1. (10 pts) Define the *stabbing* number of a triangulated polygon as the maximum number of diagonals intersected by any line. Show that every convex polygon with \( n \) vertices has a triangulation with stabbing number \( O(\log n) \).

2. (20 pts) On \( n \) parallel tracks, \( n \) trains are moving with constant speeds \( v_1, v_2, \ldots, v_n \). At time \( t = 0 \), the trains are at positions \( k_1, k_2, \ldots, k_n \). Describe an \( O(n \log n) \) time algorithm to detect all trains that at some moment in time are leading.

3. (20 pts) Let \( S_1 \) be a set of \( n \) disjoint horizontal line segments in the plane, and let \( S_2 \) be a set of \( m \) disjoint vertical line segments in the plane. Describe an \( O((n + m) \log(n + m)) \) time algorithm to count the number of intersections among \( S_1 \cup S_2 \). (Note that the number of intersections can be \( nm \), which is much larger than the time allowed for your algorithm.)

4. (25) Analyze the performance of the \( kd \)-tree data structure in \( d \) dimensions, where we treat \( d \) to be a constant.

   (a) First, describe the procedure for constructing a \( kd \)-tree for a set of \( n \) points in \( d \)-dimensional space. Prove that data structure uses \( O(n) \) space, and can be constructed in \( O(n \log n) \) time.

   (b) Describe the query algorithm for performing a \( d \)-dimensional orthogonal range query using this \( kd \)-tree, and prove that the worst-case query time complexity is \( O(n^{1-1/d} + A) \), where \( A \) is the size of the answer.

5. (25 pts) Let \( S \) be a set of \( n \) axis-parallel (possibly intersecting) rectangles in the plane. Preprocess this set for the following type of queries: *given a query rectangle \( R \), report all rectangles of \( S \) that lie entirely inside \( R \).*

   Describe a data structure that solves this problem in worst-case query time \( O(\log^4 n + k) \), using \( O(n \log^3 n) \) storage, where \( k \) is the number of rectangles in the reported answer. Include proof of correctness and analysis of query complexity.

   (Hint: Transform this problem to an orthogonal range searching problem in a higher dimensional space.)