CS 498TC, Spring 2018
Timothy Chan

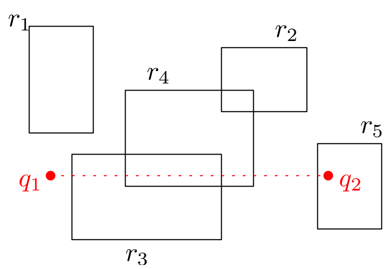
Assignment 5 (due April 27 Friday 2pm (in class))

You may work in a group of at most 3 students. Carefully read https://courses.engr.illinois.edu/cs498tc3/policies.html and https://courses.engr.illinois.edu/cs498tc3/integrity.html.

1. [*12 pts*] Given a simple polygon P with n vertices and a convex polygon R with a constant number of vertices, we want to determine whether R can be placed inside P by translation, i.e., whether there exists a translated copy of R that is contained in P. Show that this problem can be solved in $O(n \log^2 n)$ time. [Hint: use a known approach from class. Another hint: consider the complement of P.]

2. [*8 pts*] Consider the following problem: store a set S of n (axis-aligned) rectangles in 2D so that for a given query horizontal line segment $\overline{q_1 q_2}$, we can quickly report all rectangles $r \in S$ that $\overline{q_1 q_2}$ completely cuts across (i.e., $\overline{q_1 q_2}$ intersects both the left and right side of r). Give an efficient data structure for this problem.

   [Hint: reduce the problem to orthogonal range searching. How many dimensions?]



$\overline{q_1 q_2}$ completely cuts across r_3 and r_4 (but not r_5)

3. [*25 pts*] Recall that in d dimensions, the L_1-*distance* (or *Manhattan distance*) between two points $p = (p_1, \ldots, p_d)$ and $q = (q_1, \ldots, q_d)$ is defined as $|p_1 - q_1| + \cdots + |p_d - q_d|$. Our problem is to store a set P of n points in $\mathbb{R}^d$ so that given a query point q, we can find an L_1-nearest neighbor of q, i.e., a point $p \in P$ with the smallest L_1-distance to q.

   (a) [*7 pts*] Consider the problem for $d = 2$ in the special case when all query points q lie on the y-axis ($x = 0$). Show that this special case can be solved by a data structure with $O(n)$ space and $O(\log n)$ query time. [Hint: sort points according to y-coordinates, and maintain the minimum of some function over prefixes/suffixes...]

   (b) [*12 pts*] Now solve the general problem for $d = 2$ using $O(n \log n)$ space and $O(\log^2 n)$ query time. [Note: do not use Voronoi diagrams; instead, adopt an approach based on range trees, using part (a) as a subroutine. For example, if q is to the left of $x = 0$ and P_{right} is to the right of $x = 0$, how can we compute the L_1-nearest neighbor of q in P_{right}?]

   (c) [*6 pts*] Generalize part (b) to obtain a data structure for any constant d with $O(n \log^{d-1} n)$ space and $O(\log^d n)$ query time.