LAPORAN PRAKTIKUM 6

ANALISIS ALGORITMA

Disusun oleh :

Nurul Ma'arif

140810180040

PROGRAM STUDI S-1 TEKNIK INFORMATIKA

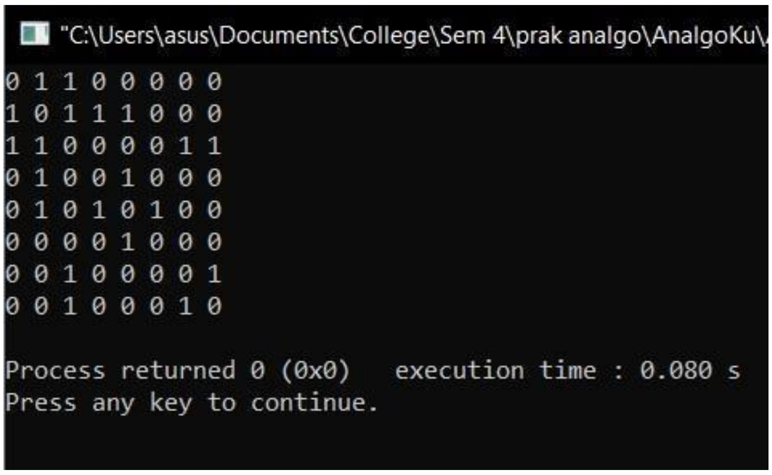
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

2020

1.

```
/*
Nama    : Nurul Ma'arif
NPM     : 140810180040
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);
}
```

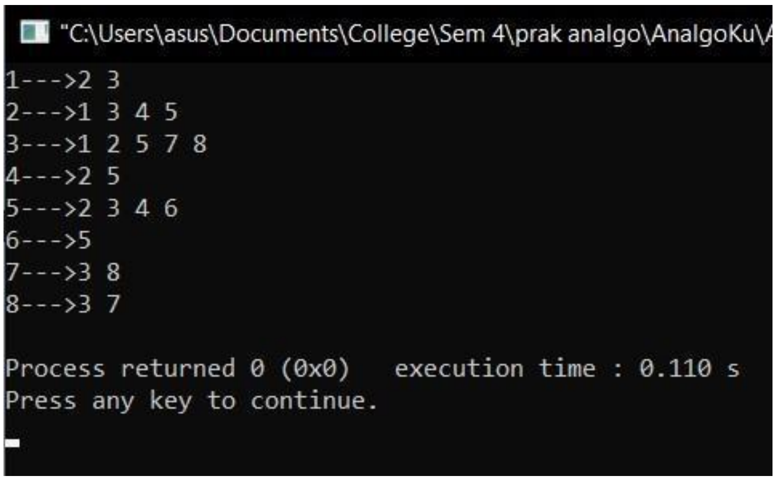


```
"C:\Users\asus\Documents\College\Sem 4\prak analgo\AnalgoKu\
0 1 1 0 0 0 0 0
1 0 1 1 1 0 0 0
1 1 0 0 0 0 1 1
0 1 0 0 1 0 0 0
0 1 0 1 0 1 0 0
0 0 0 0 1 0 0 0
0 0 1 0 0 0 0 1
0 0 1 0 0 0 1 0

Process returned 0 (0x0)   execution time : 0.080 s
Press any key to continue.
```

2.

```
/*
Nama    : Nurul Ma'arif
NPM     : 140810180040
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 << " ";
```

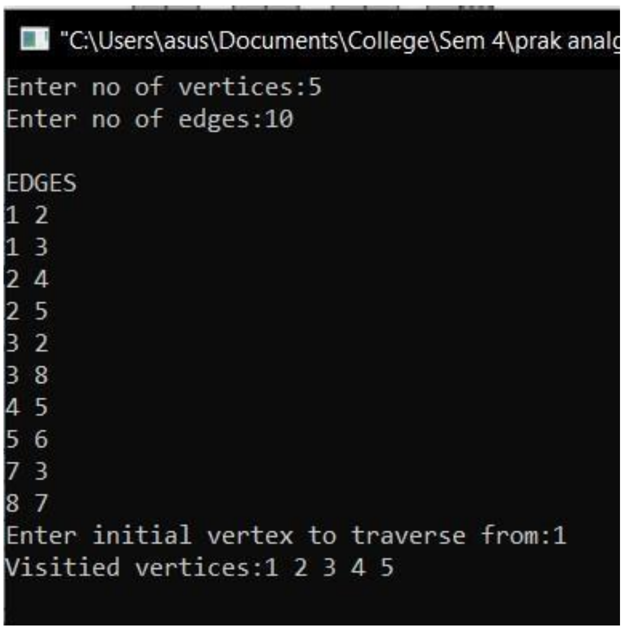
```
        }
        cout << endl;
    }
}

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);
}
```



```
"C:\Users\asus\Documents\College\Sem 4\prak analgo\AnalgoKu\
1--->2 3
2--->1 3 4 5
3--->1 2 5 7 8
4--->2 5
5--->2 3 4 6
6--->5
7--->3 8
8--->3 7

Process returned 0 (0x0)   execution time : 0.110 s
Press any key to continue.
```

3.

```
/*
Nama    : Nurul Ma'arif
NPM     : 140810180040
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;
}
```



```
"C:\Users\asus\Documents\College\Sem 4\prak analg
Enter no of vertices:5
Enter no of edges:10

EDGES
1 2
1 3
2 4
2 5
3 2
3 8
4 5
5 6
7 3
8 7
Enter initial vertex to traverse from:1
Visitied vertices:1 2 3 4 5
```

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)

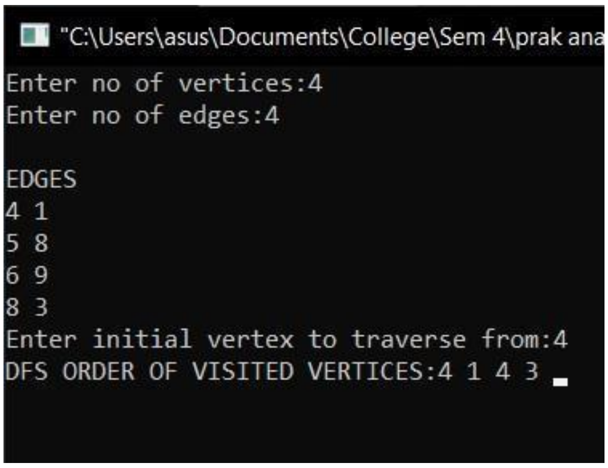
= O(max(V,V)) + O(E)

= O(V) + O(E)

= O(V+E)

4.

```
/*
Nama    : Nurul Ma’arif
NPM     : 140810180040
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;
}
```



```
"C:\Users\asus\Documents\College\Sem 4\prak ana
Enter no of vertices:4
Enter no of edges:4

EDGES
4 1
5 8
6 9
8 3
Enter initial vertex to traverse from:4
DFS ORDER OF VISITED VERTICES:4 1 4 3 _
```

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)