CS 152h: Data Structures Homework #3

SORTING

7.19 Sort 3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5 using quicksort with median-of-three partitioning.

Ans The original input is

$$3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5$$

After sorting the first, middle, and last elements, we have

$$3, 1, 4, 1, 5, 5, 2, 6, 5, 3, 9$$

Thus the pivot is 5. Hiding it gives

$$3, 1, 4, 1, 5, 3, 2, 6, 5, 5, 9$$

The first swap is between two fives. The next swap has i and j crossing. Thus the pivot is swapped back with i:

$$3, 1, 4, 1, 5, 3, 2, 5, 5, 6, 9$$

We now recursively quicksort the first eight elements:

$$3, 1, 4, 1, 5, 3, 2, 5$$

Sorting the three appropriate elements gives

$$1, 1, 4, 3, 5, 3, 2, 5$$

Thus the pivot is 3, which gets hidden:

$$1, 1, 4, 2, 5, 3, 3, 5$$

The first swap is between 4 and 3:

$$1, 1, 3, 2, 5, 4, 3, 5$$

The next swap crosses pointers, so is undone; i points at 5, and so the pivot is swapped:

$$1, 1, 3, 2, 3, 4, 5, 5$$

A recursive call is now made to sort the first four elements. The pivot is 1, and the partition does not make any changes. The recursive calls are made, but the subfiles are below the cutoff, so nothing is done. Likewise, the last three elements constitute a base case, so nothing is done. We return to the original call, which now calls quicksort recursively on the right-hand side, but again, there are only three elements, so nothing is done. The result is

$$1, 1, 3, 2, 3, 4, 5, 5, 5, 6, 9$$

which is cleaned up by insertion sort.

7.32d Suppose you are given a sorted list of N elements followed by $f(N)$ randomly ordered elements. How large can $f(N)$ be for the entire list to be sortable in $O(N)$ time?

Ans $f(N)$ can be $O(N/\log N)$. Sort the $f(N)$ elements using mergesort in $O(f(N) \log f(N))$ time. This is $O(N)$ if $f(N)$ is chosen using the criterion given. Then merge this sorted list with the already sorted list of N numbers in $O(N + f(N)) = O(N)$ time.

7.11 Show how heapsort processes the input

$$142, 543, 123, 65, 453, 879, 572, 434, 111, 242, 811, 102.$$

Ans The input is read in as

$$142, 543, 123, 65, 453, 879, 572, 434, 111, 242, 811, 102$$

The result of the heapify is

$$879, 811, 572, 434, 543, 123, 142, 65, 111, 242, 453, 102$$

879 is removed from the heap and placed at the end. We”ll place it in italics to signal that it is not part of the heap.

102 is placed in the hole and bubbled down, obtaining

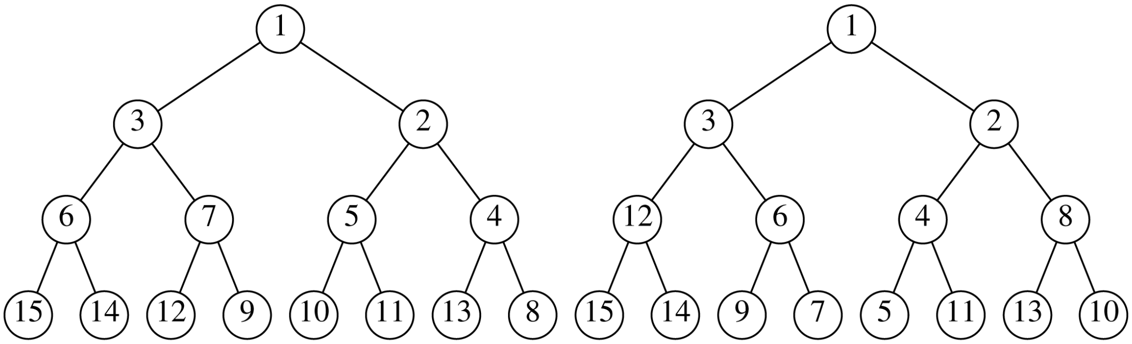
$$811, 543, 572, 434, 453, 123, 142, 65, 111, 242, 102, \mathit{879}$$

Continuing the process, we obtain

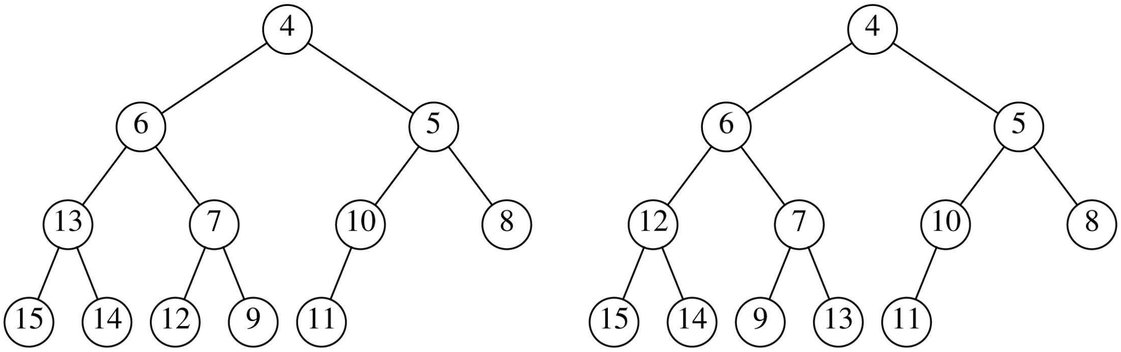
$$\begin{aligned}
572, 543, 142, 434, 453, 123, 102, 65, 111, 242, \mathit{811}, \mathit{879}\\
543, 453, 142, 434, 242, 123, 102, 65, 111, \mathit{572}, \mathit{811}, \mathit{879}\\
453, 434, 142, 111, 242, 123, 102, 65, \mathit{543}, \mathit{572}, \mathit{811}, \mathit{879}\\
434, 242, 142, 111, 65, 123, 102, \mathit{453}, \mathit{543}, \mathit{572}, \mathit{811}, \mathit{879}\\
242, 111, 142, 102, 65, 123, \mathit{434}, \mathit{453}, \mathit{543}, \mathit{572}, \mathit{811}, \mathit{879}\\
142, 111, 123, 102, 65, \mathit{242}, \mathit{434}, \mathit{453}, \mathit{543}, \mathit{572}, \mathit{811}, \mathit{879}\\
123, 111, 65, 102, \mathit{142}, \mathit{242}, \mathit{434}, \mathit{453}, \mathit{543}, \mathit{572}, \mathit{811}, \mathit{879}\\
111, 102, 65, \mathit{123}, \mathit{142}, \mathit{242}, \mathit{434}, \mathit{453}, \mathit{543}, \mathit{572}, \mathit{811}, \mathit{879}\\
102, 65, \mathit{111}, \mathit{123}, \mathit{142}, \mathit{242}, \mathit{434}, \mathit{453}, \mathit{543}, \mathit{572}, \mathit{811}, \mathit{879}\\
65, \mathit{102}, \mathit{111}, \mathit{123}, \mathit{142}, \mathit{242}, \mathit{434}, \mathit{453}, \mathit{543}, \mathit{572}, \mathit{811}, \mathit{879}
\end{aligned}$$

6.2a 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.

Ans



6.3a Show the result of performing three `deleteMin` operations in the heap of the previous exercise.



Ans