1. Solve each of the recurrences below, and give the best $O(\cdot)$ bound you can for each of them:
   (a) (2 points) $T(n) = 4T(n/4) + n$.
   (b) (2 points) $T(n) = 4T(n/4) + 1$.
   (c) (2 points) $T(n) = T(n-1) + n$.
   (d) (4 points) $T(n) = T(n/3) + T(n/2) + \sqrt{n}$.
   (e) (4 points) $T(n) = T(\sqrt{n}) + 4$.
   (f) (6 points) Suppose we have $T(n) = 3T(n/2) + g(n)$. Analyze the behavior when (a) $g(n) = n^2$ (b) $g(n) = n$, and (c) $g(n) = n^{\log_2 3}$. (This will illustrate the so-called *master theorem.*)

2. (5 points) Let $A[0 \dots n-1]$ be an array of integers, and let $M = \max_i A[i] - \min_i A[i]$. Describe an algorithm that sorts the array in time $O(n + M)$. [Source: Dasgupta et al. textbook]

3. (10 points) Suppose we are given two **sorted** arrays $A[0 \dots N-1]$ and $B[0 \dots N-1]$, and an integer k. Design an algorithm to find the kth smallest element in the union of the two arrays in time $O(\log n)$. [You get partial credit if you give an algorithm with running time $O(\log^2 n)$.]

4. Let $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ be n points in the 2D plane. The *Manhattan distance* between points $p = (x_1, y_1)$ and $q = (x_2, y_2)$ is defined to be the quantity

$$d(p, q) = |x_1 - x_2| + |y_1 - y_2|.$$

(I.e., it is the "walking distance" if we only have roads along the axes – as is the case in Manhattan, or say Salt Lake).

Given n points (all of whose x, y coordinates are different, for simplicity), our goal is to find the two *closest* points, in Manhattan distance. Consider the following algorithm:

   1. Find an x such that precisely $n/2$ points have $x_i \le x$ (i.e., a vertical line that divides the points into two). Let $L = \{(x_i, y_i) \mid x_i \le x\}$, and $R = \{(x_i, y_i) \mid x_i > x\}$.
   2. Now, recurse on sets L and R, and find the closest pairs in these two subsets of points. Let d be the smaller of the two distances.
   3. It remains to be seen whether there is a point in L and a point in R that are less than distance d from each other. Let S be the set of points whose x-coordinates are in the interval $[x - d, x + d]$. Note that only these points matter.
   4. Sort the points in S now by their y coordinates, and for each point, compute the distance of the point to the **next 13** points in the sorted list. Let d' be the minimum distance computed in this process.
   5. Return $\min(d', d)$.

Write the answers to the following as functions of k:

   (a) (1 point) Suppose we skip Steps 3-4 in the algorithm, and simply return d. Give an instance in which the answer output is wrong.
   (b) (5 points) Prove that the algorithm outputs the minimum distance. [*Hint:* prove the following statement: suppose we have 7 points $p_1, \dots, p_7$ in a $d \times d$ square, then there exist two of them that have a distance strictly smaller than d.] (You get partial credit for proving the hint.)
   (c) (4 points) Describe briefly how to implement each step of the algorithm, and prove that the run time satisfies the recurrence:

$$T(n) = 2T(n/2) + O(n \log n).$$

   Show that it leads to the bound $T(n) \le O(n \log^2 n)$.

[Source: Dasgupta et al. textbook]

5. Suppose $A[0 \dots n]$ is an array of n distinct integers. We saw in class that to find the median of $A[0 \dots n-1]$, it helps to first find a *near median.*[1] If $T_{\text{near-median}}(n)$ is the time taken for this step, we saw in class that

$$T_{\text{median}}(n) \le T_{\text{near-median}}(n) + T_{\text{median}}(3n/4) + O(n).$$

In this exercise, we show that $T_{\text{near-median}}(n) \le T_{\text{median}}(n/5) + O(n)$. We saw in class that this leads to an overall bound of $O(n)$ for $T_{\text{median}}(n)$.

Consider the following procedure:

1. Divide the elements of A into groups of 5, arbitrarily. Let us call the sets obtained $B_1, B_2, \dots, B_{n/5}$.
2. For each i, $1 \le i \le n/5$, sort the elements of B_i in increasing order.
3. Form the array C, consisting of the *middle* (i.e., the third smallest) elements of the B_i's.
4. Let M be the median of the numbers C.

For this procedure, prove the following:

(a) (2 points) Its running time is $T_{\text{median}}(n/5) + O(n)$.

(b) (3 points) The element M is a near median for the array A. (I.e., prove that there exist at least $n/4$ elements of A that are $\le M$, as well as at least $n/4$ elements of A that are $\ge M$.)

[1] Formally, the near median is an element $A[i]$ with the guarantee that there are at least $n/4$ elements of A that are $\ge A[i]$, and at least $n/4$ elements of A that are $\le A[i]$.