Foundations of Algorithms, Spring 2018: Homework 2

Due: Monday, February 12, 11:59pm

To start, work on the problems in the order: 1, 4, 3, 2, 5. We'll cover the necessary material over the next couple of days.

Problem 1 (10 points)

Consider the following divide-and-conquer algorithm that assumes a global array A of integers:

```
WHATDOIDO(integer left, integer right):
    if left == right:
        return (1, 1, 1)
    else:
        m = (left + right) / 2       (integer divide)
        (llstreak, lrstreak, lmaxstreak) = WHATDOIDO(left, m)
        (rlstreak, rrstreak, rmaxstreak) = WHATDOIDO(m+1, right)

        if A[m] == A[m+1]:
            maxstreak = max(lmaxstreak, rmaxstreak, lrstreak+rlstreak)
            if lmaxstreak == m - left + 1:
                lstreak = lmaxstreak + rlstreak
            else:
                lstreak = llstreak
            if rmaxstreak == right - m:
                rstreak = rmaxstreak + lrstreak
            else:
                rstreak = rrstreak
        else:
            maxstreak = max(lmaxstreak, rmaxstreak)
            lstreak = llstreak
            rstreak = rrstreak

        return (lstreak, rstreak, maxstreak)
```

Before running the algorithm, we ask the user to enter n integers that we store in the array A. Then we run `WHATDOIDO(0,n-1)`.

a) State the recurrence for $T(n)$ that captures the running time of the algorithm as closely as possible.

b) Use either the "unrolling the recurrence" (telescoping) or mathematical induction technique to find a tight bound on $T(n)$.

c) What does the algorithm do? Specify what the three returned values represent.

Problem 2 (8 points)

Use the Master Theorem to provide tight asymptotic bounds for the following recurrences. In each case, also specify the values of a, b, and $f(n)$, and indicate which case of the Master Theorem applies.

(a) $T(n) = 4T(n/2) + n^2$

(b) $T(n) = 2T(n/2) + \sqrt{n}$

(c) $T(n) = 7T(n/3) + n^2$

(d) $T(n) = 2T(n/8) + n^{1/3}$.

Problem 3 (14 points: 10 for implementation / 4 for writeup)

Albert Grithum teaches seven and eight year olds. Today is school picture day and everybody, including the teacher, has lined up in a single line for the class picture. Initially everyone has lined up in a random order. The photographer wants the following arrangement from left to right: first, all of the seven year olds, in order of increasing height. Next, Mr. Grithum in the middle. Last, the eight year olds in decreasing order of height. The only adjustment allowed is a swap, in which two *neighboring* people swap their positions. Design an $O(n \log n)$ algorithm that computes the minimum number of swaps necessary to get the class into the desired order.

Problem 4 (20 points: 15 for implementation / 5 for writeup)

Given is a large paper with n different points with coordinates $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. Notice that by folding the paper along a single line we can make some of the points align. For example, if the points are (1,2), (2,1), and (4,3), then if we fold along the line going through the origin at the 45 degree angle, the points (1,2) and (2,1) will align. Design an $O(n^2 \log n)$ algorithm that finds the maximum number of pairs of points that can be aligned.

Problem 5 (20 points: 15 for implementation / 5 for writeup)

(a) We are given an array of integers $A[0..n-1]$. We would like to determine whether there exists an integer x that occurs in A more than $n/2$ times (i.e., whether A has a majority element). Design an algorithm that runs in $O(n)$ time and argue its correctness and running time estimate.

Example: For $A = [3, 1, 2]$, the answer is NO. For $A = [3, 1, 3]$ the answer is YES.

(b) We are given an array of integers $A[0..n-1]$. We would like to determine whether there exists an integer x that occurs in A more than $n/3$ times. Design an algorithm that runs in $O(n)$ time and argue its correctness and running time estimate.

NOTE: remember, you are not allowed to search the internet for answers. You will not get any credit if you implement "Moore's voting algorithm" or variations of this algorithm for either part of your solution.