University of Washington

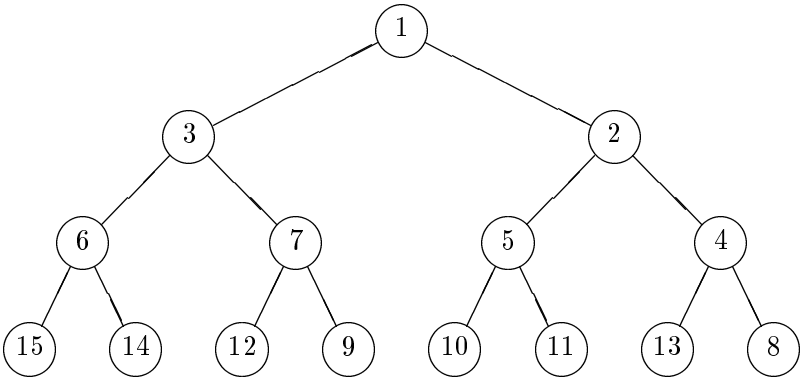
Department of Computer Science and Engineering

CSE 373

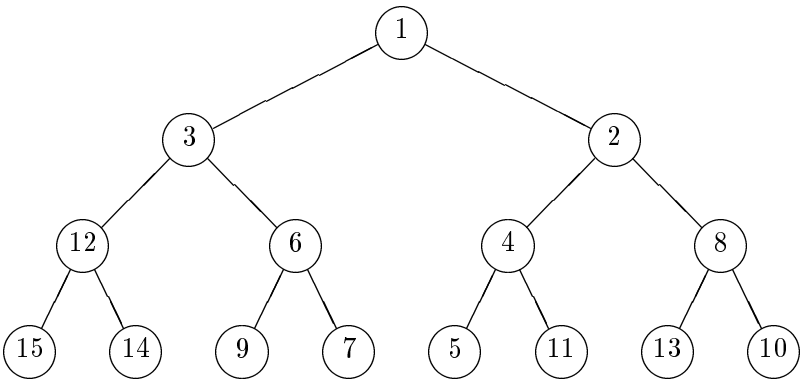
Problem Set 5 Solution

1. (2 points) Heap Insertion

(a)

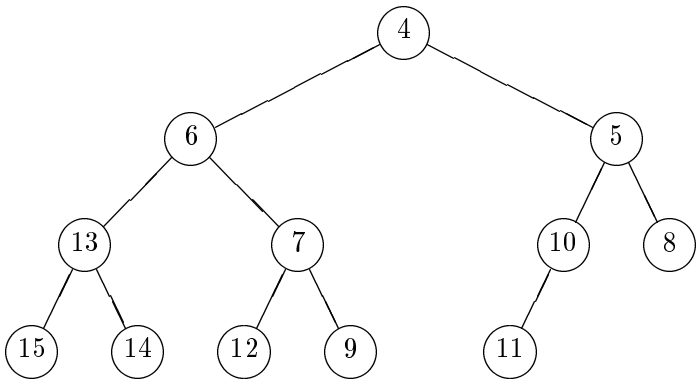


(b)

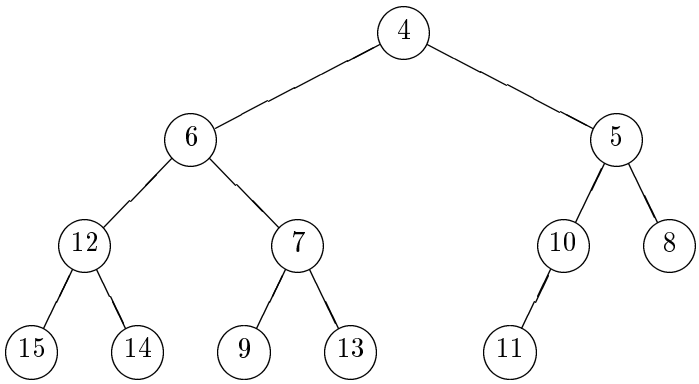


2. (2 points) Performing DELETE_MIN

   Assume we start with the solution in 6.2(a). Then the answer is



If we start with the solution in 6.2(b), the answer is



3. (2 points) Counting Nodes in a Big Tree

   There are 225 nodes. One possible derivation is to start with $i = 1$ at the root of the tree, and move towards the last leaf. Every time a left child is taken, double i; every time a right child is taken, double i and add 1.

4. (2 points) Insertion Sort on a Sorted Array

   Since the algorithm will never do more than compare the current element to its sucessor as it moves down the array, the time required is $O(n)$.

5. (3 points) Analysis of Mergesort

   The merging step always takes exactly $O(n)$ time (this is often called $\Theta(n)$ time), so the sorting process always takes $O(n \log n)$ (actually $\Theta(n \log n)$) time regardless of the input.

6. (3 points) Effects of Pivot Choice on Quicksort

   (a) $O(n \log n)$

   (b) $O(n \log n)$

(c) $O(n \log n)$

7. (2 points) Quicksort When Keys Are Equal

   $O(n \log n)$ because the pivot will partition perfectly.

8. (4 points) Stable Sorts

   Insertion sort and mergesort are stable if coded correctly. Heap sort is unstable because instances of the same key can be scattered throughout the heap, and there is no guarantee of which will be encountered first. Quicksort may reorder elements when performing swapping to partition the elements. Note that any sort can be made stable by introducing a secondary key which is the element's original position.