Algorithms and Data Structures
MSIS 26:198:685
Homework 4

Instructor: Farid Alizadeh
Due Date: Tuesday December 9, 2019, at 11:50PM

last updated on December 2, 2019

Please answer the following questions in **electronic** form and upload and submit your files to Sakai site, before the due date. Make sure to push the **submit** button.

You can typeset your answer using MS Word (and MS equation for mathematical formulas), or LaTeX, or you may simply handwrite your note and scan and turn it not a **single pdf format file** and upload to Sakai.

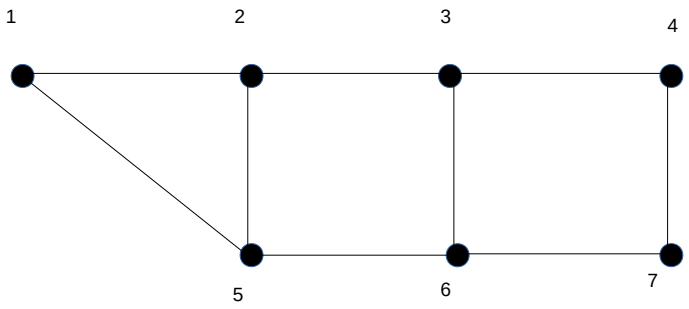
1. This question is about inserting and deleting in a balanced tree.

   1a) One by one insert the letters of the sequence of DATASTRUCTURES into a 2-tree tree. At each step show the status of the tree, and if there are 4-nodes created, show the details of their splitting.

   1b) Repeat part a) but this time use red-black trees. There should be a one-to-one correspondence between your red-black trees and your 2-3 trees.

   1c) Now delete the minimum element from the 2-3 tree. (No need to do it for the red-black tree.)

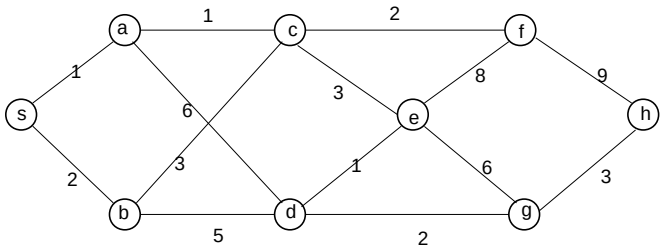
2. Consider the following graph.



2a) Apply the depth-first search algorithm with ties broken on alphabetical order. At each stage show the status of the stack, and indicate the edges used to build the depth-first search tree.

2b) On the same graph as in part a) run the breadth-first search, again showing the status of the queue at each iteration, along with the edges of the breadth-first search tree.

3. Consider the following graph with weights on the edges.



3a) show the adjacency list representation of the graph.

3b) Run the Kruskal's algorithm to find the smallest spanning tree. Assume the set of edges are sorted in increasing order of weight (so, no heap is needed.) At each iteration show the status of the forest as represented by the data structure for union-find operations. Show the *set representation of trees* only in tree format. Also show the status of this tree representation after every union-find operation.

3c) Now run Prim's "grow-a-tree" algorithm. At each iteration make sure to show the status of the heap, along with the in-heap, predecessor and index arrays.

3d) Now using Dijkstra's algorithm find all the shortest paths from vertex 1 to all other vertices. Again, show the status of the heap, and the in-heap, predecessor and index arrays at each iteration.

4. Run the Bellman-Ford algorithm to find the shortest path from vertex 'a' in the following graph. At each iteration show the ordered set of edges, and the relabeling and the predecessor of each vertex. Use the lexical ordering of the edges, for instance edge 1-2 comes before 1-3 etc. Determine if the graph has a negative cycle or not, and if so, how does the Bellman-Ford algorithm determines that.