LAPORAN PRAKTIKUM 6

ANALISIS ALGORITMA

Disusun oleh :

Muhammad Zulfikar Ali

140810180064

PROGRAM STUDI S-1 TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

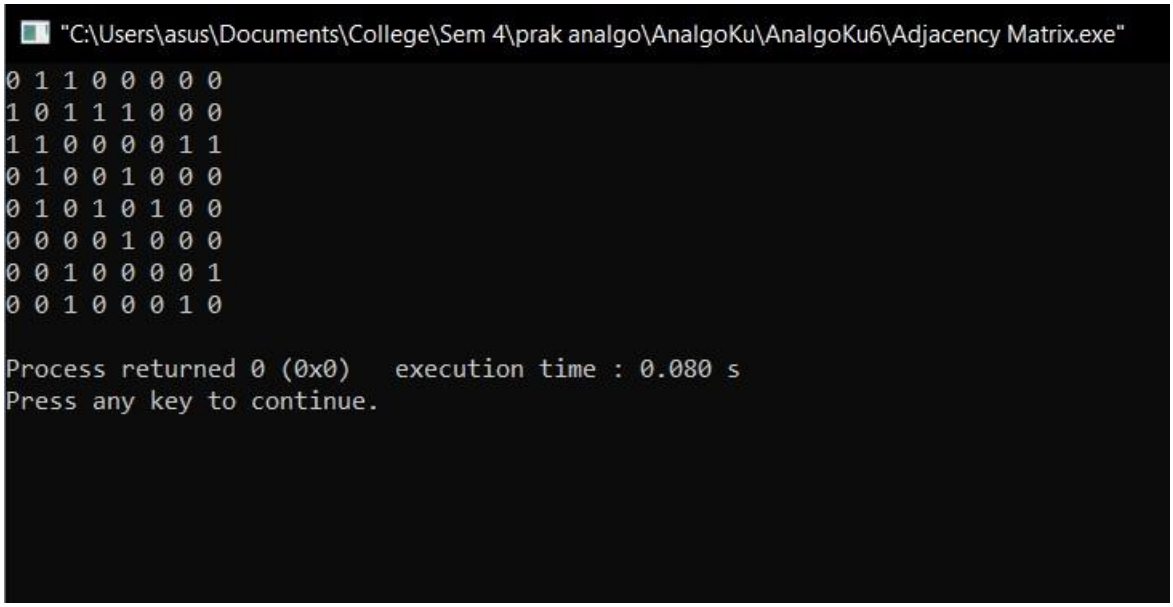
UNIVERSITAS PADJADJARAN

2020

1.

```
/*
Nama    : Muhammad Zulfikar Ali
NPM     : 140810180064
Kelas   : B
*/

#include<iostream>
using namespace std;

int vertArr[20][20];
int count = 1;

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;
}
main(int argc, char* argv[]) {
   int v = 9;
   add_edge(1, 2);
   add_edge(1, 3);
   add_edge(2, 4);
   add_edge(2, 5);
   add_edge(3, 2);
   add_edge(3, 8);
```

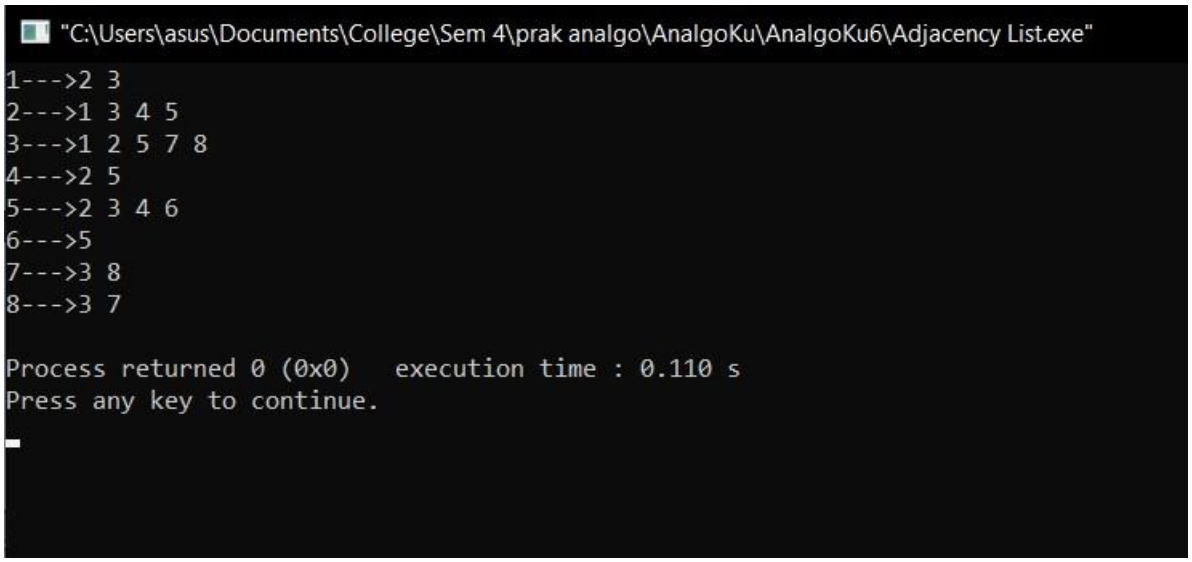
```
   add_edge(4, 5);
   add_edge(5, 6);
   add_edge(7, 3);
   add_edge(8, 7);
   displayMatrix(v);
}
```



2.

```
/*
Nama    : Muhammad Zulfikar Ali
NPM     : 140810180064
Kelas   : B
*/

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

void displayAdjList(list<int> adj_list[], int v) {
   for(int i = 1; i<v; i++) {
      cout << i << "--->";
      list<int> :: iterator it;
      for(it = adj_list[i].begin(); it != adj_list[i].end(); ++it) {
         cout << *it << " ";
```

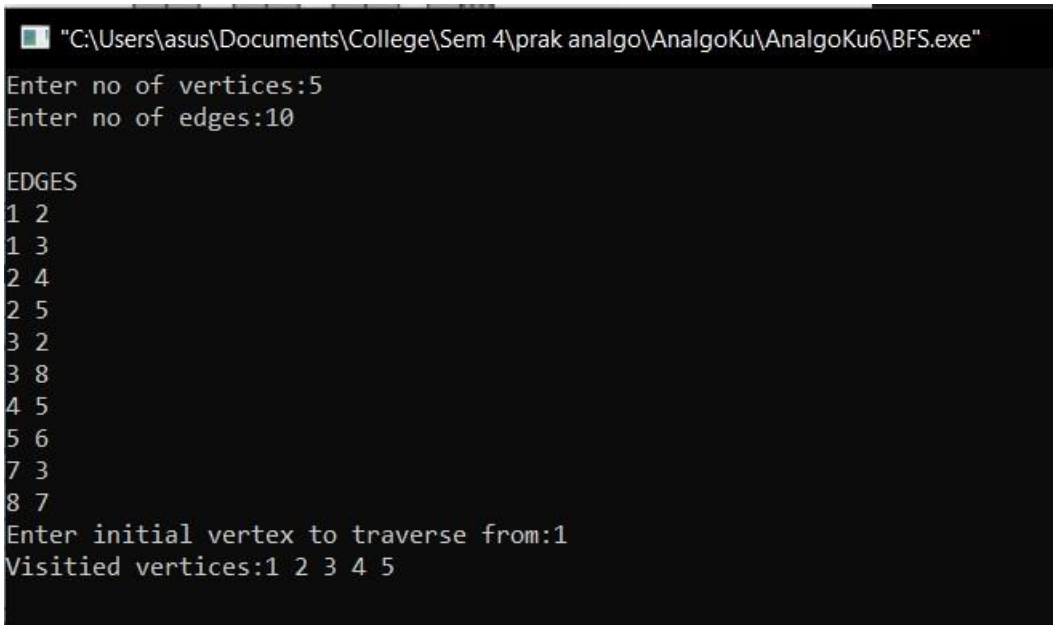
```
      }
      cout << endl;
   }
}

void add_edge(list<int> adj_list[], int u, int v) {
   adj_list[u].push_back(v);
   adj_list[v].push_back(u);
}

main(int argc, char* argv[]) {
   int v = 9;
   list<int> adj_list[v];
   add_edge(adj_list, 1, 2);
   add_edge(adj_list, 1, 3);
   add_edge(adj_list, 2, 3);
   add_edge(adj_list, 2, 4);
   add_edge(adj_list, 2, 5);
   add_edge(adj_list, 3, 5);
   add_edge(adj_list, 3, 7);
   add_edge(adj_list, 3, 8);
   add_edge(adj_list, 4, 5);
   add_edge(adj_list, 5, 6);
   add_edge(adj_list, 7, 8);
   displayAdjList(adj_list, v);
}
```



3.

```
/*
Nama    : Muhammad Zulfikar Ali
NPM     : 140810180064
Kelas   : B
*/

#include<iostream>
#include<conio.h>
#include<stdlib.h>
using namespace std;

int cost[10][10],i,j,k,n,qu[10],front,rare,v,visit[10],visited[10];
int main()
{
    int m;
    //clrscr();
    cout <<"Enter no of vertices:";
    cin >> n;
    cout <<"Enter no of edges:";
    cin >> m;
    cout <<"\nEDGES \n";
    for(k=1; k<=m; k++)
    {
        cin >>i>>j;
        cost[i][j]=1;
    }
    cout <<"Enter initial vertex to traverse from:";
    cin >>v;
    cout <<"Visitied vertices:";
    cout <<v<<" ";
    visited[v]=1;
    k=1;
    while(k<n)
    {
        for(j=1; j<=n; j++)
            if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1)
            {
                visit[j]=1;
                qu[rare++]=j;
            }
```

```
        v=qu[front++];
        cout<<v <<" ";
        k++;
        visit[v]=0;
        visited[v]=1;
    }
    getch();
    return 0;
}
```



Kompleksitas waktu asimptotik:

V: Vertex E: Edges

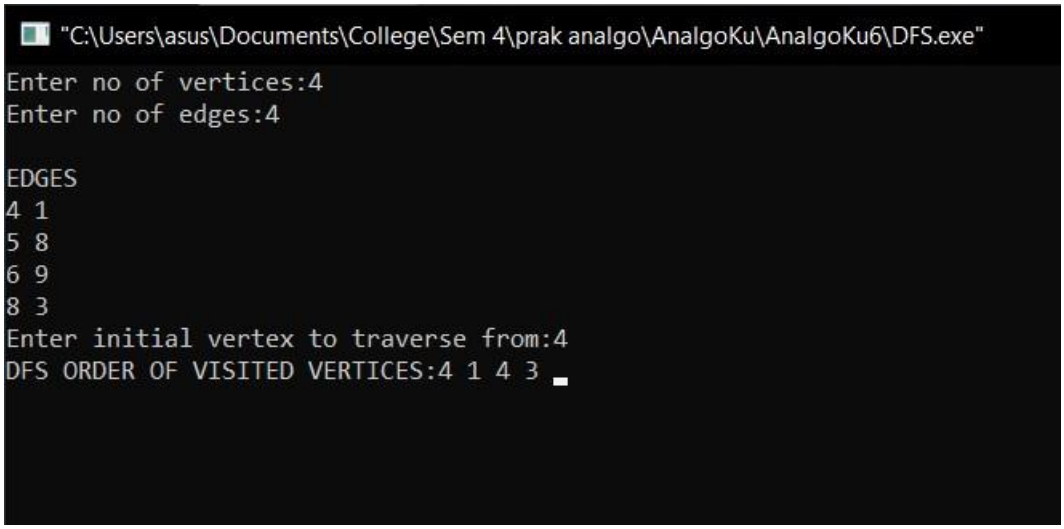
- Menandai setiap vertex yang belum dikunjungi: O(V)
- Menandai vertex awal yang telah dikunjungi dan masukkan ke queue: O(1)
- Keluarkan vertex dari queue lalu cetak: O(V)
- Kunjungi setiap vertex yang belum dikunjungi lalu masukkan ke queue: O(E)

➔ T(n) = O(V) + O(1) + O(V) + O(E)
= O(max(V,1)) + O(V) + O(E)
= O(V) + O(V) + O(E)
= O(max(V,V)) + O(E)
= O(V) + O(E)
= O(V+E)

4.

```
/*
Nama    : Muhammad Zulfikar Ali
NPM     : 140810180064
Kelas   : B
*/

#include<iostream>
#include<conio.h>
#include<stdlib.h>
using namespace std;

int cost[10][10],i,j,k,n,stk[10],top,v,visit[10],visited[10];
int main()
{
    int m;
    //clrscr();
    cout <<"Enter no of vertices:";
    cin >> n;
    cout <<"Enter no of edges:";
    cin >> m;
    cout <<"\nEDGES \n";
    for(k=1; k<=m; k++)
    {
        cin >>i>>j;
        cost[i][j]=1;
    }
    cout <<"Enter initial vertex to traverse from:";
    cin >>v;
    cout <<"DFS ORDER OF VISITED VERTICES:";
    cout << v <<" ";
    visited[v]=1;
    k=1;
    while(k<n)
    {
        for(j=n; j>=1; j--)
            if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1)
            {
                visit[j]=1;
                stk[top]=j;
                top++;
```

```
            }
        v=stk[--top];
        cout<<v << " ";
        k++;
        visit[v]=0;
        visited[v]=1;
    }
    getch();
    return 0;
}
```



Kompleksitas waktu asimptotik:

V: Vertex E: Edges

- Menandai vertex awal yang telah dikunjungi lalu cetak: O(1)
- Rekursif untuk semua vertex: T(E/1)
- Tandai semua vertex yang belum dikunjungi: O(V)
- Rekursif untuk mencetak DFS: T(V/1)

➔ 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,V)) + O(E)

= O(V) + O(E)

= O(V+E)