Comp 163: Computational Geometry
Tufts University, Fall 2022

Professor Diane Souvaine
Scribe: David Gentile

Dynamicized Convex Hulls Continued

1 Recap of Last Discussion

In the last lecture, we considered some edge cases of computing the convex hull for a set of points where it would actually be beneficial if our running time were asympototic to a function of the number of points in the hull - for example, we considered the case where we have just a few extreme outliers, where we have the vast majority of our dataset bounded by a quadrilateral convex hull whose points lie sufficiently far from the rest of the data. In this case, we saw that Jarvis's March actually performs quiet well, since for large n in this scenario, it's totally possible that we have $nh < n \log n$.

This prompted us to investiage the so-called Ultimate Convex Hull algorithm, a method which computes for a set X the hull $CH(X)$ in $\mathcal{O}(n \log h)$, which would be a significant improvement. This "prune & search" or "marriage before conequest" algorithm presented a dynamicized method for computing convex hulls, which runs on an inital input set X of size n in $\mathcal{O}(n \log n)$ time and updates in $\mathcal{O}(\log^2 n)$ time.

2 Aside: But What's the Deal with Convexity Anyways?

On the first day of class, we introduced several flavors of polygons — convex, monotone, and star-shaped. Why do we take an interest in those particular flavors? There are two main reasons, for us: point inclusion and intersection detection. These problems come up in many applications of computational geometry. For example, let's consider VLSI design for a moment — chips are made up of several layers of differing materials, and the places where different materials overlap corresponds to the presence of a transistor. Geometrically, the problem of designing a transistor then amounts to overlaying a sequence of rectangles, and finding their overlap and the outline of their union.

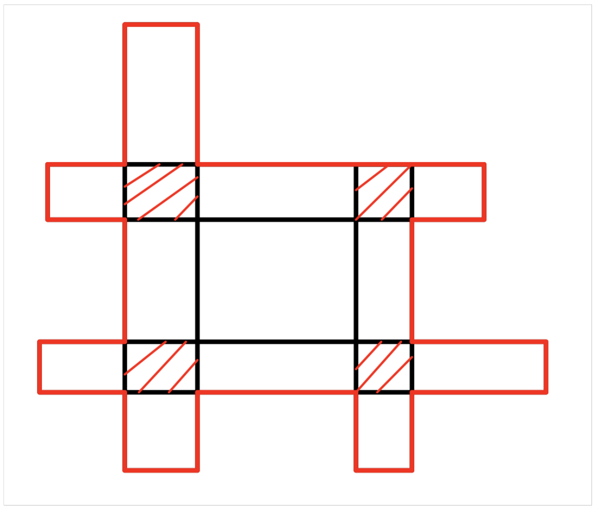


Figure 1: finding the intersections and boundary of the union for a family of rectangles

3 Back to Dynamic Convex Hull Algorithms

Suppose we are in the middle of merging a sequence of lower convex hulls together to form the final lower hull of our input set. We need to be careful wehn deleting points while merging the lower hulls if we want to be able to dynamically add and remove points from the set — in other words, we need to be keeping track of our previous work in the process of building up the hull. Furthermore, it would be very unfortunate if we required $\mathcal{O}(n \log n)$ time for evry update. Recall now a data structure introduced in CS160 - the augmented tree. There are essentially binary trees with a "pointer to anything" associated with each node (cf. void* pointers in C), conceptually a whiteboard to keep track of whatever information we need to keep track of. In the case of keeping track of convex hull computations, we could have the pointers direct us to another tree containing information about the process so far. The top node will contain a pointer which directs to a tree that encodes the actual convex hull of the set, while everything below the root will act as a live history of our dynamic hull algorithm's work. We now claim that with this data structure we can run our dynamic hull algorithm as we had hoped — that is, $\mathcal{O}(n \log n)$ to create the inital hull and $\mathcal{O}(\log^2 n)$ to update it.

4 The 8 Cases

Suppose we are trying to merge two lower hulls. In order to proceed, we'll need to consider the 8 possible configurations of a candidate bridge between two vertices and the ways we might eliminate (at least) half the points in either (or both) of the lower hulls when the candidate bridge does not belong to the merged hull. Note that these cases are discussed in [**?**]. To recap our information before diving into the cases: we are looking at two lower convex hulls, which we might label LCH_L and LCH_R. We know the median line l between the two hulls which divides them, so that LCH_L lies entirely to the left of L and LCH_R lies entirely to the right of l, and the left and right most vertices of both hulls. We also have enumerate the vertices of the lower hulls. Let us denote them by $V(LCH_L) = \{p_1, \ldots, p_n\}$ and $V(LCH_R) = \{q_1, \ldots, q_m\}$. In the following cases, we will consider the angles $\angle p_{n/2-1}p_{n/2}p_{n/2+1}$ and $\angle q_{m/2-1}q_{m/2}q_{m/2+1}$, which we well now abbreviate to L and R, respectively. We also denote the line connecting $p_{n/2}$ and $q_{m/2}$ by λ. Illustrations of these cases can be found at the end of the notes, as well as in the listed reference.

1. The first case is easiest. If λ falls below both angles then we have found our bridge and are done.

2. In the second case, we have λ passing through L below $p_{n/2+1}$ and above $p_{n/2-1}$. We see that λ cannot be the bridge, but we also know that we can safely discard $\{p_{n/2}, \dots, p_n\}$ going forward, since they cannot be vertices of the bridge.

3. This case is symmetric to case 2, but we instead discard $\{q_1, \dots, q_{m/2}\}$.

4. In this case, things get a bit trickier. We see that λ passe through both angles, meaning it is not immediately obvious what points we can discard. For this and for the next two cases, let us now extend the line segments $p_{n/2}p_{n/2+1}$ and $q_{m/2-1}q_{m/2}$ and label the point at which they intersect by r. Now we observe that the median line, together with the right most point of LCH_L and the left most point of LCH_R form a "zone of exclusion". For this case, let us assume that r lies to the left of the exclusion zone. If this is the case, it's clear that we can eliminate $\{p_1, \dots, p_{n/2}\}$ and continue.

5. Again by symmetry to the previous case, if the point r lies to the right of the exclusion zone, then we can eliminate $\{q_{m/2}, \dots, q_m\}$.

6. Now suppose that r lies within the exclusion zone itself - this is a particularly nice case, because we see that we can eliminate $\{p_1, \dots, p_{n/2}, q_{m/2}, \dots, q_m\}$.

7. In the final two cases we consider the possibility that λ lies below one angle and pass through another. If λ lies below L but passes through R, we see that we can eliminate $\{p_{n/2}, \dots, p_n, q_{m/2}, \dots, q_m\}$.

8. By symmetry to the previous case, if λ lies below R but passes through L, we can eliminate $\{p_1, \dots, p_{n/2}, \dots, q_1, \dots, q_{m/2}\}$.

We have now shown that our bridge finding process can be run in log time, and we have used the fact that we can think of the problem of finding a bridge as *order decomposable*. We presort the points with some ordering operation $ORD(n)$ and then perform a $MERGE(n)$ operation which runs in $\mathcal{O}(\log n)$.

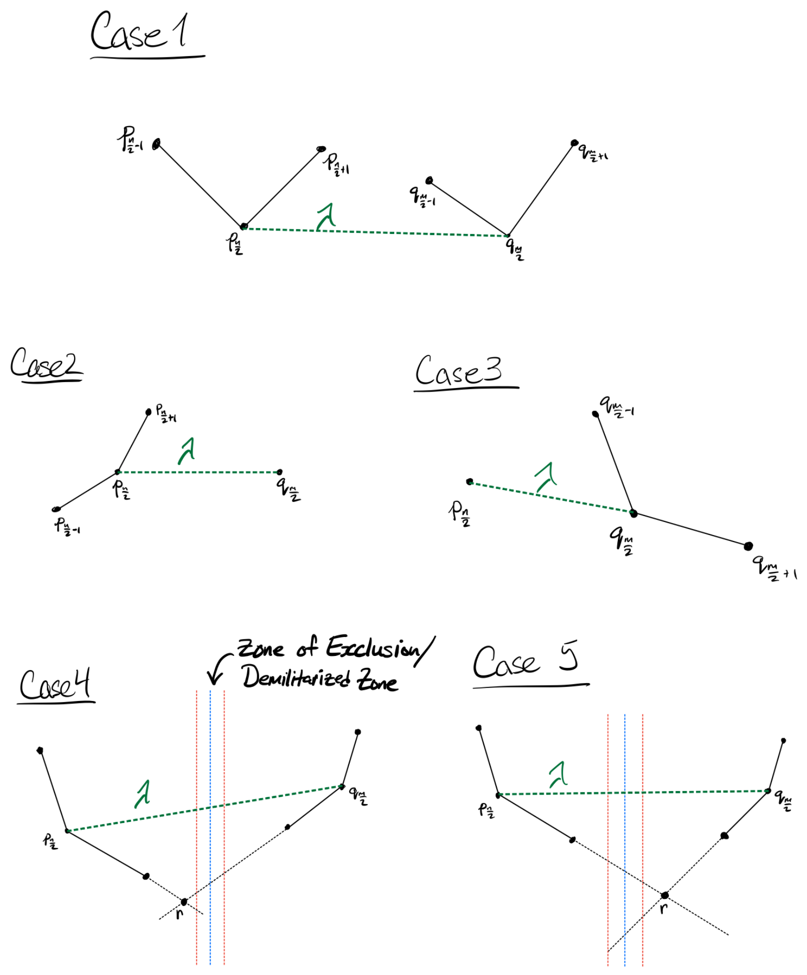


Figure 2: Cases 1 through 5

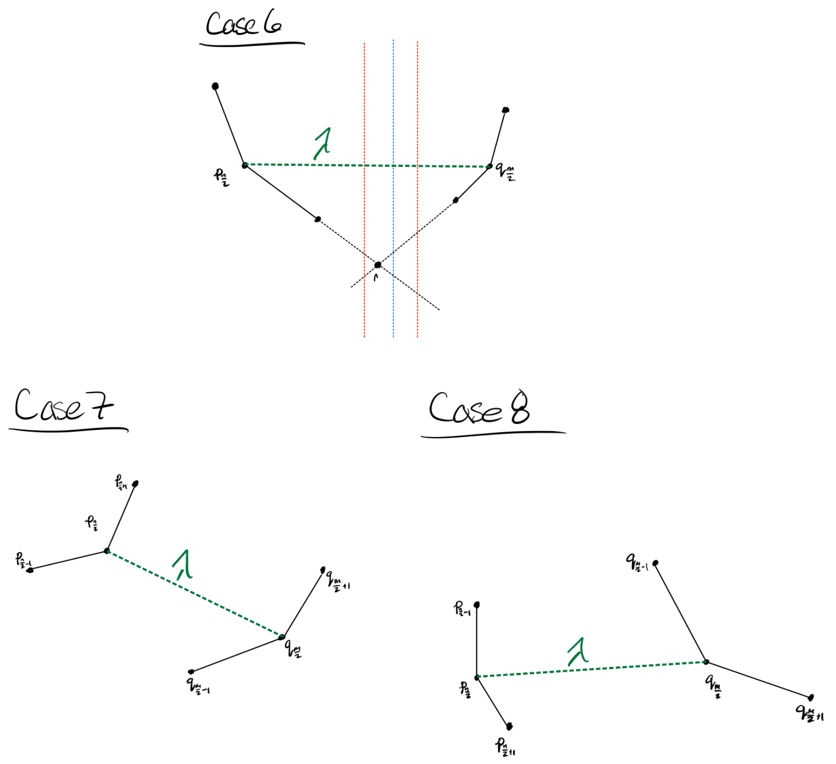


Figure 3: Cases 6 through 8

5 Order-Decomposability Discussion

Order-decomposability will be a common theme throughout the course. Let's preview a few problems that we can use this viewpoint to solve.

5.1 Finding the Intersection of Half-planes

Suppose we have a family of half planes and we want to find the intersectin of them and the boundary of that intersection. We also want to do this in a dynamic way, i.e., we want to be able to freely add/delete lines to/from the family. The solution to this problem will wind up being extremely similar to the work above, although we will be leveraging the slopes of the lines to order them instead.

5.2 Find the Dominating Set of a Set of Planar Points

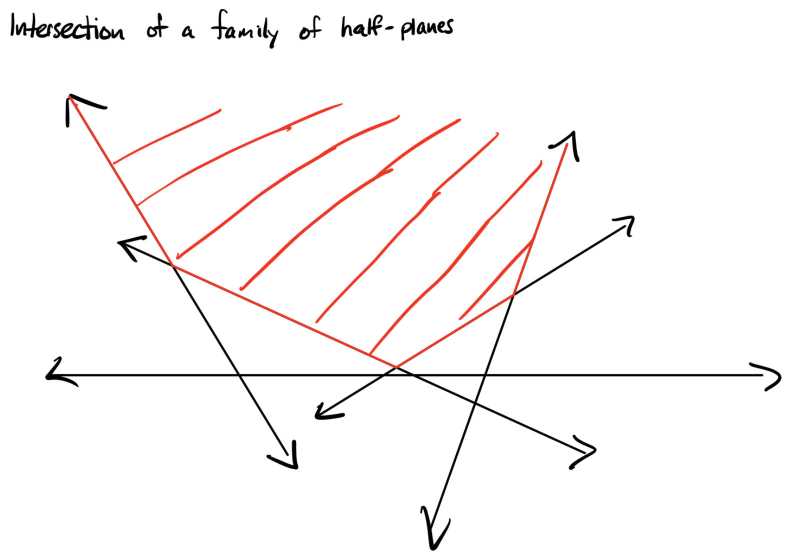
Suppose we have a set of points X in the plane. Each point then, when taken together with the origin, defines a unique rectangle. The dominating set of X is defined to be the subset of points which is not interior to any of the generated rectangles. How do we compute this set? Again, we will see that the solution is very similar to the work above.

5.3 Finding the Top and Bottom of a Polyon

Suppose we have a simply polygon P, with vertices labeled $\{p_1, \ldots, p_n\}$. How can we find the vertices of maximal and minimal height in sub-linear time? Consider plotting the height of each vertex as a function of its label/index. Then by applying a variation on the above themes with ternary search we can perpetually discard 2/3s of the points until we find the critical values.

References

[1] D. P. Dobkin and D. L. Souvaine, *Computational Geometry – A User's Guide*, Chapter 2 of *Advances in Robotics 1: Algorithmic and Geometric Aspects of Robotics*, J. T. Schwartz and C. K. Yap, eds., Lawrence Erlbaum Associates, 1987, 43-93.



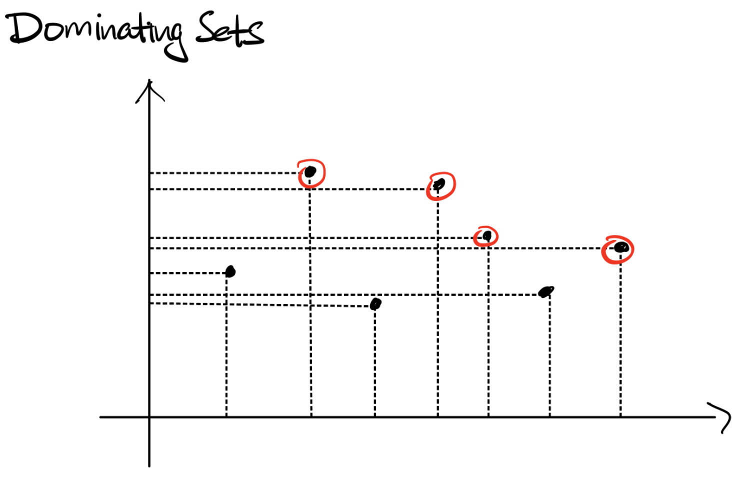


Figure 4: Finding the intersection of a family of half planes, and the dominating set of a family of points in the plane

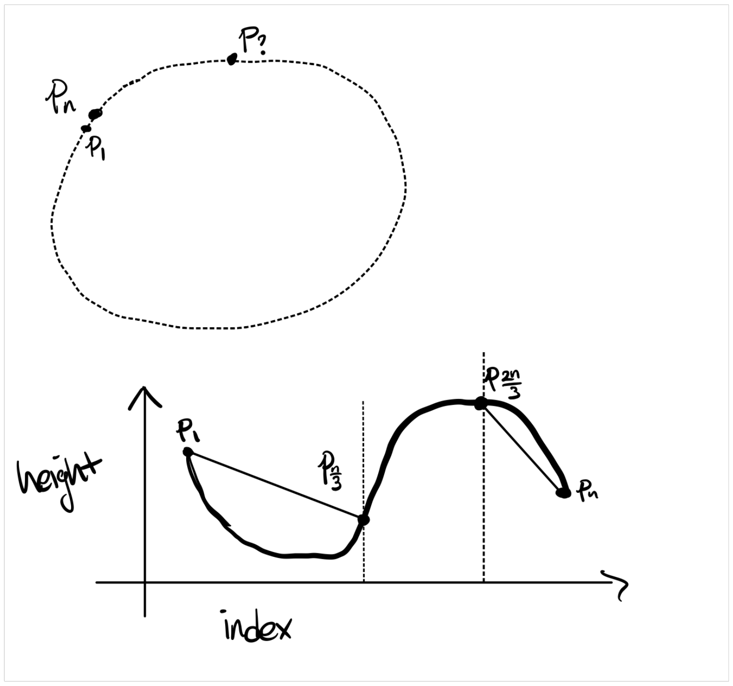


Figure 5: finding the points of maximum and minimum height for a given polygon