Graphs

# Due Date

Friday, April 27th 2018

# Description

In this project you will design a data structure to represent a directed graph as an adjacency list. Additionally you will program a breadth-first search (BFS) traversals over the graph. BFS will require you to use a queue.

# Requirements

1. Correctly implement the graph data structure. Support the following operations.
   1. Load a graph from file using a simple representation.
   2. Create the graph with a fixed number of vertices.
   3. Add an edge to the graph.
   4. For a vertex, calculate all adjacent vertices.
2. Perform a breadth-first search of the graph.
   1. Given a starting vertex, place it in the to-visit queue.
   2. Remove a vertex from the queue and print that vertex’s information. Place all vertices adjacent to the current vertex in the back of the queue.
   3. Continue removing and visiting node from the queue until it is empty.
   4. Do not add duplicate nodes to the queue. You may want to keep a binary search tree to determine which nodes you have already visited.

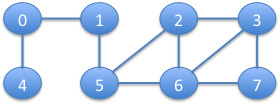
**File format**: The first number in the file is the number of vertices. Next a list of edges provides the graph.

<number of vertices>

<vertex1>[TAB]<vertex2>

<vertex1>[TAB]<vertex2>

**Example Graph**



**Example Graph File**:

8

0 1

0 4

1 5

5 2

5 6

2 3

2 6

6 3

6 7

7 3

# Rubric

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Type** | **Description** | **Value** | **Received** |
| **Compiles** | **Compiles** | **10** |  |
| **Comment** | **Useful comments in code** | **10** |  |
| **Correctness** | **Graph correctly implemented** | **20** |  |
| **Correctness** | **Graph Correctly loaded from file** | **20** |  |
| **Correctness** | **Performs Breadth-first search correctly** | **20** |  |
| **Correctness** | **Meaningful program output** | **20** |  |
|  | **Total** | **100** |  |