LAPORAN PRAKTIKUM

ANALISIS ALGORITMA

Disusun oleh

Muhammad Reza Atthariq kori

140810180060

PROGRAM STUDI S-1 TEKNIK INFORMATIKA

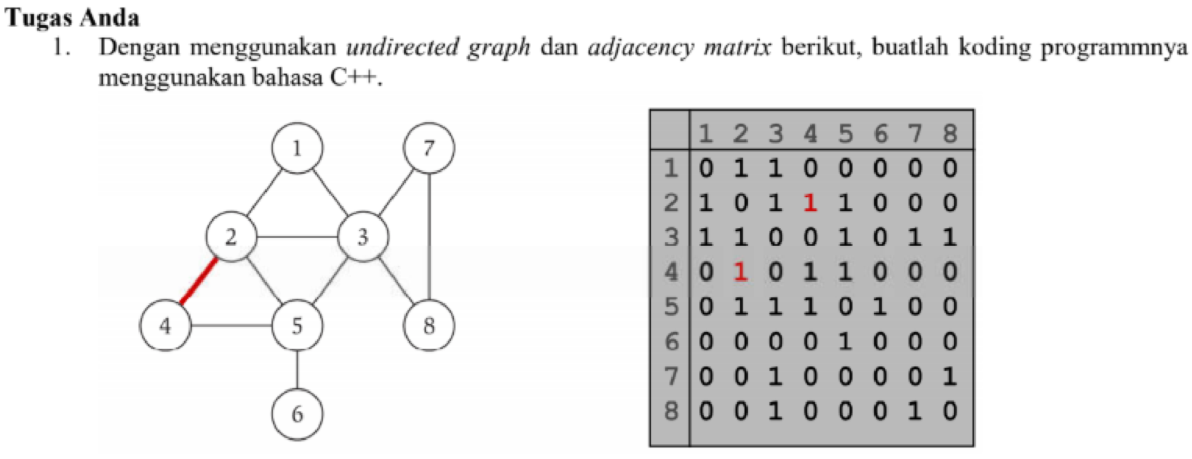
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

SUMEDANG

2020

M Reza Atthariq Kori
140810180060
Tugas 6



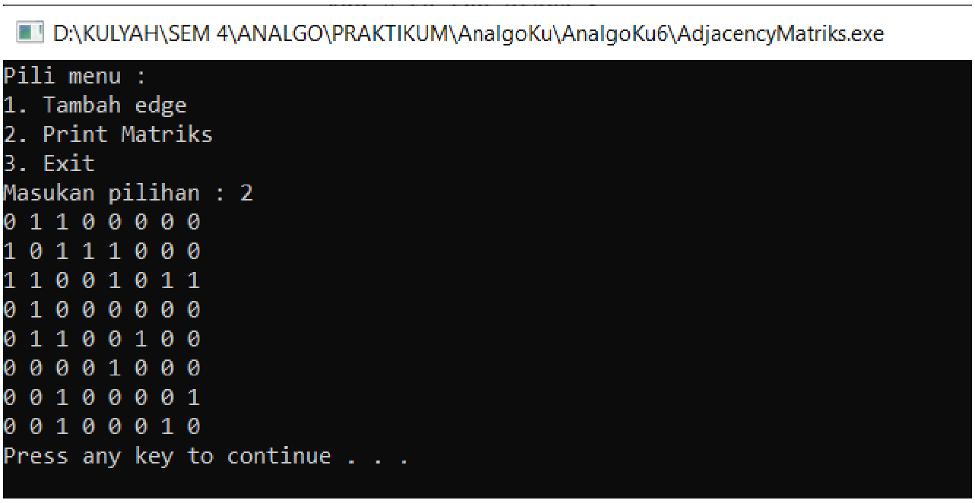
Tugas Anda

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 :

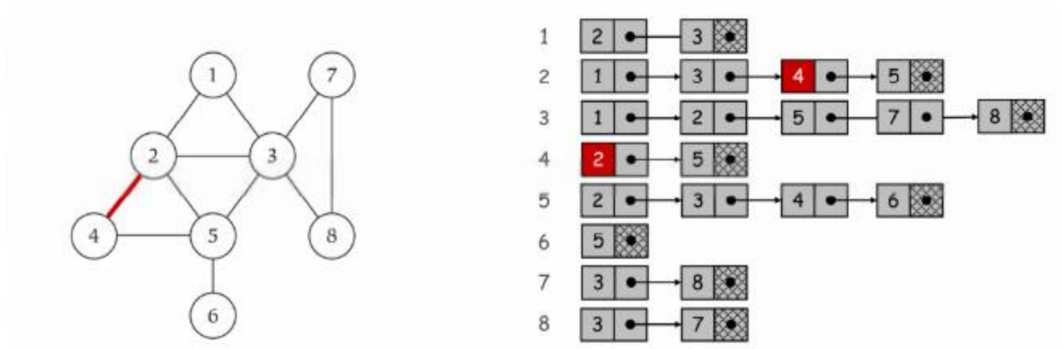
```
/*
Nama            : Muhammad Reza Atthariq Kori
NPM             : 140810180060
Kelas           : B
Program         : Program Representasi Adjacency Matriks
*/

#include <iostream>
using namespace std;

int vertArr[20][20];
int count = 0;

void printMatrix(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;
}

int main(int argc, char *argv[]){
    int v;
    cout << "Masukkan jumlah matrix : "; cin >> v;
    int pilihan,a,b;
    while(true){
        cout << "Pilih menu : " << endl;
        cout << "1. Tambah edge " << endl;
        cout << "2. Print Matriks" << endl;
        cout << "3. Exit " << endl;
        cout << "Masukan pilihan : "; cin >> pilihan;
        switch (pilihan){
            case 1:
                cout << "Masukkan node pertama  : "; cin >> a;
                cout << "Masukkan node kedua     : "; cin >> b;
                add_edge(a,b);
                cout << "Edge telah ditambahkan\n";
                system("Pause");
                system("cls");
                break;
            case 2:
                printMatrix(v);
                system("Pause");
                system("cls");
                break;
            case 3:
                return 0;
                break;
            default:
                break;
        }
    }
}
```


D:\KULYAH\SEM 4\ANALGO\PRAKTIKUM\AnalgoKu\AnalgoKu6\AdjacencyMatriks.exe

```
Pili menu :
1. Tambah edge
2. Print Matriks
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 0 0 0 0 0
0 1 1 0 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 . . .
```

M Reza Atthariq Kori
140810180060
Tugas 6

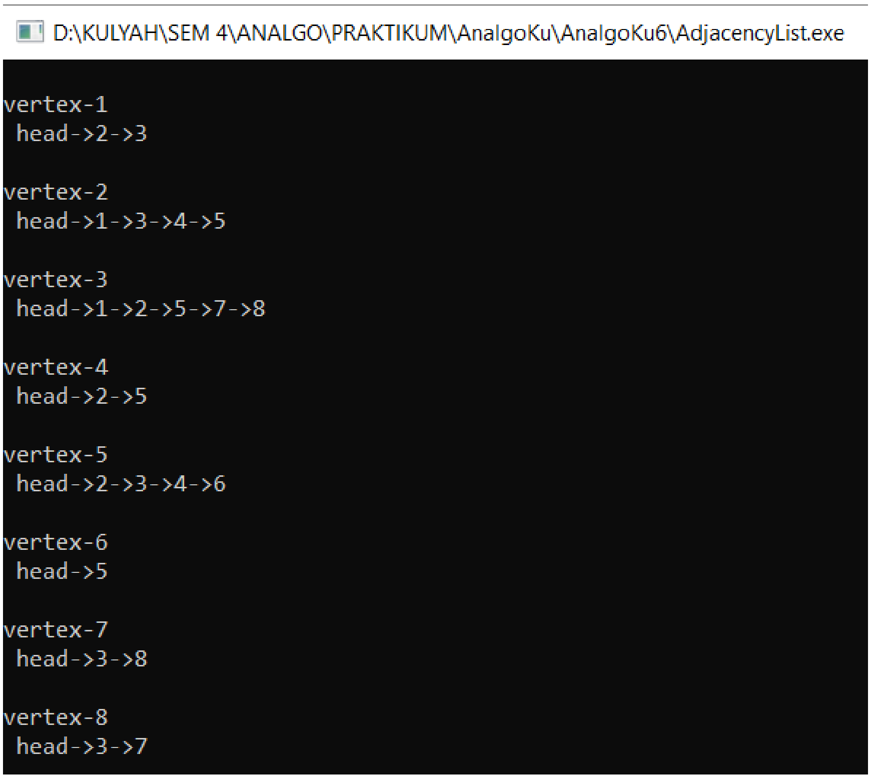
2. Dengan menggunakan *undirected graph* dan representasi *adjacency list*, buatlah koding programmnya menggunakan bahasa C++.



Jawab :

```
/*
Nama	: Muhammad Reza Atthariq Kori
NPM	: 140810180060
Kelas	: B
Program		: Program Representasi Adjacency List
*/

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

//Adjacency List Node
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];
```

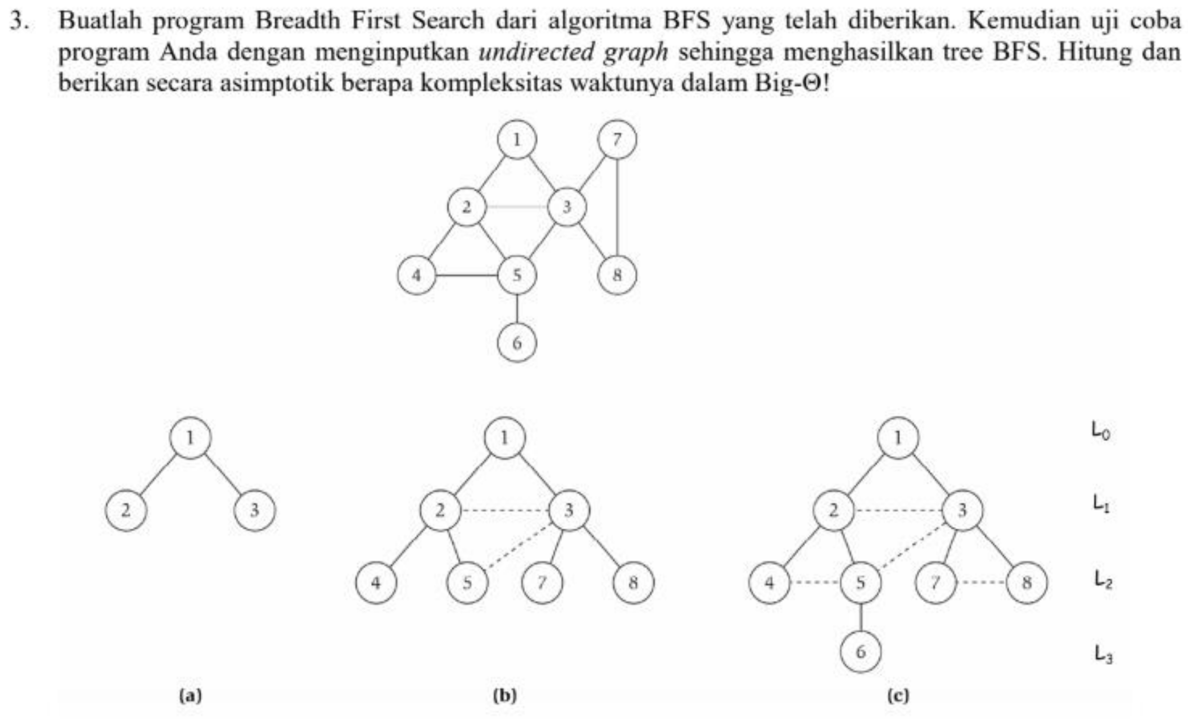
```
            for (int i = 1; i <= V; ++i)
                array[i].head = NULL;
        }

        //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 << "\nvertex-" << 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();
}
```



D:\KULYAH\SEM 4\ANALGO\PRAKTIKUM\AnalgoKu\AnalgoKu6\AdjacencyList.exe

```

vertex-1
 head->2->3

vertex-2
 head->1->3->4->5

vertex-3
 head->1->2->5->7->8

vertex-4
 head->2->5

vertex-5
 head->2->3->4->6

vertex-6
 head->5

vertex-7
 head->3->8

vertex-8
 head->3->7
```

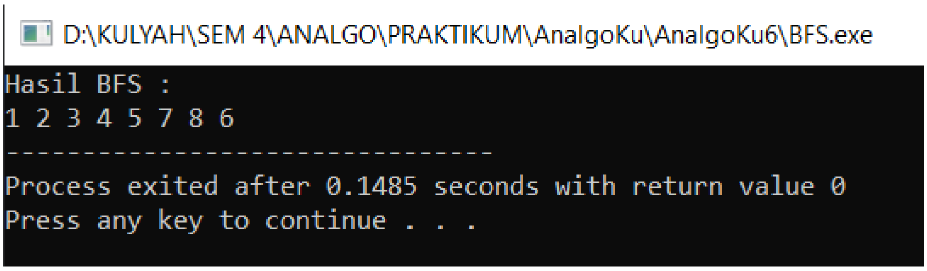
M Reza Atthariq Kori
140810180060
Tugas 6



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-Θ!

Jawab :

```
/*
Nama	: Muhammad Reza Atthariq Kori
NPM	: 140810180060
Kelas	: B
Program		: 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];
	//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<<"Hasil BFS : "<<endl;
	for(int i = 0; i < vertexSize; i++){
		cout<<output[i]<<" ";
	}
}
```



D:\KULYAH\SEM 4\ANALGO\PRAKTIKUM\AnalgoKu\AnalgoKu6\BFS.exe

```
Hasil BFS :
1 2 3 4 5 7 8 6
--------------------------------
Process exited after 0.1485 seconds with return value 0
Press any key to continue . . .
```

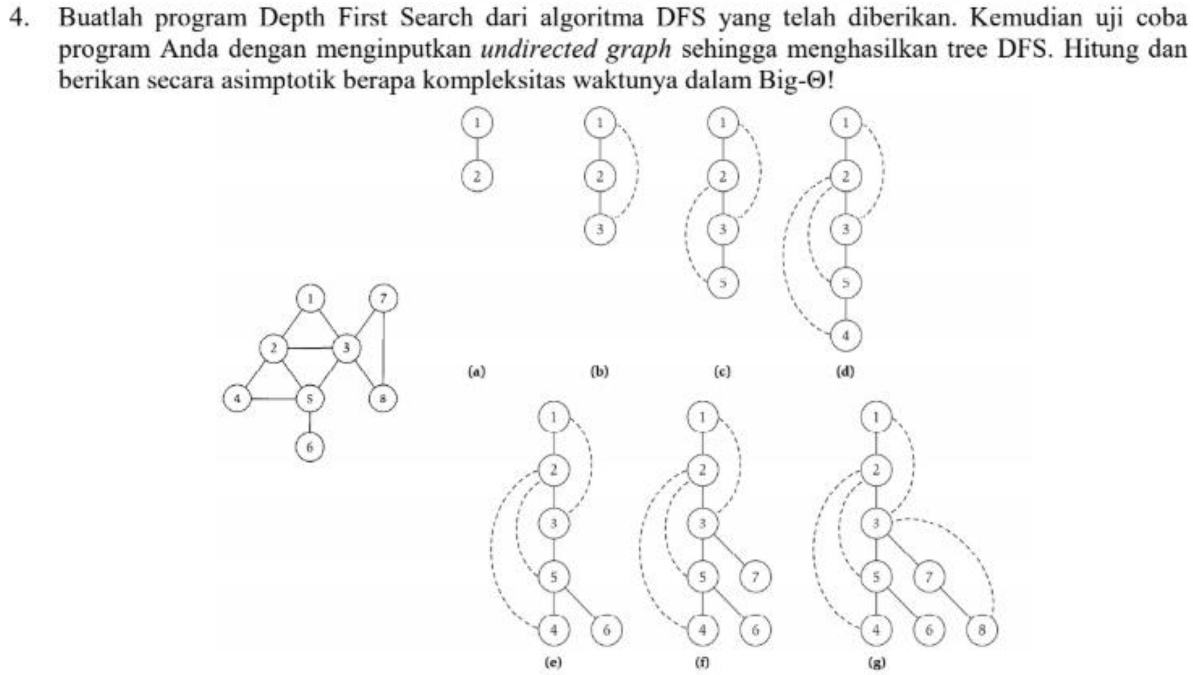
Kompleksitas waktu dari BFS adalah big O(|V| + |E|).

V = vertex
E = jumlah edges

Maka, Big-O = O(n) dimana n = V+E.

Big-Θ nya adalah Θ(n).

M Reza Atthariq Kori
140810180060
Tugas 6

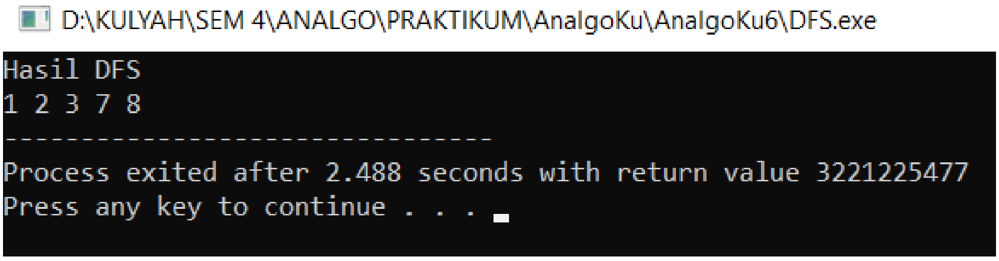


4. Buatlah program Depth First Search dari algoritma DFS yang telah diberikan. Kemudian uji coba program Anda dengan menginputkan *undirected graph* sehingga menghasilkan tree DFS. Hitung dan berikan secara asimptotik berapa kompleksitas waktunya dalam Big-Θ!

Jawab :

```
/*
Nama            : Muhammad Reza Atthariq Kori
NPM             : 140810180060
Kelas           : B
Program         : 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;
                    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 << "Hasil DFS" << endl;
          g.DFS(1);

          return 0;
}
```



```
Hasil DFS
1 2 3 7 8
--------------------------------
Process exited after 2.488 seconds with return value 3221225477
Press any key to continue . . .
```

Mengatur/mendapatkan label vertex/Edge membutuhkan waktu O(1) waktu
DFS berjalan dalam waktu O(n + m) asalkan grafik diwakili oleh struktur daftar adjacency