1. (2.5 points) Given a set of $n$ red and $m$ blue points in the plane, called $R$ and $B$. We wish to compute a line that separates the red and blue points, or determine that none exists. Assume no three points of $R \cup B$ lie on a line. A separating line may have red and/or blue points on it.

To solve this problem, we can check first whether a vertical line exists that separates $R$ and $B$. This is easy to do in linear time, so it remains to determine if such a line exists when no vertical line works. We can solve this by considering two subproblems: a separating line with $R$ above it and $B$ below it, or vice versa. Obviously, if we have an algorithm for one of these problems, we have solved the whole problem.

Therefore we restrict ourselves in (a.) to the following problem:
Find a separating line that has \( R \) above it and \( B \) below it (knowing that no vertical separating line exists), or decide that none exists.

(a.) Give an algorithm that solves the problem (the one in bold) by randomized incremental construction in \( O(n + m) \) expected time (where the expectation is over the random choices made by the algorithm only). To this end, decide how to initialize, put all remaining points of \( R \cup B \) in random order, and incrementally insert them.

Give the algorithm and analyze the expected worst-case running time. Hint: Find the largest slope separating line (maximal \( h \) for a line \( y = hx + c \)).

(b.) If no separating line exists, we can try to find the line with the smallest number of points of \( R \) below it and the smallest number of points of \( B \) above it. In particular, we want the line that minimizes the total number of points on the wrong side.

We can solve this problem using the theory of Chapter 8. First, dualize the problem and state exactly what we need to compute.

(c.) Show that with the help of arrangements and levels in arrangements, the problem can be solved in \( O((n + m)^2) \) time in the worst case. Give the steps of the algorithm and give the running time analysis.

2. (2 points) We consider a variant of kd-trees where internal nodes are not binary but ternary. In other words, an internal node has three children: left, middle, and right. If the node is on an even level, then it stores two vertical lines that split the point set in three equal-size subsets: one to the left of both lines, in the left subtree; one in between both lines, in the middle subtree; and one to the right of both lines, in the right subtree. On an odd level, the analogous thing happens with two horizontal lines. When the set to be stored has only one or two points, then we store the one or the two points in a single leaf.

Let \( P \) be a set of \( n \) points in the plane, where no two points have the same \( x \)-coordinate or \( y \)-coordinate. Assume that \( P \) is stored in a ternary kd-tree.

(a.) What is the maximum depth of a ternary kd-tree that stores \( P \)? Give a precise answer, without using \( O(\ldots) \)-notation.

(b.) Explain whether and how the query algorithm and the kd-tree construction algorithm must be changed for ternary kd-trees.

(c.) Analyze the worst-case query time for a rectangular range query in a ternary kd-tree that stores \( P \).

3. (2 points) Suppose range tree on a set of \( n \) points in the plane does not have any associated structures on the topmost \( \log \log n \) levels. So only nodes with depth between \( \log \log n \) and \( \log n \) (the leaves) have an associated structure.

(a.) Show that range queries can still be answered in \( O(\log^2 n) \) time.

(b.) Show that the storage requirements are still \( \Theta(n \log n) \).

4. (2.5 points) A set \( P \) of \( n \) points in the plane gives rise to \( \binom{n}{2} = n \cdot (n - 1)/2 \) pairs of points \( p, q \) (with \( p \neq q \)) from \( P \), each giving a distance between the points of that pair. A Voronoi diagram of \( P \) can be used to find the closest pair of points in \( P \). Namely, if \( p, q \in P \) are the closest pair among all pairs of points in \( P \), then \( p \) and \( q \) have Voronoi
cells that are adjacent. Therefore, after computing the Voronoi diagram of $P$, one can determine the closest pair in only $O(n)$ time by a straightforward traversal of the DCEL that stores the Voronoi diagram. (By the way, the fastest algorithm to compute the closest pair takes $O(n \log n)$ time, and no worst-case faster algorithm exists.)

Assume that no four points of $P$ are co-circular, and no two pairs of points have the same distance.

(a.) Give a rigorous but short proof of the fact that both the closest pair and the second-closest pair of points in $P$ have adjacent cells in the Voronoi diagram (since we consider all $\binom{n}{2}$ pairs of points, the two closest pairs may have a point in common).

(b.) Give an example showing that the third-closest pair of points in $P$ need not have Voronoi cells that are adjacent. Give exact coordinates of the points that you use in the construction, and a well-drawn figure of the situation.

(c.) Suppose that we want to compute the $k$ closest pairs in $P$, for some small value of $k$. Assume that $k$ is a constant. Give an algorithm that computes the $k$ closest pairs in $P$ in $O(n \log n)$ time in the worst case (since $k$ is a constant, factors with $k$ do not appear in the bound). Prove your time bound.