
Azoacha Forcheh, 20558994
CS 234 Fall 2017
Assignment 02, Problem 5

Consider a list **A** of length n that is already sorted, and set *low* = 0 and *high* = n-1 in the quick sort call. The sublist would then be divided into 2 sublists of lengths 0 and n-1, and the (n-1)-length sublist will be divided similarly (into sublists of lengths 0 and n-2), and this will continue recursively until we get to 2sublists of length 0 and 1, i.e. there will be n-1 recursive calls to recQuickSort within the main call in quickSort.

This means that *partition* will then have to be run on n sublists of decreasing (by 1) length. Note that since *low* = 0 and *high* = n-1, the left marker is shifted all the way to the right of the list, so the inner while loop performing this shift takes *O(n-i)* time for the ith recursive call and the inner while loop for the shifting of right marker only takes O(1) time as the right marker will then be smaller than the left one. The outer while loop will terminate after this, and hence for the ith recursive call to recQuickSort, partition will take O(n – i) time and so recQuickSort will also take O(n - i) time.

Hence, since we have a total of n calls to recQuickSort in the worst case, we get a worst case time for quick sort of:

$$\sum_{i=1}^{n}(n-i) = n(n) - \sum_{i=1}^{n} i = n^2 - n = O(n^2)$$