Module 9 - Discussion Prompt Questions
Adam Rich
April 7, 2018
Section EN.605.202.87.SP18

1. How is a priority queue different from a traditional queue? What applications would make the best use of such a data structure?

   A priority queue will "pop" the element with the highest priority, an additional data value, instead of just the one that has been in the queue the longest. Another way to look at it is that a regular queue is a priority queue where the priority equals the time in the queue.

2. Is there an implicit prioritization in an traditional queue? If so, what is it?

   Yes, it is the amount of time the element has been in the queue.

3. What data structures lend themselves to effective implementation of a priority queue?

   Min or max heaps, which in turn can be implemented using arrays.

4. How does the order of of a tree (a general tree with no special properties) affect the size of the tree and layout of the nodes in the tree. How do you decide which child path to follow?

   A general tree has exactly one path between any two nodes. However, it is not necessary rooted. Giving the tree is giving a root node and then deciding how child elements are arranged left to right. A rooted, ordered tree has only one path from the root node to any descendant.

5. How would a Huffman tree look different if we used different tie-breakers. and how would that impact the potential compression.

   The trees we studied in the lectures are "skewed right" because larger nodes are on the right. If we put them on the left instead the trees would be "skewed left". I don't know that it would affect the compression because I think that with a tree built that way, the 1/0 would have to be changed so that 1 meant go left and zero go right. Or else it would be impossible to distinguish between codes 0, 00, and 000, etc.