Part a: Trees

For questions in this part, you will use the following values: `[1,2,3,5,7,9,11,16,17,18]`

Question 1 [**5 points**]

Draw a Max Heap (as a tree) containing the values above. It can be any valid Max Heap that contains exactly the above values.

Question 2 [**5 points**]

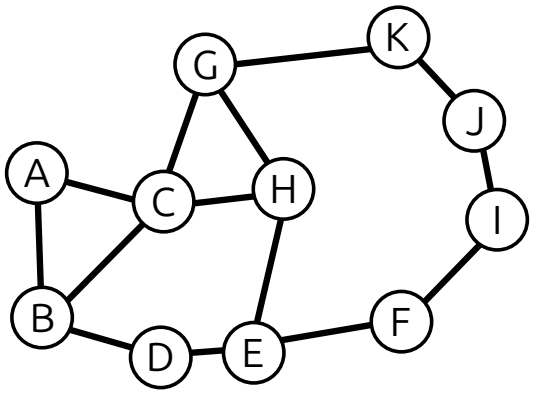
Write out the ArrayList representation of the Heap you drew in the previous question.

Question 3 [**10 points**]

Draw an AVL tree containing exactly the values above. It can be any valid AVL tree that contains exactly the above values.

Part B: Graphs

The following questions pertain to the graph below. Where relevant, assume that the `incidentEdges` method returns out-edges in **alphabetical order** of the opposite vertex. e.g., `C.incidentEdges()` would return `A, B, G, H`.



Question 1 [**10 points**]

Perform a **Breadth-First Traversal** of the graph starting at vertex **A** and provide a list of every vertex visited in the order in which they are **removed** from the Queue .

Question 2 [**5 points**]

Based on your answer above, is the undirected edge connecting vertices **A** and **C** a *spanning* edge (part of the spanning tree created by the traversal) or a *cross/back* edge (not part of the spanning tree). (Circle One)

Spanning Edge **Cross/Back Edge**

Question 3 [**5 points**]

Based on your answer above, is the undirected edge connecting vertices **J** and **I** a *spanning* edge (part of the spanning tree created by the traversal) or a *cross/back* edge (not part of the spanning tree). (Circle One)

Spanning Edge **Cross/Back Edge**

Part c: Priority Queue

In class, we designed two priority queues using a linked list, and one using a heap structure. We also discussed several forms of binary search tree.

Question 1 [**5 points**]

Explain **in at most 2 sentences**, or a small block of pseudocode, how you would implement the `add` operation of the priority queue using a binary search tree. You may call (or reference) the three BST operations discussed in class (`find`, `insert`, and `remove`) *without* having to explain how they work.

Question 2 [**5 points**]

Provide an upper bound (i.e., Big-O) on the runtime complexity of your `add` operation for (i) A generic binary search tree, (ii) An AVL tree, and (iii) A Red-Black tree. Your answers should be in terms of the number of elements in the priority queue (N).

`Generic BST:`

`AVL Tree:`

`Red-Black Tree:`

Question 3 [**5 points**]

Explain **in at most 2 sentences**, or a small block of pseudocode, how you would implement the `remove` operation of the priority queue using a binary search tree. You may call (or reference) the three BST operations discussed in class (`find`, `insert`, and `remove`) *without* having to explain how they work.

Question 4 [**5 points**]

Provide an upper bound (i.e., Big-O) on the runtime complexity of your `remove` operation for (i) A generic binary search tree, (ii) An AVL tree, and (iii) A Red-Black tree. Your answers should be in terms of the number of elements in the priority queue (N).

Generic BST:

AVL Tree:

Red-Black Tree:

Generic BST:

AVL Tree:

Red-Black Tree:

Question 5 [**5 points**]

Bonus Question Due to a hardware component called the cache, in most practical settings, it can be significantly more expensive to access an object referenced by a pointer (e.g., following a pointer in a linked list node going from one node to another in a tree) than to access a specific index in a relatively small array. Give an asymptotic upper (i.e., Big-O) bound on the number of pointers accessed when calling `add` on a priority queue implemented using (i) The ArrayList-based heap discussed in class, and (ii) an AVL Tree, as you designed above.

Heap:

AVL Tree:

Part d: Runtimes

Question 1 [**10 points**]

What is the tight worst-case runtime to find a **specific value** in each of the following data structures?

Sorted ArrayList:

Sorted LinkedList:

Min Heap:

General BST:

Balanced BST:

Question 2 [**10 points**]

What is the tight worst-case runtime to remove a **specific value** from each of the following data structures if you do not have a reference to it or know the index of it?

Sorted ArrayList:

Sorted LinkedList:

Min Heap:

General BST:

Balanced BST:

Part e: Data Structure Design

Task Scheduler

You are implementing a task scheduler for a new web application. This component will keep track of a large number of tasks that the application needs to perform. When one of the computers supporting the web application is available, it will request the next task to be performed.

The tasks need to be prioritized according to the deadline. The next task to be performed is the one with the earliest deadline. Tasks may arrive in any order.

Question 1 [**5 points**]

Identify the **Abstract Data Type** (*not* the data structure) that best aligns with the needs of the task above.

Question 2 [**5 points**]

Identify the specific **Data Structure** that you would use to implement the task above. In no more than 2 sentences, justify your choice.

Flight Departure Table

You are implementing a system for the 'big board' of flight departures for the coming day at a major airport. Your data structure must be able to efficiently (i) List all flights for a given range of times, and (ii) Move a flight from one departure time to a different departure time.

The airport operates using a fixed number of departure 'slots'. (e.g., one slot for a 10:00 AM departure, one for 10:05 AM, one for 10:10 AM, etc...). A departing flight may only be moved to an open slot.

Question 3 [**5 points**]

Identify the **Abstract Data Type** (*not* the data structure) that best aligns with the needs of the task above.

Question 4 [**5 points**]

Identify the specific **Data Structure** that you would use to implement the task above. In no more than 2 sentences, justify your choice.

Scrap Page

Scrap Page