Computational Geometry

Prof. Sandeep Sen

Department of Computer Science and Engineering

Indian Institute of Technology, Delhi

Module No. # 04

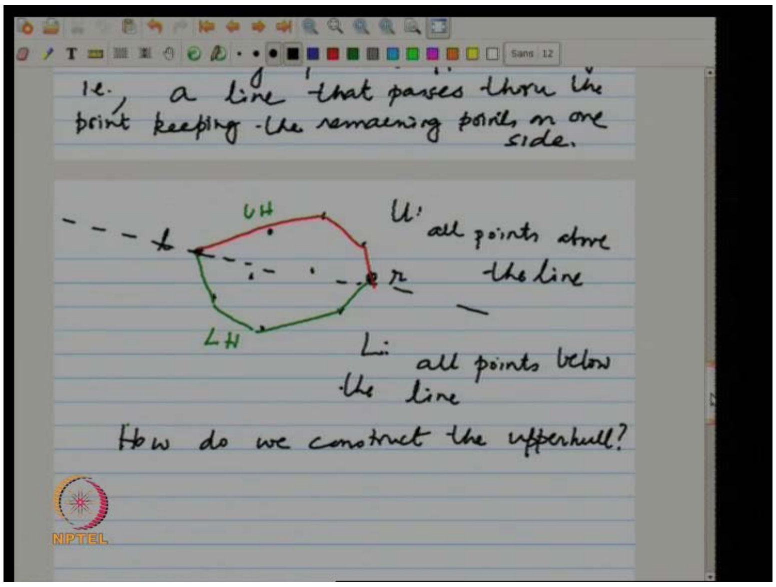
Convex Hull Different Paradigms and Quickhull

Lecture No. # 02

Convex Hull (Contd.)

So, we will continue on the topic that we began yesterday, namely convex hulls; two dimension convex hulls specifically. So, if you have any questions from yesterday, you can ask me. So, I think I just started on the second algorithm.

(Refer Slide Time: 00:54)


i.e., a line that passes thru the point keeping the remaining points on one side.

U: all points above the line

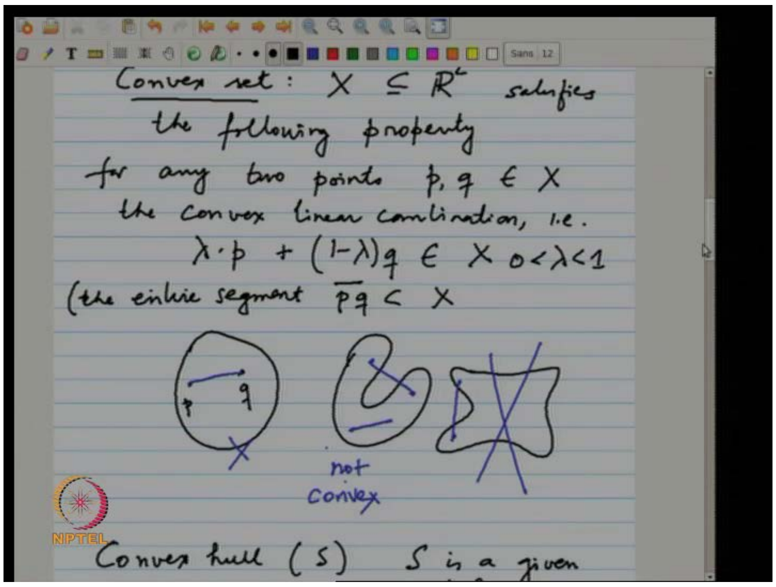
L: all points below the line

How do we construct the upperhull?

So, let me recap. So, we decided that we are going to construct the upper hull and the lower hull separately, namely; you look you consider the line joining the left most and the right most point; and the part of the hull above is called the upper hull, the part of the hull below is called the lower hull, right. I just like to add one more definition; so, another you know definition for boundary points. So, as I said yesterday, boundary points are those points through which you can draw a tangent, right. So, the tangent is such that all the points of the given set should be on one side, so that is what the tangent is; the entire convex hull should be on one side.

There is another useful definition of boundary point, which is often used when we design algorithms; and that is related to the definition of the convex hull in some sense, that. We said something about the entire segment connecting two points should be completely inside the set; I call this as the convex linear combination; just recall, let us see here right.

(Refer Slide Time: 02:11)



Convex set : $X \subseteq \mathbb{R}^2$ satisfies the following property

for any two points $p, q \in X$

the convex linear combination, i.e.

$\lambda \cdot p + (1-\lambda) q \in X \quad 0 < \lambda < 1$

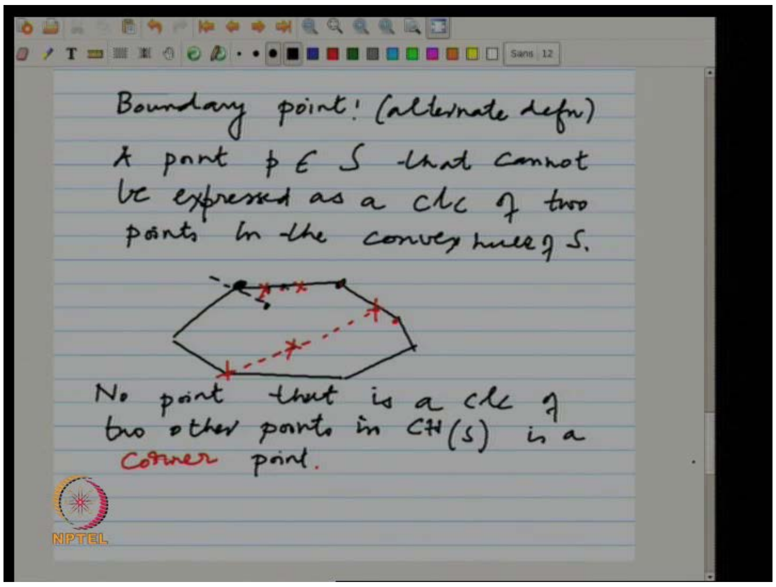
(the entire segment $\overline{pq} \subset X$

p
q
not convex

Convex hull (S) S is a given

So, this is the definition that I am referring to, lambda p plus 1 minus lambda q; so, this is called a convex linear combination. So, boundary point is the point, which actually cannot be expressed as a convex linear combination of two points in the convex hull. So, that is another alternate definition.

(Refer Slide Time: 02:28)



Boundary point: (alternate defn)

A point $p \in S$ that cannot be expressed as a clc of two points in the convex hull of S.

No point that is a clc of two other points in $CH(S)$ is a corner point.

So, boundary point; there is an alternate definition. A point p among the given set of points that cannot be expressed as a clc - convex linear combination of two points in the convex hull right; so the boundary looks like this; this is the boundary point. None of these points essentially, I cannot draw you know, this segment that contains these points. So, if I try to draw a segment, one point will be inside and the other point will be outside. So, this boundary point cannot be expressed as a clc of two distinct points; it can be expressed as a clc of itself, you know you can, it is a trivial kind of combination, that point and itself. But it cannot be expressed as clc of any other point.

Analogously, no point that is a clc of two other points in convex hull of S is a boundary point. So, every such point is an interior point; any other point. So, you are right so, maybe I should say; no no, so, what is the distinction between corner and the boundary point?

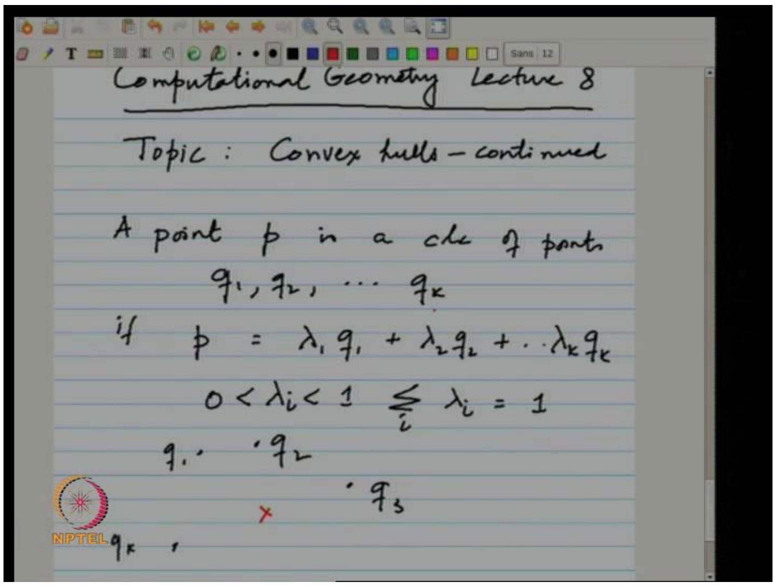
(( ))

Yeah, yeah. So, this point, right. So, maybe I should; yeah, boundary point right. So, you are calling a point on this boundary, but then if you take a point on this boundary, it can be expressed as linear combination of two other points, right. So, that is I think what you are saying. So, any other point can be expressed except this corner point that is what; yeah sure. So, no point that is a boundary point; so, corner point; you want me to be more specific, corner point. Thank you.

So, in other words, the moment we see that some point which can be expressed as clc of two points, we know that it cannot be boundary point; it is no longer a candidate for a boundary point. So, that is also an another way these algorithms work; that the moment I find that some point is a convex linear combination of two points; it means that we can discard that from further consideration, right. So, any point in the interior, see you can. So, if suppose this is the convex hull; any point, which is in the interior, you can actually figure out that it is actually convex linear combination of two points, right. So, I can take any corner point; join, draw the line and it is going to intersect the boundary somewhere, may be here. So, this point is in the convex hull because it is in the convex linear combination of these two corner points, this one and this one. And likewise, this point is in the convex linear combination of this point and this point.

So, certainly any point in the interior, you can argue is a clc. In fact, there is a more general definition of convex linear combination which is that; we are only looking at its convex linear combination of two points; you can actually extend the definition to a convex linear combination of more number of points. In other words, every point in the interior; let me just write this.

(Refer Slide Time: 07:39)



Computational Geometry Lecture 8

Topic : Convex hulls – continued

A point p in a clc of points $q_1, q_2, \cdots q_k$

if $p = \lambda_1 q_1 + \lambda_2 q_2 + \cdot\cdot \lambda_k q_k$

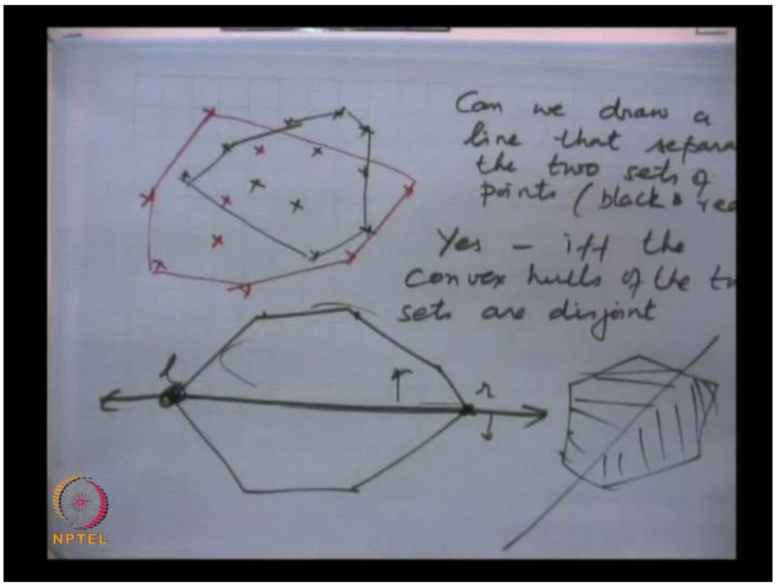
$0 < \lambda_i < 1 \quad \sum_i \lambda_i = 1$

So, point p is a clc of points q 1, q 2, q k, if p equal to lambda 1. So, that is another alternate definition that these points basically says; if you; geometrically, it means that you know I have these points q 1, q 2, q 3 upto q k and this point p will be somewhere, essentially in the interior of the convex hull of these points, certainly to be more precise, you can look at the convex hulls of q 1. So, this is a definition, algebraic definition and the geometric interpretation is that the point p is in the interior of the convex hull of q 1, q 2 upto q k, that is what geometrically, this is the interpretation of that; a convex linear combination of the boundary points.

So, with this definition in place let me go ahead. Linearly independence. See, in two dimensions how many linearly independent points can you get? No. So, the boundary the corner points of the convex hull; they are certainly not convex linear combination of the remaining points. So, you can have any number of points; I give you hundred points, then all those points can be corner points, so the convex hull of those points and therefore, all those points cannot be expressed as a convex linear combination of the other points.

Then all of them are boundary points corner points, whatever; right. So, that is the algebraic way of expressing it. So, going back to this; one more point I should make. There was a question yesterday at the end of the lecture that when I separated out the upper hull and the lower hull, why should not the upper hull and the lower hull have any kind of interaction? That was one question asked and I gave some sort of an answer, but let me elaborate on that little more, because this also happens to be one of the so, the motivating applications of you know, convex hulls; not only in two dimensions, but in higher dimensions also.

(Refer Slide Time: 11:28)


Can we draw a line that separa
the two sets of points (black & red
Yes - iff the convex hulls of the t
sets are disjoint
l
r

And that application is that if I given a set of points well, not 1 set of point, let us say two sets of points and let me use two different colors for that. Now, I want to ask the following question that can we draw a line that separates the two sets of points, namely the black and red? Exactly. So, it turns out that, yes and if and only if the convex hulls of the two sets are disjoint, but. So, in this particular case, of course, and if you look at the red point, this is the red convex hull, right. So, there is an intersection of the convex hull. So, there is no way that you can actually separate them.

And analogously, if the two convex hulls are disjoint, then you can have linear separability of point sets; and so very important primitive you know for many things including even learning and it is a very hard problem to solve in high dimensions. So, in two dimensions we can construct the convex hulls in polynomial time, whatever and find out whether they intersect or not etcetera all that can be done.

(( ))

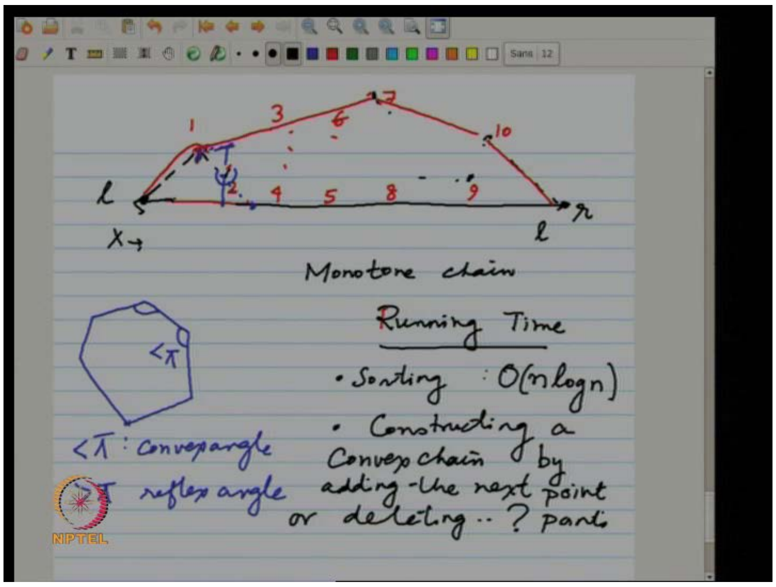
Yeah, no no. So, this is the only characterization that we know of well; there is nothing else keen as this. So, the question here is that do we necessarily have to construct the convex hulls? Well, if it were, so that then we have kind of almost a lower bound argument that you know, if the convex hulls are very hard to construct in higher dimension, then there cannot be any efficient algorithm. But there is no such result; but this is the cleanest formulation that we have at the problem, that is all.

So, it is not known how to do these efficiently in high dimensions; dimensions means arbitrary dimensions; you know it is not just 2 or 3 or 4, but let us say dimension d; do we have an algorithm that runs polynomial in number of points n and d, so, that is not known, right.

So, when we talk about the upper and the lower hull essentially what you are doing? you are actually. Looking at these convex hulls; for the final convex hull and we are essentially linearly separating them out. So, we have this linear kind of separation; this is the line that we are using to separate them out, left most and the right most point; it means, if the entire upper hull lies upwards of this line, the entire lower hull lies below this line and here talk me about you know this convex hull including these two end points and this convex hull including these two points, so there is no other intersection other than this line. So, they cannot have any kind of interference. So, we can construct them independently, because they are actually linearly separated. We have bi construction. So, you take any convex hull not this; I can draw any line and you know this part and this part, they are linearly separated. So, they are not going to interfere with each other when we do the construction; the construction can be done independently.

So, I just wanted to make this point that you know this is one of the sort of more important applications of convex hulls, you know this notion of linear separability . So, now, let us proceed with constructing the upper hull.

(Refer Slide Time: 16:30)



The first thing we are going to do is rotate the axis, so this line that we have, the separating line; let us assume that it is the horizontal line; no, this is my x axis; just two sort of; may it easier to visualize and we have these points above this line. So, you know that even if I rotate the axis, the convex hull remains the same; it is invariant.

So, we have some points. The first thing that we are going to do is look at the sorted order of these points. So, this is the one this is r and I am going to join the points in the sorted order along x axis. So, I have some kind of similarity, which I do not like; I mean, I am telling you that I do not want two points to have the same; it is cumbersome to explain, that is all, so actually sorry. So, I have joined these points in, what is called a monotone chain; this chain, if we walk along the chain, it is always going to move in the direction of the positive x-axis; it is a monotone chain. Now, this is not necessarily a convex chain. You want to now convert this or transform this into a convex chain; it means that some points will not belong to the convex hull and those are points that we want to identify.

So, we start walking along this chain, so, we start walking along this chain and the property of the convex chain is that you know it should always sort of; if we walking from towards in the positive x-direction, it should always basically turn towards the right. So, we walk and we make a right turn; that is fine. Then, we come to this point; let me number them before I lose them. So, when I come to point 1, I make a right turn, which is unacceptable because it is you know by definition, convex chain should be right turning. But the moment I reach point 2 and I try to go to 3, I have a left turn between the points 1, 2 and 3, right. So, I have made a left turn; here, which means that this point should not be on the convex hull and can you explain this on the basis of the definitions that we done just recently? Why should not point two be on the convex hull?

(( ))

No. No. Yes, triangle area is what we are using to figure out whether it is left turn or right turn; my question is why what is the logic or what is the reason that we can forget about point 2? Yes, that is certainly you know I think one of the answer; so, what is being said here is that this you know; let me use another color. So, you look at. So, what is being said here is that this point and you extend it, this point is on the boundary and therefore, two is a convex linear combination of that. Yeah, sure sure; that is another good argument. So, you are saying that I cannot draw a line through point 2, such that all points will be on one side, that is the other observation being. So, if it is a left turn, there is no way that I can draw a line through 2, such that all points will be one side of the line. Another thing I can see is that you know; well, I mean this is the good enough proof, but you know you can look at what is the relation between 1, 2 and 3? So, 1 and 3; so, this segment should be in the convex hull. So, any point on this boundary should be on the convex hulls. So, you know this thing, again two is the convex linear combination of these two points. So again it is basically drops out.

Right right. So, all those reasoning's are fine. So, yes I mean that another thing I forgot to mention that these kind of angles where. So, you look at the interior of a convex hull, you look at these angles; these angle must be less than pi and this is called a convex angle; less than pi is a convex angle and greater than pi is called a reflex angle. So, clearly any time we make a left turn, this is going to be a reflex angle and therefore, that cannot be a point on the boundary of the hull.

So, there are many reasoning's due to which you know, left turn basically means that you know we have loss that point. So, we get rid of that point and so, we can see that two cannot be on the hull. So, I am going to take out 2 and now, change my walk to go to 3 directly from 1. So, now again l, 1, 3 these 3 points form a right turn. So, there is nothing wrong with it.

So, we proceed further; again at 1, 3, 4 it is a right turn, fine. So, that is also ok and then, again we have a problem; from 3, 4, 5 it becomes a left turn. The moment it becomes a left turn, it means that I should I cannot consider 4; so, 4 goes out of the consideration; then I should go to 5 directly from 3. So, we continue like this and then again its turns out to be a left turn; so, in the next when we go to 6, you know even 5 should drop out and we should go to 6. Now, it is not clear whether 1, 3, 6 is a right turn or not; you know, let me refer to the third umpire and you know let us say is a benefit would have out; it is a right turn; all right. But you know 3, 6, 7 makes a left turn again, so that means that is form.

Now, again it refers to the third umpire; my diagram such that and this time let us say the umpire makes a decision this is the left turn. So, which means that not only we loss 6, we should also lose 3, because 1, 3, 7 or if we look backward, 7, 3, 1; so, there is a problem at 3. So, just to make it look better, let me raise the level of 7, a little bit more. Let us take 7 to be here and then I think (()); so, now, it looks little more like a left turn. So, this is the left turn at 3 from 1, 3, 7. So, we cannot have this change; so, we should also now forget about 3, which means that I should go to 7 directly from 1.

So, what this illustrates is that if there is a left turn at 1 at some point, that point gets dropped. But as a result of that, we may also lose some of the preceding points. So, let me make this example more pronounced.

So, we had a situation, let us say where it was like this, these were the points, where you made these right turns you know, suddenly we have a point here. Of course, we lose this; this goes out. So, we should consider this boundary. Now, this is a left turn again; so, this goes out; this is not out; same thing you know this is still a left turn, this is not out. So, you can see it could actually have a chain reaction where lot of things can get knocked out, you know due to just consideration of a new point. Fine, so, even if all this happens, finally, what we get you know; at the end of after walking reaching r, is a chain that is necessarily a right turning chain.

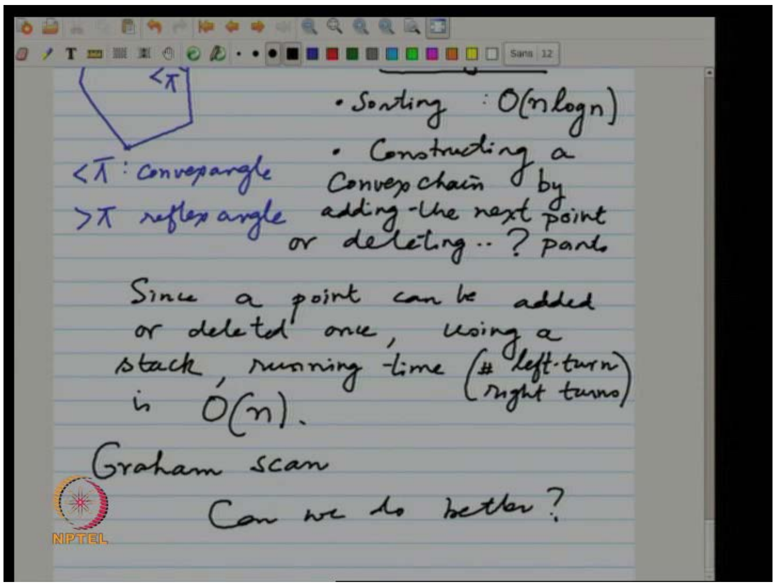
So, in this case we basically end up with this point; so, we end up with this. This is always a right turning chain, so, all internal angles are convex angles and we are satisfied with this. Any point that is knocked out once can never be a part of the convex hull because it is already a clc of some point gone, right.

So, algorithmically we will see what to do? But we are not yet reached the stage, where I am trying to count number of operations. All I am seeing is that you know, but it is nice ordered structure, right; when we go back, we are going back by one step, we do not have to jump and find out some you know; we just reverse basically from them; we do not have to chase a random point. We just reverse and we keep reversing till we find the change is right turning and prior to that also it is right turning. So, the entire thing should be right turning. In other words, what we are doing here is some kind of an inductive argument that if we have considered up to point I, we will have a convex chain up to point I. So, that is basically what we are doing.

Now, let us look at the time complexity of the whole thing what is that we are doing. So, we had sorted. so, the running time. So, sorting is going to cost us something; here we have, let us say there are n points, n log n. Next, we are doing scan you know; so, we are walking and sometimes, deleting points you know; sometimes you know adding another point to the chain. So, constructing the convex chain by adding the next point or deleting how many points? At most, what I add; the higher stage we are removing I minus 1 points, at most you know. So, what is the running time? So, you did not sum up the I, right; you did something smarter than that.

See, a point can be deleted only once, right; and what is the data structure that will support it? Yeah, stack of the points and when you have to retrace, you know it is the last in first out; so, you have to reverse and reversing is very easy in the stack. Any other kind of data structure also, if we have the right pointers you can do it; the stack or some kind of you know linked list, with the right pointers; anything is fine.

(Refer Slide Time: 30:28)



$<\pi$

$<\pi$: convex angle

$>\pi$ reflex angle

• Sorting : $O(n \log n)$

• Constructing a convex chain by adding the next point or deleting .. ? points

Since a point can be added or deleted once, using a stack, running time (# left-turn / right turns) is $O(n)$.

Graham scan

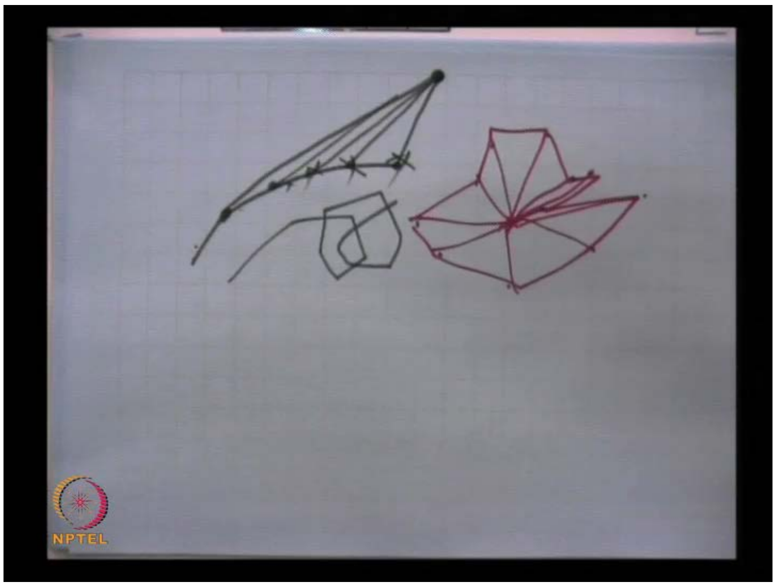
Can we do better?

So, overall then, a point; what we are saying is that since a point can be added or deleted once, using a stack, running time here is basically number of left turns and right turns test, right. So, this procedure or this particular algorithm goes under the name of Graham scan. The total running time is the sorting time whatever it takes plus linear time. Right, but it is a constant size determinant; it is a 3 by 3 determinant; so that we assume within the big O notation. So, whatever the time it takes, you know about 3 4 multiplications; 3 4 additions, so that constant will have 3 or 4 or 5 or 10, whatever. So, we are ignoring the constants.

So, now you have an ordered n log n algorithm as a post to an n times h algorithm for that the simple one, where we sort of pull a rope around all the points, which is better; of course, in the worst case this is better. But if you have very few points in the convex hull, namely you have many, let us say about constant points, then the previous algorithm is still somewhat better.

So, although we have a faster algorithm, there are some cases where previous algorithm could be better; we will address that; that is an important issue. But even prior to that, again our same question, can we do better? Yes. I mean, when you do the implementation you can do it altogether, upper and lower hull, except what you need to do is instead of doing; I avoided what is called a polar sorting. So, I would have to actually sort with respect to angle, but then again you know that we do not have to compute angles; we can do it everything with respect to left and right turns; Yeah, I just said this is easier to explain and yeah, I can actually run the whole thing at once, all I need to do is identify a point inside the convex hull. I want to sort everything inside the convex (hull); just since the point is when raised, you have these points given to us; I need to identify some point will be strictly inside the convex hull, may be this.

(Refer Slide Time: 33:47)

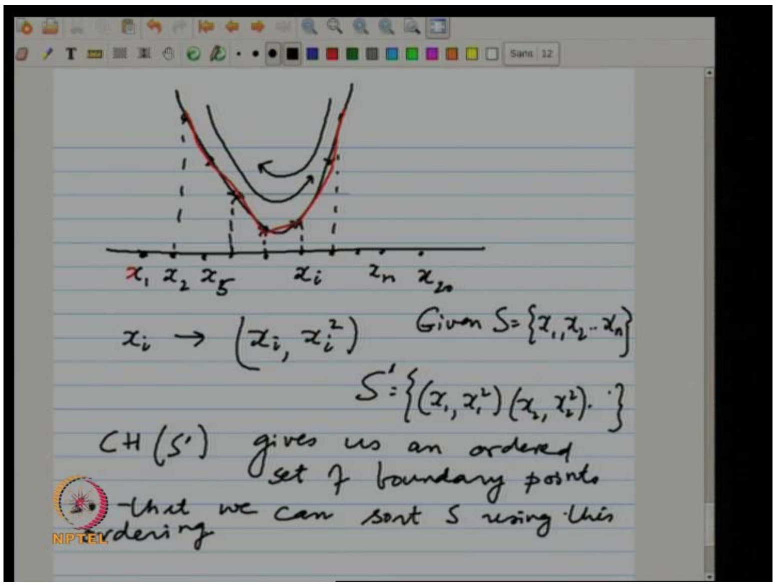


And I am going to do my sorting; my monotone chain will be essentially this, which is what I have to convert to a convex chain. So, how do you find a point inside the hull? 2 is not a good thing; take any three points and any convex linear combination of 3 points must be strictly inside the convex hull. No no, it will not work. So, I have actually; if you look at the homework problems, I have given one as the problem that you know; we have to do this kind of polar sorting; there are some complications, otherwise. You may end up getting what is called a self intercepting chain; this is a problem. So, you may take right turns, but you know you may take right turns like this; so, you have to avoid that. A right turn is not always necessarily good; it is working because we are starting with a monotone chain.

So, can we do better? Are we satisfied with n log n or can we do better?

So, let me give one construction; nothing that will convince you, at least it will partially convince you about something. So, let us consider you know something like a parabola.

(Refer Slide Time: 35:29)



$x_i \to (x_i, x_i^2)$ Given $S = \{x_1, x_2 \dots x_n\}$

$S' = \{(x_1, x_1^2)(x_2, x_2^2) \dots\}$

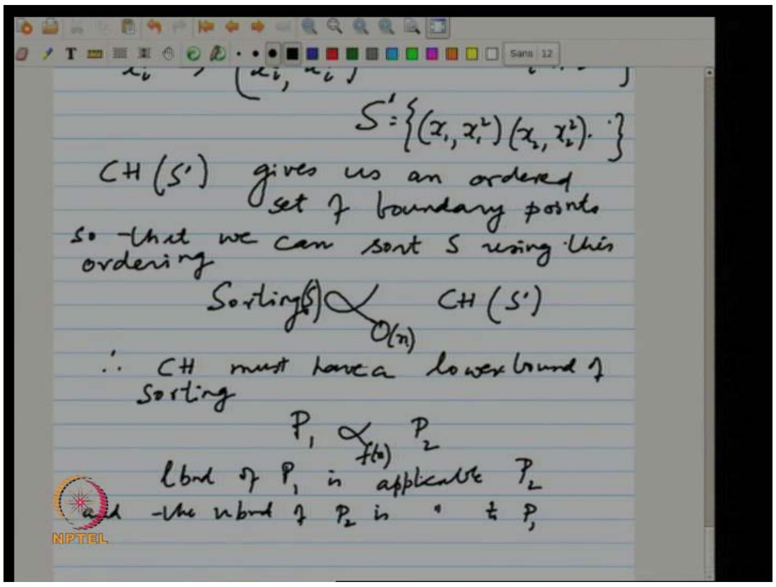
CH(S') gives us an ordered set of boundary points that we can sort S using this ordering

Let me consider a set of points given to me which are just points, you know thus just values. So, these are actually the numbers given to me and not necessarily in this ordered fashion; it is given to me in some oddball fashion x 1, x 5, x 2, x i, x n, x 20 and so on.

I have given some points; what I am going to do is I will project these points on this parabola; actually, not project, but I should use the word lift. So, I am lifting these points by 1 dimension. So, a point x i is getting mapped to x i, x i squared. So, given a point set, given s equal to x 1, x 2, x n, I am defining s prime as a two dimensional set of points x 1, x 1 squared x 2, x 2 squared etcetera.

Now, I construct the convex hull of s prime; what is the convex hull of s prime? All these points; what is the convex hull of this? So, all the points will be corner points. Are you convinced about this? All the points should be necessarily corner points. And, if our assumption or convention is that a convex hull algorithm should produce the points in whatever, clockwise or counter clockwise fashion, then I can recover from the convex hull, the sorted set of the points. So, it gives us an ordered set of boundary points. So either, it will be like this or it could be like this; either way is fine. So that we can sort s using this ordering since all points are on the boundary. So, we have essentially shown that sorting is linear time reducible to convex hull, sorting of s is reducible to convex hull of s prime. And therefore, convex hull must have a lower bound of sorting, otherwise you know I could reduce sorting to convex hull and then, sort it faster than the lower bound for sorting, which cannot be… So, the lower bound for sorting must be also applicable to the lower bound of convex hull.

(Refer Slide Time: 35:29)



$S' : \{(x_1, x_1^2)(x_2, x_2^2) \cdots\}$

CH (S') gives us an ordered set of boundary points

so that we can sort S using this ordering

$$\text{Sorting}(S) \propto_{O(n)} CH(S')$$

$\therefore$ CH must have a lower bound of Sorting

$$P_1 \propto_{f(n)} P_2$$

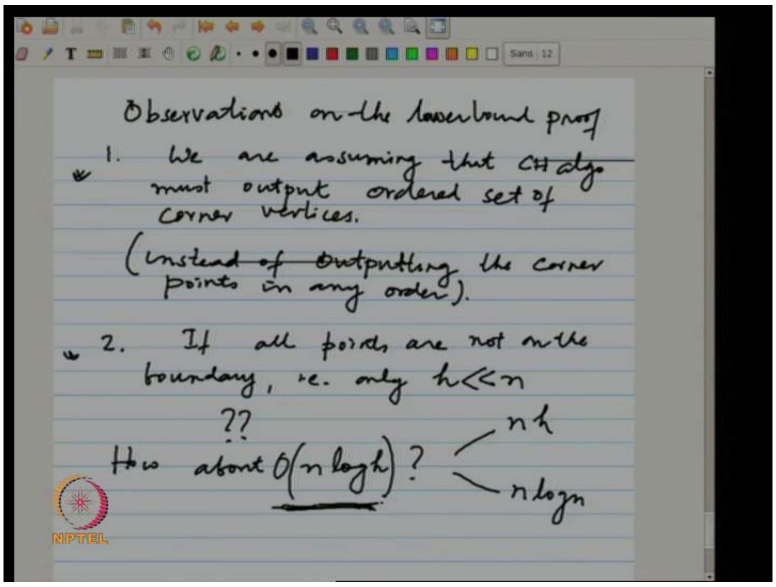
lbnd of P_1 is applicable P_2

and the ubnd of P_2 is " to P_1

These are the points; I have been given that I wanted to sort. No. It is not needed to be multiple set of points; I have reduced the problem of sorting, so, I have been given these points x 1, x 2 up to x n, which I am supposed to sort. I have shown you how to sort using a convex hull algorithm and therefore, convex hull cannot be constructed faster than the lower bound for sorting; that is all.

So, when I reduce problem p 1 to p 2, the lower bound of p 1 is applicable to p 2 and the upper bound of p 2 is applicable to p 1. Of course, this must also happen in some efficient time. So, I have not mentioned this; so, hopefully you are familiar with the reducibility in time f n or whatever. So, this f of n cannot be too large; it has to be. So, if this was not order n reducible, but let us say order n log n reducible; did not mean nothing. Yes, we will come back to that. But what this problem show what this proof shows is a very simple proof.

(Refer Slide Time: 42:12)



Observations on the lowerbound proof

1. We are assuming that CH algo must output ordered set of corner vertices.

(~~Instead of outputting~~ the corner points in any order).

2. If all points are not on the boundary, i.e. only $h \ll n$

??

How about $\underline{O(n \log h)}$? $\begin{cases} nh \\ n \log n \end{cases}$

But this proof establishes something; let me summarize what this proof establishes first. So, this proof claims that this proof or let us say some observation about the proof, let us say we are assuming that convex hull algorithm must output ordered set of corner vertices. If I did not have this constraint, suppose I was only interested instead of outputting the corner points in any order. Suppose, my convex hull algorithm was not required to produce this vertices in counter clockwise or clockwise order, but just either set of point sets; tell me what are the corner points? If my convex hull problem was specified like that, this lower bound will not hold.

And number 2, what was just observed here is that; I could be interested in an algorithm where I only have to identify which are the corner points. I do not have to output them in any given ordering. No, then this previous reduction does not hold because I have to assume that the convex hull gives us an ordered set of boundary points, so that from there I can reconstruct the sorted set. If the output points when not to be produced in any sorted order, then I cannot reconstruct the convex hull from there; the sorted points from there.

So, the proof uses that strong assumption that the output of the convex hull; for most algorithms of convex hull, they produces them in sorted order. But then someone may have a different specification; that is what it is. So, most natural algorithms you will design, will produce them in sorted order and therefore, this proof hole goes through.

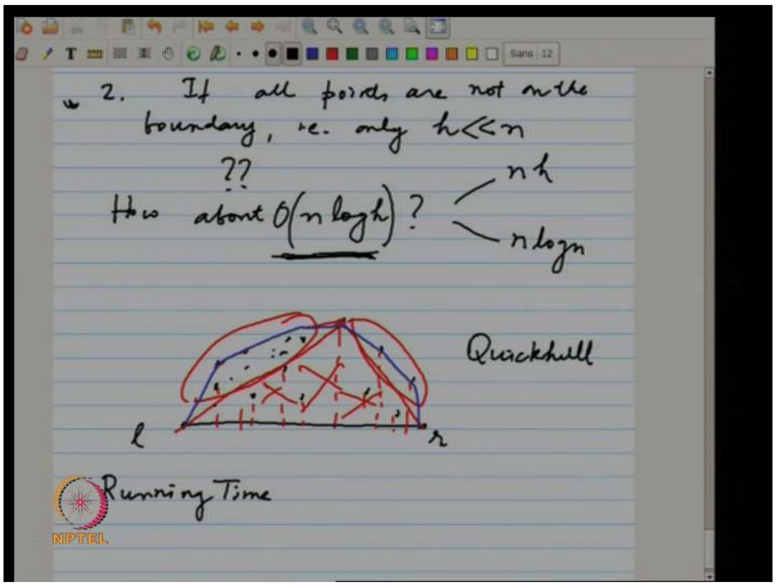
But, then theoretically why do you always want the points to be in sorted order? I may be only interested to know, what are the corner points? And therefore, then in that case this reduction does not hold. Number 2, if all points are not on the boundary i.e., only h, where let us say it is much smaller than n, again this lower bound does not give us anything. To the contrary, we have already seen we have an algorithm that works in n times h time; we have an algorithm that works in n log n time and they are incomparable depending on the size of the output. If the output size is very small, one is better; the output very large, the other one is better.

And, we only know that if the output size is large, then log n is optimal, if my convex hull is required to produce the points in the sorted order. So, I will address this one again later, I will postpone this discussion when I get into more formal models of lower bounds. So, about point 2, it basically leads or gives us hope or you know whatever you call it or you know with just we have to break our heads again to figure out if we can have something better than n log n. n h and n log n are not comparable; you want something that will behave, whatever you call it optimally, for whatever the output size is; you remember what we did for the two dimensional maximum? What do we get eventually? n log h, right. So, that would be a good guess; if we can get an algorithm that is n log h, so, the question I am raising here is this: how about n log h?

So, can we get an algorithm that works in n log h? Because if it does, then this seems to this would actually subsume both the cases n h and n log n, because when h is large then it will be no more than n log n and when h is small then again it will be better than n times h. However, as I said the optimality of this again will be left; I am not going to address at this point; we will just try to see if you can obtain an algorithm for which the running time is what, n log h.

No, but that only give you h log h bound, right. Yeah, that is what I am saying. So, this both for maximal points and this one; it turns out that you can actually prove this sort of a lower bound n log h.

(Refer Slide Time: 48:38)



2. If all points are not on the boundary, i.e. only $h << n$

??

How about $\underline{O(n \log h)}$?

nh

$n \log n$

Quickhull

l r

Running Time

So, this is a tight bound actually, but that discussion and will have to wait. So, let us first try to design something better; I will give you the idea and then we will carry on next time. So, historically there was a very sort of very well known algorithm for convex hulls and that is very simple idea; what it does is it identifies the left most point; again, let us look at only upper hulls; right most point. And then, what it does is, we are given other points. It finds out the point that is furthest from this line passing through l r. In other words, and if you translate, you think about parallely translating the line l r, what is the furthest in terms of perpendicular distance? Let us say.

So, you can find out the distance of this point from this line; again, we are using some formula of the analytical geometry. And suppose this is the farthest point in this case, then this point will necessarily be on the convex hull; that is known, fine; that is nothing new. Then, what you do subsequently is that look at this triangle. Any point in the triangle will not be on the convex hull; gone, so, these points are all gone. So, what you are left with: points here and points here. So, what you can do now is try to construct recursively, try to construct the convex hull of this side and construct recursively the convex hull on this side again; this can be simply pasted because that is the separating boundary.

And you keep repeating it; this is a very simple thing, you just find this left most and right most point, draw the line, find the point farthest from there and you know throughout all the points in the triangle; you can modify these by figuring out some trapezoid and throwing out all the points from within, but you know this is I am saying historically, this was identified as a very simple procedure for finding convex hull of point sets and this goes under the name of quick hull.

So, quick hull was discovered even before this Graham scan etcetera was discovered. Now, what do you think, it should be the running time of quick hull, this kind of quick hull? Average over what? All possible what? Yes. So, how would you even sub why are you even thinking about average because you do not even know how to express that average; till now we are never talking about how the point sets are distributed; point sets, how point sets are distributed? We do not even have a measure or notion of that yet.

So, let us not bring that into; n h? No, in the worst case what will be the running time? Let us forget about h. No, but then you can also be dividing it very nicely, right. So, why is it called quick hull? Can you make a guess? Why this is called quick hull? Yes, let it be a quick sort; what do you do in a quick sort? You identify a splitter and divide the points into two sets and sort them recursively and that is you quick sort.

Here, it is called quick hull because it reminds you of one of that. I have basically, now I have two sub problems, where I am recursively sorting and doing. So, sometimes people say that you know quick sort has the running time of N Square in the worst case; why do people say that? Because if the splitted point or the pivot point is very skewed to one end, you are not actually doing much justice to dividing the problem. So, quick sort works very nicely when the point sets are more or less equally divided and that can be actually almost guaranteed by if you choose a splitter at random.

Here, by the way we are not choosing anything at random; we cannot choose anything at random, everything is defined on the basis of the point sets. Here is the left point, here is the right point and it is the point that is furthest from this line. So, there is nothing probabilistic or randomized about this. So, we cannot even ensure like we do in quick sort, by picking the revert element at random.

Well, in this description, there cannot be any randomness, clearly. Then, you have to prove certain properties about it, right; if you pick a point at random, how is going to behave? How is going to split up? We have no idea. So, it is not clear, she is just saying; you know that we just pick a point random; how we do it, you know? We have to think about it what you know. Somehow randomness is some kind of magic that will work; we will come to that. So, I will show you that it does work as magic also, but not a very straight forward manner.

Here, you know we have the same thing again, that I can have a situation where you know all the points will be on one side and therefore, I am not going to divide the problem into; doing any justice about you know fairly dividing the problem, so, it run into the same problem of you know. When you do not use randomized quick sort, you may have the problem. So, these are the same problem. But let me just make this one modification and again the solution is to use some kind of randomness, but I think we are out of time today, I will take it off next time.