CS 3358 Assignment 5

Due: 11:55pm Wednesday, May 1, 2019

Instructor: Kecheng Yang (yangk@txstate.edu)
Instructional Assistant: Cody Blakeney (cjb92@txstate.edu)

In this assignment, you are asked to implement several functions in a `Graph` class, which uses an adjacent matrix to represent a directed graph, in `graph.cpp`.

1. Implement the public function `BFTraversal()`, which outputs a breadth-first traversal of the graph. To do so, you need to implement a private function `BFS()`, which performs breadth-first search from a given vertex. In `BFTraversal()`, you may need to call `BFS()` multiple times (in loops) in order to perform breadth-first search in sub-graphs that are not connected.
   NOTE: in both `BFTraversal()` and `BFS()`, whenever you can choose from multiple un-visited vertices to proceed, please choose the one with the lowest index for better grading consistence. For example, you always start `BFTraversal()` by performing `BFS()` on vertex 0, and then other vertices if necessary.
   Hint: You will need to use a queue to implement the function `BFS()`. You may use an `int` queue to store the indices to represent vertices in queue. The C++ Standard Template Library (STL) provides the template class queue for you to use by `#include<queue>`. See http://www.cplusplus.com/reference/queue/queue/ to learn the syntax of the declaration of an `int` queue and its member functions.

2. Implement the public function `DFTraversal()`, which outputs a depth-first traversal of the graph. To do so, you need to implement a private function `DFS()`, which performs depth-first search from a given vertex. In `DFTraversal()`, you may need to call `DFS()` multiple times (in loops) in order to perform depth-first search in sub-graphs that are not connected. Furthermore, you are asked to implement `DFS()` using recursion.
   NOTE: in both `DFTraversal()` and `DFS()`, whenever you can choose from multiple un-visited vertices to proceed, please choose the one with the lowest index for better grading consistence. For example, you always start `DFTraversal()` by performing `DFS()` on vertex 0, and then other vertices if necessary.
   Hint: You are expected to implement `DFS()` using recursion. Nonetheless, a `for` loop is also expected within `DFS()`, i.e., the recursive calls are within the loop. (This is a counterexample to disprove "a recursive function must have absolutely no loop inside".)

Submission:

You should submit your work via the assignment tag in the TRACS system.
You should pack `graph.cpp` and an optional README plain text file into a single .zip file to upload to TRACS. The .zip file should be named as a5_yourNetID.zip, such as a5_zz567.zip

Sample tests:

Note that successes in getting the following test results do not guarantee the correctness of your work and therefore do not guarantee you a satisfactory grade, whereas failures in getting the following test results probably do indicate flaws in your work and you may lose points.

```
Constructing a directed graph...
Please enter the number of vertices in this graph: 6
Adding an directed edge...
Trying to add an edge (-1,-1) indicates the end of adding edges and prints
outputs.
Please enter the vertex index of the source of the edge to be added: 4
Please enter the vertex index of the target of the edge to be added: 1
Adding an directed edge...
Trying to add an edge (-1,-1) indicates the end of adding edges and prints
outputs.
Please enter the vertex index of the source of the edge to be added: 1
Please enter the vertex index of the target of the edge to be added: 2
Adding an directed edge...
Trying to add an edge (-1,-1) indicates the end of adding edges and prints
outputs.
Please enter the vertex index of the source of the edge to be added: 2
Please enter the vertex index of the target of the edge to be added: 4
Adding an directed edge...
Trying to add an edge (-1,-1) indicates the end of adding edges and prints
outputs.
Please enter the vertex index of the source of the edge to be added: 0
Please enter the vertex index of the target of the edge to be added: 3
Adding an directed edge...
Trying to add an edge (-1,-1) indicates the end of adding edges and prints
outputs.
Please enter the vertex index of the source of the edge to be added: 0
Please enter the vertex index of the target of the edge to be added: 1
Adding an directed edge...
Trying to add an edge (-1,-1) indicates the end of adding edges and prints
outputs.
Please enter the vertex index of the source of the edge to be added: 0
Please enter the vertex index of the target of the edge to be added: 4
Adding an directed edge...
Trying to add an edge (-1,-1) indicates the end of adding edges and prints
outputs.
Please enter the vertex index of the source of the edge to be added: 1
Please enter the vertex index of the target of the edge to be added: 2
This edge already exists. No edge has been added.
```

```
Adding an directed edge...
Trying to add an edge (-1,-1) indicates the end of adding edges and prints
outputs.
Please enter the vertex index of the source of the edge to be added: 1
Please enter the vertex index of the target of the edge to be added: 1
No self loop! No edge has been added.
Adding an directed edge...
Trying to add an edge (-1,-1) indicates the end of adding edges and prints
outputs.
Please enter the vertex index of the source of the edge to be added: 8
Please enter the vertex index of the target of the edge to be added: 2
Invalid vertex index! No edge has been added.
Adding an directed edge...
Trying to add an edge (-1,-1) indicates the end of adding edges and prints
outputs.
Please enter the vertex index of the source of the edge to be added: -1
Please enter the vertex index of the target of the edge to be added: -1
A directed graph has been constructed:
This directed graph has 6 vertex (vertices), indexed from 0 to 5
This directed graph has 6 edge(s):
(0,1)
(0,3)
(0,4)
(1,2)
(2,4)
(4,1)
Breadth-First Traversal: 0  1  3  4  2  5
Depth-First Traversal: 0  1  2  4  3  5
```