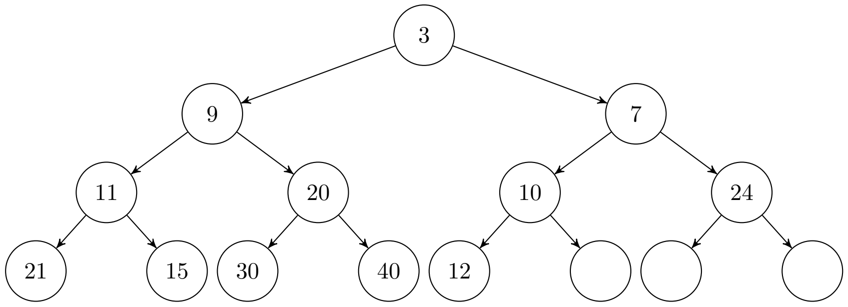
CS350 — Winter 2019
Homework 4

Due Tuesday 12th March, on paper, at the start of class. This assignment *will* be graded.

1. **Priority Queues.** The figure below shows a *min*-heap used to represent a priority queue.



   (a) Insert an item with priority 2 into the priority queue; draw all steps for the insertion, showing the swaps at each step.

   (b) What is the average-case time-efficiency of performing an insert operation in a *min*-heap?

   (c) What is the average-case time-efficiency of removing the minimum item from the heap?

   (d) Imagine that you are using this priority queue in an algorithm that requires the priorities to be *updated* as the algorithm runs. (Dijkstra's shortest path algorithm is an example of an algorithm that does this; in this algorithm the "priority" of an item is the shortest path found so far from that item to the start.) Give pseudo-code for an efficient algorithm for implementing the *decrease*(v) operation that changes the priority of item v in a min-heap. You may assume that you are given a handle to v, and operations to find the parent and the left and right children of any tree node.

   (e) What is the time-efficiency of your algorithm (in big-oh notation).

2. **Huffman Codes.**

   (a) Construct a Huffman code for the following data:

symbol	X	Y	B	D	#
frequency	0.45	0.16	0.06	0.08	0.25

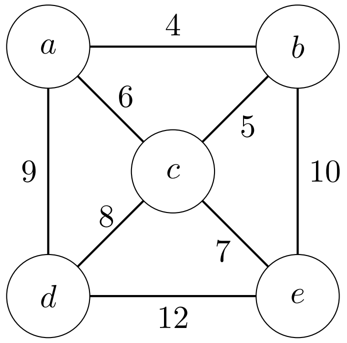
   Follow the algorithm on slide 14 from Lecture 14. Assume that the item on the *left* branch of the Huffman tree is represented by a 0 bit, and that on the *right* branch by a 1 bit.

   (b) Encode DD#B#YBX#Y using the code that you created for part (a).

   (c) Decode the text whose encoding is 10001011100101000 using the code that you created for part (a).

3. **Prim's Algorithm.**

   Apply Prims algorithm to the graph below, starting with vertex a. Construct a table that shows the vertices added to the spanning tree in one column, and the priority queue containing the remaining vertices in a second column, as Levitin does in Figure 9.3. (It is *not* necessary to illustrate the graph at each step.) Label each vertex with (1) the nearest vertex in the spanning tree so far, and (2) the weight of the edge to that vertex.



Tree vertices	Priority queue of remaining vertices

   When the algorithm is complete, show how to construct the list of edges that makes up the spanning tree from the information in your table, and write down this list.