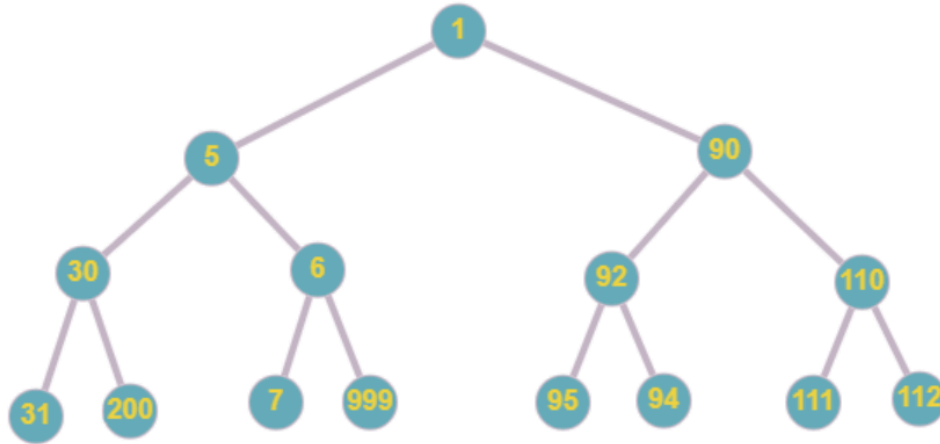


CS 231 Assignment 9: HashedHeap

In this assignment, we'll investigate one of the annoyances of the Heap structure you've built so far. In particular, imagine you have a heap like:



If you had to find the value 999, you would have to start from the root, but there is really no way of knowing which direction you should walk. You basically have to try both directions and see if you find it.

This is annoying particularly for the methods of `contains`, `remove`, and `updatePriority`, which need to find the given object in the structure. To assist with this we'll include a `HashMap` under the hood of `Heap` to make finding specific objects easier. Specifically, this `HashMap` will map each object to the index in the heap-array that holds that object. To help get you started, we've started [this file](#) (you'll also need this [PriorityQueue](#) interface): since it's a set, you can ignore duplicate offers. Your main goal is to implement the `HashedHeapSet` so that `contains`, `remove`, and `updatePriority` all take at most log time (assuming $O(1)$ runtimes from the `HashMap`). Actually, `contains` should take constant time. **Note that this means you need to include a `contains` and `remove` method.**

To submit, include your finished `HashedHeapSet.java` file and the associated `PriorityQueue.java` file in your homework folder.