

Solutions to Cs445 — Homework #8

Closest pair and convex hull

Due last day of class

This homework is optional — if you opt to submit it, its grade would replace the second lowest grade of your previous homeworks.

1. Run the algorithm studied in class for computing the convex hull on the “A” of the blue part of the logo of the University of Arizona. Your drawing does not have to be exact, but should be correct in terms of the order at which points are processed. Show the data structure you are using after each stage.
2. The following theorem can be proven using similar techniques to the ones we have used to prove the lower bound on sorting:

Element Uniqueness Theorem. Let $X = \{x_1 \dots x_n\}$ be a set of n real numbers, and let A be an algorithm that returns YES if and only if there are two elements $x_i, x_j \in X$ ($i \neq j$) such that $x_i = x_j$. Then there is an input X for which the running time of A is $\Omega(n \log n)$.

Show that an algorithm for finding the closest pair of points between a set of n given points in the plane runs in $\Omega(n \log n)$ time.

Answer:

Assume we are given an instance of the problem $X = \{x_1 \dots x_n\}$. We define the set of points $S = \{(x_1, 0), \dots, (x_n, 0)\}$ and apply the algorithm to find the closest pair. The distance between the closest pair is 0 iff two values x_i, x_j are identical.

3. Suggest an algorithm that accepts a set $P = \{p_1 \dots p_n\}$ and finds the three points $p_i, p_j, p_k \in P$ that the triangle that they define has the smallest perimeter among the perimeters of all $\binom{n}{3}$ triangles defined by triples of points of P . The algorithm should run in time $O(n \log n)$.

(The perimeter of a triangle is the sum of lengths of its edges)

Answer:

We define $d(S)$ to be the perimeter of the triangle with the smallest perimeter among all triangles whose vertices are in S . Our algorithm is a slight modification of the one studied in class for finding the closest pair. We divide the points into S_L and S_R , such that we have $n/2$ points on the left and $n/2$ on the right, and then find $d = \min(d(S_L), d(S_R))$. Next we construct a vertical strip of width $2d$ centered at the line separating S_L from S_R . We sweep this strip with a window of width $2d$ and height d as described in the algorithm in the slides, and check every point that enters the window with the triangles it defines, together with other points inside the window. Note that there is no need to check triangles defined by points outside the strip, and that the window contains at each instant only $O(1)$ points (why?).

4. Describe a version of the algorithm for computing the convex hull of a set of points, that use a queue, rather than a stack, and still runs in $O(n \log n)$.

Answer:

After sorting the points by their angle with p_0 , we create a simple polygon by connecting each vertex to its two neighboring vertices, and maintain this list as a doubly-connected link list. Note that this polygon might not be convex. Next we check each vertex from this list in its turn, and place in the queue the ones which define a concave angle. We then iteratively pick a concave vertex from the queue, for example the one at the head of the queue, and verify that it still makes a concave angle with its neighbors. If yes, we delete it, and insert its neighbors into the queue. The algorithm stops when the queue is empty.

5. Let P be a polygon, whose vertices $\{v_1, v_2 \dots v_n\}$ are given to you in the order they appear along its boundary, in a clockwise order. Suggest an algorithm that finds $CH(\{v_1, v_2 \dots v_n\})$ in time $O(n)$. Note that P does not have to be convex.
6. Let A be an array containing the vertices of a convex polygon, in the order they appear along its boundary, in counterclockwise order. Assume that array is given to you by pointers pointing to the first and last cell in the array. Explain how to find the leftmost vertex and rightmost vertex, in time $O(\log n)$. Here n is the number of vertices.
7. Let $P = \{p_1 \dots p_n\}$ be a given set of n points. Consider a very long corridor whose width is d . Suggest an $O(n \log n)$ time algorithm that finds whether P after being appropriately rotation, can be translated along the corridor. Hint — compute first its convex hull.

Answer:

The minimal width of the corridor is the width of P . We can find this by observing that it is obtained between a vertex v of $CH(P)$, and an edge of $CH(P)$. To observe this fact, assume that two parallel lines are passing through two vertices of P , such that P is between these lines. If neither of these lines passes through another vertex of P , then we can rotate P , either clockwise or counter-clockwise, and push these lines closer to each other. Hence the algorithm is based on computing $CH(P)$, and rotating P , where at each even we check the distance between the vertical line passing through the left endpoint of $CH(P)$, and the line passing through the right endpoint of $CH(P)$ (i.e. the lines analogous to the corridor). We stop and check this distance each time that one of these lines passes through an edge of $CH(P)$.