Homework 3
Due Date: February 15, 2018 at 11:59 PM on Blackboard.

Question 1
Consider a set \( p_1, \ldots, p_n \) of 3-dimensional points \( p_i = (x_i, y_i, z_i) \) and define the distance between two points \( p_i, p_j \) to be \( d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \).

Part 1 (15 points) Suppose we sort \( p_1, \ldots, p_n \) by x-coordinate and split the set evenly into \( Q \) (the first \( \lfloor n/2 \rfloor \) elements in the sorted list) and \( R \) (the remaining \( \lceil n/2 \rceil \) elements). Let \( q_0^*, q_1^* \) (resp. \( r_0^*, r_1^* \)) denote the closest pair of points in \( Q \) (resp. \( R \)) and let \( \delta = \min \{ d(r_0^*, r_1^*), d(q_0^*, q_1^*) \} \) denote the minimum distance between these two pairs. Given a point \( p_i = (x_i, y_i, z_i) \) show that there are at most 256 points in the region
\[
R(p_i, \delta) = [x_i - \delta, x_i + \delta] \times [y_i - \delta, y_i + \delta] \times [z_i - \delta, z_i + \delta].
\]
(Note: You will likely be able to establish a better constant than 256. The key is that the upper bound is \( O(1) \). To simplify analysis you may assume that all points have a unique x value.)

Part 2 (10 points) Develop an \( O(n \log^2 n) \) time algorithm to find the closest pair of distinct points \( p_i \) and \( p_j \) minimizing \( d(p_i, p_j) \). You should argue that your algorithm is correct, and prove that the running time is \( O(n \log^2 n) \). Hint: Partition space into cubes of length \( \delta/2 \) aligned with the plane separating \( Q \) and \( R \) and associate each point \( p \) with the corresponding cube containing that point. You may assume the existence of a function \( \text{label}_{\delta/2}(p) \) which outputs the index of the cube containing the point \( p \), and you may assume that the cost to query this function is \( O(1) \).

Question 2
Analyze the asymptotic complexity of each of the following functions (e.g., if \( T(n) = 2T(n/2) + n \) then \( T(n) \in \Theta(n \log n) \)). You should aim to provide tight upper and lower bounds on \( T(n) \) and you should prove your answers are correct e.g., using induction or unrolling. In each case you should assume that \( T(n) = 1 \) for \( n < 3 \).

- \( T(n) = 3T(n/2) + n \) for \( n \geq 3 \).
- \( T(n) = T(n/3) + T(2n/3) \) for \( n \geq 3 \).
- \( T(n) = T(n/2) + \log_2 n \) for \( n \geq 3 \).
- \( T(n) = T(n - 1) + n^2 \) for \( n \geq 3 \).
- \( T(n) = T(n - \sqrt{n}) + n^{1/3} \) for \( n \geq 3 \).
Question 3

In a marketplace, there are $n$ sellers $s_1, \ldots, s_n$ selling oranges. You may buy at most $k$ oranges from each of the sellers. Depending on how many oranges you buy, the sellers may give you some discount or charge a bit extra. If you buy $j$ oranges from seller $s_i$, you need to pay $c(i, j)$ to that seller. You may not assume any logical relations between prices of different numbers of oranges, even from the same seller.

a. Your goal is to buy exactly $M$ oranges, for some $0 \leq M \leq kn$, while spending the least amount of money. Design an $O(n^2k^2)$-time dynamic programming algorithm for the task. Prove its correctness and analyze its running time.

b. You love eating oranges so much that all the sellers have become your friends. They will feel bad if they see you are buying too many oranges from neighboring sellers. Formally, if you buy $m_i$ oranges from seller $s_i$, that seller will feel bad if you buy $m_i + 2$ or more oranges from either of the sellers $s_{i+1}$ or $s_{i-1}$ (if they exist). Design an $O(n^2k^2)$-time dynamic programming algorithm to minimize your cost of buying $M$ oranges, where $0 \leq M \leq kn$, under this restriction. There is no need to prove the algorithm’s correctness or analyze its running time.

Question 4

In a distant country, a wide river crosses the country east to west. There are $n$ cities $v_1, \ldots, v_n$ on the north bank of the river, and $m$ cities $u_1, \ldots, u_m$ on its south bank. We are also given a set $E$ of pairs $(v_i, u_j)$ of cities that we would like to be connected with bridges. But since the country is not very advanced technologically, the bridges that we build cannot cross each other (though we do allow several bridges to connect one city to several other cities). We clarify what it means for a pair of bridges to cross. Bridge $v_i u_j$ crosses bridge $v_{i'} u_{j'}$ if $j' < j$ and $i' < i$ or if $i < i'$ and $j' > j$. Our goal is to connect as many pairs of cities from $E$ as possible (a pair $(v_i, u_j)$ of cities is considered connected iff there is a bridge between them). Design an efficient ($O(mn)$ time) algorithm to solve this problem, prove its correctness and analyze its running time. You can receive partial credit for a polynomial time solution.