Computational Geometry

Prof. Sandeep Sen

Department of Computer Science and Engineering

Indian Institute of Technology, Delhi

Module No. # 02

The Maximal Points Problems

Lecture No. # 01

2D Maxima

So, welcome to lecture 3. Before I begin - are there any questions from what we did in the last lecture? I think we essentially ended up proving the art gallery theorem, so any confusion, any doubts? So, that me let me proceed with the new problem, it is it is a fairly simply stated problem and may not sound particularly useful, I do it, I will use that problem, yes. One is for three-dimensional objects, what will be, I mean what will be the number. So, the question is that if the art gallery problem is posed in three dimensions what happens?

So, clearly the problem becomes even harder, I mean so there is no question of having a fast polynomial type solution, if you are asking about essentially the results about you know, do you have a tight bound like this one we have, I must confess that I do not know and I have I doubt that there is any such type result in the three dimension. Another question is suppose the at the rate is like a unlike a normal gallery. Yeah, that is how you pose the problem as an orthogonal polygon, simple polygon initially; pose the problem like an orthogonal simple polygon.

If that is the case, if the result.

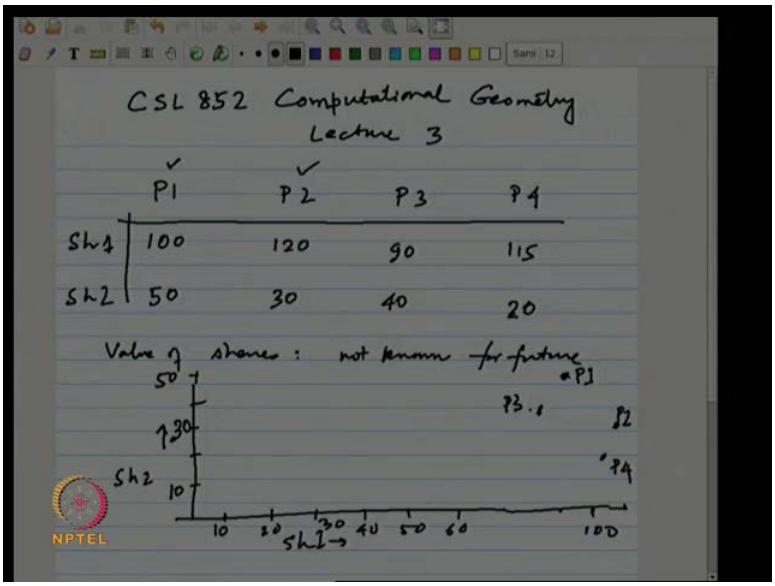
No, the results remain the same.

Let me let me let me retract that, so the results may not remain the same, for the simple reason that the lower bound construction, that is you need at least so many, so I will take it back, so the result would differ by small amount, yes I agree with that, if it is a polygon.

So, coming back to this new problem that I want to discuss today, so let me just give some kind of motivation before I pose the problem. So, it is like this, you know suppose you know, so people do own you know shares from different companies, right and you know the shares are priced you know according to the market in every day the price changes. So, suppose there are you know ten kinds of shares available and there are many people who own such shares and the pricing, the the cost of less price the shares change every day, but at on any specific day depending on the price of the share some person can claim to be the richest person right, depending on in what the share, how many shares we have, how many share values, what is the share value and so forth. So, then, if some person is is a richest person one certain day, the the then next day you know because the change in share prices, you know someone else could become the richest person.

Now, given a population, if you want to keep track of the fact that who are the potential richest person, what would you do. So, what we do in the sense that, you know who the potentially richest person are. So, how do you approach this problem? So, I am not trying to do any prediction, I only trying to get a set of persons who could be the richest person on some day depending on some values of the shares.

(Refer Slide Time: 04:33)



Let me give an example, suppose you have this you know four persons you know P 1, P 2, P 3, P 4 and let us say that only two kinds of shares, share one and share two, alright.

So, person one owns let us say you know 100 shares of this and let us say 50 shares of this, person two owns may be 120 shares of this and may be 30 shares of this, person three owns let say you know 90 shares of type one and let us say 40 shares of type two, person four owns let say 115 shares of type one and 20 shares of type two, ok.

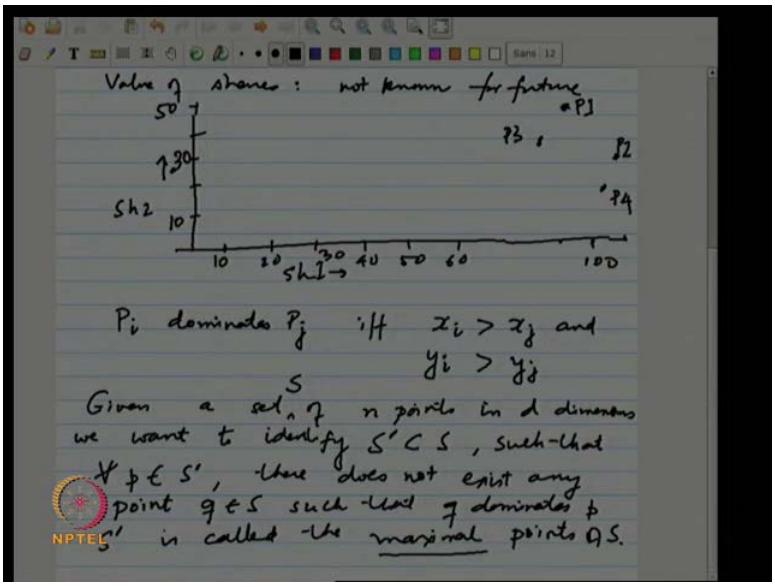
Now the value of a share is unpredictable, let us say you not know on you know you cannot even predict the future, not known for future. So, if you if you have this kind of data, if we have this sort of data, then can you say anything about who could be the potentially the richest people. So, you are saying that p 1 and p 2, why are we excluding p 3 and p 4? Because p 3 has 90 shares 90 of but p 1 has got more than more shares than… So, whatever is the price of the share on a certain day, p 1 because it has more shares of both types than p 3, so that person you know what ever be the it will always kind of be richer than the person three and likewise the number of shares own by p 2 of each type exceeds the number of shares own by p 4 of each type and therefore on any particular day whatever be the value of the share p 2 will always be richer than p 4. So, therefore, p 4 and p 3 are certainly not among the potential richest person, only p 1 and p 2 and depending on the price the share may be on a certain, the p o1 is richer than p 2 or vice versa, ok, but then certainly p 3 and p 4 cannot claim that position on any any day, because they are, they have fewer shares of any type than another person in the set.

So, this kind of relation, so I have I have discussed only with respect to two kinds of shares, so I can, you know if I if i draw this, you know you know you know is a is a graph, so let us say this share one and this is sharing two, right, so let us say some obtain 20, 30, 40, 50, 60, let us say about 100 and here we have 1, 2, 3, 4, 5 let us say up to 50, right. So, I can actually plot this, so person 1 has 100 shares of type 1 and 50 of type 2, so person p 1 is a point here, person two has you know let us say X coordinate 120 and Y coordinate 30 right, so some where here and 20.

Person p 3 has 90 has X coordinate and 40 right, so it will be somewhere here; the p 4 has 115 and only 20, right, so just a minute, so then now 150, I am sorry, so this was incorrect, that that was 30 right. This is 30, so 100 and p 2 is 120 and 130, so it is about here, this is p 2, this is p 1, this is p 3 and p 4 is 115 and 20, right, so somewhere here as before. Sorry, the example is of all the points are clustered on this end. Now, geometrically you know what is happening is p if you look at p 1 and p 3, both

coordinates of p 3 are smaller than p 1, right and both coordinates of p 4 are smaller than p2 right.

(Refer Slide Time: 09:30).



So, in this relation where both coordinates are smaller we have a name for it, so you say dominate. So, we say p i dominates p j, x i is greater than x j and y i is greater than y j, right now there is of course this a thing that what happens if it is equal to, let me ignore it for the time being and that is not get in to that, it can also be taken care of with the appropriate definitions.

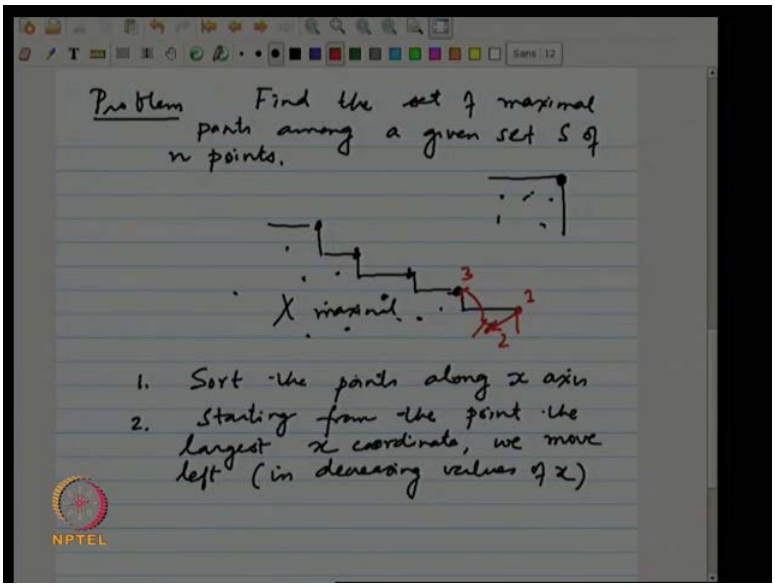
So, say we are only having this inequalities, so p i dominates p j only if and only if both coordinates of the i th point dominates the, the dominates the coordinates are, is greater than the coordinates of the other point. And this a course you know where this definition extends to high dimensions, where you can have an three dimensions of depending on how many kinds of shares it has, these kinds of shares go to three dimension takes place the same information, right. So, then what are we what is this problem is about? So, this problem is about, the following that given a set of n points, so in the special case on a plane, otherwise in the in d dimensions. In d dimensions we want to identify, let us say call this, call this set s, identify let us say s primes sub set of s such that for all points in s prime there does not exist any point q belongs to s such that q dominates p, right.

So, those points from this sub set s prime, so those, that set of points, so in in the given example, you know p 1 and p 2 basically comprise that set s prime, so because p 1and p

2 they are not dominated by any other point. So, this is what we are calling this s prime and the the name for this set s prime, so s prime is called the maximal set of point's maximal points.

It is not dominated by any other point, right, so why you are saying only s. So, the question is that why I was chosen q to be in s, it cannot be dominated by any point in the set, not just s prime, sorry s minus s prime, you know I cannot let it dominated by any another point, no way. So, a maximal point is the one that is not dominated by any point in the set right and the problem is, the (( )) of problem is, to given a set of n points, find the set of maximal points.

(Refer Slide Time: 13:24)



So, problem, find the set of maximum points, fine that is a problem, this is the problem would like to solve algorithmically and you know of course has with with a algorithm as fast as possible. So, from the definition of the problem, what would be the straight forward solution? So, the straight forward solution has, some one just point it out is n square or whatever, n chose two y, I can take every pair of points and find out if it if one point is dominated by the other, any point that is dominated by another point cannot be a candidate for the maximal and it is get thrown out. So, any point that survives all these tests will be maximum and clearly at least one point must be maximum, right.

The point having the highest x coordinate points, having the highest y coordinate, those points must be the and this this can be the same point, right. So, what I am saying is that

essentially your set could be like all the points are here, so this is the only maximal point, that is one extreme, another could be that you know I have things like this, right. So, this kind of looks likes a staircase, so all these points must be maximum, because they are not dominated by each other, right.

And any other point inside this stair case you know not maximum clearly, so this is thus also the structure of the solution right, I mean it is like a falling staircase in the increasing x direction. This just look at it carefully, so having doubts. that any any other point inside the staircase must have, must be dominated by at least one point, must have both x and y coordinates smaller than some other point in the set.

So, for example this point, so this point and this point you know clearly the the x coordinate of this point and y coordinate of this point is smaller than x and y coordinate, is more over the x and y coordinate of this point is also smaller than x and y coordinate of this point. So, a point may be dominated by more than one point, but certainly it is dominated by at least one point and none of these points in the stair case can be dominated by each other, because it is in the increasing y direction, a sorry x direction and falling y direction.

So, this is the structure of the final solution and we want to essentially identify this. And this straight forward simple solution is, take every two point, compare their x and y coordinates and if one is dominated by the other just throw it out. That is to ensures to well that is, but you know maybe we can do better. It can be done or analog mean the x coordinates and then take the point maximum in the y coordinate and start going back for two point is x coordinate until we find a y coordinate .

So, I you you know I think you knew the solution, because you have seen this problem before right if I am not wrong, it is obvious, fine it may be obvious. So, let me iterate this one proper solution which seems to be a nice and fast solution. So, you know the the is that you know you sort the points along along x axis and then starting from the point with the largest x coordinate right, we move left I in decreasing value, values of x, right, so this is what we we do. So, in this particular case, you know suppose this is the highest x coordinate and there are many other points, the point here, point here, so what we do is you know from here we this is the next point that we must consider in the decreasing y coordinate.

Now, because we considering them in the decreasing x coordinate, we know that any point that we encounter later must is already dominated on the x coordinate, it now is the question of whether or not, the y coordinate will also smaller than this point, so if it is so we discard this point, so this point gets discarded, then we move further and we come here. So, this has is smaller x coordinate then then one, at this is the two, this is three, but three has a larger y coordinates, so three survives and three will always survive because all other points have smaller x coordinates, so no other point can dominate three, alright.

So, this is the way we progress and so what is the information that we require to maintain? So, when we are here, I know that this point has a smaller x coordinate than all the points we are seeing so far, so the only thing we need to find out if there is any of these points that we have seen till now has a larger y coordinate, it means the only information that we need to maintain is the largest y coordinate of the points scanned so far. We are scanning from from in the decreasing x direction, so this is usually done right, so staring from the point you largely move and maintain the value of the largest y coordinate.

So, if the y coordinate its. So, let us call it something, say y max, if for the current point p x sorry, y of p is less than y max, then p is not maximum, else p is maximum, that is all. And what is the running time of this? Well, in step one we sort, so sorting will take whatever time it does, let us say n log n. If you are using comparison trees, so this will take n log n and then the step two, which is the iterative step, you know we look at every point one after the other, what is the information that we we maintain? Just the maximum y coordinate, y max, y max have to be, may have to be updated.

So, in fact, I missed out something here, so else if y else p is maximal, which means that we must update, so y max must now we set to the the y coordinate of p, because that exceeds the previous y max. So, the only change that we have to make is is this this one value, so it is basically constant amount of work for every iteration for and therefore for for all the n points we need only a total of, so for each point additional order one implying total order n.

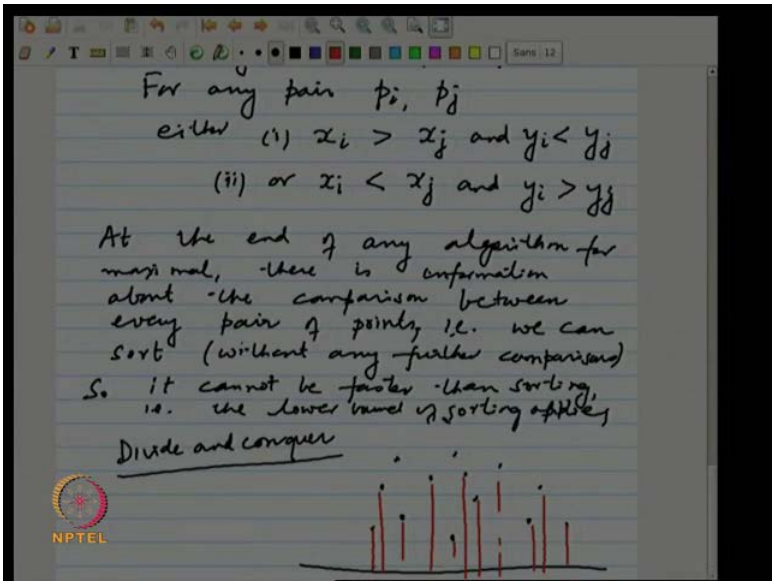
So, the whole thing is actually dominated by this n log n to the first step is the dominating step, I mean interns of algorithmic complexity and the second step is only linear time. So, if someone had given you these points in a sorted manner, then you

would only need alternate time linear, so nothing can be better than linear, but of course the first step is n log n, you can do better than this, one sorted points, right.

So, answering a question like can you do better means that we have to somehow argue against all possibilities, right, in all all all the intelligent people in the world. Let you know someone says no I can design a linear time algorithm, you know then I have to somehow we able to argue without even knowing what that person knows, no, impossible, you cannot design anything faster than something, which is essentially we can we do better means we have to prove if form a lower bound, right, can we do better right. So, some kind of lower bound argument is necessary, right, which must hold against all possible you know possible strategies of designing algorithms, right.

So, for this particular thing let me give you an argument that we cannot do better than n log n. So, here is the idea, so the idea of so this is the notation for big omega is rotational lower bound, right.

(Refer Slide Time: 24:52)



So, will construct some kind of an input, so construct an input x i, y i, i it one and n such that x i plus y i is some constant c, large enough constant. Now, what does it mean? In this particular input how many maximal points? So, I will just so, actually let me not even bother about the large enough, you know everything could be let us say between zero and one, that is also any any constant, I do not require this. How many maximal points are their? All points are maximal, right, because from this equation x i plus y i is

constant if we take so far any pair p i, p j either x i is greater than x j and y i is less than y j or the other way around or x i is less than x j and y i is greater than y j. So, no point dominates another point essentially and therefore all the points are maximum, right.

So, if we have if we design an algorithm, I mean what ever be the algorithm for maximal, that algorithm should be able to detect this condition, if it is a correct algorithm, it should produce the final answer, ok. Now, produce the final answer, it should have been able to conclude that for all players of points, either this is true, either this is true or this is true, what does it mean? It means that it actually must gather information, enough information about all points so that it knows the is knows the answer whether you know x i is greater than x j or x i is smaller than x j, because without figuring out this, it it will not know whether or not it dominates. So, at the end of the algorithm, whatever be the algorithm for maximal, the algorithm should have gather this information or deduce this information, which means that we have enough information by which we can compare any pair of points and therefore with that information, we should be able to sort.

And therefore, the lower bound for sorting should apply here. So, at the end of the algorithm, of any algorithm basically, right, i.e. we can sort without any further comparisons, right. It does not mean that you know every pair of points has been compared, but there should be enough information end end so that it will pick up any two point, we should know either condition, one is satisfied or condition two is satisfied.

Which mean that now with this information one can sort, right, so it cannot be faster than sorting i to a lower bound of sorting applies. So, this even, this simple algorithm is actually up to mark, so we cannot do anything better than sorting and we know that whatever is a lower bound for sorting in whatever model. So, in the comparison model the lower bound for sorting is n log n, so this also n log n.

So, I will just mention one word about the model, we haven't quite actually looked in to the model that we using, I mention in the first lecture that we dealing with real numbers and therefore we actually also we have to deal with real valued functions and so on and so forth. In the normal model of computation, we have only dealing with you know some comparisons, may know perhaps have some addition subtraction and so on so forth, but normally you analyze only with respect to comparisons.

And I have to do this kind of geometry computing, we have to we have to resort to other kinds of arithmetic operations which may be when if you want to find the distance, you may have to compute this complete the square root because the the l 2 norm, which is this you know the root of the some more squares requires last two, not only multiply, was also take square root. Some of these problems presumably could also depend on something like angle computations, you know we do not, you would not rule it at this stage ok.

Right now I do not want to get too much in the model of competition, but you must keep in mind that the you know the the the the available operations you know are you know most of these basic you know arithmetic real valued functions.

So, what exactly we will use I will come to that preferably much later in the course, but I will just I just making this this point, alright, so this is an optimal algorithm for finding maximum. Let me also mention another possibility that because I said you know the techniques that we learn in other algorithm course can also be apply, suppose we have to apply divide and conquer to this problem, right, so how would the divide conquer work? So, let us say divide and conquer, you normally would divide in to two parts, solve the problem in the two parts and then combine them somehow. So, in this particular case, you know we have a set of points in the plane, what could be natural way of dividing this problem into two parts which so let lets go with this that you know we project the points on the x axis. So, I am not drawing all, the everything, some of them and then after the projection we can find the median point and that will partition the point in to two parts.

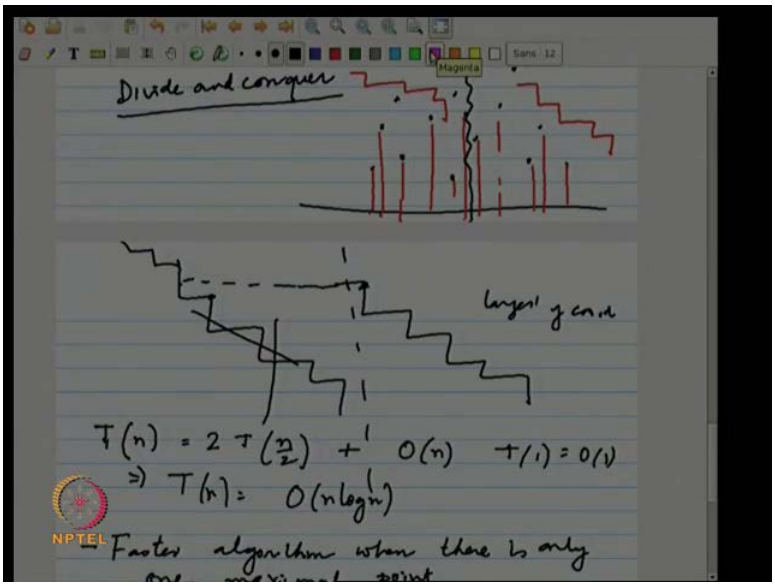
So, that is a clean way of partitioning that there is you know so one part is completely on one side, the other part is another side, of course, when you do some like merge sort you do not do this, you know you actually partitioning any arbitrary way, I can just take any n over two points and take the compliment of those n over two points. Here, this way of partitioning is actually find it by finding the median, I am actually finding a kind of a in separated way, all the points on one half have larger x coordinates than the other half.

And because this is for maximal as well we have seen the the previous approach that it helps us you know to figure out you know if some point can be for maximum. For instance, all the points in the left half, so suppose this is the median, then we know for sure that the points on left half cannot dominate any point on the right half, because they

have smaller x coordinates, right. So, then if I am trying to solve this by dividend conquer, I find the maximal on right half, I find the maximal on the left half, which would look like let us say some staircase here, some staircase on this side, another stair case on this side, right and somehow we need to combine this staircase, right. And we know that the staircase on the right must survive right, if it is maximal on the on the right half it has to survive, it is only the points in the left half that may be dominated by some points on the on the right side.

And that shows easy to figure out exactly which one by just observing that if you know if you have these two staircases is linearly separated staircases, because there is a separation between them.

(Refer Slide Time: 35:26)



Then all we need to do is find out what is the maximum for any point, if the maximum y coordinate to the on the right half is larger than this one, right. So, that means, basically I just extend this and all this is not maximal and anything above this is maximum. So, how do we find this thing, simple right, find the largest y coordinate among largest which is actually this point, we know that it is a falling stair case and then you compare the y coordinates of all these points on left half and filter them out.

So, how much time does it take? Right, so then if I have to write a recurrence, then I can write the time to compute maximal of n points is two times the time to compute maximal of n over 2 plus big o of n, this is your very familiar recurrence. So, if I am not writing

the terminating condition and that implies t's of n equals, so that is another n log n algorithm, right, good. So, n log n is a lower bound and we cannot do better fine, but let me scroll back to this situation, what happens here, the point there is only one point that is maximal.

Alright, so once we have find this point, which is the maximal point, when you compare this point with any other point, it dominates every other point and they go out, gone, right. So, if I have this algorithm that I compare at I find the y coordinate, the maximum y coordinate ok and when I do my pair wise comparisons, I always do my pair wise comparisons starting with the maximum x coordinate and then when I finish this, then I take the next point and then do the comparisons from that.

So, if run that algorithm, it is an n its, it is an n square algorithm potentially, but in this particular situation when this only one maximal how much time will it take to run. Just m comparison or n minus one comparisons will surprise to eliminate all other points, right, so then we have beaten the beaten the n log n bound at least in this one special case. So, of course, you can say that you know this is only for this one special case where it does not hold in the general case, you are true, but with this observation does it give us any hope that there is any furthers scope of improvement and how.

In this what I am saying is that faster algorithm when there is only one maximal point this is what I just pointed out. You can actually extend the same arguments say that you know you can put it a design in faster algorithm when there are even two maximal points perhaps by the some kind of a similar argument, because in one point you eliminate everything, you pick up the next maximal point and with that you may have eliminated within else, so again.

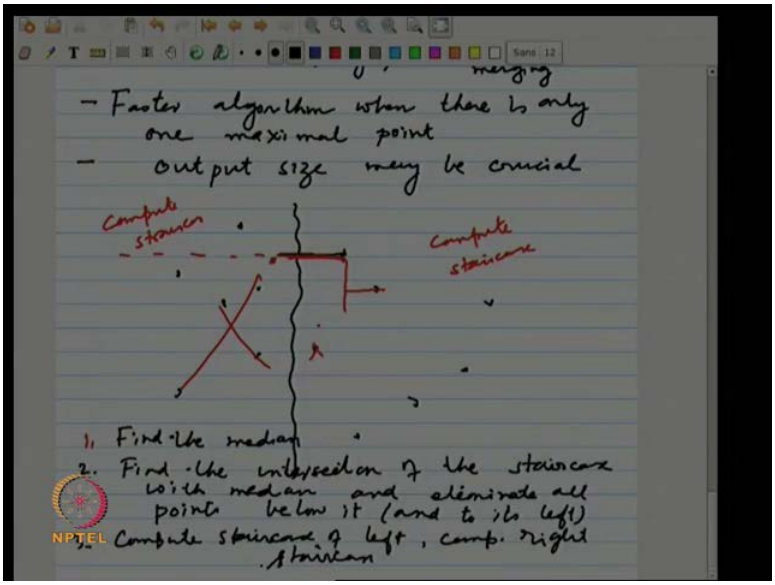
So, what it amounts to saying is that, so what would you deduce from that, I mean what what is the lesson that we can take from this or just not just one and see when there is few maximal points then there is probably potential for improvement and of course we do not know initially whether there are one maximal point or there is one or more than one or even may be all the inner maximal points, but it seems to be depend on what is basically what we can call the output size, output size may be crucial, but then I do not know what the output sizes, you give me set of points, how the hell do I know that how many points they are going to maximal, I if knew the reason only one maximal point you

know I could run this algorithm by pick picking up the maximum y coordinate and running that algorithm and getting it, but I have no idea whether how many maximum points are going to be their, right.

So, here is another idea, another algorithm let us say and what I am going to do is the divided conquer that, we just designed, in I am going to modify it a little bit, where the divided conquer what did I do. We took all the points projected on the on the x coordinate, you know found the median point so that I could divided two parts, found the maximal points on the right half, found the maximal points in the left half and then we merge them and merging was very simple.

Now, instead of actually finding out the all the the maximum points, are seem to be kind of you know you know wasted work, because some of these points are not maximal in the final output, right, even before I compute these maximal points on the left half, if I could somehow find the way of filtering them out, then I wouldn't have actually do it recursively. See the reason we got this n log n kind of thing, because we have to write down two times n over two, we solve the right half recursively, we solve the left half recursively, you know what I am saying is that instead of actually solving the left half recursively, if we can somehow find out in a what the maximum point on the right half is and just use that to filter out this, then perhaps there is some savings and of course, you to analyze you know how that c v is going to happen, right.

(Refer Slide Time: 42:26)



In other words, when we have these points, suppose this is my median line, I would the final, so I want to find out that to the right of this median line what is the maximum y, what is the maximum y coordinate and for that do I have to sort? No, right, I only need to find the median which takes place order in time by the way I forgot to mention that. you know so this order n term, this order n term not only contains, not only has the component for merging, but also has the median, finding the median, right, median plus merging, it has both these components. So, I can find the median in in linear time and after having found the median, I can separate out all these points and did not know the you know the relative ordering of those points. I know the right half of the points and once I know the right half of the points, I can find the maximum y coordinate, right, I can find the maximum y coordinate, the whole thing is order n.

So, in order n time I know essentially what is the maximum y coordinate to right which is which amount to saying that I actually know where the stair case is going to cross this median line, because the shape is such that is a falling staircase. So, essentially what we know is that the median line is going to intersect the stair case here, so we have actually computed some part of the output because this thing let me use red.

So, this thing actually is a part of the final out final stair case, now once I know this then all these scenes basically disappear, right. So, all these seems basically disappear and then I continued doing the same. So, I know this information and of course, with this, I can also may be you know eliminates this some points here also, that is fine, I can eliminate these points also and then I have to find the reaming stair case, I have to find out you know what is the remaining stair case, which is basically you know some like this. So, I have to compute this staircase here to write and compute staircase to left.

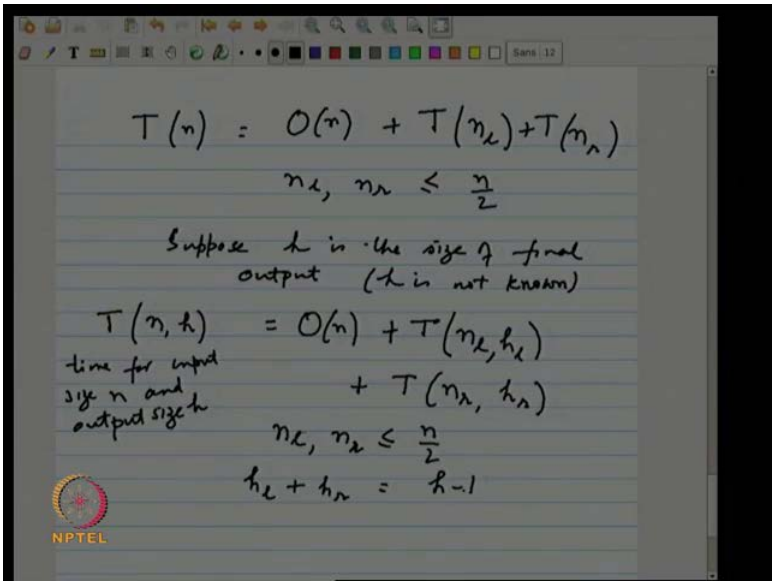
And whatever I compute, thus the left left staircase that I compute will survive, that is a beauty of it. Now, I do not have to do any merging, I just take take together, it is like, something like the sort, you know if I once I have found the partitioning element, I know these that the points larger than that, I know the points smaller than that and I just paste them together, I do not have to merge like, I do a merge sort.

So, similarly here, I do not have to actually find out whether the left staircases will survive, it will survive a simply have to paste it together or else that becomes trivial alright, so this is the algorithm. So, I compute, so I am I just write down the steps. Find

the median, then find the the find the intersection of the staircase with median and eliminate all points below it and to its left and then compute staircase of left, compute this the right staircase and that is it, you are done, you finally the output is simply just joining these two staircases.

Fine, it is easily stated, you can also write a program, your challenging part is how do we analyze, this what can we write when when we have to analyze this, what do we write?

(Refer Slide Time:47:22)



So, T's of n is the time to compute the entire staircase and that is essentially the order n that you require for finding the median, finding the where the staircase intersects intersects the median and then t of n l plus T's of n r, left hand side, right hand side, all we know is that n l and n r both are of the less than n over 2.

But then we do not know exactly what the values are, you do not know what the values are and that is what actually complicates matters. You know if you do not know what the values are, then I if I write T's of n over two plus T's of n over two back to this, so there is something else that should come in to this, something else that should come in to the our our calculation analysis and our intention was that something it has got, something to the output size.

So, this recurrence does not capture anything about the output, it still talking about the set of input points, n input points and then again when we are doing on the left and write

half, we are only the applying the recurrence to the point that is survive in the left half and the points in the right half. We are nothing more than I am, so this is the best that we can do if you write the recurrence in terms of input, somehow will have to bring in the notion of output here, so for that I will do the fallowing.

I will say suppose h is the size of final output, h is not known, we do not know h, but then I can may be just say if the running time is the function of both input and output, I should actually now meet my recurrency little bit more complicated. So, time for input size h, sorry n and output size h, right. So, this again the order n term remains the order n, but now I can see something like T's of n l comma h l let us say, plus T's of n r comma h r and then what do we have? Yeah, I surely we have this condition n l, n r are both less than n over two, the something more I can say about it, h l and n r, what can I say.

So, h l plus h r equal to h minus one, because we have found one stair one stair of the stair case. Now, this recurrence you know if you look in to it, i you n i I may be I should actually ask you to try to solve this, but I will give the solution. Now, how you solve this? We know we will take it as an exercise, how you are going to solve it, how does one solve references. The solution to this happens to be how you are going to verify this, you know I leave it up to you, how do you solve it, you may not have seen this references before.

Now, if you look closely at the solution n log h, what is the maximum value of h n? Right, so the maximum running time is n log n, which is what we found out even by doing this lower bound argument and and also the other algorithm give us n log n. When h is small, for instance h is constant, this is order n, so it it interpolates nicely between all kinds of an output sizes, small to large, all right. Not only that, you know you can also show you know this is something that you know outside this scope of the discussion, right, now can be shown that this is optimal with respect to input and output, oh sorry output sizes, so will stop here today.