Module 9 - Discussion Prompt Questions
Jot down short answers for each prompt question as you go along. Review and update your answers after finishing and submit before Tuesday midnight. Click on "Module 9 - Discussion Prompt Questions" above to submit.
The prompt questions for this module are:

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

A traditional queue is FIFO so that a push operation puts the item in the back of the queue, behind all other items in the queue currently. Items pushed first and popped first. A priority queue is a queue where each item has a priority, and items with higher priority are closer to the front of the queue (will be popped off first) than items with lower priority. The priority queue push operation inserts a new item such that the item is in front of all items of lower priority, and in the back of all items with equal or higher priority.

When pushing items of equal priority, the first-in-first-out rules apply. The 2nd item is after the first item and will be popped later than the original item with the same priority.

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

Traditional queues are FIFO so that any newly pushed items have lower priority than all other items in the queue currently. I guess you can say stacks are also an implicitly priority queue and are LIFO so that newly pushed items have higher priority than all other items in the queue currently.

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

And unordered array or linked list can both support the priority queue. Each array item could hold both the data and the priority and inserts would have to traverse the array to search for the array item with a lower priority than the item being inserted and insert the item ahead of it. Linked lists could do the same; each node holds data and the priority and inserts would adjust node pointers to include the new item.

A heap can also support a priority queue. A Heap is a tree data structure in which the tree is a complete binary tree. A max heap where the key present at the root node must be greatest among the keys present at all of its children is appropriate. Since there is an order in a heap (and it's often implemented using arrays) a heap is a good candidate for the data structure for the priority queue.

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?**

So, trees are a non-linear data structure. They don't store data in a linear way. They organize data hierarchically. The height of a tree is the length of the longest path to a leaf (a node without children). The depth of a node is the length of the path to its root.

- Root is the topmost node of the tree
- Edge is the link between two nodes
- a Leaf is a node that does not have a child node in the tree

- Height is the length of the longest path to a leaf
- Depth is the length of the path to its root

If the tree is unordered, I'm not sure that the size or layout of the nodes are affected. If we order the tree in a binary search tree, then the tree's height at minimum is logN base 2. The max is N-1 if the nodes are inserted in sorted order.

Without knowing more about the tree, unsure which path to follow. You can do a breadth first search or an in-order traversal of the tree. If it's a BST, then you can find the node you're looking for by realizing the rules of the BST: lesser nodes are to the left of any parent node, and greater nodes are to the right of any parent node.

To grader, can you give me a better answer here? I am a tad lost admittedly on this question.

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

If we used different tie breakers, then the nodes on that level would switch in the binary tree. What was previously on the left is now on the right and vice versa. Then the 0s and 1st would be on different nodes than before. So now a code that was 110 might now be 111 if the last level tiebreakers were switched.