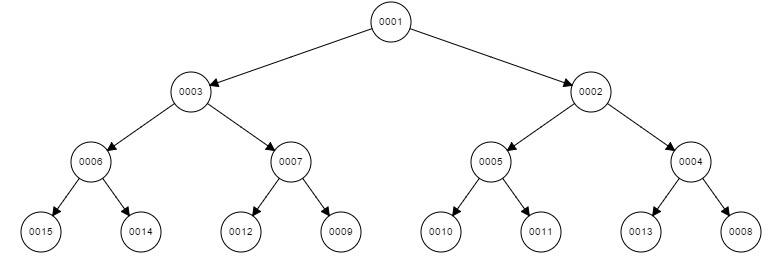
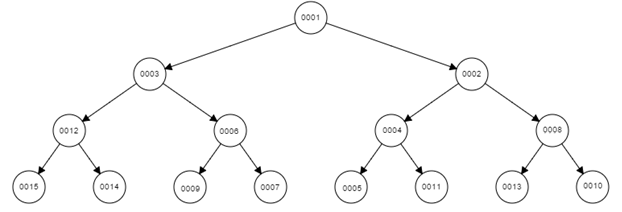
Neal Noble  
IT333  
Exercises - Heaps

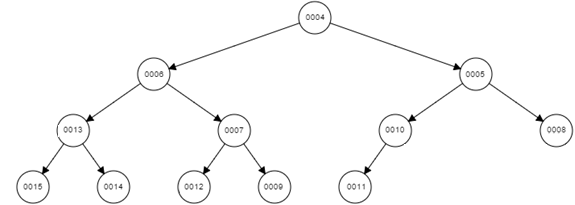
1a. Show the result of inserting 10, 12, 1, 14, 6, 5, 8, 15, 3, 9, 7, 4, 11, 13 and 2, one at a time, into an initially empty binary heap

1,3,2,6,7,5,4,15,14,12,9,10,11,13,08 

1b. Show the result of using the linear-time algorithm to build a binary heap using the same input.



2. Show the result of performing three deleteMin() operations in the heap of the previous exercise.



3. Suppose that we instead placed elements in a binary heap starting at index zero in an array. Given the index of an element in the tree, what would be the calculation to find the:

a) index of the parent node

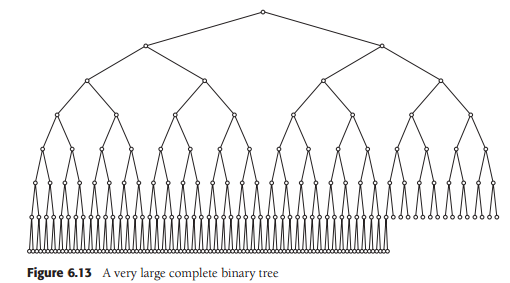
The index of the parent node is found by dividing the current index by 2(i/2)

b) index of the left child node

The index of the left child node is found by multiplying the current index by 2 (2i)

c) index of the right child node

The index of the left child node is found by multiplying the current index by 2, then add 1(2i + 1)



4. How many nodes are in the large heap in Figure 6.13 of your book?

5. A d-heap is a more general version of a binary heap where each node has up to d children. The heap property of the tree would be similar to a binary heap in that each of the d children of a node must be larger (for a min-heap) than the parent node. How would this design affect the runtime analysis for a heap structure?

A d-heap has a shallower depth than a binary heap, which results in running time of O(logd N) when inserting elements into the d-heap. When deleting an element from the heap, it increases the cost for the size of d because all children must be found and evaluated for deletion, increasing the cost to O(d logd N). A d-heap is best used when there is likely to be a lot elements added to the heap, and very little items removed.