Mamoutou Sangare

V00010526 CSC 226

Assignment 1

1- Show that any comparison based sorting algorithm still requires Ω(n log n) time to sort $X_1, x_2 \ldots, x_n$ even if it is given this additional information

Let the input be a sequence of n numbers $x_1, x_2 \ldots, x_n$.
Using a comparison based sorting, the output should be a permutation of $x_1, x_2 \ldots, x_n$ such that $x_1 \leq x_2 \leq x_2 \ldots \leq x_n$. We know that the number of possible permutation of the size n is n! .
We can use a decision tree to represent our comparison based sort. The execution of the sorting algorithm corresponds to finding a path from the root of the decision tree to a leaf since a leaf represents the ordering. Therefore, the decision tree will have n! leaves.
Now we claim: The height of a decision tree with n! leaves is Ω(n log n)
Proof:
Let h be the maximum number of comparisons in a sorting algorithm. The maximum height of the decision tree would be h. A tree with maximum height h has at most 2^h leaves.

$$\text{Leaves} \leq 2^h$$
$$n! \leq 2^h$$
$$\log_2(n!) \leq \log_2 2^h$$
$$\log_2(n!) \leq h$$

Since we know that $\log_2(n!) = O(n \log n)$ thus, we can conclude that $h = \Omega(n \log n)$
So a comparison based sorted requires Ω(n log n). Now, with this additional we can say that the input elements are partially sorted since x1 < x3 < x5 . . . and x2 < x4 < x6 . . . (input example: 1,4,2,5,3,6 . . .). Since the elements are partially sorted, we can think about a sorted algorithm which can minimize the number of comparison so that we can improve the running time. Therefore, we can use insertion sorted to reduce the number of comparison. With this approach, we reduce the number of comparison but we still have to swap (4,2 output of input example) the elements. Since we compare elements and swap them when we must do so to sort properly, the algorithm doesn't run on linear time. Thus, with this additional information, the sorted will still be Ω(n log n).

.2- Show that the modified algorithm of LinearSelect does not run in O(n) time if we use groups of 3
We will use a prove by contractiction to show that the LinearSelect algorithm does not run in O(n) time if we use groups of 3. Let's assume that the LinearSelect algorithm runs in O(n)time if we use groups of 3. The recurrence $T(n) = T(n \div 3) + T(2n \div 3) + 5n$
Let's find a k such that T(n) = kn.

Guess T(n) = Kn

T(n) = T($n \div 3$) + T($2n \div 3$) + 5n

Kn = (kn $\div$ 3) + (2kn $\div$ 3) + 5n

3 kn = kn + 2 kn + 15 n

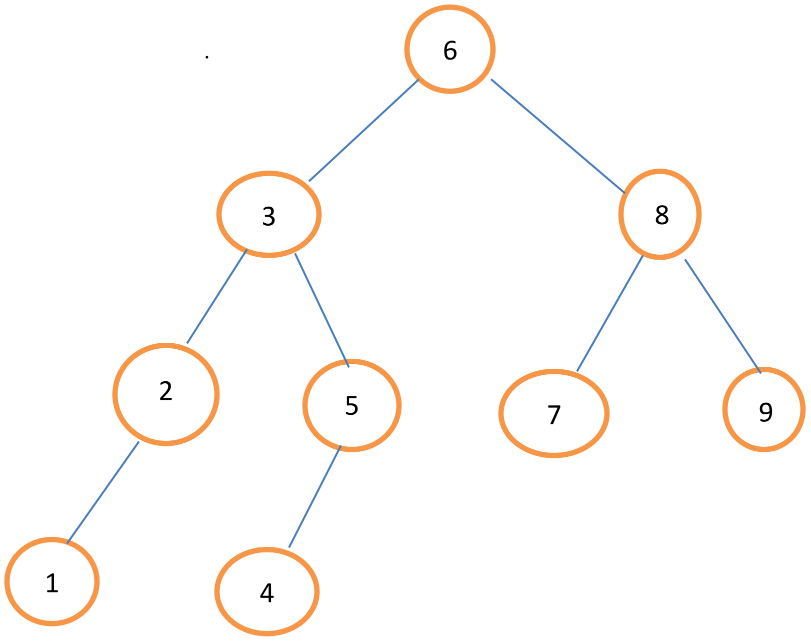
3 kn = 3 kn + 15 n

3 kn – 3kn = 15 n

0 = 15 n which means n must be 0. We can clearly see that there is not such a k such that kn = T(n) for n≠ 0. Since n ≠ 0, we conclude that there is not a natural number k such that T(n) = kn, therefore the LinearSelect does not run in O(n) time if we use groups of 3.

3 - AVL construction with the following keys in the given order: 2, 3, 5, 6, 9, 8, 7, 4, 1

The nodes which caused rotation are 2,5,6 and 9



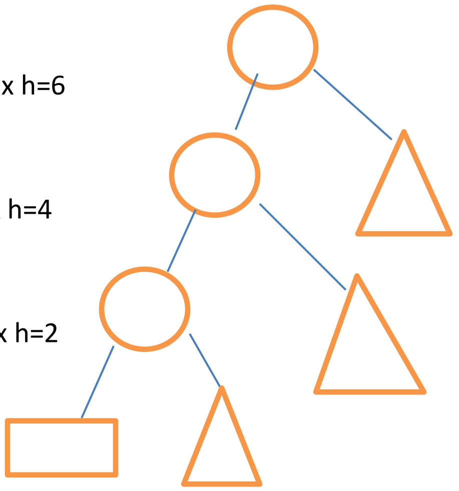
4 - Show that the height of this AVL tree T is at most 2k-1. Let draw the tree to find a pattern starting at k (bottom to go up)

Level k-3
(h = 5 or h=6) Max h=6

Level k-2
(h=3 or h=4) Max h=4

level k-1
(h=1 or h=2) Max h=2

level k
h = 0



If we continue until k becomes 0, we can see that the pattern for max height will be 0,2,4,6, 8,10 . . . 2k. Therefore max h $\leq$ 2k. Because the max height alternates between 2k-1 and 2k, we can say max h $\leq$ 2k-1 as well.