CISC 380 **Instructor: Dr. Sarah Miracle**

Homework 6

Student: Shijun Yang, Haoyuan Du

Purpose:

- Understand and apply graph algorithms presented in class.
- Design and analyze your own graph algorithms.
- Learn about properties of shortest path trees and spanning trees.

General Homework Policies:

- This homework assignment is due by the deadline given in Canvas. Late assignments will not be accepted and will receive a 0.
- Submit **two** files through Canvas:
  1. The `pdf` for the written portion of the assignment (it should generated by modifying the the LaTeX file for this assignment).
  2. The completed `DirectedGraph.java` and `ColoredGraph.java` files. Before submitting please zip all the java files together and submit one `.zip` file. See the *Additional Programming Instructions* section below for more detailed instructions.
- You may choose to work with a partner on this assignment. If you choose to work with a partner, only one assignment should be submitted and you and your partner will both receive the same grade. Make sure to include your partner's name on the homework.
- If you choose to work with a partner, you must add yourself and your partner to an empty group in Canvas one week before the assignment deadline (this is available from the `People` tab in Canvas).
- If you choose to work with a partner, your assignment should be a true joint effort, equally created, and understood by both partners.
- You are not allowed to consult outside sources, other than the textbook, your notes, the Java API and the references linked from Canvas (i.e., no looking for answers on the internet).[1]
- Getting *ANY* solutions from the web, previous students etc. is *NOT* allowed.[1]
- You are not allowed to discuss this assignment with anyone except for your partner (if you have one) or the instructor.[1]
- Copying code from anywhere or anyone is not allowed (even previous code you have written). Allowing someone to copy your code is also considered cheating.[1]
- Your work will be graded on correctness and clarity. Write complete and precise answers and show all of your work.
- Questions marked (PRACTICE) will not be graded and do not need to be submitted. However it's highly recommended that you complete them.

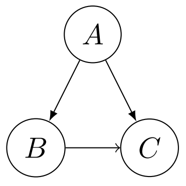
[1]*See the section of the syllabus on academic dishonesty for more details.*

Homework Problems:

1. (3 points) Consider the following algorithm for finding the shortest path from node s to node t in a directed graph with some negative edges: add a large constant to each edge weight so that all the weights become positive, then run Dijkstra's algorithm starting at node s, and return the shortest path found to node t.

   Is this a valid method (assuming the graph has no negative weight cycles- cycles whose total weight is negative)? Either explain why it works correctly, or give a counter-example (and explain clearly why your graph is a counter-example).

   Solution: No. There is a counter example as below.

   AB is -2, BC is 6, AC is 10. According to the question, we add a large constant to make sure each edge is positive. In this case, assume we add 100 for each edge. Then AB = 98, BC = 106, AC = 110. If we use Dijkstra's Algorithm, start from A, then AB will be chosen because $98 < 110$. Next, AC will be chosen because $98 + 106 > 110$. However, the correct answer should be AB+BC, since $-2 + 6 < 10$. So this algorithm is false.

2. (5 points) Write a method to determine if a directed graph is acyclic. Implement your solution by adding code to the `isAcyclic` method in the `DirectedGraph.java` file.

3. (5 points) Given is a colorful maze such as the one shown here:

   `maze.pdf`

   To pass through the maze, you start at the top left corner and want to get to the bottom right corner, and as you pass through the edges, they have to alternate colors in this order: red, yellow, blue. (That is, you have to start with a red edge, end with a blue edge, and after each red edge you need to take a yellow edge, after each yellow edge you need to take a blue edge, and after each blue edge you need to take a red edge).

   We will represent the colorful maze as an undirected graph G, where each of its edges has been colored by one of these colors: red, yellow, or blue. Given are also two of G's vertices s and t. Design an $O(m + n)$ algorithm that will find the smallest number of edges one needs to take to go from s to t while obeying the rules of the maze.

   Consider the following example:

   You are given a colored graph in adjacency list format as a 2d array of integers, `adjList`, with edges colored according to the corresponding 2d String array `colors`:

   `adjList` = [[1, 3], [0, 2], [1], [0, 4], [3, 7], [], [], [4]]

   `colors` = [["blue", "red"], ["blue", "yellow"], ["yellow"], ["red", "yellow"], ["yellow", "blue"], [], [], ["blue"]]

If $s = 0$ and $t = 7$, the shortest colored path is [0, 3, 4, 7].

You will implement your solution by adding code to the `coloredMaze` method within the `ColoredGraph.java` file. Note that if your code does not implement a solution with the required linear running time $O(n + m)$ you will lose credit regardless of whether the test cases execute correctly.

(EXTRA CREDIT) Implement the `getSolution` method which returns the actual shortest path as an integer array containing the vertices along the shortest path, with s as the first element and ending with t in linear time. If no valid path exists, then this method should return null.

Note again that if your code does not implement a solution with the required linear running time $O(n + m)$ you will lose credit regardless of whether the test cases execute correctly.

4. **(PRACTICE)** For this problem, assume that the graph $G = (V, E)$ is undirected, connected and has at least 2 vertices. Do not assume that the edge weights are distinct. **PROVE** the following statements about minimum spanning trees.
   (a) There exists a graph G which has a cycle C, and e is the **unique lightest** edge in C so that e is a part of **at least one but not all** MSTs of G.

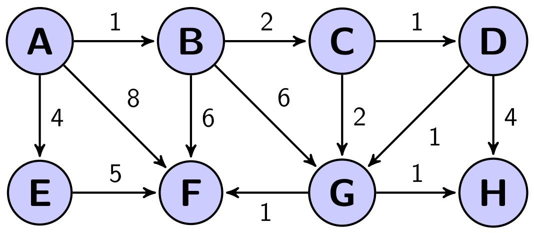
   Solution: YOUR SOLUTION HERE

   (b) There exists a graph G which has a cycle C, and e is the **unique lightest** edge in C so that e is **never** a part of any of the MSTs of G.

   Solution: YOUR SOLUTION HERE

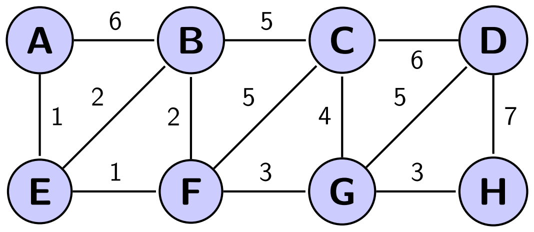
   (c) For any graph G, if G has a cycle C, and e is the **unique heaviest** edge in C then e is never a part of any MST on G.
   (d) For any graph G, let e be an edge of **minimal** weight then there is some MST which includes e.

5. **(PRACTICE)** Suppose Dijkstra's algorithm is run on the following graph, starting at node A.



   (a) Draw a table showing the intermediate distance values of all the nodes after each iteration of the while loop.
   (b) Show the min-heap (or priority heap) after each iteration of the while loop. List the heap operations that were performed (e.g. DeleteMin(Q), DecreaseKey(Q, v, 5)).
   (c) Give the final shortest path tree.

6. **(PRACTICE)** Run **Prim's Algorithm** on the following graph.



(a) Show the tree at each step. Indicate the final spanning tree.

(b) Draw a table showing the intermediate cost values of all the nodes at each iteration of the algorithm.

(c) Show the min-heap (or priority heap) at each iteration of the algorithm.

(d) How many minimum spanning trees does the graph have? Show each one.

Additional Programming Instructions:

Note that your code will be automatically run on a standard set of test cases. In order to ensure that you do not lose points, follow the instructions below.

- Your code must compile without any errors using the version of Java on the lab computers. If your code does not compile you will not receive any points for the assignment.
- Do not modify any of the methods signatures (i.e. name, return type or input type). Note that you are always welcome (and encouraged) to add additional methods but these will not be run directly by the test code.
- You are not allowed to use packages (e.g. no statement `package ...` at the top of your file).
- No extra folders or files in your submission. Zip up only the files you need to submit not the folder they are in.
- Your solution should not print anything unless explicitly instructed to.

Grading Criteria:

Your work will be graded on both correctness **and clarity**. Write complete and precise answers and show all of your work. Your pseudo-code and proofs should follow the guidelines posted on Canvas and discussed in class.
Evaluation Rubric for Problem 1.

Component	Requirement for Full Credit
Solution Correctness (1pt)	The answer is correct.
Written Explanation (2pt)	Your solution includes a logical argument explaining why your answer is correct. The explanation is clear, correct and complete. It includes full sentences and is easily understood by any student in the class. Think of this as a proof and remember what you learned in Math 128.

Evaluation Rubric for Problem 2.

Component	Description
Style & Documentation (1 pts)	I'll be watching for style issues as well as correct output. See the syllabus for some general style guidelines (e.g. your code should be well-documented).
Correct Output on Test Cases* (4 pt)	Your code will be run on a standard set of test cases. Make sure to test your code thoroughly!

*If your code does not run in linear $O(n + m)$ time you will lose at least 2 points.

Evaluation Rubric for Problem 3.

Component	Description
Style & Documentation (1 pts)	I'll be watching for style issues as well as correct output. See the syllabus for some general style guidelines (e.g. your code should be well-documented).
Correct Output on Test Cases* (4 pt)	Your code will be run on a standard set of test cases. Make sure to test your code thoroughly!
Extra Credit (2 pt)	Your method will be run on a standard set of test cases. To receive ANY points this method must run in $O(n+m)$ time and follow the class style guidelines.

*If your code does not run in linear $O(n + m)$ time you will lose at least 2 points.