Questions

1. (a) Use the `upHeap()` algorithm to build a heap out of the characters of the word: `computed`.
   (b) What is the result of applying the `removeMin()` method to the heap in (a) ?
   (c) Give an example of a heap that uses the characters of the word `computed`, and that has the following additional property: for any node with two children, the left child is less than the right child.
   (d) Give an example of a complete binary tree that contains the characters of the word `computed` and is also a *binary search tree*. (Such a tree is not a heap.)

2. Suppose you have an array with the following characters in it.

   `f b u e l a k d w`

   Show how the array evolves after each of the $n = 9$ loops of the slow `buildHeap` method.

3. The `removeMin()` method puts the last element of the heap into the root. Will this element necessarily need to be moved down in the heap? The answer is obviously no if there were only two elements in the heap before the call. Consider the more interesting case that the heap had at least four elements in it when the `removeMin()` was called.

   If this is obvious to you, then you should still try to write down your argument why.

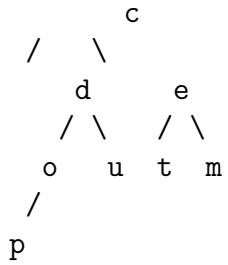
4. Give a tight $O(\ )$ bound for heapsort. Explain when the worst case occurs.

5. Suppose you want to see if a heap contains some element and, if so, then return the index of the element. Give an algorithm for `find(element)` that is as efficient as you can.

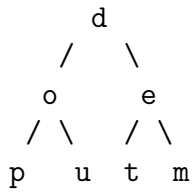
Answers

Have you given an honest shot to the questions or did you jump here right away? In the latter case, stop reading. Get your hands out of the cookie jar!

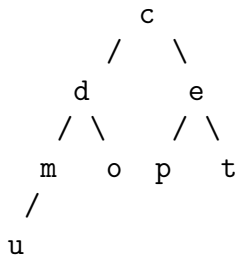
1. (a)



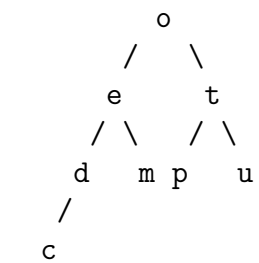
(b)



(c) The easiest way to solve this one is to sort the elements from smallest to largest. The resulting sequence, if written as a array, defines a heap.



(d) The tree node structure is determined by the number of nodes. The only question is where to place the elements. But remember that when you traverse a binary search tree "in order", you visit the elements in order. So you need to sort the elements of the string: `cdemoptu` and place them into the tree in order.



2. The boundary between the heap and non-heap elements is marked with |.

```
1   2   3   4   5   6   7   8   9
-----------------------------------
| f   b   u   e   l   a   k   d   w
f | b   u   e   l   a   k   d   w    (added f)
b   f | u   e   l   a   k   d   w    (added b)
b   f   u | e   l   a   k   d   w    (added u)
b   e   u   f | l   a   k   d   w    (added e)
b   e   u   f   l | a   k   d   w    (added l)
a   e   b   f   l   u | k   d   w    (added a)
a   e   b   f   l   u   k | d   w    (added k)
a   d   b   e   l   u   k   f | w    (added d)
a   d   b   e   l   u   k   f   w |  (added w)
```

3. Yes, it will necessarily need to be moved down. If the last element in the heap is at level l where $l > 1$, then it will have an ancestor that is a child of the root. (This ancestor may be its parent, or grandparent, etc.) But in a heap, a node is always greater than all of its ancestors. So, when the last element is moved to the root and becomes a parent of one of its (former) ancestors, this new root will be greater than its (former) ancestor, which is now its child, and so this new root will not satisfy the heap property.

4. We can ignore the `buildHeap()` step since I showed in class this was $O(n)$.

   The heapsort algorithm consists of a loop that passes n times. In pass k, starting at $k = 0$, there is a heap with $n - k$ elements and the algorithm removes the minimum element. Removing the minimum element takes the last element, moves it to the root, and then down heaps it, which swaps it at most $floor(log_2 n)$ times. So, we just need to sum this total number of swaps. But this is the same sum we saw in the slow buildheap algorithm which was $\Theta(nlog_2 n)$.

   Verify with examples the worst case indeed occurs if the initial built heap happens to produce an array of elements that is sorted from smallest to largest.

5. Unlike binary search trees, heaps do not have enough structure to support finding a particular element. Instead, one has to traverse the whole tree in the worst case, which is $\Theta(n)$.