

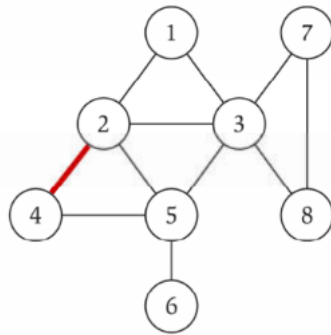
Nama : Hana Meilina Fauziyyah

NPM : 140810180012

Tugas 6

### Tugas Anda

1. Dengan menggunakan *undirected graph* dan *adjacency matrix* berikut, buatlah koding 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 :

```
/*
Nama      : Hana Meilina Fauziyyah
NPM       : 140810180012
Kelas    : B
Program   : Adjacency Matriks
*****/

#include <iostream>
using namespace std;

void input(int &n){
    cout<<"Masukkan banyaknya simpul : ";
    cin>>n;
}

void input(int arr[], int size){
    for (int i = 0; i < size; i++) {
        cout<<"Masukkan simpul "<<i+1<<" : ";
        cin>>arr[i];
    }
}

void input(int &num, int value){
    num=value;
}

void print(int arr[], int n){
    for(int i=0;i<n;i++) {
        cout<<arr[i]<<"\t";
    }
}

bool check(int arr[], int n, int value){
```

Nama : Hana Meilina Fauziyyah

NPM : 140810180012

### Tugas 6

```
    for(int i=0;i<n;i++) {
        if (arr[i]==value)
            return true;
    }
    return false;
}

int main(){
    int n, val;
    cout<<"===== "<<endl;
    cout<<"                Adjacency Matriks"<<endl;
    cout<<"===== "<<endl;
    input(n);

    int simpul[n];
    input(simpul, n);

    int garis[n][n];
    cout<<"===== "<<endl;
    for (int i = 0; i < n; i++) {
        int edge;
        cout<<"Masukkan banyak garis pada simpul ke-"<<i+1<<" ("<<simpul[i]<<") : ";
        cin>>edge;

        for (int j = 0; j < edge; j++) {
            bool found = false;

            do {
                cout<<"Simpul garis ke-"<<i+1<<" : ";
                cin>>val;

                found = check(simpul, n, val);
                if (!found)
                    cout<<endl<<"Simpul tidak ditemukan!"<<endl;
            } while (!found);

            input(garis[i][j], val);
        }
    }

    cout<<"===== "<<endl;
    cout<<endl<<"Adjacency matrix : \n\t";
    print(simpul, n);

    for (int i = 0; i < n; i++) {
        cout<<endl<<simpul[i]<<"\t";

        for (int j = 0; j < n; j++) {
            bool found = false;

            for (int k = 0; k < n; k++) {
                if (garis[i][k] == simpul[j])
                    found = true;
            }
        }
    }
}
```

Nama : Hana Meilina Fauziyyah

NPM : 140810180012

Tugas 6

```
        if(found)
            cout<<"1\t";
        else
            cout<<"0\t";
    }

}

cout<<endl;
cout<<"===== "<<endl;
}
```

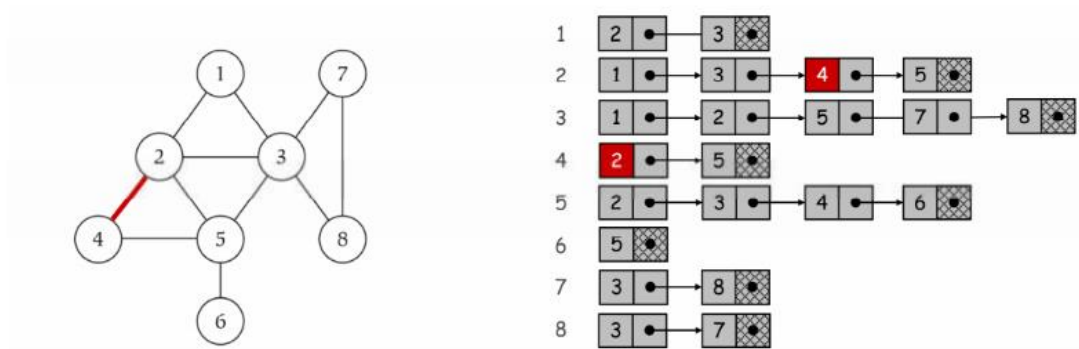
```
"D:\Semester 4\Analgo\Praktikum\New folder (2)\matriks.exe"
=====
Adjacency Matriks
=====
Masukkan banyaknya simpul : 8
Masukkan simpul 1 : 1
Masukkan simpul 2 : 2
Masukkan simpul 3 : 3
Masukkan simpul 4 : 4
Masukkan simpul 5 : 5
Masukkan simpul 6 : 6
Masukkan simpul 7 : 7
Masukkan simpul 8 : 8
=====
Masukkan banyak garis pada simpul ke-1 (1) : 2
Simpul garis ke-1 : 2
Simpul garis ke-1 : 3
Masukkan banyak garis pada simpul ke-2 (2) : 4
Simpul garis ke-2 : 1
Simpul garis ke-2 : 3
Simpul garis ke-2 : 4
Simpul garis ke-2 : 5
Masukkan banyak garis pada simpul ke-3 (3) : 5
Simpul garis ke-3 : 1
Simpul garis ke-3 : 2
Simpul garis ke-3 : 5
Simpul garis ke-3 : 7
Simpul garis ke-3 : 8
Masukkan banyak garis pada simpul ke-4 (4) : 2
Simpul garis ke-4 : 2
Simpul garis ke-4 : 5
Masukkan banyak garis pada simpul ke-5 (5) : 4
Simpul garis ke-5 : 2
Simpul garis ke-5 : 3
Simpul garis ke-5 : 4
Simpul garis ke-5 : 6
Masukkan banyak garis pada simpul ke-6 (6) : 1
Simpul garis ke-6 : 5
Masukkan banyak garis pada simpul ke-7 (7) : 2
Simpul garis ke-7 : 3
Simpul garis ke-7 : 8
Masukkan banyak garis pada simpul ke-8 (8) : 2
Simpul garis ke-8 : 3
Simpul garis ke-8 : 7
=====
Adjacency matrix :
      1      2      3      4      5      6      7      8
1      1      0      1      0      0      0      0      0
2      0      1      0      1      1      0      0      0
3      1      1      1      0      0      1      0      1
4      1      1      0      0      1      0      0      0
5      0      1      1      1      0      1      0      0
6      0      0      1      0      1      0      0      1
7      0      0      1      0      0      0      0      1
8      0      0      1      0      0      0      1      1
=====
Process returned 0 (0x0)   execution time : 65.387 s
Press any key to continue.
```

Nama : Hana Meilina Fauziyyah

NPM : 140810180012

Tugas 6

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



**Jawab :**

```
/*
Nama      : Hana Meilina Fauziyyah
NPM       : 140810180012
Kelas    : B
Program   : Adjacency List
*****/

#include <iostream>
using namespace std;

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

struct AdjList{
    struct nodeList *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;
    }

    nodeList* newAdjListNode(int dest)
    {
        nodeList* simpulBaru = new nodeList;
```

Nama : Hana Meilina Fauziyyah

NPM : 140810180012

Tugas 6

```
        simpulBaru->dest = dest;
        simpulBaru->next = NULL;
        return simpulBaru;
    }

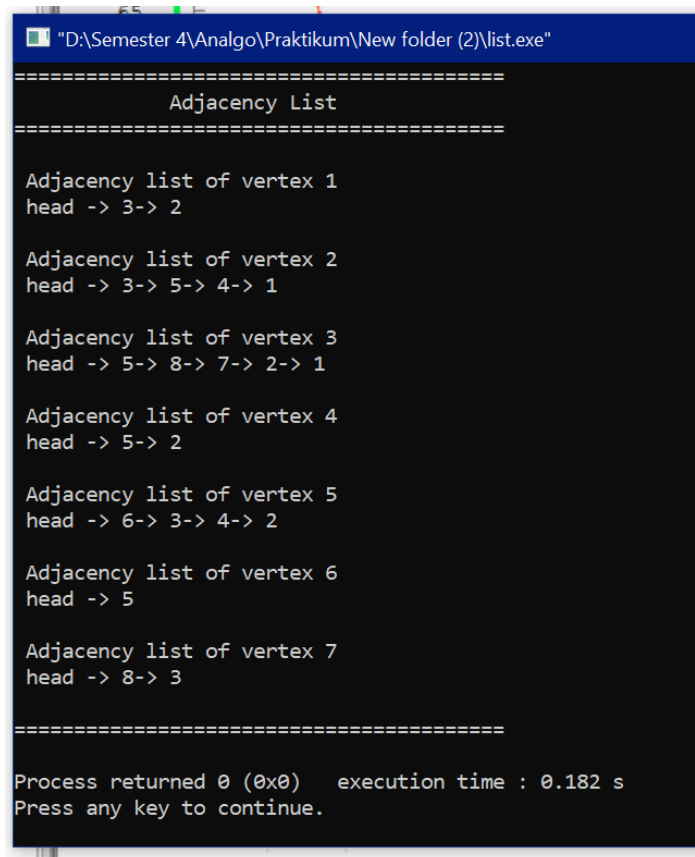
    void addEdge(int src, int dest)
    {
        nodeList* simpulBaru = newAdjListNode(dest);
        simpulBaru->next = array[src].head;
        array[src].head = simpulBaru;
        simpulBaru = newAdjListNode(src);
        simpulBaru->next = array[dest].head;
        array[dest].head = simpulBaru;
    }

    void printGraph()
    {
        int v;
        for (v = 1; v < V; ++v)
        {
            nodeList* pCrawl = array[v].head;
            cout<<"\n Adjacency list of vertex "<<v<<"\n head ";
            while (pCrawl)
            {
                cout<<"-> "<<pCrawl->dest;
                pCrawl = pCrawl->next;
            }
            cout<<endl;
        }
    }

};

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

    return 0;
}
```



```
"D:\Semester 4\Analgo\Praktikum\New folder (2)\list.exe"

=====
Adjacency List
=====

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

=====

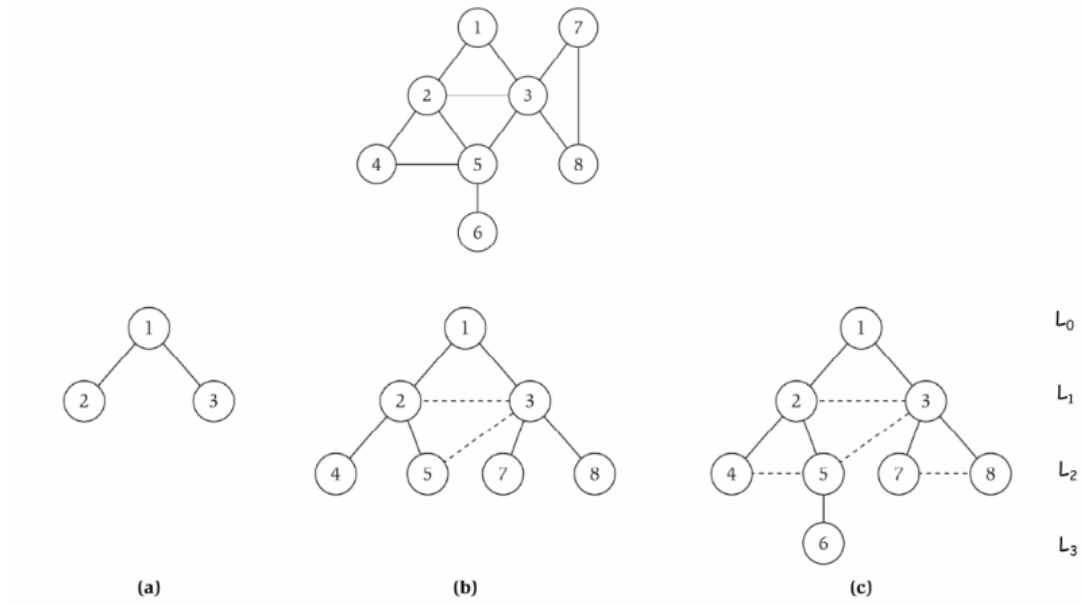
Process returned 0 (0x0)   execution time : 0.182 s
Press any key to continue.
```

Nama : Hana Meilina Fauziyyah

NPM : 140810180012

Tugas 6

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- $\Theta$ !



**Jawab :**

```
/*
Nama      : Hana Meilina Fauziyyah
NPM       : 140810180012
Kelas    : B
Program   : Breadth First Search
*****/

#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];
    discovered[0] = true;
    output[0] = 1;
```

Nama : Hana Meilina Fauziyyah

NPM : 140810180012

### Tugas 6

```
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<<"===== "<<endl;
cout<<"          Breadth First Search"<<endl;
cout<<"===== "<<endl;
cout<<endl;
cout<<"BFS : "<<endl;
for(int i = 0; i < vertexSize; i++){
    cout<<output[i]<<" ";
}
cout<<endl;
cout<<endl;
cout<<"===== "<<endl;
}
```

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 kemudia masukan ke queue :  $O(E)$

Maka :

Nama : Hana Meilina Fauziyyah

NPM : 140810180012

Tugas 6

$$T(n) = O(V) + O(1) + O(V) + O(E)$$

$$= O(\text{maks}(V,1)) + O(V) + O(E)$$

$$= O(V) + O(V) + O(E)$$

$$= O(\text{maks}(V,V)) + O(E)$$

$$= O(V) + O(E)$$

$$= O(V+E)$$

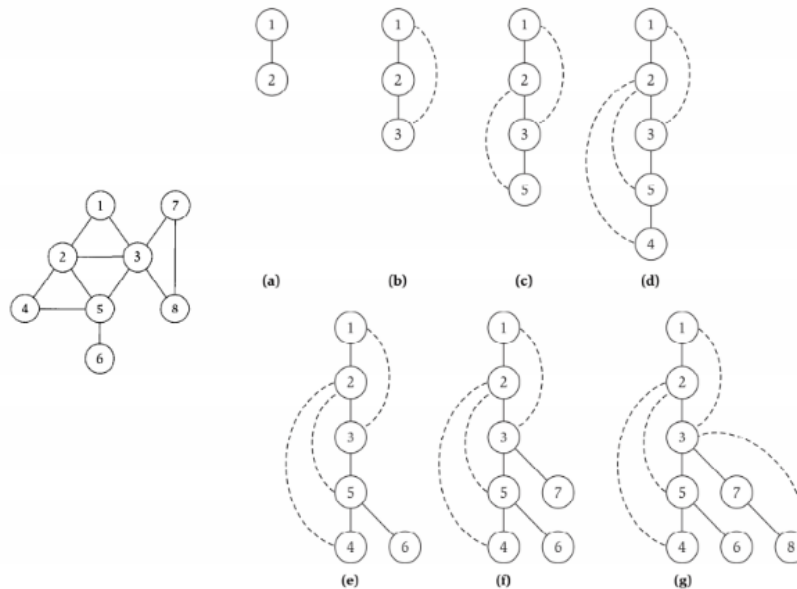


Nama : Hana Meilina Fauziyyah

NPM : 140810180012

### Tugas 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- $\Theta$ !



**Jawab :**

```
/*
Nama      : Hana Meilina Fauziyyah
NPM       : 140810180012
Kelas    : B
Program   : Depth First Search
*****/

#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;
    }
};
```

Nama : Hana Meilina Fauziyyah

NPM : 140810180012

### Tugas 6

```
        adj = new list<int>[N];
    }

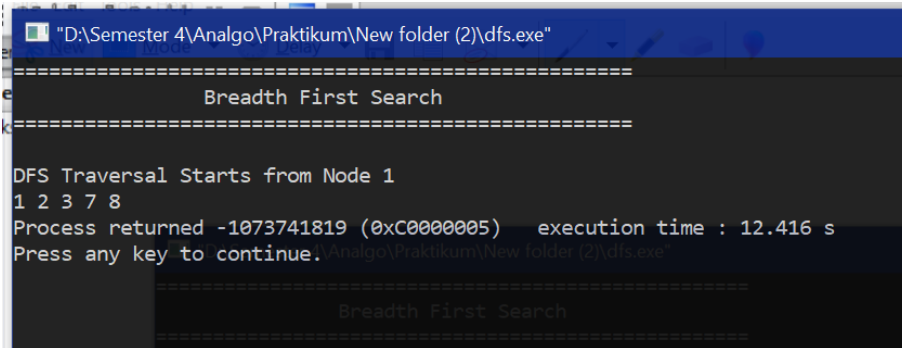
    void addEdge(int u, int v){
        adj[u].push_back(v);
    }

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

main() {
    cout<<"===== "<<endl;
    cout<<"          Breadth First Search"<<endl;
    cout<<"===== "<<endl;
    cout<<endl;
    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 << "DFS Traversal Starts from Node 1" << endl;
    g.DFS(1);
}
```



```
"D:\Semester 4\Analgo\Praktikum\New folder (2)\dfs.exe"
=====
Breadth First Search
=====

DFS Traversal Starts from Node 1
1 2 3 7 8
Process returned -1073741819 (0xC0000005)   execution time : 12.416 s
Press any key to continue.
```

Kompleksitas Waktu Asimptotik :

V : Jumlah vertex

E: Jumlah Edge

- Menandai vertex awal telah dikunjungi kemudian cetak :  $O(1)$

Nama : Hana Meilina Fauziyyah

NPM : 140810180012

Tugas 6

- Rekursif untuk semua vertex :  $T(E/I)$
- Tandai semua vertex yang belum dikunjungi :  $O(V)$
- Rekursif untuk mencetak DFS :  $T(V/I)$

Maka :

$$\begin{aligned}T(n) &= O(1) + T(E/1) + O(V) + T(V/1) \\&= O(1) + O(E) + O(V) + O(V) \\&= O(\max(1, E)) + O(V) + O(V) \\&= O(E) + O(V) + O(V) \\&= O(\max(V, E)) + O(E) \\&= O(V) + O(E) \\&= O(V+E)\end{aligned}$$