CSCI-GA.1170-001/002 Fundamental Algorithms October 14, 2015

Solutions to Problem 1 of Homework 5 (8 points)

Name: GOWTHAM GOLI (N17656180) *Due: Tuesday, October 13*

Consider a sorted array A of n elements and two integers x and y not in the array, with $x \leq y$. A comparison based algorithm computes how many elements in A are less than both x and y, how many elements are between x and y and how many are bigger than both x and y. What is the best lower bound (precise answer, not asymptotic) you can prove (using the decision tree technique) for the time complexity of the algorithm?

Solution:

Each leaf node in the decision tree can be viewed as a sequence of elements of A that are divided into 3 parts.

- The first part corresponds to all those elements of A that are lesser than both x and y (could be empty)
- The second part corresponds to all those elements of A that are in between x and y (could be empty)
- The third part corresponds to all those elements of A that are greater than both x and y (could be empty)

Therefore the number of leaf nodes in the decision tree will be equal to the number of ways in which we can place two partitions among the n sorted elements of A which can be viewed as selecting two elements from $n+2$ elements $= \binom{n+2}{2} = \frac{(n+2)(n+1)}{2}$. Let it be l

The lower bound on the height of the decision trees in which each permutation appears as a reachable leaf is a lower bound on the running time of any comparison sort algorithm. So let h be the height of the binary tree and since a binary tree of height h can have no more than 2^h leaves, we have

$$l \leq 2^h \implies \frac{(n+2)(n+1)}{2} \leq 2^h \implies h \geq \log_2 \frac{(n+2)(n+1)}{2} = \log_2(n+2)+\log_2(n+1)-1 \implies h = \Omega(\log n)$$

Therefore, the precise lower bound is $\log_2(n+2) + \log_2(n+1) - 1$

□