Lecture Transcript
Convex Hull

Hello and welcome to this next lecture on Data Structures and Algorithms. We will continue our discussion of geometric algorithms. In the last discussion, we tried answering question such as whether a point p1 turns clockwise or counterclockwise from a point p2 with respect to some point p0 and we had a very simple cross product or determinant based formula for answering such questions based on which we also answered slightly a trickier question such as whether two line segments p1 p2 and p3 p4 intersected at all. Today, we'll discuss another flavour of computational geometry algorithms, where the task is to generate a new geometric form from a given set of points or a structure and the specific form we talk about today is the convex hull. So the convex hull of a set of points Q is the smallest convex polygon P for which each point in Q is either on the boundary of P or in its interior. Let's illustrate with an example.

So let's say these were the points. So the convex hull which is the tightest smallest convex polygon could be obtained by enclosing these points with a rubber band stretched around all of them. So you want to enclose all of them, but as tightly as possible. So you probably need to ensure that you select the top most point and the leftmost point, the rest can be successively covered. You can find that these two points should be covered, so should these two points and these two. One can imagine the rubber band that has been stretched to just enclose these points to be the convex hull. Now, this can be computed using a technique called rotational sweep. We'll discuss one flavour through the well known Graham's scan algorithm. So, Graham's scan does the following. Given a set of points Q. It starts with a point p0, which has a lowest y-coordinate which is the point which is at the bottom. So, as already motivated on the previous slide you might want to first include the points that are either extreme left or on the extreme right or on the extreme top or on the extreme bottom. What Graham's scan does are as what one version of Graham's scan does is starts with the bottom most point. We leave it as homework to develop variants of Graham's scan that start with the left most point or the right most point or the top most point. For the sake of the remaining discussion, we'll start with the bottom most point, the point with the lowest y-coordinate. In case there is a tie, which is in case there are two points which have the same smallest y-coordinate which could be the case here. You will break the tie by selecting the one with the smallest x-coordinate as well.

So let's stick to the story, what would you do next? Well, you would like to look at the orientation of all the other points with respect to the point that is already chosen. That is with the new point. What would we like to do next? We can do the following. We can look at the orientation of the remaining points, sort them in some order of orientation and thereafter decide how to construct the convex hull in an incremental manner. In particular, we'll resort to a counterclockwise inclusion of points. Let me explain this a little bit more carefully. So, we're going to sort the remaining points of Q by their polar angle related to P0 in counterclockwise order by comparing cross products. What does this mean? So let's say the point that we have already included here P0, which is the point with the smallest y-coordinate is here and let's look at p1 here, p2 here, p3 and so on. What we're are going to do is compare cross products, which is p1 with the smallest x coordinate - p0 cross product with (p2 - p0). (p1 - p0) cross product with (p3 - p0) and so on.

So very implicitly we have chosen as our frame of reference the point to the lowest y-coordinate and the point to the lowest x-coordinate, p0 and p1 respectively. And we're going to look at cross products of (pi - p0) with (p1- p0) and we're going to sort those remaining points. What would we expect? Well, we'll expect that any point (pi - p0), its cross product with (p1 - p0), that's (p1 - p0) cross product (pi - p0). This we expect to be positive, if and only if p1 is clockwise from pi with respect to p0, and the value of this cross product something that we have ignored so far will give you how much p1 is clockwise from pi with respect to p0. So, if you have way to compute this cross products for all points pi and we of course expect to sort the points and if we were to sort these points based on increasing values of this cross product then we would actually get the points sorted in counterclockwise order from p1.

So, the larger the magnitude more is pi counterclockwise rotated from p1 with repect to p0. So again recall that p1 is the point with smallest x-coordinate. What do you do next? Well, now we will be interested in inspecting points in the order in which they are sorted. So we're going to include p1, we look at p2 and then look at p3 for inclusion. When would we include p2 and when would we consider including p3, given that p0 and p1 are already included. So we will look at the angle suspended at p2 from p3 and from p1. What we find here is with respect to p2, p3 is counterclockwise from p1. On the other hand, if p2 had been up here, then one could find that p1 is actually counterclockwise from p3 with respect to p2 or rather p3 is clockwise from p1 with respect to or rather p3 is clockwise from p1 with respect to p3. So, p2 will be excluded if the new point happens to be counterclockwise from the previous point with respect to the point p2.

So in this specific case given this counter clockwise rotation, we will drop p2 and move on to p3 to decide whether to include p3 will of course, need to look at p4 and see if p4 is rotated clockwise from p1 with the respect to p3. If p4 is rotated clockwise from p1 with the respect to p3 you will include p3 else will excluded. So the way do we keep track of the last three points or the three points on which we need to may comparison is through the stack so we will at any point of time maintained in this stack points which we think could belong to the convex hull. So stack S is a subset of points which could potentially belong to the convex hull of (Q). We know for sure p0 belongs to a convex hull and so does p1. So what needs to be inspected is p2 and we will compare p2 for its inclusion by looking at the angles suspended at p2 by the new point p3 which is a query point. So again recap we have p2, we have p1 and we have p3 and angle of the sort which integrates a counter-clockwise rotation of p3 from p1 with respect to p2 will mean that p2 should get excluded.

On the other hand, if p2 were such that p3 is rotated clockwise from p1 with respect to p2 then p2 should also get included. This is summarized here, if the angle form by points pi, we start with i = 3 and the first two points on a stack in that order is in the clockwise direction, then pop the topmost element from the stack, else push pi into the stack. So finally, the stack contains all the points from bottom to top, exactly the vertices of the convex hull (Q). We'll illustrate this. So here is the complete example we've included p0 and p1 should be included p2 as discussed. We 're going to look at the angle suspended by p3 at p2 and look for the rotation of p3 from p2. We find that p3 is counter-clockwise rotated from p1 with respect to p2 and therefore, p2 can be dropped. So this angle isn't working in favor of p2. We dropped p2, include p3 in the stack look at p4. So now the stack S has a following elements S is [p0 p1 p3]. We know that p0 and p1 should be included.

So we always look at the top element as a candidate for inclusion or exclusion. In conjunction with the new point that is being considered which is p4, but we will use the previously included point p1. This is the second last element on this stack to check for whether p3 should be included or not and that's we'll do here. The angle suspended at p3 we find that p4 is rotated clockwise from p1 with respect to p3. So this means p3 will be included. What about p4? Well, we find that p4 simply cannot be part of the convex hull because it's actually in the interior of the convex hull. By virtue of this p5 being counter-clockwise rotated from p3 with respect to p4. So we dropped p4 include p3 move on to p5. We next check if p5 should be included and we find that indeed p6 is oriented clockwise from p3 with respect to p5. Whether to include p6 or not we'll know soon but you proceed and then we check for p7 and again we find that p7 is oriented clockwise from p5 with respect to p6 so we continue our journey. Look at p8. We seem to be heading wrong. Are we going to correct this?

Well yes, pretty soon, so at p8 we still don't detect the problem, but when we consider p9 and look at the angle suspended by p9 at p8 with respect to p7 we'll find that now suddenly p9 is oriented counter-clockwise with respect to p7 at p8. So this means we need to pop p8. Drop p8 and in fact, when you go to p10 we're going to drop a whole lot of the other points. We're going to drop p7, p6, p5 and so on and actually connect p3 to p10 directly. So we keep popping points out of the stack as long as they result in a counter-clockwise motion because we found new point that can be included to be consistent in the counter-clockwise inclusion, and as we find points that lead to clockwise motion around the point. We keep them in the stack for verification, later on and that's what happened here. We included p10 in the stack, we've dropped all the other points now we look at p11 should p10 continue remaining on the stack. Well, p11 is oriented clockwise from p3 with respect to p10, because p3 is still what we see the second last element in the stack. So one might want to recall the stack now contains p0, p1,p3 and p10.

The new point that we're looking for inclusion is p11. You know for sure that p0, p1 and p3 are part of the convex hull queries about p10 and p11 and yes there is a clockwise motion around p10 strong enough case for p10 and we included. Well, p11 will also be included as long as there is a case for its exclusion and in the next step, we find that p11 should be excluded because p12 is oriented counter-clockwise from p10 with respect to p11. So we are going to drop p11 and the last point p12 will be tested for inclusion with respect to p0 and yes, we find that p0 is oriented clockwise from p10 with respect to p12. So p12 will be included. So here's the algorithm of finding convex hull, we start with the reference point p0 with the lowest by coordinate. In some sense we also have p1 as the next reference point, we push p0 and p1 and then the subsequent points p2 onwards will be queried for. So we start with p2 and the next point p3 and we check if p3 is oriented clockwise from p1 with respect to p2 and here's the depiction again.

We've already included p1, here's p2 for being queried, here's p3 should be included p2 or not. Well, we will include p2 and continue looking at p3 for inclusion, if p3 is oriented clockwise from p1 with respect to p2. We would now generalise this two not just p1 and p2, but the last two elements of the stack. So what was p3 is now basically the Pi. What was p2 is the top of the stack S and what was p1 is now the second last element of the stack. We retrieve that second last element by this call next to top(s). We refer to the counter-clockwise orientation of Pi with respect to the next to top as a case for dropping top(s). We refer to this as a non-left turn. So what the non-left turn means is that Pi is oriented counter-clockwise from the second last element, with respect to the top of the stack. We keep doing this, keep popping elements as we saw in the illustration. We kept dropping elements p9, p6 and p5, when we encountered p10 and we keep doing this till we encountered a clockwise orientation. At this point, we basically insert Pi, ns and we continue. The complexity of this algorithm is the complexity of sorting and that turns out to be nlog n. There are other algorithms that give you complexity of the order of n times h, where h is the size of the convex hull. So if the convex hull turns out to be very small, Graham-Scan turns out to be expensive, because its nlog n. However, if it turns out that most of the points belong to the convex hull, then Graham-Scan turns out to be a very good choice.

Thank you.