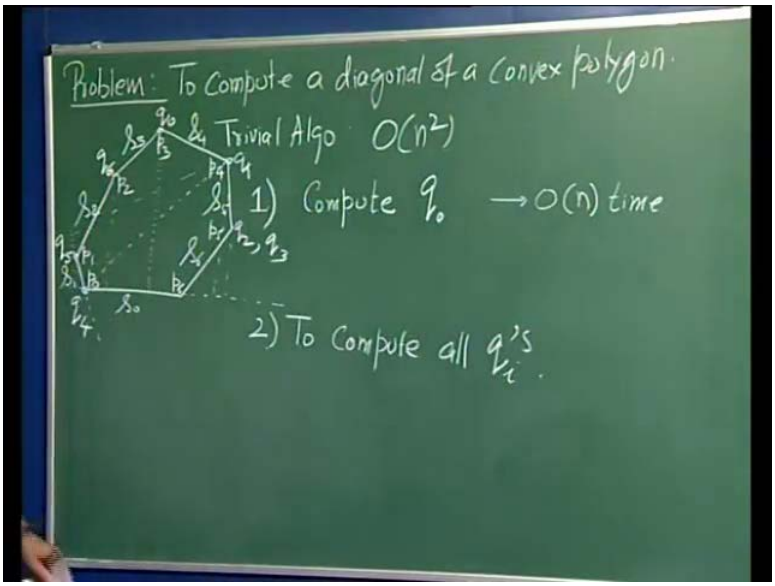
Computers Algorithms - 2
Prof. Dr. Shashank K. Mehta
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 31
Geometry III

(Refer Slide Time: 00:30)



Hello, today again we are working on geometrics algorithm and today's problem is to compute the diagonal of a convex polygon. Compute a diagonal of a, so suppose we are given a convex polygon, and we would like to compute that pair of vertices whose distance is maximum. So, for in this case perhaps it looks like this is the diagonal. Now, the trivial way out is to be consider all pair of vertices and compute the distances and pick the 1 which has the maximum distance. So, the trivial algorithm will cost as order n square, because it takes only the constant amount of time to compute the distance of 2 points, we wish to do better than this.

So, the way we will approach is we will first try to compute the point which is furthers from each of these lines segment. So, let us just label this segment as s 0, s 1, s 2, s 3, s 4, s 5 and s 6, now the perpendicular drop from this point on this line is maximum compare to perpendicular drawn from any other point. So, this is the point which we will call q naught, we are going to also label theses vertices, so let us keep them label p 0 p 1 p 2 p

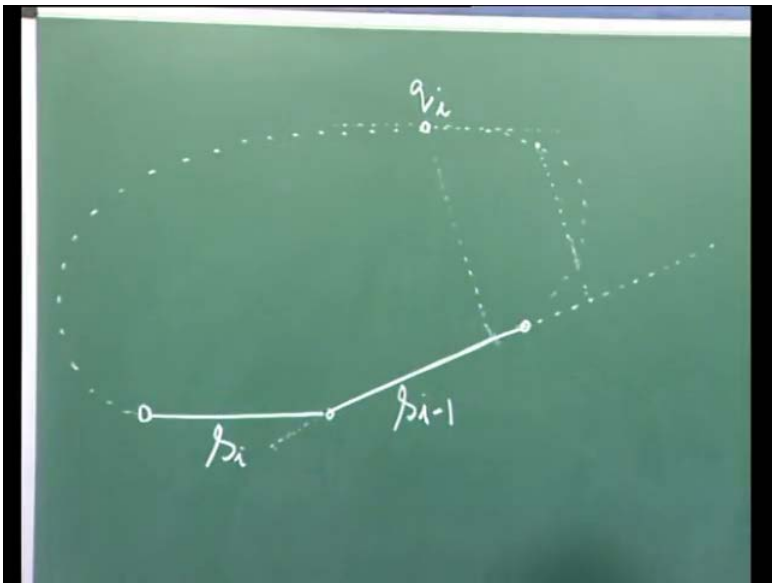
3 p 4 p 5 and p 6 and this p 3 is also my q naught the point has maximum distance from segment s naught.

Let us just go with this picture for a minute, let us look at this segment and it appears that this is this has the perpendicular from here is perhaps this way and this is going to be largest. So, we will call this as q 1 perpendicular from this will be for just of here will be the longest from the segment as 2, so this will be q 2. Perhaps that may be true that the perpendicular from s 3 is maximum from the same point, so may be this is also q 3 and so on.

Can we perhaps finish hopefully this q 4, because that seems like be for this from here, and this is q 5 and perhaps this is q 6. The purpose for working with this diagram is to give some indication which seems that if we label our segments in a clock wise order, then our q points are also in anti clockwise order it is possible that some of them may be to the same point, but they appeared to be also ordered in the same fashion.

Now, what we are going to do is we are going to show that this is the case and later on we will use that fact. Now, so the first step I would like to do is to compute all these q is so, the first step is to compute q naught well this is not too difficult. All we have to do is draw perpendicular from each of the points on to this line, and measure the length of the perpendicular and pick the 1 which has got the maximum value and this can be done of course, in order n times. So, this is not a problem, now our next task is to compute all q i s and we would like to do this efficiently, for this purpose I would like to establish that the q's are also increasing in the same order as our segments are. If we can prove that, that will make it very easy to compute all the q is efficiently, so let us first try to prove that q's are in the same order.

(Refer Slide Time: 07:09)



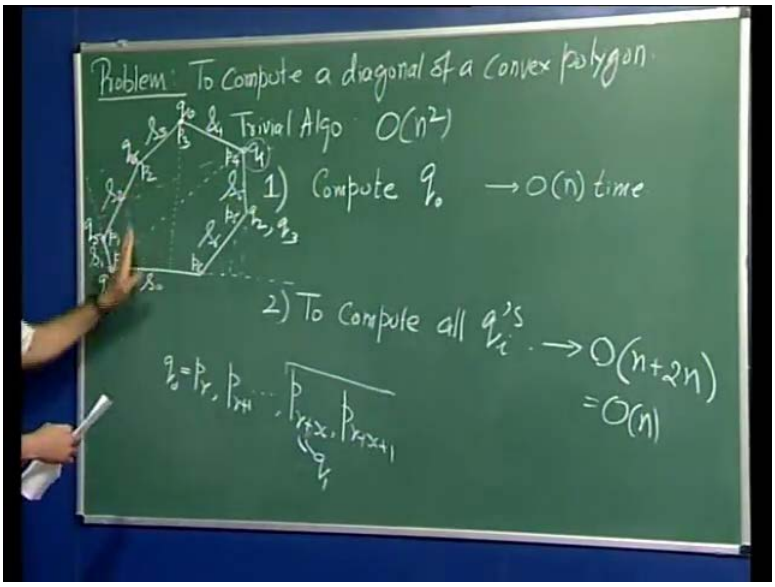
So, let us come to that problem, let suppose we have a some segment s i with end points as shown, and let suppose somewhere here we have a q i the point which has got maximum perpendicular distance. So, if I draw a horizontal line from here, what we can say is that all vertices of this polygon must be between these two parallel lines. They can not be beyond this and of course, can not be beyond this point, let suppose the segment s i minus 1 this would be s i minus 1 is this.

Now, let us partition imagine this is the rest of the convex polygon, let us partition the points of the polygon into two parts, those which are on the right side of this point and those which are on the left side. Now, consider we want to find out what is actually q i minus 1. So, we will consider this line, the line passing through s i minus 1 and we will have some perpendicular does not has to be the same point say this is the perpendicular.

Now, what we notice is that the entire set of points of the right side the polygon, are below this horizontal line. So, there any point here when you draw the perpendicular cannot be greater than this, because this is coming closer to this line. That is to say the horizontal line and this line are coming closer as we approach this side, so the perpendicular from here will only give a value less than this value. Hence, q i minus 1 cannot be on this side of the a polygon, it has to be either same point as this or on this side. So, what we conclude is that our q i minus 1 is anywhere from this point on this, or

further on this side. And that is how we wanted to establish that as we take the successive line, segment the corresponding q i will also move on this side.

(Refer Slide Time: 10:33)



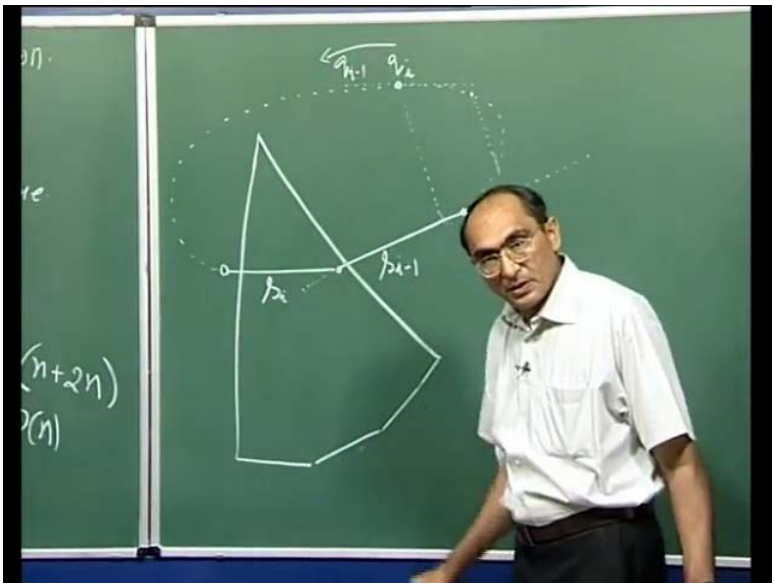
Now, let us talk about the second problem how to compute all the q i, now we know that this sis q 0 this is segment as 0 this is q 0, and this is my s 1. So, what we have just shown is that q 1 has to be either here or here or here, it has to be going down this way. So, what we can do is we consider this line and determined the perpendicular distance from this point namely q 0 on to this and then we compute distance from here, until we find the farthest point will come to this point and we find that the distance has decreased.

Hence, we determine that this is the maximum, and we label this as q i when we go to the next line our search will begin from here from q i and will compute this and this. Now, in the process in the search for each q i, we will compute distances from each point starting from the previous q i. So, if we have q 0 equal to p r then we are going to search distance from p r p r plus 1 up to p r plus x p r plus x plus 1.

Well this would be q 1 because we have to go 1 further point and then determine that this is the point which we are looking for, in this second class we will starts search from here and proceed. So, if we discounts these two from our calculations, then what we notice is that we performs search from each point exactly once. If we discount these 2 computations then we do exact search from one each point exactly once. And we are going to do n such searches, so there will be two more searches per search. So, the entire

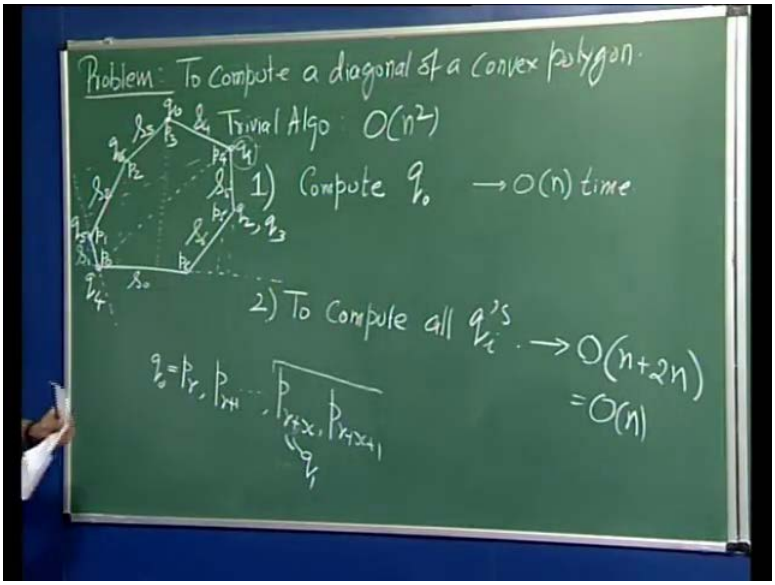
computations will cost as n plus 2 n which is order n, so this is happening simply because we know the direction in which we have to search, we do not have to go backwards, we are to just proceed forward from each point. So, considering this also takes linear amount of time, so does this we have all the q's in linear amount of time.

(Refer Slide Time: 14:12)



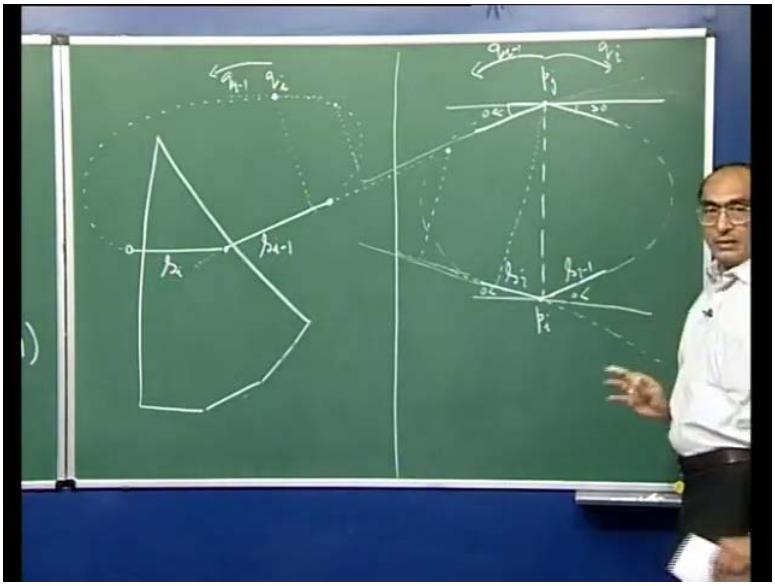
So, can be a point be maximum from 3 line segment, so this is maximum from this and this all the 3, so it is possible no you do not have to do that.

(Refer Slide Time: 14:35)



The thing is that you just start from the previous point, and you will find that the next one itself is decreasing, then you will conclude that this is also q 2. Then for q 3 will again start from here and go to this and you find again it is less than this, so this is also q 3. So, it does not matter how many q is cons or concentrated on the same point, but you will have to do up to one point beyond the point that is designated as q i. So, the total number of searches, that we perform, so total number of perpendicularly compute and we measure are only n plus 2 n, so the total thing is order n. So, the last step in this process is that we are arm with the information about all the q i's how do we find out the diagonal from this information.

(Refer Slide Time: 15:45)



So, now, let us go to generate a problem, where we have a some convex polygon in which the diagonal is this line where we have p i here and p j there, and of course, the rest of the convex polygon is here. Now, one thing we can do is we draw a perpendicular to this line segment and over here will do the same thing, we have segment here is here, it does not look terribly set. May be I should just draw again here is a perpendicular line to this segment, and we assume that this is the diagonal.

So, this is the farthest pair of vertices of the polygon, now there are segments passing from this vertex and there are segments going from this. Now, why should they go below this lines here which is perpendicular to our diagonal, the reason is if line say this side of the polygon were parallel to this or further away from this. Then our distance to the next

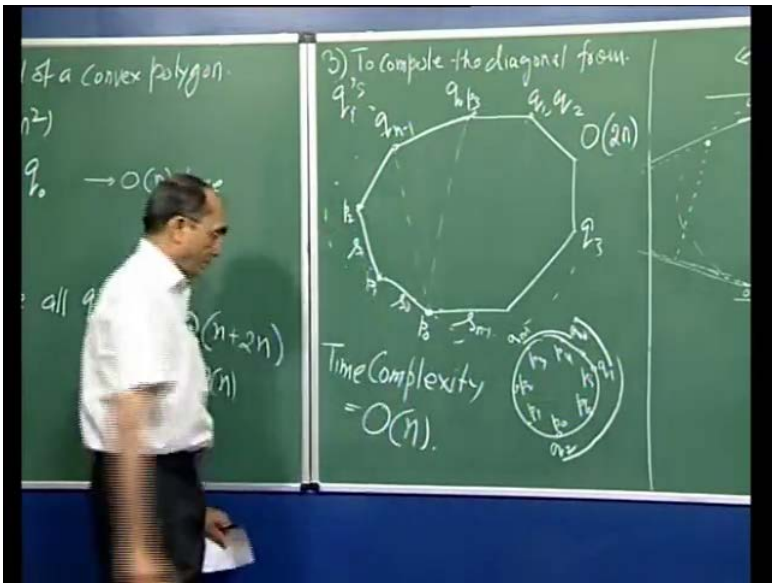
vertex, that is this vertex which is p j minus 1 would have been further from p i then but p j is, so it is not possible by our assumption this is the diagonal.

So, this has to be an angle strictly greater than 0 strictly larger than 0, this is inside the segment by the same token all these angles must be greater than 0, every one of here is greater than 0. Now let suppose, so based on the labeling we have done say this is s i minus 1 and this is s i, what I am going to show is that q i can be either p j or some vertex on the right side of the diagram. And I will try to show that similarly q i minus 1 can be this vertex or something to the left of the diagram.

So, how do we do that q i is a point which has the maximum perpendicular on this line, the line passing through the segment s i. Let suppose first we consider the points on this side of the polygon, let us drop a perpendicular from here on to this on to the line passing through s i. Once, again it is a very similar situation we have this horizontal line and we have this line passing through s i, this is a non 0 angle they are approaching each other.

Hence, every point on this side of the polygon must be inside, well it is bound to be inside this and this is not it, because these are the two line segments and this is a convex a polygon, so everything is inside this. So, no point inside this no point we chose can be farther than this. So, if we have to choose only this part of the polygon, this is of course, the farthest point. So, q i can be either a point namely p j or it could be something on this side. So, we conclude that q i is from here onward somewhere, because of the symmetry of the picture by the same token q j sorry q i minus 1 to be either this vertex or something on this side of it. In other words the 2 q’s are split around this what could be at of course, here that is the possibility. Now, this gives us a very easy way to determine the diagonal.

(Refer Slide Time: 21:37)



So, now, we have third step, step 3 is to compute the diagonal from q i's, so we have let us say this is the picture. This is my p 0 p 1 p 2 s naught s 1 s 2 and so on, let us consider first p 0 and consider these two line segments, we already know our q, so this may be q 0 this is q 1 may be this is also q 2 may be this is q 3 and so on and this is how it goes. We will consider this is s n minus 1, so will consider q 0 and q n minus 1 somewhere here.

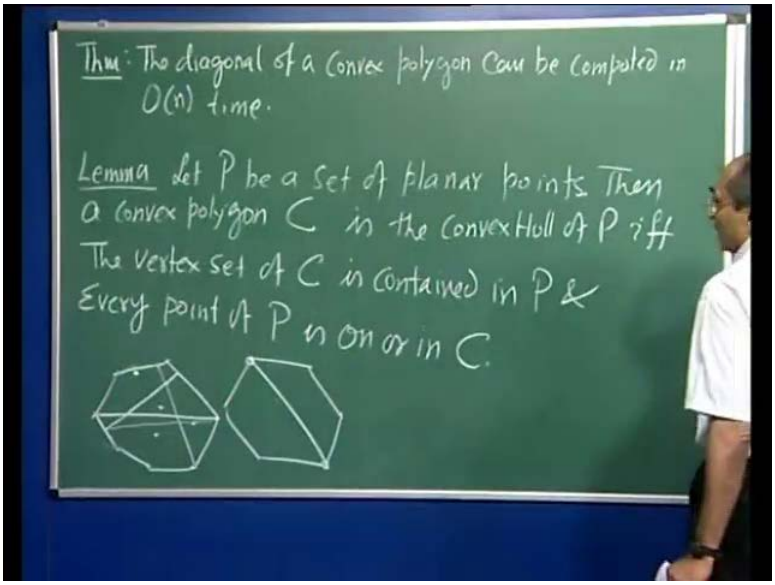
So, this is say q 0 negate, this is also well just for the sake of drawing a picture here I am let us say this is my q n minus 1, this could be of course, at the same point. So, we know that if the diagonal passes through p naught, then the other side of the diagonal must be any vertex from here to here, because what we have just shown is that. If this was diagonal the other vertex of the diagonal, well in the picture it looks like more of it candidate, but that is not the point. If this was the other side then what we notice is both q 0 and q n minus 1 on the same side which is not allowed.

So, the better be the case that the other side either this or this, so we will just compute the distance from p 0 to q 0 and p 0 to q 1, and find the larger of the two. Well, continue to do the same thing then will do the same thing from p 1 will compute the distance from p 1 to q 0 and every successive vertex until we reach q 1 because we have the 2 segment as s 0 and s 1. So, the corresponding span will be q 0 to q 1, then from q 1 to q 2 all the vertices will measure the distances from p 2 will continue to do that.

So, the picture is we have this we have a p 0 p 1 p 2 p 3 and so on, p 4 maybe p 5 p 6 and will go from q n minus 1 to q 0. There could be several vertices in this range will computed the distances from p naught, then we will measure from q naught to q 1 this is q 0 to whatever may be several vertices in between q 1. Then we measure distances from q 1 to q 2 their distances all these vertices will be computed from p 2 and so on.

And pick the largest because the diagonal has to be the largest of all, and we know one of these pairs is the diagonal, how many computations are, we doing here. Notice that if there are once again what we are doing is if we start from this end go to q naught, then again we are starting from q naught and go to q 1. Then we are going from q 1 to q 2. So, at most same vertex is computed is twice, hence the entire task should take order 2 n which is of course, order n.

(Refer Slide Time: 27:18)



Hence, the entire algorithm time complexity of the algorithm is linear in the number of size of the, so our result is theorem is that the diagonal of a convex polygon can be computed in linear time. Now, we have a simple consequence of this result, now one of the characterization of convex hull of a set point is that, let p b a set of planar points. Then a convex polygon C is the convex hull of p if and only if the vertex set of C is contained in p.

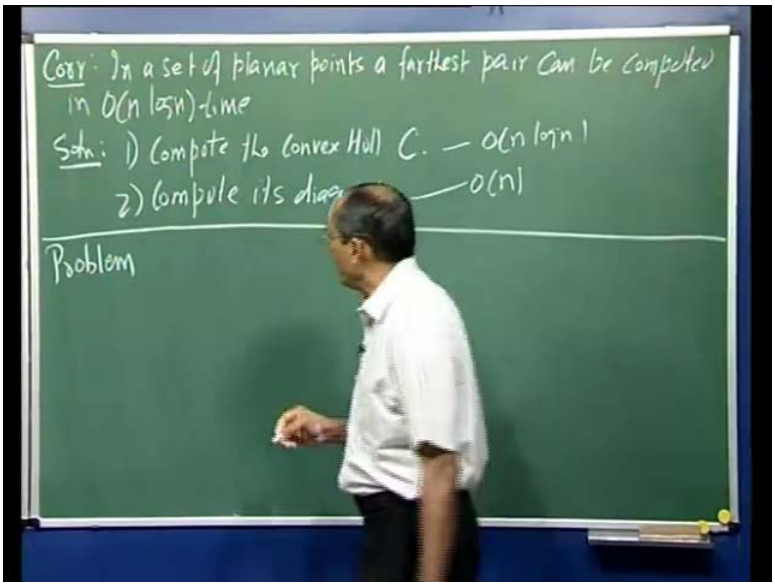
And this is all vertices of c are actually members of p and every point of p is on or in C. So, this is one possible characterization of the convex hull of a point set, now suppose

we have a certain set of points here, and we want to find out a pair of points which are farthest from each other. So, suppose we compute the convex hull that is here and the let suppose we say the farthest pair of points of this points set is say somewhere from here to here or say here to here.

Then what we notice is that no end can possibly be inside the region the region is and we can always extend it. And we can find a longer line, which ends in the two points on the hull somewhere even if it is not the vertex, we can always extend this. So, we can always show there is a line segment which is longer than such a segment and ends on the two points on the hull. Then one can also show that this line segment, when we move this point to this or to this and 1 of the sides it will increase or at best it will remain equal.

So, let us say this is larger and by the same token when we move this to this side, in this side one of the sides it will further increase. So, thing is that the longest line segment touching two end two points on the hull, will be either equal to or less than the diagonal of the convex polygon. So, this is the longest point inside this region the longest line segment inside this region, this is the diagonal. Hence, the farthest pair of points must be these two cannot be anybody else, since the computation of convex hull takes n log n time or n h, so we can also claim.

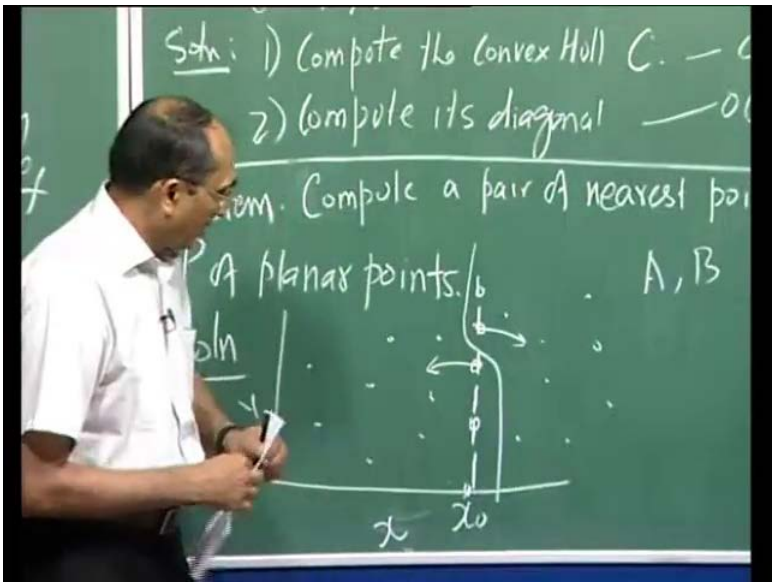
(Refer Slide Time: 32:28)



So, we have a corollary that in a set of planar points a farthest pair can be computed in order n log n time, so solution is first compute the convex hull C and two compute it is

diagonal. This takes order n log n time and this takes order n times and that will be the answer of the problem of computing farthest pair of points from a set of times.

(Refer Slide Time: 33:55)

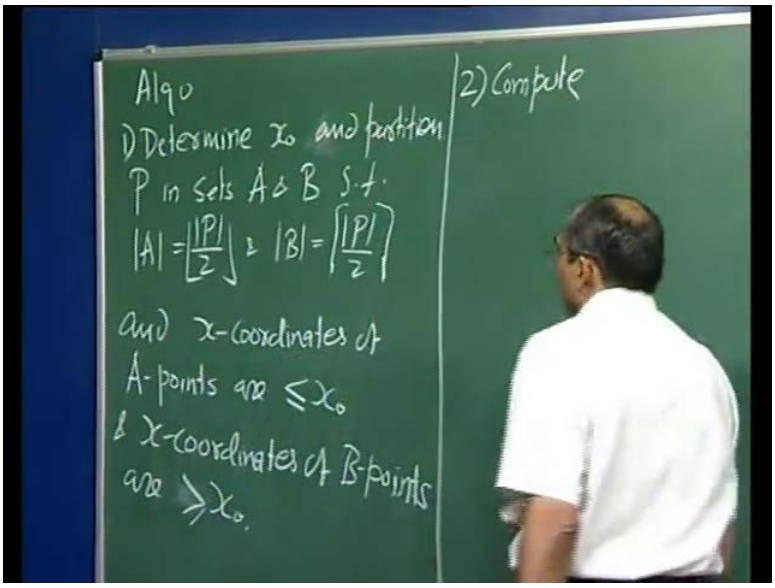


Now, in the remaining time we are going to address another problem of similar nature, this time I want to compute a pair of nearest points from a set of planar points. So, in a compute let us say compute a pair of nearest points from a set p of planar points, so this is a completely different problem although very similar sounding as this. This time we are going to take a very different approach, this we are going to use a divide and conquer approach.

So, let us first partition the set, so let us say we have plane x y and we have the points here, then determine a coordinate x equal to x naught. Let us say may be this one which may pass through more than one point, but the idea is that we split the point set into almost equal parts. So, we will split it into set a and b such that size of a is equal to mod of p by 2 flour and size of b is mod of p by 2 sealing, that we can do.

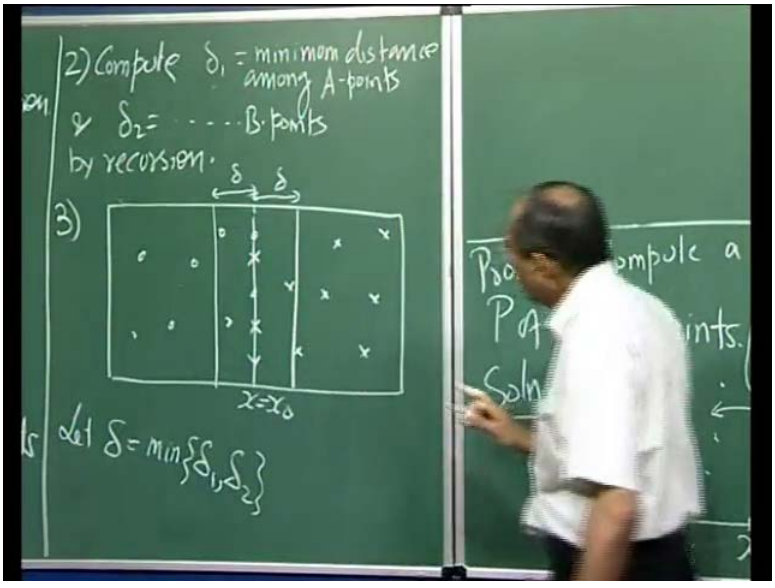
Now, notice that such a situation may arise that you have several points on this, so you may have to assign some of them this side and some on we should may need to put on the other side that does not bother us. So, now, we begin with, so may be you have actually a partition like this, but we make almost exact division on the set of points. Our step two will be to inductively compute a pair of minimum separation on this set, and similarly we solve the same problem on this set.

(Refer Slide Time: 36:53)



So, our algorithm goes as follows first determine x naught and then and partition p into sets a and b such that size of a is equal to p by 2 flour and b is equal to p by 2 sealing. And x coordinates of A points are less than or equal to x naught and x coordinates of B points are less or greater than or equal to.

(Refer Slide Time: 38:12)



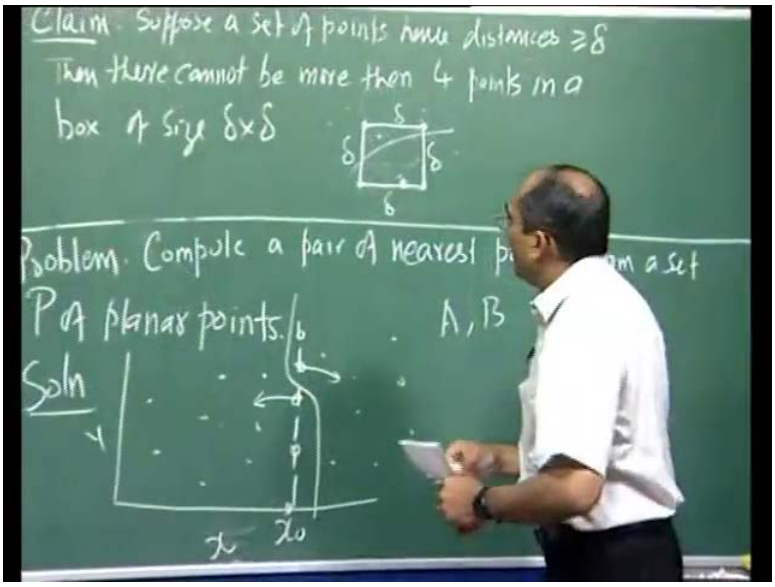
Step two compute although we can actually compute the actual pairs, but for simplicity I am just assuming us computing only the distance between the nearest pair of points. So, we compute the minimum distance delta 1 minimum distance among A points, and delta

2 from B points by recursion. So, when the set is only of pair of vertices, we can directly determine that otherwise this recursive call each time we are reducing the size to half.

So, we can do this finally, the picture is a smallest, let us say we have a bounding box in which all the points are lying and we have x equal to x naught line here. So, some of the points of set A or these and the set B points are let us say are here. Now, the minimum distance pair in this entire set P is either the pair corresponding to delta 1 or delta 2 or the pair is need up of one point from set A and one point from set B. And notice that if I drop, so let us say let delta B minimum of the 2 min of delta 1 and delta 2. Hence, to pair wise distances of set a is no more than no less than delta and similarly here.

So, now suppose I draw parallel lines to x equal to x naught with distance delta from the middle line, any point of this side beyond this point can never have distance less than delta to any B set point. By the same token no point in considering any of these points for in it is more distance, so we have to only focus on this strip of width delta plus delta. Now, we are going to show how to determine instead of computing pair wise distances, between every x and every o point we can do a better job.

(Refer Slide Time: 41:58)

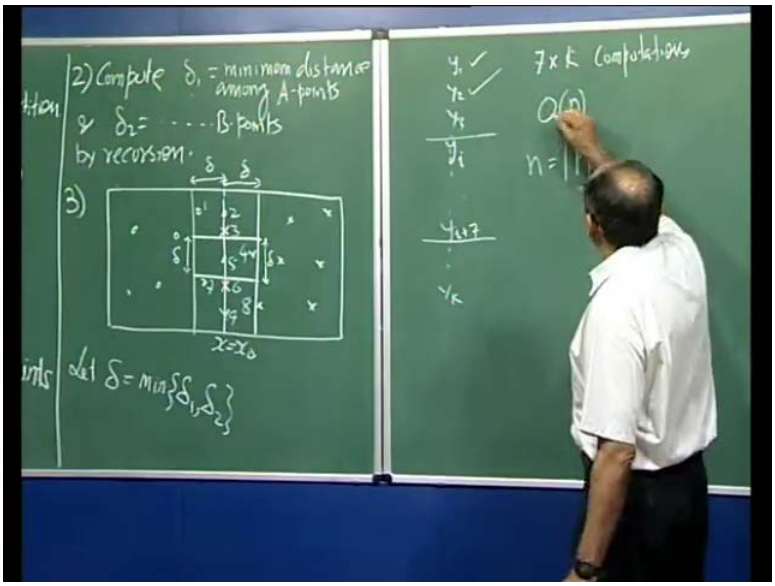


And this is as follows A claim here, suppose A set of points have distances greater than equal to delta, then their cannot be more then 4 points in a box of size delta by delta. So, suppose we have a box of size delta, delta a square of size delta and all points the given set of points, there cannot be more than 4 points inside the box or on the box. Now,

suppose I put a point here and here, and here, and here, then I cannot put any more points the reason is this will cover this region and you cannot have any other point inside this. Similarly, this will cover this region there will a region here there will be a region here this will cover the entire area of the square you cannot put any extra points.

On the other hand, suppose we put a point on this side, then that will cover this region. Now, if I put a point this point that will cover this region I will be able to put one more point, but I cannot put more, because this covers everything and this point will cover this region that will cover the entire space of the square. If I put any point in the interior then that will cover much larger region and only two points will be allowed to put, one can actually case by case and show that you cannot put more than 4 points of it is.

(Refer Slide Time: 44:29)



Now, let us consider some region like this, where this is a of size delta as well then we know that there cannot be more than 4 x's inside this, there cannot be more of than 4 o's inside this. Hence, totally there can be at most 8 points in this suppose we sought these points by their y coordinates, so let us take they are 1 2 3 4 5 6 7 8 9 and so on. Let us write down those points as y 1 y 2 y 3 y k, only those which are inside this region.

Now, we notice that if we look at some y i and we just look at the points below y i which can be at distance delta or less. My claim is no more than next 7 points that is starting from y i at most y i plus 7 only these many points can be within the delta re distance

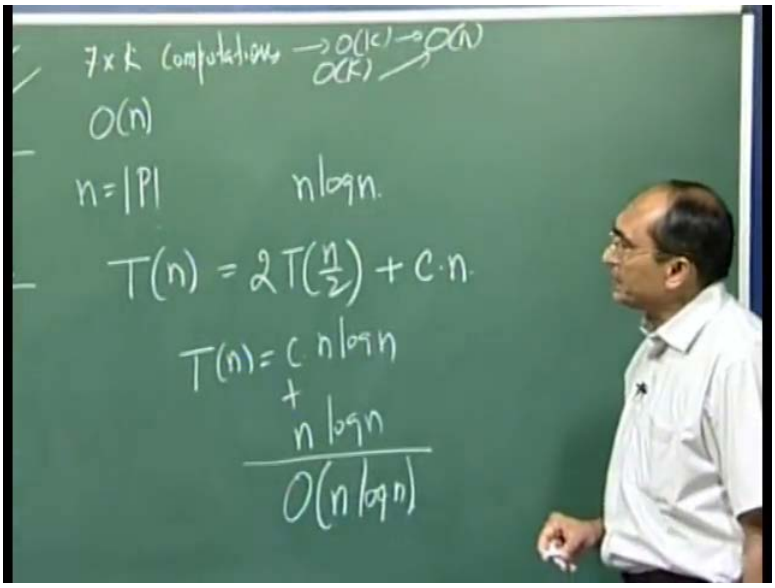
from that point. That is because if the point that we are considering y i is on this line somewhere on this line.

We know that inside this box there are at most 8 points including that point there can be 7 more points inside this. So, and anything below this can not be less than delta distance, if i cannot be even delta distance. So, if i in really in care to find out what potentially what points can be at that is distance delta or less than delta we need to only consider up to y i plus 1. So, starting from y 1 we will only compare it is distance up to y 7 y 8 and then from y 2 we will go up to y 9 and so on.

Hence, we will have to on addition to the initial 2 computations of delta 1 and delta 2, we will have to compute in step 3 totally 7 into k computations in the entire process. And the shortest of all these distances if it is less than delta, then that will be our answer if none of these is less than delta then we already have delta as our minimum distance, so that will be the answer. So, in step 3 we are now finding the distances across the two sets in this fashion, now notice that this costs as at most order n, the reason is k cannot exceed n, and n is of course, mod p.

So, let me go ahead and show that n is mod p, now there is one problem and that is sorting these points, now if we repeatedly sorting these points in each alteration, in each recursion that will be very expensive, but that does not have to be done. What we can do is we sought all the points according to their y coordinates once, and whatever subset that we want to reproduce here in that order we can simply scan through the sorted list. Check whether it is in side this by looking at their coordinates and output them. So, initially we have to short, but subsequently extracting this set will take order k time, this is k order k time and the extracting this set is also order k which is order n.

(Refer Slide Time: 48:58)



So, the entire process as the following time complexity T of n is 2 times T of n by 2 or without loss of generality I am assuming n is power of 2. So, we have this plus in the third step we or going order n time, so some C times n, in addition to this overall cost will be n log n for sorting with respect to y coordinates. If we simplify this we get T of n equal, to you know this is C n combined cost of this side will be C n and so on, and there are only log n levels. So, if this is cost as some C times n log n plus and overall global cost of n log n, hence the total cost is n log n, to compute the nearest pair of points in the sets. So, this is a in algorithm based on divide and count, so this is where we close.