CSL 852, Computational Geometry: Practice Problems

Convex hull different paradigms and quickhull

1. For a given set S of n points let $CH(S)$ be the convex hull of S. Consider the line-segment $s_{i,i+1}$ connecting the consecutive boundary points i and $i+1$ and the corresponding half-plane $H_{i,i+1}$ defined the by line $l_{i,i+1}$ containing the remaining points. (A boundary edge is such that the remaining points lie on one side of the line containing this edge).

   Now consider the intersection of all such half-planes $R = \cap_i H_{i,i+1}$. Prove that $R = CH(S)$.
   Hint: Half-planes are convex and intersection of convex sets are convex and show that $CH(S) \subseteq C$..
   Argue that every point in C can be generated by convex linear combination of two points in the CH(S) and therefore $C \subseteq CH(S)$

   Solution:
   Claim 1 : $CH(S) \subseteq C$
   Proof : Since all $H_{i,i+1}$ either contain points of S on boundary or inside them, $\Rightarrow$S $\subseteq$ C
   All half planes are convex and intersection of convex sets is convex $\Rightarrow$ C is a convex set containing S. But the smallest convex set containing S is CH(S), so $CH(S) \subseteq C$

   Claim 2 : $C \subseteq CH(S)$
   Proof : Take a point p_2 which belongs to C. Suppose p_2 cannot be possibly generated by a convex linear combination of points in S. Now take a point p that can be generated by a convex linear combination of S and thus lies inside CH(S). Join the points p and p_2 . If the line segment pp_2 lies completely inside CH(S), then p_2 could have been generated as a linear combination of S but it is not true. So pp_2 line segment cuts CH(S), say at line segment $l_{i,i+1}$. Now from claim 1, since CH(S)$\subseteq$C thus CH(S)$\subseteq H_{i,i+1} \Rightarrow p \in H_{i,i+1}$
   $\Rightarrow p_2 \notin H_{i,i+1} \Rightarrow p_2 \notin C$.
   So there is a contradiction. Hence assumption that p_2 cannot be generated as a CLC of S is wrong. Therefore p_2 can be generated as a CLC of S and is hence inside CH(S) $\Rightarrow C \subseteq CH(S)$. So by claim 1 and 2, $C = CH(S)$.

2. If you run Graham's scan on a *non-monotone* chain, show that it may not succeed by a counterexample that produces a self intersecting polygon.

   Solution: If we take points in such an order that all 3 consecutive points always make a right turn, then Graham's scan will return all points in the same order as given. But if the points are not monotone in x-direction, then joining the points in order, we may get a self intersecting loop.

3. Describe an $O(n)$ time algorithm to compute the intersection of convex polygons C_1 and C_2 where n is the cumulative number of vertices.

   Solution: Idea is to divide the plane into different regions separated by vertical lines, compute the intersection of polygons in each region and join them.
   First, break the boundary points of each polygon into 2 monotone chains between the left and right extreme points. Since the points are given in a sequence, this operation requires to binary search the extreme points, hence $O(\log n)$.
   Now merge all these 4 lists to give the sorted list of all event points for line sweep. This operation takes $O(n)$ time.
   Begin sweeping points from left to right and at each stage maintain the line segments in all 4 lists intersecting the vertical line through current point. At each point, compute intersection of polygons

between previous vertical line and this line. Merge this intersection polygon to the cumulative intersection achieved so far. This is a constant time operation.
So, overall running time of algorithm = $O(\log n) + O(n) + n.O(1) = O(n)$

4. Complete the algorithmic details, including data structures needed to implement the *insertion hull* algorithm in $O(n \log n)$ steps.

   Solution: Insertion Hull works very similar to insertion sort. Start with 3 points that define a trivial convex hull, the triangle. Then insert one point at a time, updating the convex hull, until you have exhausted all n points. Suppose you are considering point p_{i+1}. If the point falls within the interior of CH_i (the convex hull of 1st i points), then ignore and move forward. Else construct CH_{i+1} by finding the two tangents from point p_{i+1}.

   Construction of tangents from a point : It can be shown that the upper tangent intersects the hull at a point where the turn (made by two consecutive points on hull and p_{i+1}) switches from Left to Right. So we can do a binary search on the points and find this point in $O(\log n)$. Similarly the lower tangent occurs at a point where the turns switch from Right to Left.

   Checking whether the point is in interior of the CH_i : For finding if the point is included in the convex polygon CH_i we will use a generalization of Ray Shooting. The idea is to draw a horizontal line through the point p_{i+1} and find which two sides of CH_i it intersects. If the point lies between the two sides, it is in the interior of CH_i else it lies outside CH_i.

   The steps of algorithm are :
   Find the points having the highest and lowest y-coordinates. Since the points are given to us in a sequence as they appear in CH_i, we can find these points by doing binary search. So time taken is $O(\log n)$.
   Then break the sequence of points into two parts. One sequence contains point from the highest to lowest and the other contains points from lowest to highest, call them CH_R and CH_L. Find the two points in both lists whose y-coordinates are just greater and just lesser than the y-coordinate of point. The line segments, l_L and l_R formed by these two points in both lists will be those line segments that are intersected by the horizontal line through p_{i+1}. Since the list contains points in a sorted order, finding the above mentioned points require a simple binary search. So again time required is $O(\log n)$.
   Now check whether p_{i+1} lies on the same side of both l_L and l_R. This can be done by simply plugging the point into the equation of these two line segments. If both give the same sign, then the point is on same sides and is hence exterior to CH_i else it lies in between the lines and is hence interior. This step is clearly an O(1) step.
   Thus we can solve the problem of point inclusion in CH_i in $O(\log n)$.

   We have seen that for a new point, checking its inclusion costs $O(\log n)$ and if it is exterior then finding two extreme tangents also costs $O(\log n)$.
   So total running time, $T_n = n * O(\log n) = O(n \log n)$.

5. Given a set of points P on the plane, design an efficient algorithm to compute the *diameter* $D(P)$ and width $W(P)$. The diameter of a P is defined as $\max_{x,y \in P} dist(x, y)$.
   The width is defined as the minimum distance between two parallel lines such that all points are lie in the region between the parallel lines.
   Hint: Prove that the pairs of points that determine the diameter and the width respectively are points on the convex hull. Modify the line sweep to an angular sweep method (also called the rotating calipers method).

6. Design a binary-search based strategy to find the closest vertex of a convex polygon from a given input line ℓ.
   Hint: The distances of the vertices from the line increase and subsequently decrease. Also note that if ℓ intersects the polygon, then one of the two edges that ℓ intersects contain the closest vertex.

   Solution: If ℓ does not intersect the polygon, then a modified binary search can be used to find the vertex closest to the line, the one having minimum distance from line. Maintain a and b to be the 2 ends of the list of points left to be searched. Find the middle point of list as $c = \frac{a+b}{2}$. Now 6 distinct cases are possible depending on whether a, b and c lie on increasing or decreasing side of hull and in each of the cases either the new list is taken as (a, c) or (c, b). For example if a is on down side and c on up, then (a, c) is chosen. If a, b and c are all on up side and c is above both, then (c, b) is chosen and so on.
   In case ℓ intersects the polygon, the same algorithm can be used to find one of the local minima, after which the 2 line segments from it can be checked for intersection with ℓ. If so, the other intersecting line segment can also be found by a similar algorithm and the minimum distance vertex from these 4 can be reported.

7. Complete the analysis of the $O(n \log h)$ quickhull algorithm.

8. An alternate analysis of the output sensitive algorithm can be done using the following formulation. Suppose there are n_i points under the i-th step,then argue that the running time can be bounded by $\sum_i = 1^i = hn_i n_i \log(n/n_i)$.

9. It is known that if the n input points are distributed uniformly at random in a unit disk then the expected number of points on the hull is less than $\sqrt{n}$. Based on this property, design an $O(n)$ algorithm for convex hull. Note that the running time will be expected $O(n)$ over the distribution of the input points and not for the *worst case* input.
   Hint: A subset of uniformly distributed points is also uniformly distributed in a disk and has the same property.
   Solution: You may think about the each point as being generated independently at random from the distribution, so the first $n/2$ points have identical distribution to the last $n/2$. Therefore construct the convex hull of the first $n/2$ points $CH1$ recursively and the remaining $n/2$ points $CH2$ recursively, and merge them. Note that the expected number of points in $CH1$ and $CH2$ are $O(\sqrt{n}$. So we can construct the hull of the union of $CH1$ and $CH2$ in $O(\sqrt{n})$ steps - even though they may not be linearly separable.
   This leads to $T(n) = 2T(n/2) + O(\sqrt{n})$ where $T()$ is the expected running time. The solution for this is $T(n) = O(n)$ and it can be verified by induction. CH

10. **Dynamic Convex hull** In many applications, we are required to maintain the convex hull of a set of points that could be changing over time, i.e., points can be inserted or deleted. Try to construct cases where a single insertion/deletion can lead to large changes in the size of the hull. Therefore, we can try to maintain an *implicit* representation of the convex hull so that we can report it in time $O(h)$ where h is the size of the present hull.

    Describe all the details of maintaining a convex hull under a sequence of insertions and deletions such that the cost of each insert/delete operation is bounded by $O(\log^2 n)$ and the space of the data structure is $O(n)$. Moreover the data structure should also support query of the type - "Is a given point inside the convex hull?" in $O(\log n)$ time.

    Hint: Devise a scheme based on a primary tree structure whose leaves correspond to the existing points and each internal node stores a data structure representing the upper/lower hull of all the

points in the subtree rooted at that point. This data structure must support fast *join* and *split* operation also known as a concatenable queue. Each internal node contains a bridge (tangent) of the two linearly separable convex hulls of its two subtrees - it does not contain an explicit description of the convex hull to save space. Recall that such a bridge can be found in $O(\log n)$ steps. Argue that any insertion or deletion of a point can affect bridges along a single root-leaf path of this tree and the corresponding $O(\log n)$ bridges can be recomputed on the fly. CH

11. *Convex layer* The i-th convex layer if a given set S of points is the convex hull of the (remaining) set of points after deleting the points in layers 1 to $i-1$. The *first* layer is the convex hull itself. Describe an $O(npolylog(n))$ algorithm to determine all the layers.
Hint: Can you apply the result of the previous problem ?