1. Suppose a binary maxHeap is represented as an array: 93 85 50 72 40 45 10 64 68 12. What are the left and right children, respectively, of the element 50?

2. Draw the tree representation of the heap that results after inserting a node with key = 42 into the heap of the preceding question.

3. BuildHeap constructs a heap from a random array of length $n$ by running MaxHeapify on $\lfloor n/2 \rfloor$ non-empty subtrees. As MaxHeapify can require as many as $\lg x$ operations to complete, when operating on a subtree of height $x$, the obvious upper bound for BuildHeap is $O(n \lg n)$. Is there a tighter (that is, lower) upper bound? If so, what feature of the tree enables you to reduce the $n \lg n$ estimate?

4. Assuming random permutations of $n$ distinct values as inputs, what are the complexity classes for the worst case, average case, and best case for the heapSort algorithm? Name another sorting algorithm that has this same pattern of behavior.
Solutions

1. Suppose a binary maxHeap is represented as an array: [93, 85, 50, 72, 40, 45, 10, 64, 68, 12]. What are the left and right children, respectively, of the element 50?

The heap, as a binary tree, is filled by level, top-to-bottom, left-to-right, as in the following sketch. Therefore the left child of 50 is 45; the right child is 10.

![Binary MaxHeap Diagram]

2. Draw the tree representation of the heap that results after inserting a node with key = 42 into the heap of the preceding question.

Node 42 starts as the right child of node 40 in the original tree. It then swaps up one level to the position shown in the above sketch.

![Updated MaxHeap Diagram]

3. BuildHeap constructs a heap from a random array of length n by running MaxHeapify on ⌊n/2⌋ non-empty subtrees. As MaxHeapify can require as many as $\lg x$ operations to complete, when operating on a subtree of height $x$, the obvious upper bound for BuildHeap is $O(n \lg n)$. Is there a tighter (that is, lower) upper bound? If so, what feature of the tree enables you to reduce the $n \lg n$ estimate?

There is a tighter bound, namely $O(n)$. While MaxHeapify is indeed run on $O(n)$ subtrees, the vast majority of those trees are short. On the lowest level probed by BuildHeap, MaxifyHeapify finds subtrees of height 1. On the next level up, it finds subtrees of height 2, and so forth. Hence most of the subtrees result in MaxHeapify counts in the range 1, 2, 3. Only one subtree has height $\lg n$. This distribution causes the instruction count to look roughly like $\sum_{k=1}^{\lg n} k 2^k$, which is a geometric sum variation that sums to a multiple of $n$.

4. Assuming random permutations of $n$ distinct values as inputs, what are the complexity classes for the worst case, average case, and best case for the heapSort algorithm? Name another sorting algorithm that has this same pattern of behavior.

The heapSort algorithm is $\Theta(n \lg n)$ for worst-case, average-case, and best-case input instances. Another sorting algorithm with the same pattern is mergesort.