McGill University COMP251: Assignment 2

Worth 10%. Due October 1 at the beginning of lecture (10am)

Question 1 Suppose A is already sorted in increasing order. Prove that the running time of Quicksort on input A is $\Omega(n^2)$.

Question 2 Consider the following problem:

Input: An array $A[1, 2, \ldots, n]$ of distinct integers.

Output: The number of pairs (i, j) such that $i < j$ and $A[i] < A[j]$ (i.e., the number of pairs of elements in A that are in sorted order).

For example, on input $A = (1, 5, 3, 7, 2)$ the output is 6 (the pairs are $(1, 2)$, $(1, 3)$, $(1, 4)$, $(1, 5)$, $(2, 4)$, $(3, 4)$).

(a) Suppose that A is in increasing order. What is the output?

(b) Now design a algorithm that solves the problem using divide-and-conquer technique. Your algorithm must run in time $\mathcal{O}(n \ln n)$. (Hint: Review the Mergesort algorithm.)

(c) Use the Master Theorem to verify that the running time of your algorithm is $\mathcal{O}(n \ln n)$.

Question 3 A ternary heap is like a (binary) heap that we discuss in lecture, but (with one possible exception) non-leaf nodes have 3 children instead of 2 children.

(a) How would you represent a ternary heap in an array?

(b) What is the height of a ternary heap of n elements? Give a precise answer in terms of n.

(c) Give pseudo-code for Heapify3(A,i). This procedure assumes that the sub-trees rooted at the children of node i are ternary heap, but $A[i]$ might be smaller than some of its children (thus violating the heap property). The procedure makes the sub-tree rooted at node i become a ternary heap by letting the value at $A[i]$ "float down" in the ternary heap which is rooted at the largest child of node i. (This is a modified version of the Heapify(A,i) procedure.)

(d) Give pseudo-code for a sorting algorithm that uses Heapify3. The worst-case running time of the algorithm must be $\Theta(n \ln n)$.

(e) Verify that the worst-case running time of your algorithm is indeed $\Theta(n \ln n)$.

Question 4 Give an algorithm that, given an array A of n integers in the range 0 to k, preprocesses its input in such a way that any query about how many of the n integers fall into a range $[a..b]$, (i.e., how many i such that $a \leq A[i] \leq b$) can be answered in time $\mathcal{O}(1)$. Explain how such a query is to be answered. Your algorithm should use $\Theta(n + k)$ preprocessing time.