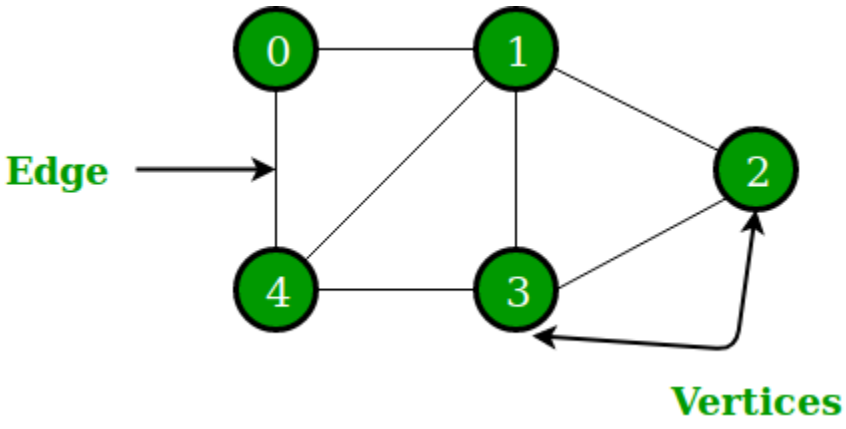
The School Electrical Engineering and Information Technology
Computer Science Department

CS223 Lab 8
Graph

Definition:

A graph is a non-linear data structure that enables representing relationships between different types of data.

There are two main parts of a graph G (V, E):
- The vertices (nodes) where the data is stored (V).
- The edges (connections) which connect the nodes (E).



A structure, which contains a vertex and a pointer to the connected vertex, is created as follows:

```
struct node{
   int vertex;
   node *next;
};
```

Main Operations

- **Graph Traversal**
  - Breadth-First-Search (BFS)
  - Depth-First-Search (DFS)
- **Minimum Spanning Tree:**
  - Kruskal
  - Prim

Lab Assignment

Consider the following C++ code, which contains the following :

- Define a structure node
- All Queue implementation needed to implement BFS
- Graph creation function using adjacency list
- Main function

```
#include <iostream>

#define MAX_NODE 50

using namespace std;

struct node{

   int vertex;

   node *next;

};

node *adj[MAX_NODE]; //For storing Adjacency list of nodes.

int totNodes; //No. of Nodes in Graph.

////////////Queue Operation\\\\\\\\\\\\\\\\\\\\\\\\

int Qarray[MAX_NODE];

int  front = - 1, rear = - 1;

int isfull(){
```

```
   if (front == ((rear + 1)%MAX_NODE ))
      return 1;

   return 0;
}
int isempty(){
   if (front == -1)
   {
      return 1;
   }
  return 0;
}
void enqueue(int value) {
    rear=  (rear + 1) % MAX_NODE ;
     Qarray[rear] = value;
  if (front == - 1)
     front = 0;
}

int dequeue() {
   int removed;
      removed = Qarray[front];
      // if there is one value
      if (front == rear)
         front = rear = -1;
      else
     front = (front + 1) % MAX_NODE ;
```

```
   return removed;
 }
void createGraph(){
   node *newl,*last;
   int neighbours,nv;
   cout<<"\n\n---Graph Creation---\n\n";
   cout<<"Enter total nodes in graph : \n";
   cin>>totNodes;
   for(int i=1;i<=totNodes;i++){
      last=NULL;
      cout<<"\\nEnter no. of nodes in the adjacency list of node \"<<i<<\"\n";
      cout<<"--> That is Total Neighbours of "<<i<<" : ";
      cin>>neighbours;
      for(int j=1;j<=neighbours;j++){
         cout<<"Enter neighbour #"<<j<<" : ";
         cin>>nv;
         newl=new node;
         newl->vertex=nv;
         newl->next=NULL;
         if(adj[i]==NULL)
            adj[i]=last=newl;
         else{
            last->next = newl;
            last = newl;
         }
      }
   }
}
```

```
/////// Breadth First Search
void BFS_traversal(){
   bool *visited = new bool[totNodes];
   int start_node;
   for(int i = 0; i < totNodes; i++)
      visited[i] = false;
  cout<<"Enter starting node : ";
   cin>>start_node;
   // Mark the current node as visited and enqueue it
   visited[start_node] = true;
   enqueue(start_node);
   while(!isempty())
   {
      int N,v;
       node *tmp;
      // Dequeue a vertex from queue and print it
       N= dequeue();
     cout << N<< " ";
      // Get all adjacent vertices of the dequeued
      // vertex s. If a adjacent has not been visited,
      // then mark it visited and enqueue it
      tmp = adj[N];  //for status updation.
      while(tmp!=NULL){
         v = tmp->vertex;
         if(!visited[v]){//check status of N\'s neighbour.
            visited[v] = true;
            enqueue(v); //insert N\'s neighbour who are in ready state.

         }
```

```
        tmp=tmp->next;
      }


   }


}int main(){
   cout<<"*****Breadth First Search Traversal*****\\n";
   createGraph();
   cout<<"\\n===BFS traversal is as under===\\n";
   BFS_traversal();


}
```