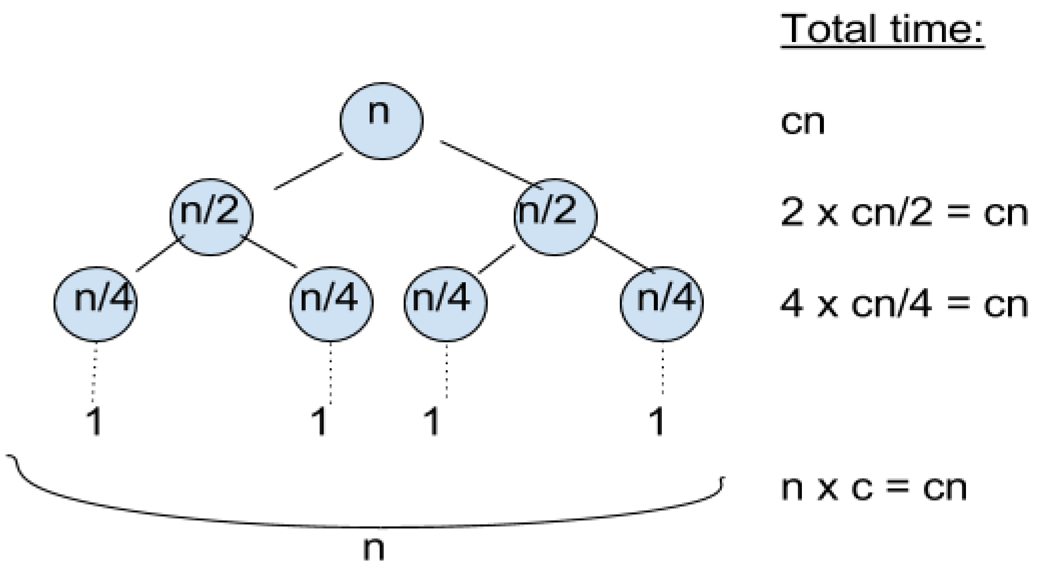
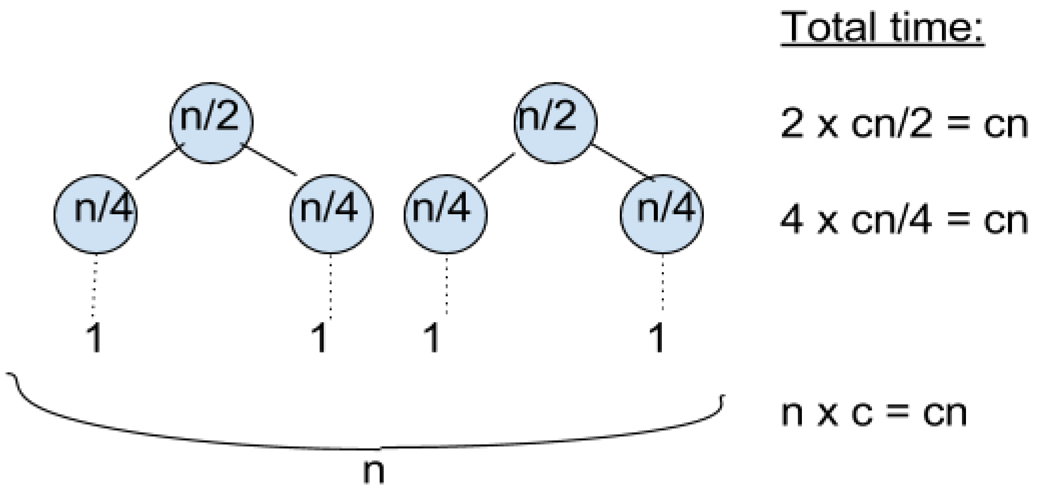
Nigel Decontie
V00853112

CSC 226 Assignment 1

1. Consider the fastest practical comparison based sorting algorithms. The running time would be Ω(nlogn) for the same reason it would be that even with the added information. Separating the elements into two lists immediately would be linear time. This allows us to initially call the sorting algorithm on 2 x (n/2) lists. So instead of:



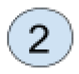
we have:

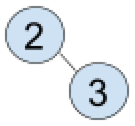


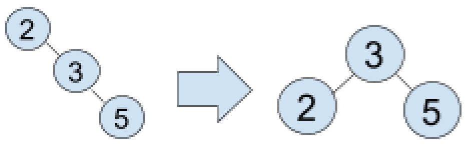
Eventually, we get to the base cases, of which there are n. Each are constant time. Now the total time would be the sum of all the levels = *L x cn. L = lg(n) + 1* (ex. *n = 8, lg(8) + 1 = 4 l*evels*; 8, 4, 2, 1*) so *lg(n) x n.* Removing the initial call on the list of size n simply results in one less level: *L - 1*. So (*L - 1*) x n = ((*lg(n) + 1*) - 1) x *n* = nlg(n). Therefore, the time complexity is still Ω(nlogn).

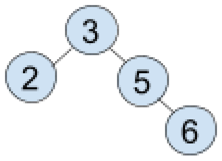
2.
   a. Groups of 5 (or 7) work because the list of medians is never more than 30% and so the recursive calls are never called on more than 70% of the list. So we get a time complexity of $T(2n/10) + T(7n/10) + cn$. Since $2n/10 + 7n/10 < 1$, we get a linear running time as the worst case.
   b. For groups of 3, the medians comprise 2/6, giving a 4/6 split resulting in $T(n/3) + T(2n/3) + cn$. So $n/3 + 2n/3 = n$, reducing the time complexity to $O(n\log n)$.

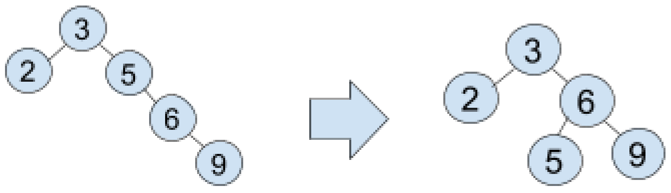
3. Inserting in order 2, 3, 5, 6, 9, 8, 7, 4, 1:

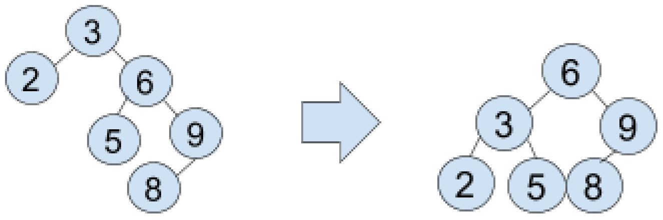
   a. 

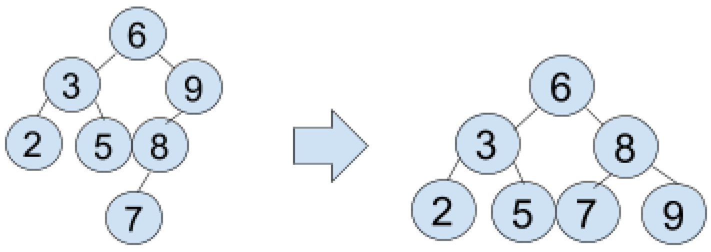
   b. 

   c. 

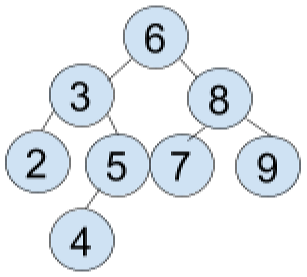
   d. 

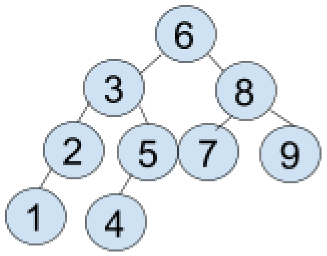
   e. 

   f. 

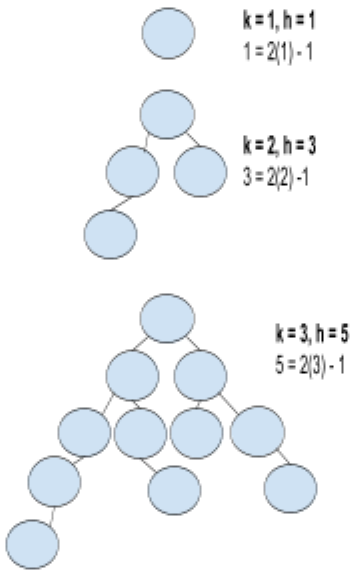
   g. 

Nigel Decontie
V00853112

h. 

i. 

4. Every time you add 2 new levels, the maximum height can become +1 more than the level from the closest leaf to the root. Every new k allows an increase in height of 2. That's where the 2k comes from. However they are initially equal at 1, so we get h = 2k - 1.


k = 1, h = 1
1 = 2(1) - 1
k = 2, h = 3
3 = 2(2) -1
k = 3, h = 5
5 = 2(3) - 1