Fall 2019 CS 316
Homework 6 Worth 21 points
Due Date: Monday, December 2, 2019, beginning of class

A hardcopy of your answers must be submitted to me at the beginning of class on the due date. Late assignments are not accepted. Electronic copies of the assignment are not accepted. The point value for each answer is given in brackets [].

A reasonable level of professionalism on submitted homework assignments is expected. This includes:
1. Your name at the top of every page of the assignment.
2. Paper size should be on standard typewriter paper, 8.5 x 11 inches, or standard notebook size, approx. 8 x 10.5 inches. You are permitted to type some answers and handwrite others if you prefer.
3. All pages stapled (preferred) or paper clipped, not folded at the corner.
4. If handwriting your answers, the ink or pencil should be dark, legible and of a normal size print (please do not write very small). Do not use highlighter pens; dark blue or black ink for pens.
5. The problems should appear in the order of the assignment and numbered accordingly. Grading your answers should not require leafing through pages or searching for your answers out of order.
6. Scribble is not professional. You may use pencil so that you can erase providing your print is dark.
7. If you are typing your answers, make certain the font is readable and LARGE; i.e. your font size should be size 12. A good font to use is Calibri.
8. The ratted edges of paper that remain after tearing paper from a spiral notebook is not acceptable.

Questions:
1. [2] Given the state of array s below used in the Union-Find “Quick-Find” algorithm,

   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
---|---|---|---|---|---|---|---|---|---|---|----|
   s | 3 | 4 | 2 | 3 | 4 | 2 | 3 | 2 | 3 | 9 | 2 |

   a. Draw the tree abstraction for the state array s above as done in class
   b. Show the new state of the array s after the Union method “unionSet(8,1)” is executed.

2. [2] Given the state of array s below used in the Union-Find “Quick-Union” algorithm,

   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
---|---|---|---|---|---|---|---|---|---|---|----|
   s | 1 | 4 | -1 | -1 | -1 | 2 | 3 | 1 | 6 | 5 | 2 |

   a. Draw the tree abstraction for the state array s above as done in class
   b. Show the new state of the array s after the Union method for the component that 8 belongs to and the component that 1 belongs to “unionSet(8,1)” is executed.

3. [2] Given the state of array s below used in the Union-Find “Quick-Union by Size” algorithm,

   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
---|---|---|---|---|---|---|---|---|---|---|----|
   s | -1 | 4 | -1 | -2 | -5 | 3 | 4 | -1 | -1 | 4 | 9 |

   a. Draw the tree abstraction for the state array s above as done in class.
   b. Show the new state of the array s after the Union method the component that 5 belongs to and the component that 10 “unionSet(5,10)” is executed.
4. [3] Show the topological sort order for the graph directly below using a queue to implement the algorithm. For credit, you must show your QUEUE states as done in class. Recall, for topological sorting weights are not used.

5. [3] (a) Show the topological sort order for the graph directly above using a stack to implement the algorithm. For credit, you must show your STACK states.

6. [3] For the graph directly below, ignore the edge weights and give the breadth-first-search unweighted shortest paths from vertex A to every other vertex. For credit, you must show for each vertex, its predecessor states and distance states as done in class. Your final result should give listing of the edges in the shortest path tree for starting vertex A.

7. [3] For the graph directly above, give a trace of Dijkstra’s Shortest Paths starting from vertex A. For credit, you must show for each vertex, its predecessor (a.k.a. path) states and distance states as done in class. Your final result should give listing of the edges in the shortest path tree for starting vertex A.

8. [3] For the graph directly below, give a trace of the Acyclic Shortest Paths algorithm starting from vertex A. For credit, you must show for each vertex, its predecessor (a.k.a. path) states and distance states as done in class.