

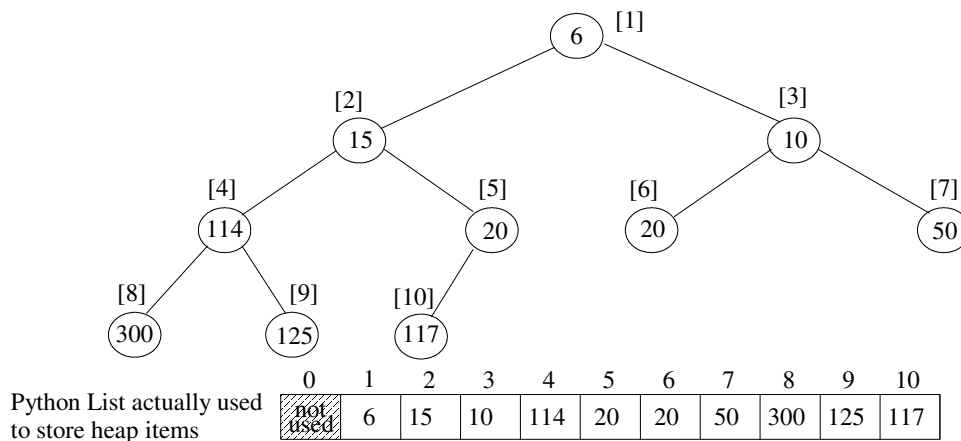
**Objective:** To understand recursion by writing simple recursive solutions.

**To start the lab:** Download and unzip the file at: [www.cs.uni.edu/~fienup/cs1520s13/labs/lab5.zip](http://www.cs.uni.edu/~fienup/cs1520s13/labs/lab5.zip)

**Part A:** Complete the recursive `searchHelper` function in the `search` method of our `OrderedList` class in `ordered_linked_list.py`. Test it with the `listTester.py` program.

**Raise your hand when done. Demonstrate and explain your code to an instructor or TA.**

**Part B:** Recall that Lecture 7 and Section 6.6 discussed a very “non-intuitive”, but powerful list/array-based approach to implement a priority queue, call a binary heap. The list/array is used to store a *complete binary tree* (a full tree with any additional leaves as far left as possible) with the items being arranged by *heap-order property*, i.e., each node is  $\leq$  either of its children. An example of a *min heap* “viewed” as a complete binary tree would be:



Recall the General Idea of `insert(newItem)` :

- append `newItem` to the end of the list (easy to do, but violates heap-order property)
- restore the heap-order property by repeatedly swapping the `newItem` with its parent until it *percolates up* to the correct spot

Recall the General Idea of `delMin()` :

- remember the minimum value so it can be returned later (easy to find - at index 1)
- copy the last item in the list to the root, delete it from the right end, decrement size
- restore the heap-order property by repeatedly swapping this item with its smallest child until it *percolates down* to the correct spot
- return the minimum value

Originally, we used iteration (i.e., a loop) to percolate up (see `percUp`) and percolate down (see `percDown`) the tree. Now I want you to complete the recursive `percUpRec` and recursive `percDownRec` methods in `binHeap.py`. Run this file to test your code.

**Raise your hand when done. Demonstrate and explain your code to an instructor or TA.**

If you have extra time, work on homework #3!