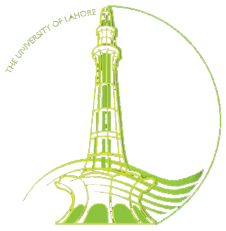

THE UNIVERSITY OF LAHORE
ISLAMABAD CAMPUS

DATA STRUCTURE AND ALOGRITHUM

Lab Report

Name:	SAMI ULLAH
Registration #:	SEU-S17-030
Lab Report #:	10
Dated:	6-29-2018
Submitted To:	Mr. Usman Ahmed

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

Experiment # 1
GRAPH

Objective
To understand the Implementation of BREADTH FIRST SEARCH

Software Tool
1.
DEV C++

1 Theory

Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key'[1]), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level.

It uses the opposite strategy as depth-first search, which instead explores the highest-depth nodes first before being forced to backtrack and expand shallower nodes.

2 Task

2.1 Procedure: Task 1

```
#include<iostream>
#include <list>
 using namespace std;
 class Graph
{
    int V;
    list<int> *adj;
public:
    Graph(int V);
```

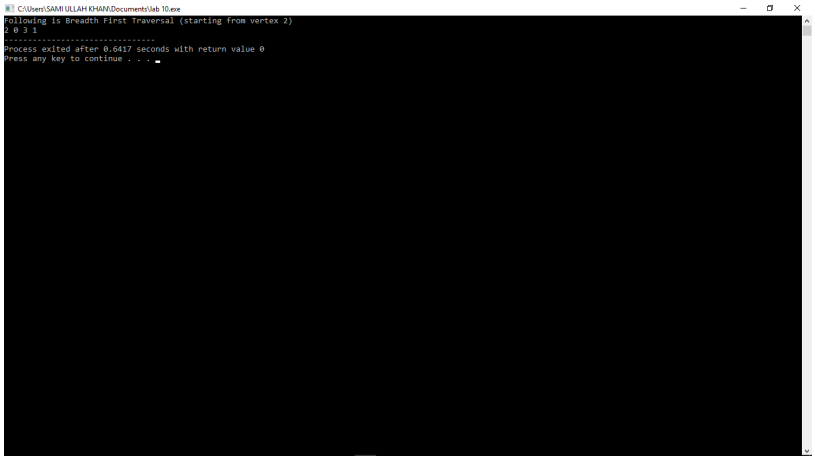

Figure 1: Time Independent Feature Set

```
    void addEdge(int v, int w);
    void BFS(int s);
};
 Graph::Graph(int V)
{
    this->V = V;
    adj = new list<int>[V];
}
 void Graph::addEdge(int v, int w)
{
    adj[v].push_back(w);
}
 void Graph::BFS(int s)
{   bool *visited = new bool[V];
    for(int i = 0; i < V; i++)
        visited[i] = false;
    list<int> queue;
  visited[s] = true;
    queue.push_back(s);
   list<int>::iterator i;
     while(!queue.empty())
    {       s = queue.front();
        cout << s << " ";
        queue.pop_front();
```

```
        for (i = adj[s].begin(); i != adj[s].end(); ++i)
          {
              if (!visited[*i])
              {
                  visited[*i] = true;
                  queue.push_back(*i);
              }
          }
    }
}
int main()
{ Graph g(4);
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 2);
    g.addEdge(2, 0);
    g.addEdge(2, 3);
    g.addEdge(3, 3);
  cout << "Following is Breadth First Traversal "
          << "(starting from vertex 2) \n";
    g.BFS(2);

    return 0;
}
}
```

3 Conclusion

in this lab we Understand

To implement and understand blind searching techniques such as Breadth First Search.

Understand the searching in BFS

Program BFS Technique