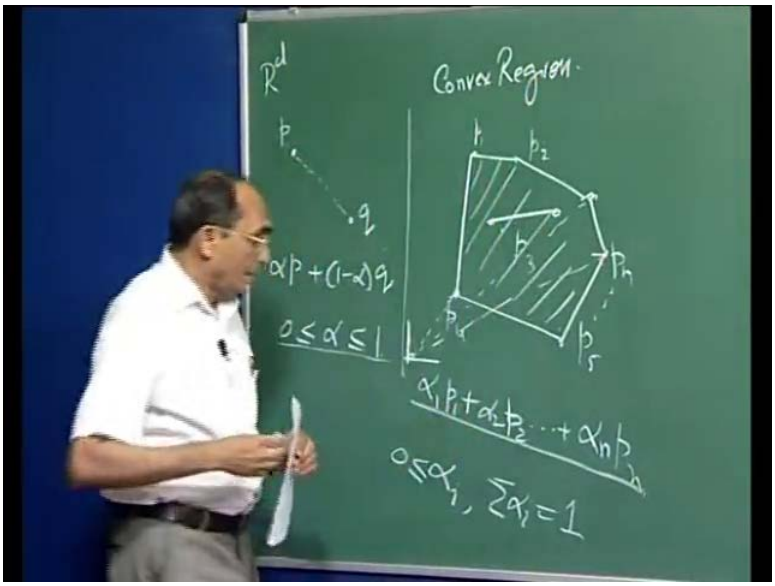
Computer Algorithms-2
Prof. Dr. Shashank K. Mehta
Department of Computer Science and Engineering
Indian Institute of Science, Kanpur

Lecture - 29
Geometry I

(Refer Slide Time: 00:31)

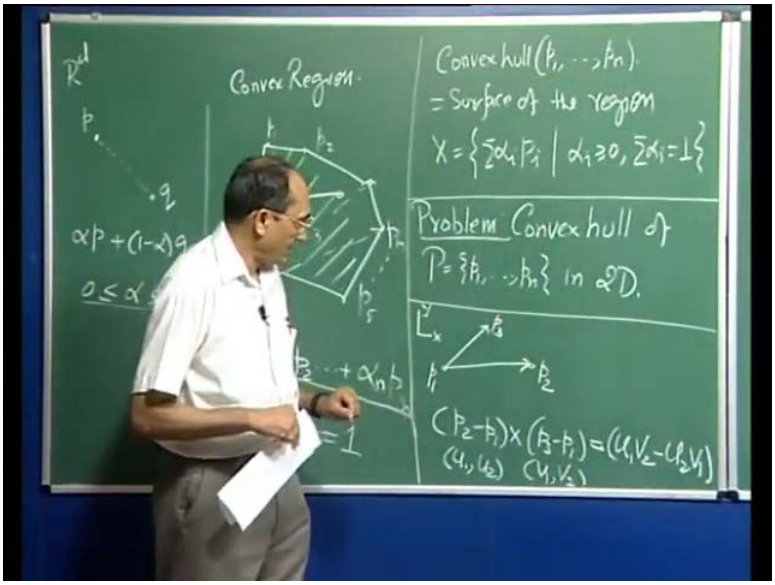


Hello today and in next 2 lectures we are going to discuss some elementary Geometric algorithms. Let us first begin with some basic ideas, let us suppose we have a point in some space d dimensional space d R d, we have 2 points p and q, then this segment is defined by alpha p plus 1 minus alpha q where alpha varies anywhere from 0 to 1. If we will remove the restriction given here then this describes the entire line passing through these 2 points. If we have several points P 1, P 2, P 3, P 4, P 5; then the same expression, where we express alpha 1 P 1 like alpha 2 P 2, alpha n P n say we have some n points.

This spans a space, if we do not put any restrictions on alpha, then it spans the space described by these points. So, if we have an some origin, with respect to which we have these vectors, if we restrict ourselves to such a constraint on the other hand, over here we had... So, each of these alphas are non negative and the sum of these alpha is 1, this is equivalent to this constraint, then it describes a convex region, defined by these points say we had these points, then this entire region is span by this expression. Now, such a region is called a convex region and that is because you pick any 2 points and you

consider the segment, line segment between them this entire segment is inside this region. So, this is called a convex region. So, a convex region defined by n points in R d is given by this, this surface of such a convex region, is called the convex hull.

(Refer Slide Time: 03:37)



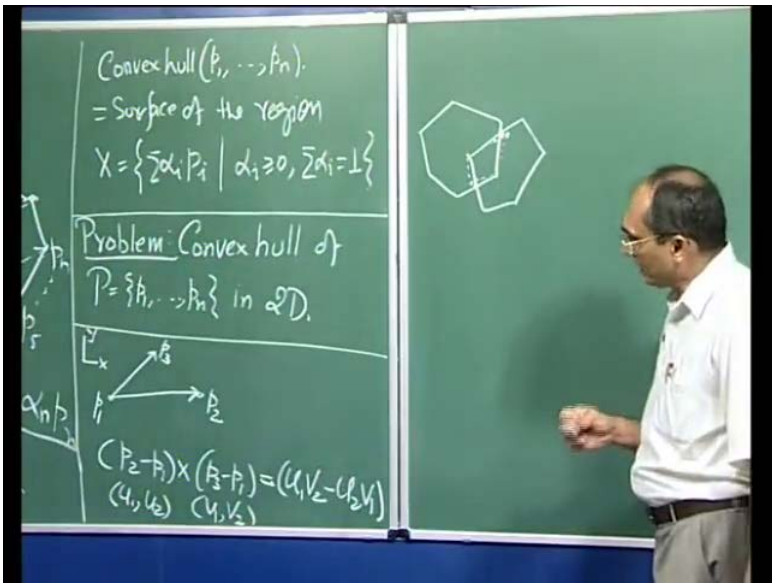
See the convex hull of a set of points is the surface of the region, let us call it x which is alpha i P i where, each alpha is non negative and the sum of alpha i is 1. Now, in many applications, one would like to compute the hull of a set of points. Today, we are going to discuss efficient algorithms to compute hull in two dimensional space, it is indeed much more difficult in higher dimension, but in two dimensional case there are several very efficient algorithms.

So, our problem is convex hull of a set of points P in two dimensional space. Now, there is one small problem that we will need to solve often, which I am going to just state here, it may happen that, we have a point P 1 and P 2 and there is a vector P 2 minus P 1. So, these are currently points in two dimensional space, then we may want to know, whether the position of P 3 is to the left of this vector or to the right of this vector. So, one possible way is to compute the cross product of P 2 minus P 1 P 3 minus P 1.

And if we think of this as a x y plane and in right handed coordinate system the z is coming out of the plane, in that case this vector will be pointing towards positive z x's. If this is to the left and it will be pointing towards a negative z x's if it is to the right, this actually will be. So, suppose the 2 coordinates of this are u 1 comma u 2 and v 1 comma

v 2 then this would be u 1, so we have a i j k. So, u 1 v 2 minus u 2 v 1 it is important in geometric problems, to try to solve problems with minimum amount of errors, due to computations the reason is that often the quantity that we compute, loses it is meaning because of the errors.

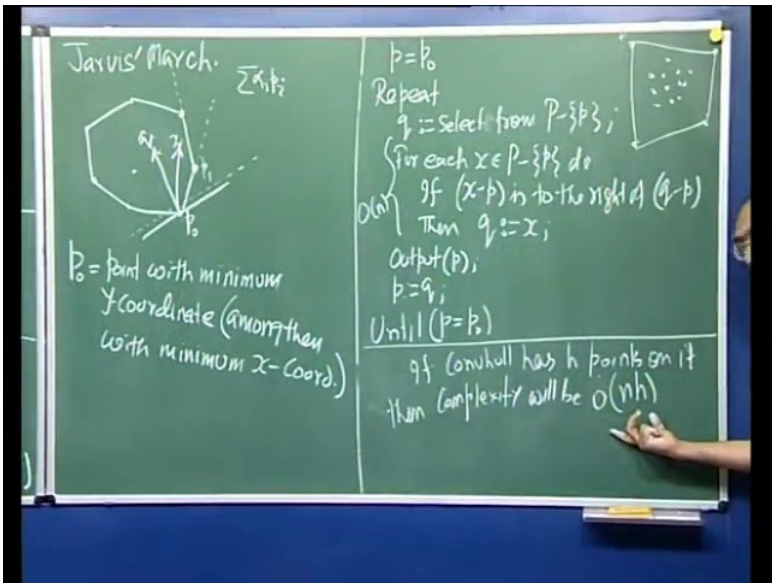
(Refer Slide Time: 07:58)



For example, if somebody wishes to compute the intersection of 2 polygons, then it is inhabitable, that you will have to compute these 2 coordinates and that may result into errors. And if you end up getting a point somewhere here, then the polygon that you will compute, will not be contained completely and may be here, is the other point. So, what turns out is that the intersection that we have computed is nether inside this, nor inside this.

So, one would like to ensure that the points that we compute is approximated if not exactly here, but inside this and so on. But more than that a problem such as this, which has yes or no answer, is the direction of this arrow towards the left or towards the right of this, one should ensure that one does not compute, any expressions to which may have errors, in deciding the answers; anyway, coming back to our problem of convex hull. We are now, given a set of n points, the algorithm that I am going to discuss, is the one that I am going to discuss is called Jarvis march.

(Refer Slide Time: 09:45)



The basic idea, behind this algorithm is the following imagine that this is a plane and you have put nails on these points, then all you need to do is rap a string around it; if you do that, you are going to get the shape that will be the hull of the convex region found by these points, this is exactly we will like to capture. Now, if I take a horizontal line at minus infinity y equal to minus infinity and bring it up, that is going to effect the lowest, if there are more than one lowest points then pick one with the minimum x value and let us call that point P naught P naught is the point which minimum y coordinate and among them, with minimum x cord. Now, such a point has to be on the hull the reason is that, if we choose a suitable line we can separate this point from the rest of the set of point. If we can separate this point from the rest of the points and that will possible with these this particular point.

Then we notice that, low amount of combination of these points can express this point. The combination that we have described here earlier, is one in which we have alpha i p i. So, the convex hull of the rest of the points is bound to be excluding this point, the hull because we have this region which contains all the points. So, the hull must be inside this; so if we include this the hull must change and if the hull changes, then this must be on the hull.

So, we begin with a point a priory known to be on the hull and our idea is, to traverse through the potential points which should be on the hull. And to do that, essentially what

we have done is we started with this string and tied it with this nail and then we, locate suppose we stretch it and we try to find, the first point that will hinder this string that will come and it will hit this. Once, we find this, this will be the next point and then we will repeat the process now, you take this line and try and move this way it will be hit by this one. So, we will have string going like this then we will try and move it with this.

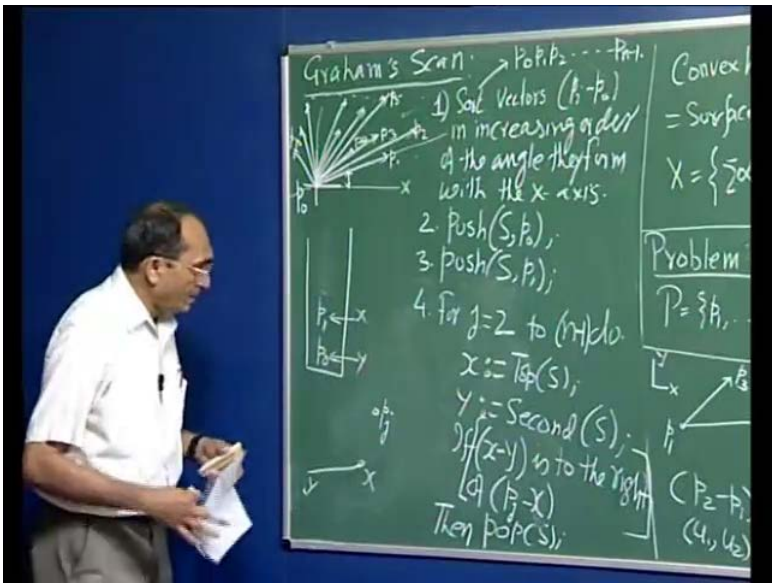
So, how do we find these points well, so now we began with this point which we know is on the hull now, we are going to start with a reference point P. So, our first point is P naught, and we are looking for the next point. We are going to alternatively do this search for these points, until we come back to this. So, the process will be then let us repeat, these steps in these steps we are going to look for a point which is to the right the vector from P naught to which is right towards the right of everybody else, notice that the entire set of points are on this half plane.

So, we will take in any point q let, q be select from P minus P we pick any point from this set. And then for each x in the set P minus P comma q P minus P do. If x minus P is to the right of q minus P; that means, we happen to have found an x here, then I would make this q because this is now a potential next point for us, then set q to be equal to x. So, after this probability is over we must have found this point as our q.

So, we will then say, output P and set P equal to q and we repeat this process until, our P naught is equal to P naught and I think at the end we are going to output the last one now, well there has been already done. So, we will. So, this would complete the entire process. Notice that, this test takes order one time hence, the for loop purses order n, n being the size of the point set.

Now, if the hull has h points, if the convex hull has h points on it then the complexity will be order n h. This algorithm, will be efficient if the number of points on the hull are much lesser than the total number of points for example, if you happen to have several points in the set, but very few points actually end up deciding the hull. In this case the hull is decided by just these 4 points while there are several points over here, the cost will be relatively low in computing this hull. Now, let us go to another algorithm and this is known as graham scan.

(Refer Slide Time: 19:04)



Now, in this case without loss of generality we will assume that all the points are in the first quadrant, here is the original coordinate system and we have various vectors to the points. So, in this case the first step we are going to take is to sort once again, I have I forgot what we have to do is, select this origin as the point which has the lowest y value and among them, if there are several points with the same y value then the minimum x value. So, I am going to choose this P naught exactly the same way that I choose in the other algorithm.

So, this is one of the points of the hull and then the rest of the points will be in the first quadrant, it is possible that some of them are on the x's, but they can be here of course, in this case I must apologize it is not necessary that they are in the first quadrant, but they will be in the first and the second quadrant. So, we solve these vectors in the increasing angle, that they make with the x axis. So, let these be P 1, P 2, P 3, P 4, P 5 and so on P x.

So, sort vectors P i minus P naught in increasing order of the angle they form with the x axis. So, once again we know that, this is on the hull, but what is more we also know, if this is the right most the first one of points in this order, then this entire line this straight line is such that all the points of the one side of this. So, once again you can imagine when you are raping, this point must be on the hull. So, this second step is, to put p 0 and p one in a stag.

So, we push in this stag P 0 and we push in the stag P 1 we will use this data structure and eventually all the points of the hull will remain in the stag. So, we have at the moment P naught and P 1 and then we move on to the next point. So, we will we put P to here. Now, what we have to check, is that if this is P naught this is P 1 then P 2 must be to the left of this direction, if P 2 were on the right of this direction, in case of P 2; obviously, not possible, but subsequently such a situation could arrive, then this cannot be a convex polygon in one place.

So, that is not possible. So, we will have to eliminate, from now one will alternatively put a point and verify whether it bends properly towards the left, but what happens, if it bends to right. So, let us try and look at that case, let suppose we have a situation where we have at the top of the stag this one subsequently these points are in the stag and we have been computing the hull and then the next point we find remember the points are sorted in the angle in the this way. So, the next point let us say and may be since P naught is at the bottom I will assume, we have these 3 points and the next point let us say is here.

So, with respect to p naught these are increasing angles. So, it is perfectly possible that the next point is here and then we put this in the stag. So, we find that with respect to this direction the new vector. So, let this be say P i this is some P j the previous one this is P k even before that, with respect to direction P j minus P k the direction P i minus P j is to the right. Hence, we cannot accept this situation. We will remove P j and consider now, this hull.

Now, imagine that this appears to be a straight line maybe this is still bending here and this is my P j this is still bending to the right, in that case I am going to again remove this and connect directly to the one before it. So, what may happen is that we lose some of the points, which were earlier part of the hull. So, in this alternative process each time we are going to insert the next point and verify whether it makes the right angle, if it does not then we are going to remove starting from the top of the stag one point at a time and verify eventually that the bend correct and stop.

So, let us try and capture this here. So, for j equal to 4 to n 1 at a time, I am assuming that after sorting the vertices are labeled in that order. So, after sorting these are in order P after P 0, we have P 1 P 2 in the increasing the indices are accordingly as per the

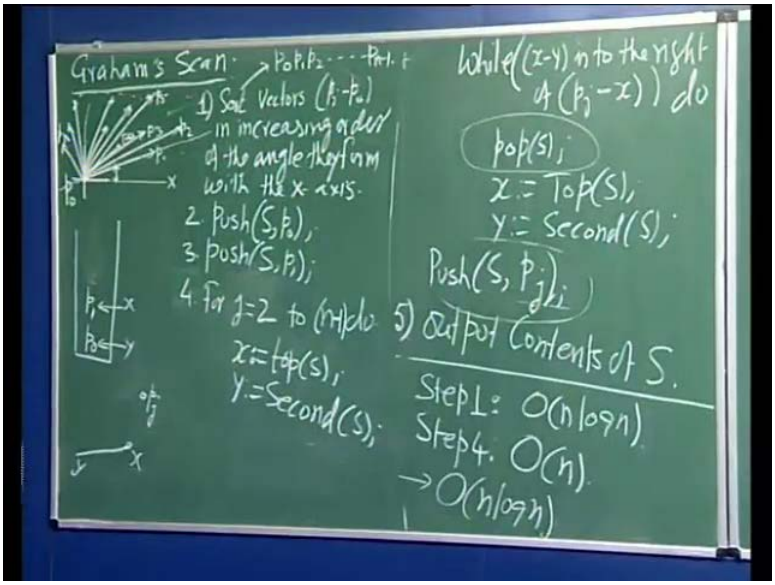
sorting, I am sorry it should be 3 2 n minus 1, because we are using P naught. So, we have labeled 0 through n minus 1 P n minus 1. So, we will do is, while now I am assuming that whatever we have in the stag is properly well, may be we can just start with 2. So, let us say start with 2, because it is possible that the very first bend is P is 1 P 2. So, this is alright.

We have put one inside it and now, before we put two or the next point we will check, while check the condition, well we need to give some label to those. So, let us say x is top of the stag and y is second from the top of this point we have these two. So, right now we do not have P 2 in this we have x here and y here and we check, if the bend from y to x. So, we have y x and then we have P j. So, x minus y vector is to the right to the left of P j minus y vector then we have to remove this.

So, we will say if x minus y is to the right of P j minus y this is the condition. Then pop, but we have to do this alternatively, if is to be repeatedly done, this is y oh sorry P j minus x this is P j minus x this step has to be then repeatedly until, this condition fails to happen. We have x minus y is.

Student: To the left sir. To the left of this if it is to the right then this is the, if this is to the left we must top the top of this we got to remove this. So, now, we have to do this alternatively. So, we will have to put this in a loop. So, let us just put this entire process, inside a loop.

(Refer Slide Time: 32:01)



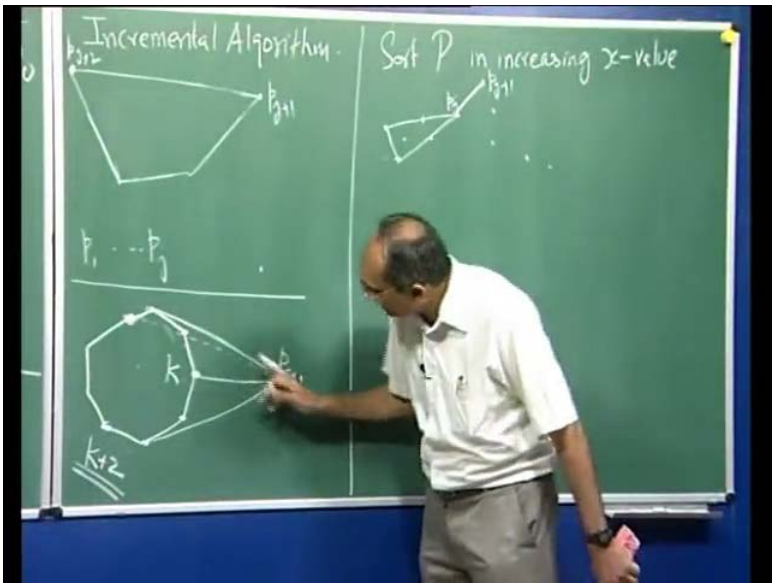
And then we are going to have while x minus y is to the right of P j minus y x do, we are first remove the top of the s and then compute x as the new top of the stag, y is the second of S. Once, we are done with this, the fifth step is to push P j into this stag. So, the entire search. So, this search here ends at sorry x y and the P j then will continue, to the for loop will then search for the position for the next one and this process will go on. Finally, in the six step in a output the contents of the stag, the algorithm in the first step in the step 1, involves sorting of n elements. So, it costs us order n log n.

Now, let us look at what does it charge, in executing this for loop which contains this while. Now, in the face of it is not clear how long this while loop will run, but then there is one obvious fact that we can see, that once a point leaves the stag it is a one of the earlier point than P j the one that we are computing, any of the earlier points never enter the stag, the notice that this is the only way it enters the stag. So, subsequently plus one will enter and P j plus 2 will enter this way.

So, any point can will enter once, and will exist at most once because it is not going to enter again. So, the cumulative cost of all the pops can not be more than order m cumulative cost if all the pushes, will be order n hence the step 4 and step I think, I have to sorry this is inside this for loop I would not name. So, this is a push S P j this is inside the for loop and I would make this step 5. So, the total cost of this step, the is step 4 is going to cost this order n, hence the total cost, is order n log n.

So, unlike well, this is taken over by this is the dominating term. So, the total cost is only n log, unlike the previous algorithm, the cost here is n log n it is independent of the number of points on the hull. So, this can be more profitable, in case there are several points on the hull, where the other algorithm will not be variable. Now, I am going to also give you one incremental algorithm. While this is to the I keep making these mistakes. So, yes this should have been to the left, because this is what we do not want well. So, let us now talk about an incremental algorithm.

(Refer Slide Time: 31:22)



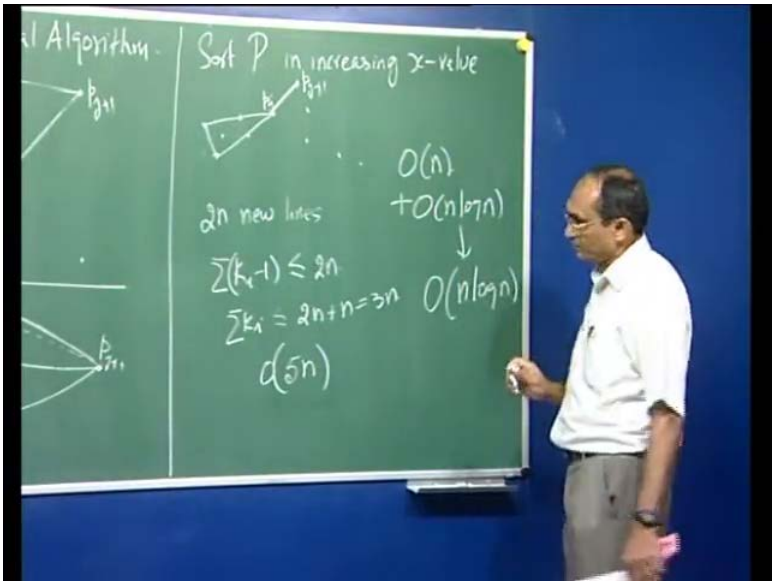
In this case we are going to build a hull, one point at a time. So, let suppose you have a hull made up of say first j point say we have point P 1 through P j, if point P j plus 1 is inside the hull, then it is not going to modify the hull will remain intact for the set 1 P 1 through P j plus 1, this in case P j plus 1 is outside the hull, let us say it is here, then we have to see which all sides that are visible from P j plus 1. So, this is visible and this is also visible, but these are all hidden.

So, if I draw a tangent, it will touch these 2 points and then we are going to remove this portion, that will form the convex hull for the first j plus 1 points, then of course, if I find now it is let us say for the sake of or in general case this is P j plus 2 in this case the visible edges are these 3. So, the hull may be found by connecting these 2 vertices with P j plus 2 and we will be removing these edges. So, this is an incremental algorithm, what does it involve is to identify one visible edge.

So, let suppose you have a hull and first thing we have to figure out whether the point is inside or is it outside, if we find that it is outside then we will have to figure out 1 edge which is visible from the new point P j plus 1 say we may figure out this edge or a vertex, which is by visible which means that this line segment does not cut the convex hull the current hull, then we have to go on walking until the point that we go to is the last point from where we can see this point. Similarly, we have to walk on the other side and to the same thing. So, there are 2 parts to this entire process, one is to find first point

which is visible from the new point and two scan through this on the either side of this point, find out what is the last point which is visible from here and the last point visible. Now, there are several ways to do these, but I am going to just discuss one of them.

(Refer Slide Time: 41:07)



Suppose, we sort the set P in increasing order of x value, increasing x value or x coordinate of those points. So, we have points and so on. So, suppose we have found the hull of these points, then observe that and say this is the last point P j and this is P j plus 1, in this case notice that always P j will be visible from P j plus 1 the point which is next to P j to its left, must be visible from here the reason is that if I draw a straight line from here to here, that line cannot cut through this the rest of the hull simply because everybody else is x coordinate is less than or equal to the x coordinate like this.

So, this victory sorts out the first part of the problem of finding out the first visible. Now, as we walk, as we go on one of the sides we have to now, determine whether this is visible from here or not. So, how do you figure that out, it can be what happens when it is no longer visible. So, when we come here what we notice is, that the bend that we have, this line segment and this line segment as long as this line segment from here is to the left of this line segment this is, but the moment here, we what we find is this lien segment is to the left of this line segment, we say it is no longer visible this is not visible from here.

So, it takes order one time to decide each of these vertices, whether they are visible from P j plus as we walk along, every point that is found to be visible, we move on we have to move up to this point, which is the first point which is not visible. So, if we find k points which are visible, in this entire segment of the hull, there are k points then we are going to test k plus 2 points, for visibility one more from itself and this one, we are going to test k plus 2 points. Each time, we are going to delete the k minus 1 line segment is between them the k points which are visible. So, there are k minus 1 segments which are in between and we are going to introduce 2 new line segments into it.

So, I am not going to explicitly write down the algorithm, but let us try and analyze this algorithm, we notice that in each step we are going to introduce at most 2 new line segments, if the point is inside we are not going to introduce anything else we are going to introduce 2 new line segment that much. So, we can introduce, at most to 2 n well 2 n new line segments, lines into the hull. So, the number of lines that we can delete from this entire process cannot exceed what we introduce into this.

So, the sum total of test that we do for visibility, will be the sum of these k's and the sum of these sum of the sum of k i minus 1 in i'th step if we have eliminated, if we have found k i visible vertices, then we have eliminated k i minus 1 lines from it. So, this sum have to be less than or equal to 2 n because that is how we introduce, well initially we actually form a triangle. So, there will be one more, hence this sum k i is less than equal to 2 n plus n this sum 1 over n 3 n and we do k plus 2 tests in each step. So, this adds up to the total cost adds up to and the 2 each time. So, the total cost is 5 n order 5 n tests we will do for visibility here. So, the total cumulative cost, will be order n, but there was a initial cost of sorting. So, this plus order n log n and hence again this incremental algorithm will cost us order n log n.