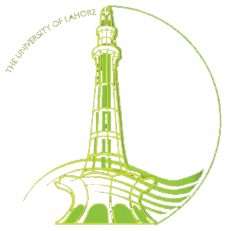

THE UNIVERSITY OF LAHORE
ISLAMABAD CAMPUS

Data Structure And Algorithum

Lab Report

Name:	Ch Hasnain Zafar
Registration #:	SEU-F16-104
Lab Report #:	08
Dated:	28-06-2018
Submitted To:	Mr. Usman Ahmed

The University of Lahore, Islamabad Campus
Department of Computer Science & Information Technology

Experiment # 1
Undirected Graph and its Implementation

Objective
To understand and implement Undirected Graph and its Implementation Problem.

Software Tool
1. DEV C++

1 Theory

Undirected graphs representation
There are several possible ways to represent a graph inside the computer. We will discuss two of them: adjacency matrix and adjacency list.
Adjacency matrix
Each cell aij of an adjacency matrix contains 0, if there is an edge between i-th and j-th vertices, and 1 otherwise The graph presented by example is undirected. It means that its adjacency matrix is symmetric. Indeed, in undirected graph, if there is an edge (2, 5) then there is also an edge (5, 2). This is also the reason, why there are two cells for every edge in the sample. Loops, if they are allowed in a graph, correspond to the diagonal elements of an adjacency matrix.

Advantages. Adjacency matrix is very convenient to work with. Add (remove) an edge can be done in O(1) time, the same time is required to check, if there is an edge between two vertices. Also it is very simple to program and in all our graph tutorials we are going to work with this kind of representation.

2 Task

2.1 Procedure: Task 1

```
#include<iostream>
#include<iomanip>

using namespace std ;
void PrintMat(int mat[][20] , int n,int weight[][20])
{
int i, j;
cout<<   \n\n  <<setw(4)<<       ; for(i = 0; i < n; i++)
cout<<setw(3)<<   (   <<i+1<<   )   ; cout<<   \n\n   ;
for(i = 0; i < n; i++) {
cout<<setw(3)<<   (   <<i+1<<   )   ; for(j = 0; j < n; j++)
{
cout<<setw(4)<<mat[ i ][ j]<<setw(3)<<    cout<<   \n\n   ;
2
weight is=
     } }
}
void PrintMat(int mat[][20] , int n) {
int i, j;
cout<<   \n\n  <<setw(4)<<       ; for(i = 0; i < n; i++)
cout<<setw(3)<<   (   <<i+1<<   )   ; cout<<   \n\n   ;
for(i = 0; i < n; i++) {
} }
int main() {
char int int
}
ch ; choice ;
mat[20][20] ,
weight [20][20];
cout<<setw(3)<<   (   <<i+1<<   )   ; for(j = 0; j < n; j++)
{
cout<<setw(4)<<mat[ i ][ j ]; cout<<   \n\n   ;
do{
cout<<  1.undirected graph <<endl; cout<<  2.directed graph <<endl;
switch( choice ){
```

```
case 1:{
int i, j, v;
     cout<< Enter the cin>>v;
int mat[20][20] ,
number
of vertexes :
    ;
     weight [20][20];
3

cout<<   \n   ;
for(i = 0; i < v; i++) {
for(j = i; j < v; j++) {
i f ( i != j ){
cout<< Enter 1 if the vertex    <<i+1<<    is c i n >> m a t [ i ] [ j ]
       } }
PrintMat(mat, v);} break ;
case 2:
{
}
else
mat[j][i] = mat[i][j];
mat[i][j] = 0;
int i, j, v; cout<< Enter the number of vertexes:    ;
cin>>v;
int mat[20][20];
cout<<   \n   ;
for(i = 0; i < v; i++) {
for(j = 0; j < v; j++) {
     cout<< Enter 1 if the vertex    <<i+1<<    is c i n >> m a t [ i ]
       } }
PrintMat(mat, v);} break ;
case 3:
4
a
d

{{
int i, j, v;
cout<< Enter the number of cin>>v;
```

```
int mat[20][20];
cout<<   \n   ;
for(i = 0; i < v; i++) {
vertexes :
    ;
     for(j = 0; j < v; j++) {
cout<< Enter 1 if the vertex    <<i+1<<    is c i n >> m a t [ i ] [ j ]
i f ( mat [ i ] [ j ]==1){
       } }
PrintMat(mat, v,weight);} break ;
default :
c o u t <<    i n v a l i d    << e n d l ;
}
}
cout<<  d o  you want to continue Y/N=   ; cin>>ch;
} w h i l e ( ( c h==    Y    ) | | ( c h==    y    ) ) ; }
```