MCS 401 Spring 2020
Homework 3

February 28, 2020

1. Problem 1 [Exercise 7-2-3, page 178]:

   Since the array is already sorted in decreasing order, the pivot element is less than all the other elements (if we use the textbook implementation, i.e., the pivot is the first element of the array). The partition step always takes $\Theta(n)$ time. Then, what remains is one subproblem of size $n-1$. Therefore, we would have
$$f(n) = f(n-1) + \Theta(n).$$
   We know that such a recurrence has a solution $f(n) = \Theta(n^2)$.

2. Problem 2 [Problem 7-4-4, page 184]:

   If we use the expression for $E[X]$ on page 184, then we have:
$$E[X] = \sum_{1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \geq \sum_{1}^{n-1} 2(\ln(n-i)-1) =$$
$$2((1-n) + \ln\left(\prod_{i=1}^{n-1}(n-i)\right)) = 2(\ln((n-1)!) - n + 1) =$$
$$2(\Theta((n-1)\ln(n-1)) - n + 1) = \Theta(n \ln n)$$
   But then, $E[X] = \Omega(n \ln n) = \Omega(n \lg n)$, as well.

3. Problem 3 [Exercise 8-1-3, page 194]:

   Suppose that there exists $c > 0$, such that for every $n \geq k$, at least half of the inputs of length n have depth at most cn. However, there are less than 2^{cn+1} elements in the tree of depth at most cn. We must have that $2^{cn+1} \geq \frac{1}{2}n!$, but for big enough values of n, we have $\log(\frac{1}{2}n!) = \Theta(n \log n)$ and $\log(2^{cn+1}) = \Theta(n)$. Contradiction.

4. Problem 4 [Exercise 8-2-2, page 196]:

   Suppose that $A[i] = A[j] = k$, for $i < j$, where A is the initial array. Since $j > i$, the loop on lines $10-12$ will examine $A[j]$ first, before examining $A[i]$. When this happens, A[j] will be placed in some position $m = C[k]$ in the final output array B (line 11) and on line 12, $C[k]$ is decremented. The elements of C are never incremented, so we are guaranteed that when the for loop examines $A[i]$, it will be placed at position $C[k] < m$. This proves that COUNTING-SORT is stable.

5. Problem 5 [Exercise 9-3-6, page 223]

   The idea is to use Divide and Conquer and the SELECT procedure which works in linear time. Assume that n and k are powers of 2. Find the $\frac{k}{2}$-th quantile in time $O(n)$ using SELECT. What remains is to find the $\frac{k}{2}$ different quantiles in the left and in the right subarray which are both of length $n/2$. Let $T(n)$ denote the time complexity of the proposed algorithm on input of size n. Then, $T(n/k) = O(1)$ and $T(n) \leq cn + 2T(n/2)$, for some constant $c > 0$. Therefore, we have:

$$\begin{aligned} T(n) \leq cn + 2T(n/2) \leq cn + 2(c(n/2) + 2T(n/4)) = \\ = 2cn + 4T(n/4) \leq 3cn + 8T(n/8) \leq \\ \leq \cdots \leq (\log k)cn + kT(n/k) = \\ = (\log k)cn + O(k) = \\ = O(n \log k). \end{aligned}$$