 0; i < vertexSize; i++)
    {
        cout << output[i] << " ";
    }
}
```

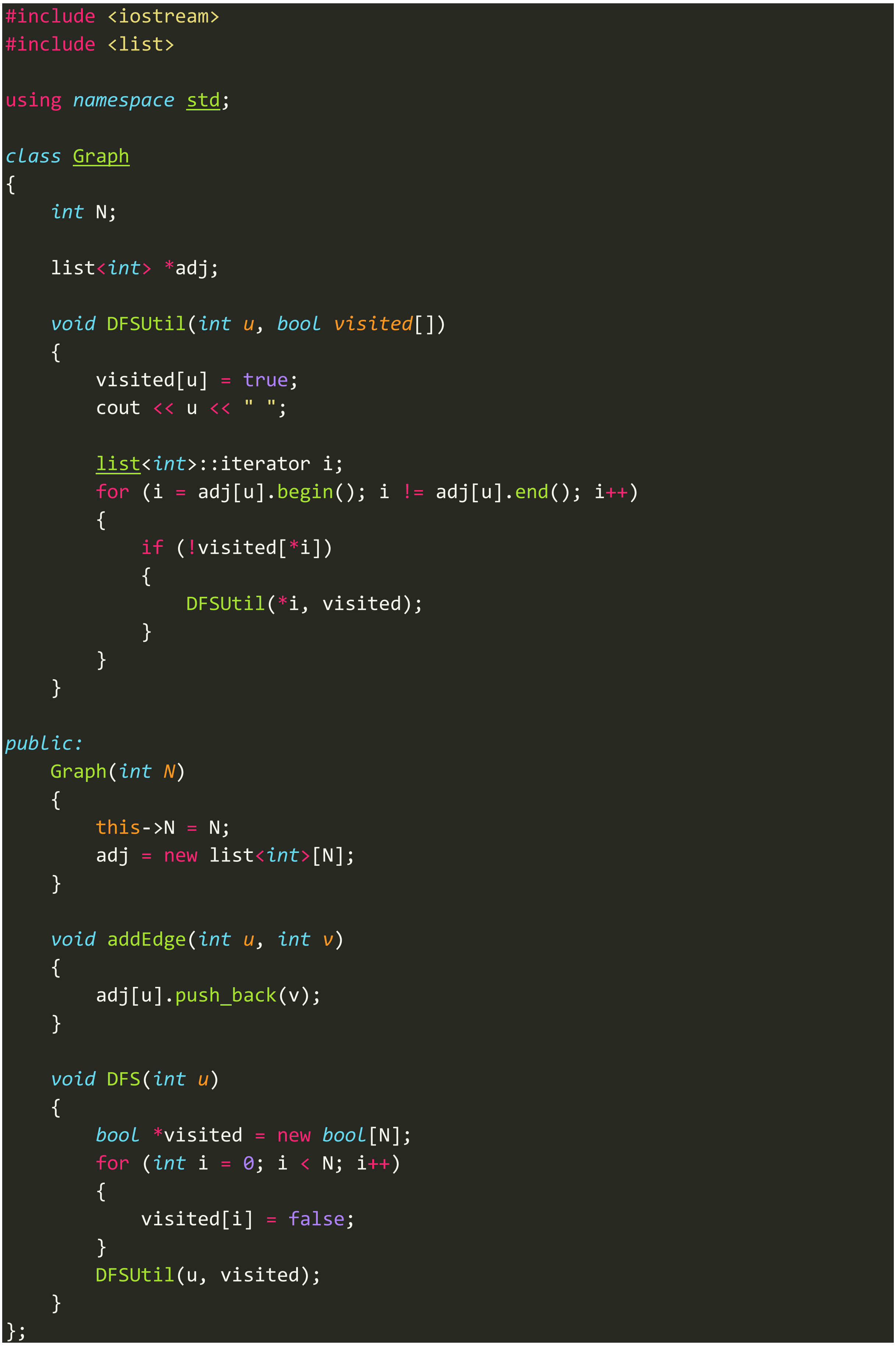
Screenshot



```
C:\Users\Libra\Documents\MEGA\Semester 4\Analgo\Praktikum\Analgoku\Analgoku6>.\"bfs.exe"
BFS :
1 2 3 4 5 7 8 6
```

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!

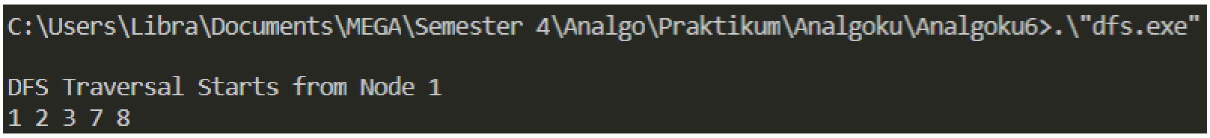




```
#include <iostream>
#include <list>

using namespace std;

class Graph
{
    int N;

    list<int> *adj;

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

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

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

    void addEdge(int u, int v)
    {
        adj[u].push_back(v);
    }

    void DFS(int u)
    {
        bool *visited = new bool[N];
        for (int i = 0; i < N; i++)
        {
            visited[i] = false;
        }
        DFSUtil(u, visited);
    }
};
```

```
int main()
{
    Graph g(8);

    g.addEdge(1, 2);
    g.addEdge(1, 3);
    g.addEdge(2, 3);
    g.addEdge(2, 4);
    g.addEdge(2, 5);
    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 << "\nDFS Traversal Starts from Node 1" << endl;
    g.DFS(1);

    return 0;
}
```

Screenhot



```
C:\Users\Libra\Documents\MEGA\Semester 4\Analgo\Praktikum\Analgoku\Analgoku6>.\"dfs.exe"

DFS Traversal Starts from Node 1
1 2 3 7 8
```