1. (20 pts) Many applications require range searching over non-point objects. The following formulation considers one such case. Be sure to prove the correctness of your scheme and analyze the space and query complexity of your data structure.

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.

(Hint: Transform to higher-dimensional range searching over points.)

2. The $kD$-Tree data structure can also be used for non-rectangular queries. For instance, the query is answered correctly if the range is a triangle.

(a) (20 pts) Show that the query time for range queries with triangles is $\Omega(n)$ in the worst-case, even if the range contains no data points. Give an explicit construction that works for any value of $n$.

(b) [Extra Credit: 20 pts] Suppose our queries are restricted to triangles whose edges are horizontal, vertical, or have slopes $+1$ or $-1$. Develop a linear-space data structure that can answer such range queries in $O(n^{3/4} + k)$ worst-case time, where $k$ is the size of the output.

3. (20 pts) Let $H$ be a set of $n$ disjoint horizontal line segments in the plane, and let $V$ 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 $H \cup V$. (Note that the number of intersections can be $nm$, which is much larger than the time allowed for your algorithm.)

4. (20 pts) Given $n$ (possibly overlapping) rectangles in 2D, propose an $O(n \log n)$ time algorithm to compute the area of their union? (Hint: Use plane sweep algorithm, and 1-dimensional segment unioning algorithm.)