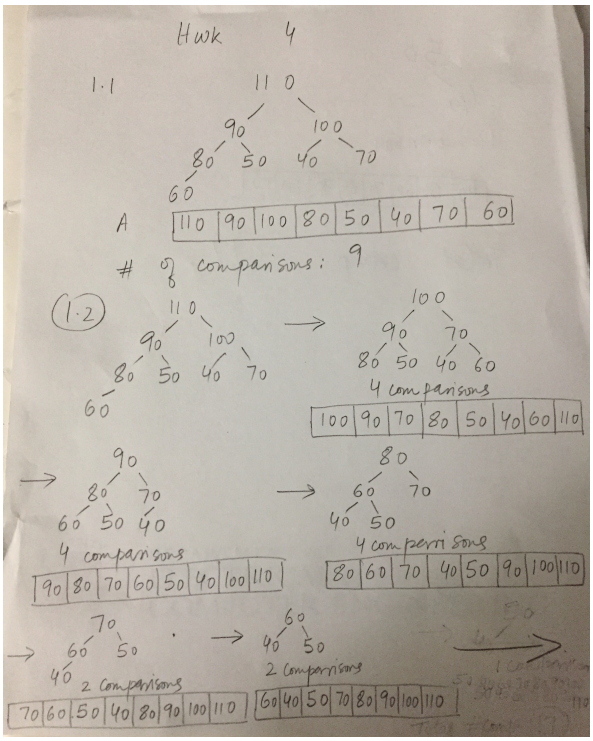
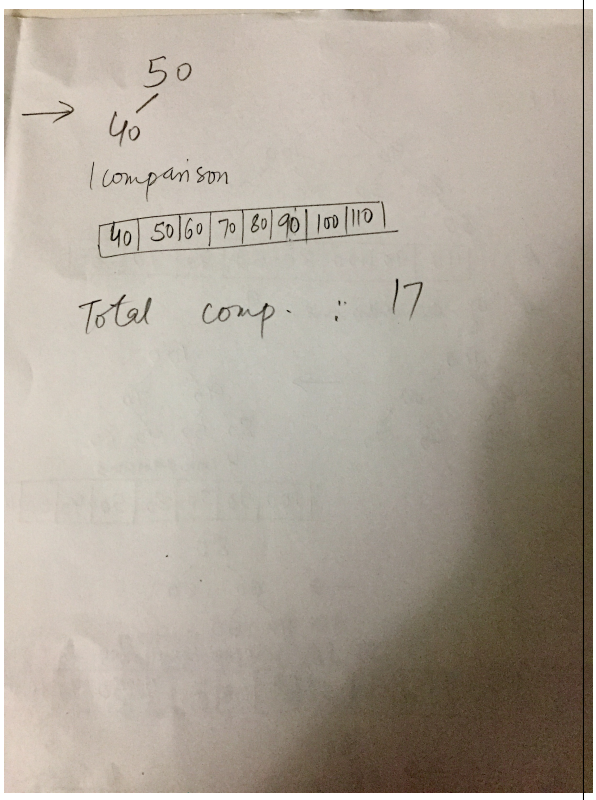
Problem 1. We want to sort the following list, $[70, 50, 110, 60, 90, 40, 100, 80]$, of numbers using heap sort algorithm that we covered in class. It will involve two steps, first, building a max heap, second, sorting, by extracting the root and removing any max-heap violations. Apply this algorithm to the given array and answer the following questions.

1. Show the max-heap as a tree. Show the array corresponding to this max-heap. Exactly how many comparisons did it take to build the max-heap?
2. Starting with the max-heap built in the previous step, show the array after each max-heapify. How many comparisons does each max-heapify use? What is the total number of comparisons for just this part, excluding the number of comparisons in Part(a).

Solution:





Problem 2. For this question, we will have to modify Merge sort algorithm. In the divide and conquer step, you will divide the array into thirds, recursively sort each third, and finally combine the results using a three-way merge routine. Answer the following question:

1. Write pseudo code for the modified Merge sort algorithm

Solution:

```
mergeSort(A,first,last):
    if first >= last:
        return A
    p = first + (last−first)/3
    q = first + 2(last−first)/3
    a = mergeSort(A,first, p)
    b = mergeSort(A, p+1, q)
    c = mergeSort(A, q+1, last)
    d = merge(a,b)
    return merge(c,d)
```

the regular merge routine shown in the class to merge two arrays first followed by merging the third array. The number of comparisons to merge these three arrays is:

(a) Merge a with b each containing $\frac{n}{3}$ numbers. The output array will have $\frac{2n}{3}$ numbers Thus, the number of comparisons is:

$$\frac{2n}{3} - 1$$

(b) Merge the result with the third list. The output array will have n numbers. Thus the number of comparisons is:

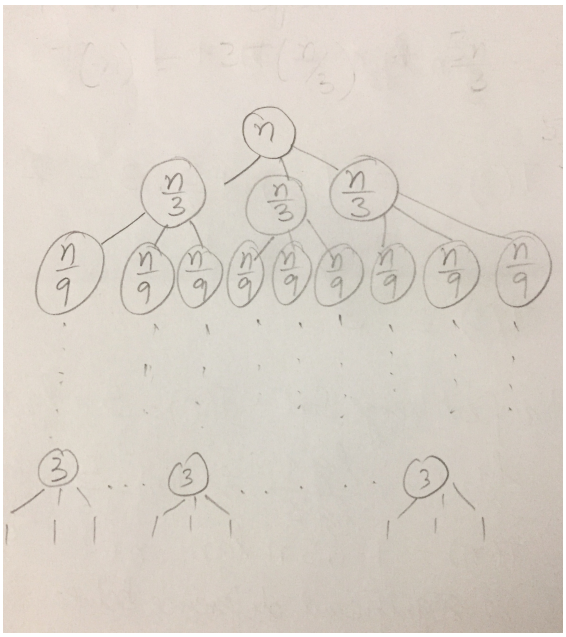
$$n - 1$$

(c) The total number of comparisons to merge the three sorted lists is:

$$\frac{2n}{3} - 1 + n - 1 = \frac{5n}{3} - 2$$

2. Show the recursion tree with at least 3 top levels and 2 bottom levels.

Solution:



3. What is the recursion equation?

Solution:
$T(n) = 3T\left(\frac{n}{3}\right) + \left(\frac{5}{3}\right)n - 2$

4. Solve the recursion equation using the recursion tree approach. Exactly how many comparisons does this modified merge sort algorithm use in the worst-case? Show your work.

Solution:

$$
\begin{aligned}
T(n) &= \sum_{i=0}^{lg_3 n-1} 3^i\left(\frac{5}{3}\cdot\frac{n}{3^i} - 2\right) + T(1), \qquad T(1) = 0 \\
&= \frac{5}{3}n\sum_{i=0}^{lg_3 n-1} 1 - 2\sum_{i=0}^{lg_3 n-1} 3^i \\
&= \frac{5}{3}n\Big[lg_3 n - 1 + 1\Big] - 2\left[\frac{3^{lg_3 n}-1}{3-1}\right] \\
&= \frac{5}{3}nlg_3 n - n + 1
\end{aligned}
$$

5. What is the value of T(3)?

Solution:

$$\begin{aligned} T(3) &= 3T\left(\frac{3}{3}\right) + \frac{5}{3} \cdot 3 - 2 \\ &= 3 \cdot 0 + 5 - 2 \\ &= 3 \end{aligned}$$

6. In terms of runtime, would this modified Merge sort be faster or slower than the regular algorithm?

Solution: The exact number of comparisons for the modified merge sort are:

$$T(n) = \frac{5}{3} n lg_3 n - n + 1$$

$$lg_3 n = \frac{lgn}{lg3}$$

$$T(n) = \frac{5}{3} n \frac{lgn}{lg3} - n + 1 = 1.05 nlgn - n + 1$$

There is no significant difference between this and the regular merge sort algorithm.

Challenge Problem (Not graded). You are given an array of n distinct unsorted numbers. Write an algorithm pseudocode or a description of the approach in concise english that finds the second smallest number in this array using at most $n + lg_2 n - 2$ comparisons.

Solution:

Hint: Find the smallest number in $n - 1$ comparisons.
Then use the tournament approach to find the second smallest from the losers (lgn of them) in the first round in $lgn - 1$ comparisons.