Computational Geometry

Prof. Sandeep Sen

Department of Computer Science and Engineering

Indian Institute of Technology, Delhi

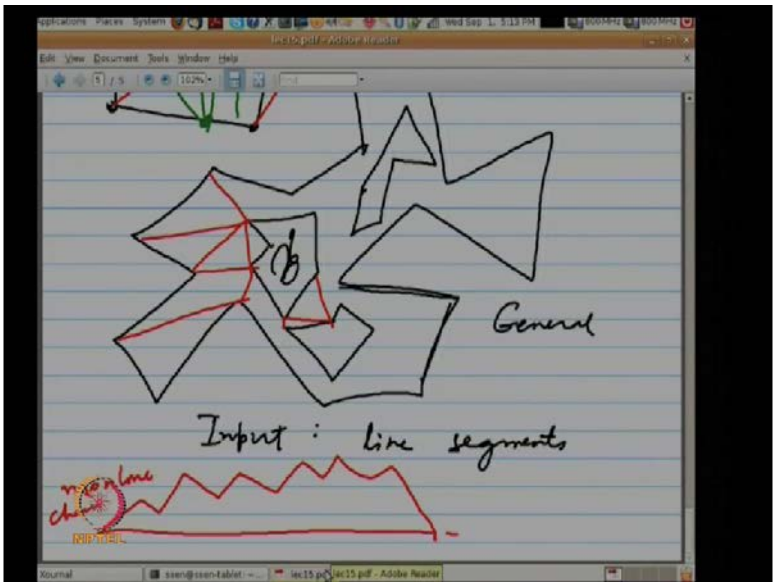
Module No. # 9

Randomized Incremental construction and Random Sampling

Lecture No. # 1

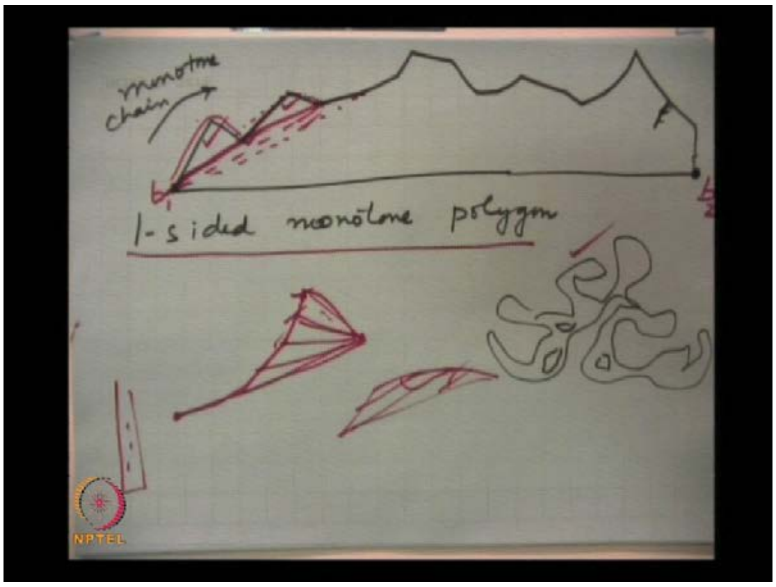
Triangulation of Arbitratry polygon

(Refer Slide Time: 00:35)



We posed a problem in the last lecture, I do not know how many people had the time to think about it namely, that is, we started on the problem of triangulation, then I posed this special case, this triangle which I am calling a one, so this triangle is called a one sided monotone triangle. Let me shift to the example (( )).

(Refer Slide Time: 00:55)



It has to be a monotone chain; so one-sided monotone polygon. So, this is exactly one edge and this is a monotone chain, monotone with respect to the edge. Suppose, we were asked to triangulate this particular object, so certainly not as bad as you know, something like, nothing like, nothing of compared to this; this is, this could be a simple polygon, and if you have some holes inside, there could be more some structure like that. So, from this object, this, this messy looking thing, you know, I am now resigned to looking at something like this as my as my first step. So, how would you attack this problem, I want to triangulate this?

So, since this is monotone maybe we should just start reversing this chain, that would be, at least it could be a reasonable thing do. So, I start reversing this chain; let us call this the base things, b 1, b 2. So, I start reversing from here, nothing to do yet and then, make, I make a right turn. So, once I make a right turn can I, basically, have I found a triangle essentially? So, maybe I have actually, I can add this edge; now, once I add this edge, once this triangle goes out I am looking at smaller polygon again, a monotone, one-sided monotone polygon, but with one less triangle. So, I can again, sort of resume a journey. So now, I again start from here, this is gone, this triangle is already part of my triangulation, so, I can resume my journey back from this b 1 again, I move, so just assume that this is not a right turn, but a left turn. Now, if it is a left turn, I cannot add this edge because this edge is not guaranteed to be within the polygon.

So, I continue my journey, then I again get lucky, I hit a right turn, the movement I hit a right turn, right turn basically involved in the last three vertices I visited, I join this again. Now, once I join this, then I should actually again, this taken care of; then I should look at what this is like, maybe this is also a right turn, if this is a right turn, then I should join this, and well this also happens to be a right turn. So then, I find another triangle. So, these are all basically edges that I can add and I am creating more and more triangles. So, now, you get the basic idea right? So, we just continue on this process and you know, we are going to generate some triangles. So, the thing is that you know when we can actually, when we have actually found a triangle it is good and when we do not find a triangle, you know, we sort of continue further hoping that at some point we will get a right turn.

And you know, in a bad case it could happen that, you know, hit the chain like this, it is a, you know, it is an upward convex chain, so all the way from here to here, you know, we cannot add anything, but at some point it has to go down, so, the moment it goes down like this then, you can basically, have found all these triangles, till basically it is, you know, and till a point when this is again a left turn instead of a right turn.

So, the point is that, you know, we will be able to, when we reach this point either, so, will have an upward concave chain like this, which is, which I can complete with a fan or I will basically, you know keep generating this triangles with a right turn and those basically keep going out so, I am dealing with one less vertex all the time. So, I am not going to do a formal group of correctness, but this is what the procedure is. And what is the time that we have taken for this, n square, n log n, n log star n?

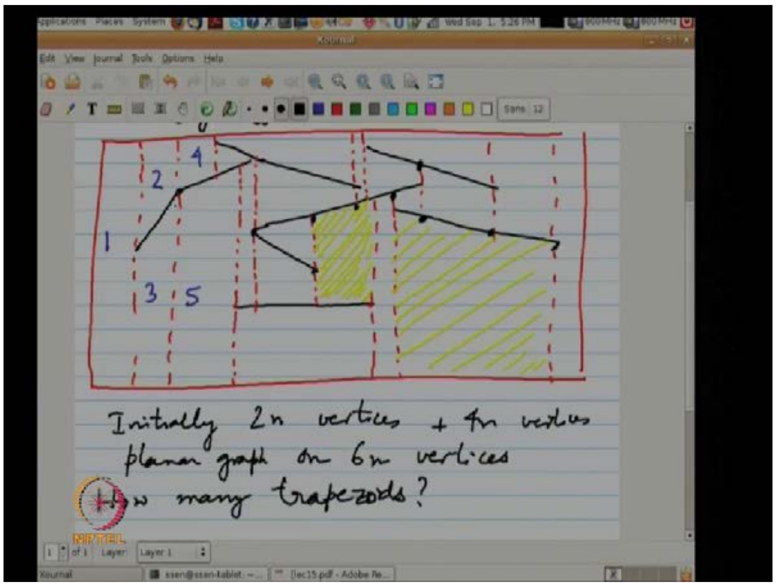
(No audio from 06:22 to 06:31)

How different is this from our Graham scan I mean, the analysis of Graham scan?

Yes we will do, the same thing, I mean, I do not see any difference between Graham scan and this, so, if that is linear time, then this should be linear time. See, you can, you can keep adding a vertex when you are doing this left turns, but then, once, so you can add it, let us see again, you maintain, show these vertices in a stack, so but once, you know, we hit upon something like this, these vertices disappear, and once they disappear they will never come back; so, we are going to insert or delete insert at most ones delete at most ones. So, this should also be linear time, though what I have not done rigorously is a proof of (( )) that this will generate the entire triangulation, so you will have to somehow combine. So, this is little bit of inductive proof essentially, you have to just, I given you all the steps, just you to compute the (( )).

So, this is my easy case where it is one sided monotone polygon. But then, this brute is far from a one sided monotone polygon, so what do you do? So, that is what we will take up today. So, we want to deal with an arbitrary polygon even with holes and somehow reduce it to the case of one sided monotone polygons. For that we will actually revisit our notion of ray shooting. So, I will, my input now let say, is only a set of line segments, so this arbitrary polygon can be looked upon as a set of line segments.

(Refer Slide Time: 08:33)



Initially $2n$ vertices + $4n$ vertices
planar graph on $6n$ vertices
How many trapezoids?

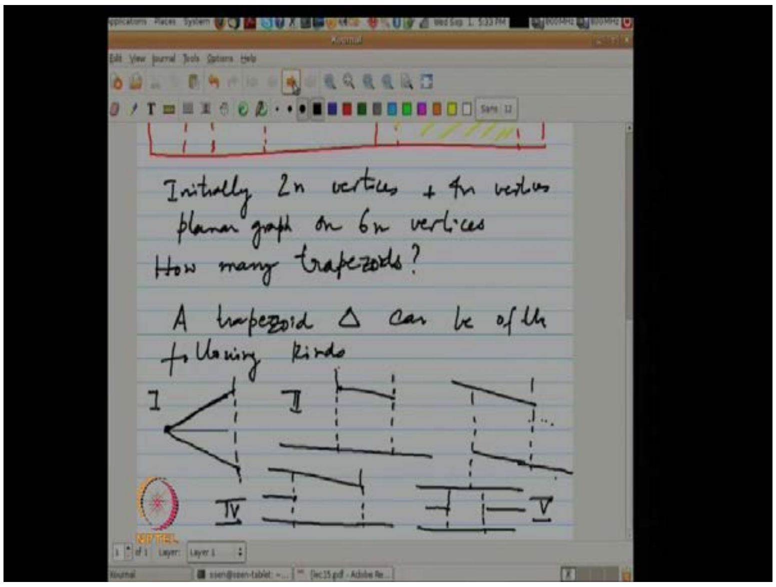
So, I am given a set of line segments, null intersecting, but possibly intersecting at the end points, something like this sort, let me stop here. So, this is supposing the set of lines segments that define that polygon. What I am going to do is from every end point of this, these set of line segments I am going to shoot or draw a vertical line, which means I am basically shooting ray upward and a ray downwards and extend it till it hits another line segment. We will begin, you know, our familiar trick is that you know, we will just enclose everything, pretend everything is in a box, so we do not have to go infinity. So, from every end point shoot ray upwards and down wards till it intersects, hits another line segments; now, actually, there is another one here which I am not drawing. So, this figure where you have drawn all the vertical, extended the vertical lines, what does it remained you of? Trapezium.

Right, exactly. So, if you look at a portion like this say, so, this shaded thing, lightly shaded thing, it is bounded by two parallel lines on the two vertical sides and you know it has a top edge and a bottom edge- say, someone suggested this is a is a trapezium or trapezoid. So, what we have effectively ended up doing is that by drawing these vertical lines, we now have a set of these trapezoids, large number of trapezoids. So, I can number them may be even, so, 1, 2, 3, 4, 5 etcetera. So, how many trapezoids do you think we can have, given that we have n line segments, from n line segments? N line segments mean 2n end points. So, we have, we would have drawn at most 2n vertical lines. And this 2n vertical lines could have well, I mean, 2n vertical lines would have 4n additional points, so, these additional points are let us say, these, and that kind of, so, in addition to the initial end points we have at most another, so, initially there were 2n, let us call it vertices and then, to that we have added at most 4n vertices, so, again it is a planar graph, this whole structure, the set of trapezoids, plus 4n vertices, so, about 6n, a planar graph on 6n vertices.

So, how many trapezoids? Well, if you go by our asymptotic arguments, it will be some big o of n, the vertices and faces and edges are all deleted linearly, so certainly no more than some constant time n number of trapezoids. But you may actually end, you can probably do a better counting, can you just think about it?

(No audio from 14:25 to 14:31) Yes, some of this trapezoids for instance like this one, sorry, like one, this is a kind of a, you know, a special kind of trapezoid where one of the sides is zero length, a triangle is also looked upon as a trapezoids. (No audio from 14:54 to 15:16) So, I will leave it as, you know, a kind of a relatively simple exercise. So, I would like you to do the counting even without using that toiler thing. So, maybe you can just look at a vertex and look at, you know, how many, so, this is a vertical line that have drawn to the vertex and you know, so, maybe I should actually, let me define a few things so that it will be useful for other purposes.

(Refer Slide Time: 15:50)

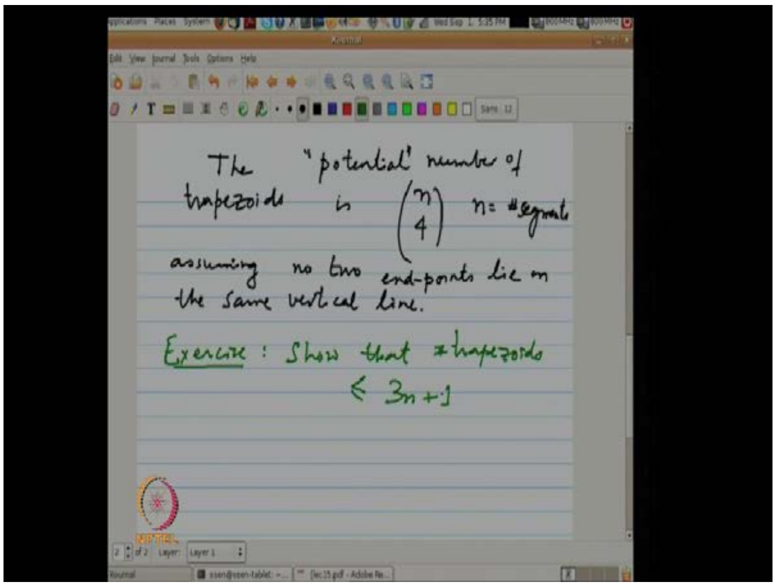

Initially 2n vertices + 4n vertices
planar graph on 6n vertices
How many trapezoids?

A trapezoid Δ can be of the following kinds

A trapezoid let us call it this, it can be of the following kinds- this is some observations that we make. And so, one is this kind of a, it is a kind of degenerated trapezoid where one of the lens is zero. So, I am just looking at shapes of this trapezoid, how are they being basically, being defined. So, other possibilities are that I have are something like this. Perhaps, the most complicated would be defined by four line segments. Why I am distinguishing between these trapezoids is a essentially to give an idea of, you know, trapezoid can be defined by, sometime, I am trying to get a kind of a bound on the number of trapezoids and to do that I am looking at in how many ways a trapezoid can be defined. So, if you look at this, the first one, so, it is a special kind of a trapezoid where it is degenerated, one of the sides is, so, one of the side is of zero length that is, the two line segments meet at a point; the second one is defined by two line segments. So, the difference between 2 and 3 is that, you know, one is, the shorter segment is above, the shorter segment is below, something like that.

No, sorry, just a moment, no, no, wait, wait... So, the difference between 2 and 3 is that the two vertical lines are defined by the one of the segments; in 3 the two vertical lines are defined by the two segments; in number 4 the vertical lines, the trapezoid is defined by the three segments- there is a segment, there is an upper and lower segment and then, there is a left, a separate third segment that defines the left boundary; in the fifth kind you have the upper and lower segments defining the two bounding edges, and left and the right defining the two vertical edges. And you will be, if you think about it, so, the maximum number of line segments that define a trapezoid is given by number five, so, there are at most 5, sorry, at most four lines that can define a trapezoid.

(Refer Slid Time: 20:27)



The "potential" number of trapezoids is $\binom{n}{4}$ n = #segments

assuming no two end-points lie on the same vertical line.

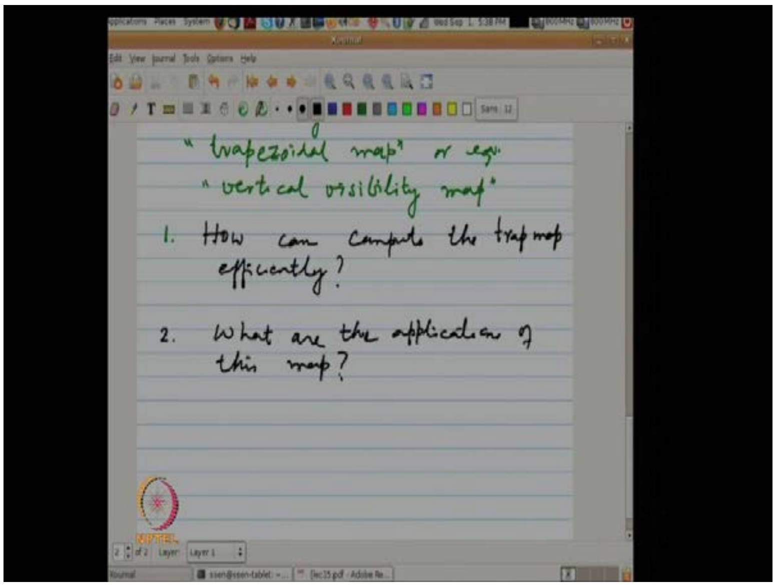
Exercise : Show that #trapezoids $\leq 3n+1$

If you go by this I can, we can state the following that the potential number of trapezoids, I will just say what I mean a potential number of trapezoids, is n choose 4 n say, number of segments, assuming again, our simplifying assumption that no two points, end points lie on the same vertical line. Why am I saying n choose 4? See, the way I have defined the trapezoids here are those trapezoids those do not contain any other line segments inside, these are the kind of empty trapezoids, but you know, I could define a trapezoid let us say, using this and this, and if I drew that, you know, so, if you drew that trapezoidal, just that inside that there will be many line segments, but we know that a trapezoid can be defined by a maximum of four line segments therefore, I am just saying that potentially there are n to the power 4 trapezoids. However, this particular diagram that I have drawn using the rule that I extend the vertical line till it hits the first line segment, with that it is a planer structure and therefore, it has a big o of n trapezoids.

And I am claiming that, you know, there is a simpler method to count where you can basically show the following that, so, this is an exercise to show that number of trapezoids less than or equal to 3 n plus 1; so, to get this number I do not want you to use Euler's formulae, you know, just do some kind of charging arguments that look at the how the left boundary can be defined, how many left boundaries can a single point define, just count it that way- so, with that you should be able to get this figure. So, it is a fairly small constant, 3 n plus 1. If so, this is the number of trapezoids if you follow the rule that you know we are not going to extend the line segments till it hits the first trapezoid.

And this n to the power 4 can come only if you are allowed to draw these vertical lines even through those line segments and still there can be at most n to the power 4. Why this number is, you know, we will see later, you know, why it is useful, it is still not a huge number, it still a polynomial, that is what we should keep in mind that given n line segments the potential number of trapezoidal is still polynomial by the way we have defined the trapezoids. We will not really try it right now, but I just want you to keep track of that.

(Refer Slide Time: 24:14)



" trapezoidal map" or equi.
" vertical visibility map"

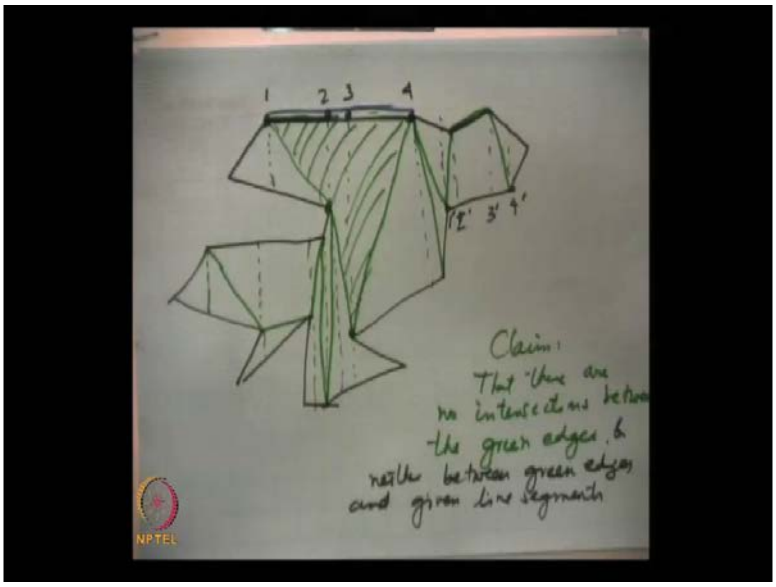
1. How can compute the trap map efficiently?

2. What are the application of this map?

So, here is how then, this diagram is called the trapezoidal map or equivalently vertical visibility map. So, the first question is how can we compute the trapezoidal map efficiently? That we can do it in polynomial time, there should not be any doubt. So, for every point I draw the vertical line I just have find out, it is a ray shooting problem, somehow I have to find out the first segment that it intersects. So, on every vertical line I can actually sort the line segments and I can get all that information, but you know, that will be kind of expensive, may be n square or n cube, something like that. We want to do it faster, so, that is the first problem, so, how can I compute the trapezoidal map efficiently, and second, you know, what is our interest in this problem, you know, why are we looking at it? So, I will try to answer the second question first and then we will make a few observations about the first one, you know, how do we actually construct it. So, what is an application and what are the applications? And in particular we want to see how is triangulation related to this problem?

Well, on a very abstract level, you know both of them, you know, triangulation and in vertically visibility map or trapezoidal maps, have got something to with visibility after all, triangulation also, whenever you are drawing a diagonal it should not intersect any of the edges of the triangle, or other triangles, so that has got some relevance to visibility. Similarly, in the visibility map of course, it is a direct, you know, it is just a vertical ray shooting, that is a, that is certainly a visibility problem, so, there is some connection, but we will establish a, you know, very nice relation between two problems. So, what we do is, so, let me shift to the other one.

(Refer Slide Time: 27:26)



Claim:
That there are no intersections between the green edges & neither between green edges and given line segments

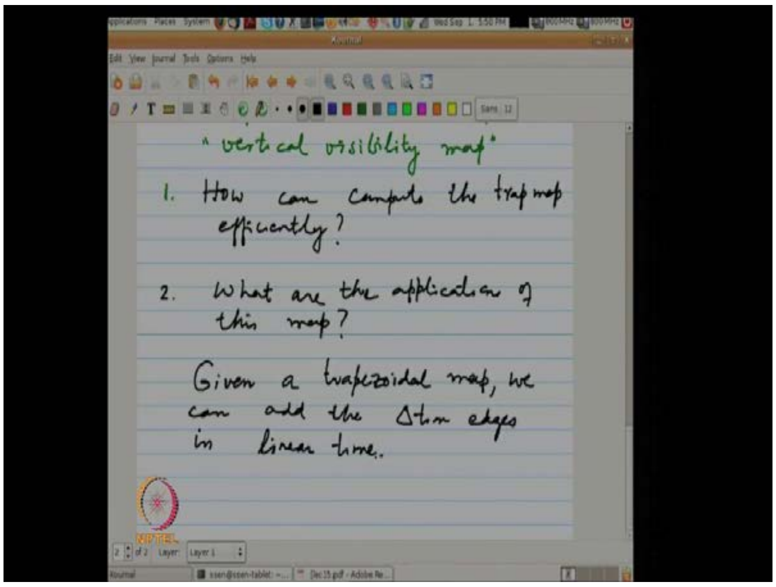
Now, when we do the vertical visibility map we are drawing these vertical things. Now, look at any edge suppose, I consider this edge, this is the edge, and look at all the vertical lines that are getting terminated at this edge, so, we have actually, and let us number them, let us number them in a, in the order that they intersect, so, let us call it, so, for this particular edge, this is of course, two end points like this, so, I start with some end, one side, it is 1, the next vertical line that intersects that this, 2, this one then, we have 3 and then we have 4 and that is all, those are all that the four vertical lines that terminate on that edge. Likewise, we can define for every other edge, so, if you look at this edge, we will find, you know, again, let us say, this is 1prime, 2prime, 3prime and 4prime. So, every edge, on every edge we look at the ordered sequence of the vertical lines that terminate on that edge.

Now, these are by definition trapezoids, this region between two consecutive vertical lines they are trapezoids. Now, what we do is starting from one end point in the order that they intersect I will just join these diagonals, so, I will join 1 with the vertex that generated this vertical line, the next vertical line is generated by this point. So, I will join with this and then the next because there is nothing else basically, it just the end point, so, I am going to join it with this. And I am going to do it simultaneously for all the edges; so, for this edge again, then I am going to join it with this and this is of course, by this vertex and then essentially it bends like this. I am not drawing all the diagonals because you know it is becomes too cumbersome to club the diagram. So, when we draw this, you know, this green line suppose, this already exists, I do not have to draw it, I claim that in our observation that there are no intersections between the green edges. Could you believe this? Just see it for a while see, why, perhaps I should also add that, and neither between green edges and given line segments. Let me try to complete this diagram when you think about it.

(No audio from 32:52 to 34:07)

Exactly, the trapezoids are something non-overlapping, so, when I am drawing a line segment it is completely confined within the trapezoid, so, there will not be any intersections. And what I have defined, if you look at this thing, now, the way I have drawn these diagonals in order of the vertical lines this region is- what is this region, what kind of a polygon is this region? So, you know, something more than that, exactly, exactly, yes. So, this is a one-sided monotone polygon, this is the one side and this is the monotone part of it, because there are all in increasing order. So, what do we do, what is the next step? We just triangulate them separately. So, what we have done is by drawing this green diagonals we have decomposed this figure into non-overlapping monotone, one-sided monotone polygons. So, given that, given a trapezoidal map suppose we could actually construct the trapezoidal map, or someone just gave it to us, gave us this information. Then the next step is rather easy, that is, you know, we can actually triangulate each of them in, each of them in linear time and so when you some up all these things it will still be linear because an edge is, edge is shared between at most two polygon- so, even if you sum it up, it will still be linear.

(Refer Slide Time: 36:29)



"vertical visibility map"

1. How can compute the trap map efficiently?

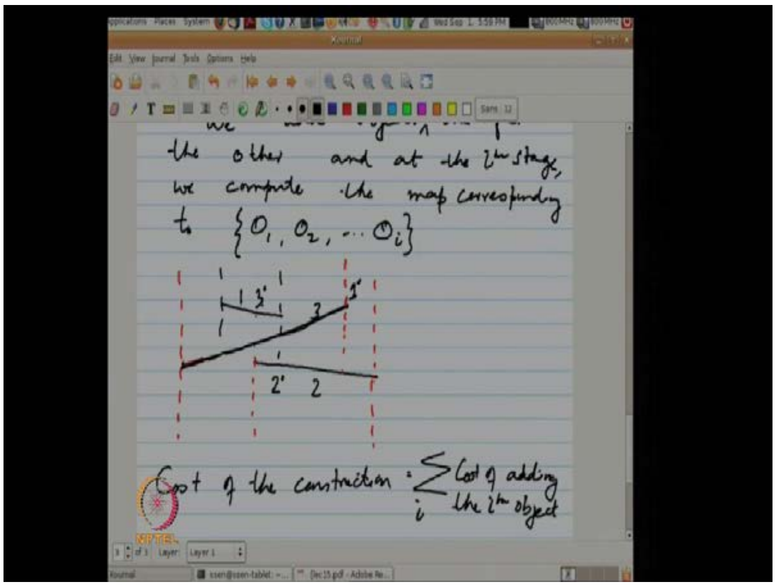
2. What are the application of this map?

Given a trapezoidal map, we can add the Δtion edges in linear time.

So, the moral of this story is that given a trapezoidal map, we can add the triangulation edges in linear time. One information will be required necessary that is, you know, whenever we are adding these edges and drawing the vertical lines, you know, what is the inside and outside of the polygon, you know, so, with every edge, you know, we should know - which is the inside, and which is the outside? So that information is, you know, I am not saying that we have to compute it, but that information will be required to do all this things. So, the vertical visibility map should be consistent with the inside and outside of the polygon. Now, so therefore, the time, one way to triangulate would be to compute the visibility map and then just follow this procedure. But now, how do we, so, how much time do we need to compute the trapezoidal map and how do we do it?

So, this problem turns out to be not very difficult, but you know, it is quite interesting and there are many ways it can be done. I will introduce a method which we will, it will be, it is like machinery that we will develop later. So this is a kind of an introduction to the that the general mechanism of you know, solving many problems in computational geometry and that is the method of what we call incremental construction.

(Refer Slide Time: 38:45)



the other and at the i^{th} stage, we compute the map corresponding to $\{O_1, O_2, \dots O_i\}$

Cost of the construction : $\sum_i$ Cost of adding the i^{th} object

So, the generate method is a following, as the name suggests what you do is we add objects, in this case line segments, one after the other and at the i eth stage we compute whatever is required, compute the map corresponding to, so, let us say, we have actually numbered these objects x1, x2, O1, O2 till On, corresponding to O1, O2, Oi. So, for a trapezoidal map we start with let us say, one line segment and that is nothing but this, we add the next line segment, it could be, you know, it could be anything let us say, it is this, if it is this, then what we do, we have to erase this part introduce. So, this is the, this is the trapezoidal map corresponding to two line segments with, add the third line segment, could be something this. So, once it is that, we have to make some modification. What are the modifications we have to make? We need to remove some of the extensions which cannot go beyond this and keep continuing that. So, when we introduce a new line segment it can describe some of the existing trapezoids, and it will create some new trapezoids.

So, this is this is what is called incremental construction, we just take up these objects, these line segments one after the other and do this. Now, the same thing could also be applied to convex hull constructions you know, we can pretend that we are picking up points one after the other. And I probably had made a comment like this in the class that we could consider the convex hull in incremental fashion which is nothing, but I was trying to mimic the insertion sort. So, this can be thought of a general paradigm, we are going to add these objects one after the other and whatever this is the final, sorry, the intermediate construction. We must have the correct construction corresponding to that, and then we add the next object, and then we update whatever construction we had to have the correct construction for the next object. So, this is by the way not a dynamic situation because these are given to us; so it is not that I do not know what the next object is?

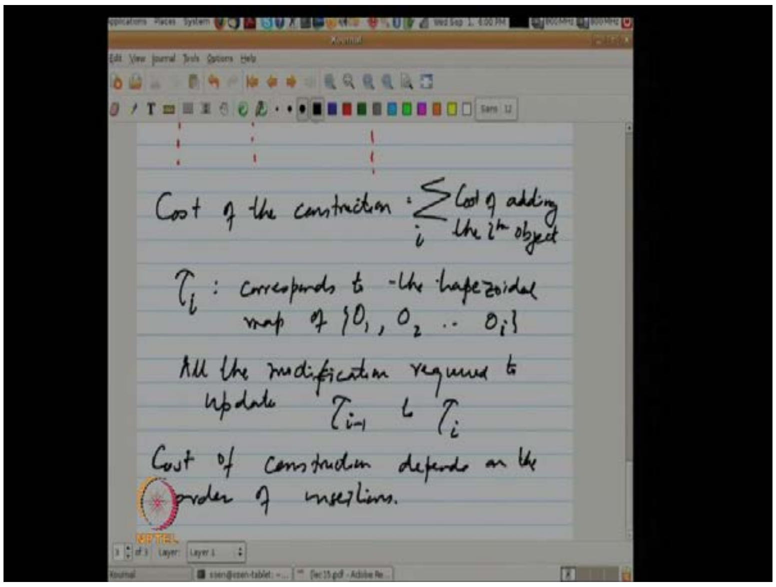
So, let us not confuse with the dynamic situation, where we do not know what the next object is, you know, that is a different problem altogether and could be a hardware problem. Here we have, we know all the objects, we just kind of pretending as if that you know, we are that they are coming one by one. But we have it available in the first place, and then when we are constructing it, you know, by just inserting the next object in the next object and so on and so forth, we are never deleting an object, so it is certainly not fully dynamic and it is not even like semi-dynamic, because as I said all the objects are available in the beginning, right in the beginning.

So, essentially, the cost of this construction is nothing but summation over i cost of adding the i eth object- cost of adding the i eth object is basically, updating the so, let us say, we call it, let us say, t is of i corresponds to the trapezoidal map of O1, O2, Oi. So, the cost of adding the i eth object is all the modifications required to update t i minus 1 to tau i it is just, you know, writing this summation what we have actually, we have no idea what his bound will be like. However, you can imagine that this bound could possibly just reflect on a little bit, could possibly depend in the order that we are going to add the object. Can you think about situations where something could be bad and something could be good, I mean better? Yes, so we could, exactly, so, we could pick up a segment which intersects lots of trapezoids. So, in fact, here when we are drawing, looking at this diagram, you know, this third segment that added actually went through a lot of trapezoids however, if I added that trapezoid, sorry, I consider that as a first segment.

So, suppose this was the first, the order in which we insert was 1, 2 and 3, you know, instead of that if I change the ordering to let us say, this becomes, I am calling it 1 prime, let us keep 2 as 2 prime and this as 3 prime clearly, when we add the 3 prime the number of things that are changing is very small, you know, just one trapezoids is getting changed. But if we add this long one, long one in the end, then you know more trapezoids could be affected. And it is not necessarily that, you know, we should be looking at long and shot, you know, it depends really on the actual configuration, the actual (( )) configuration is what is going to basically make a difference, the long one because actually goes through all these trapezoids; the long one was, you know, we off somewhere I did not matter how we, when we inserted that.

Yes, I mean sure, I mean, but the point is, I mean, how do you know beforehand, you know, which is the good ordering to introduce good ordering for the insertion.

(Refer Slide Time: 47:15)



Cost of the construction : $\sum_i$ Cost of adding the i^{th} object

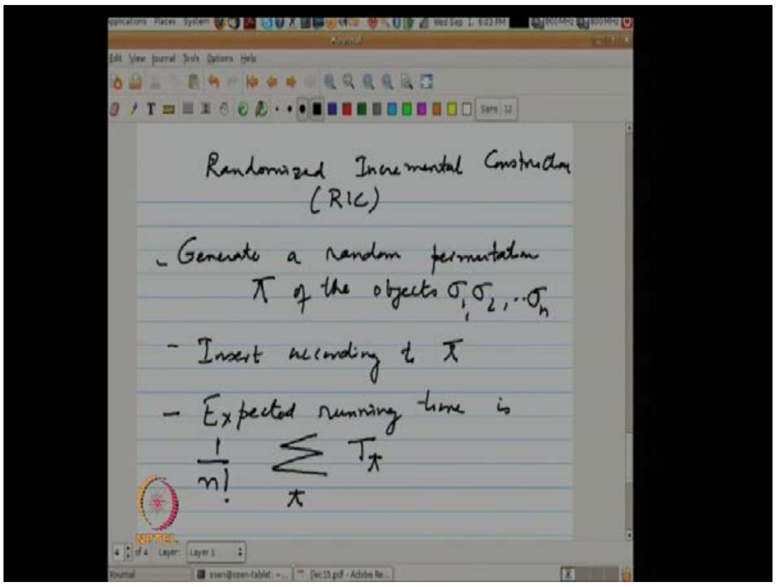
T_i : corresponds to the trapezoidal map of $\{O_1, O_2 \ldots O_i\}$

All the modification required to update T_{i-1} to T_i

Cost of construction depends on the order of insertions.

So, all I am saying is that the cost of construction depends on the order of insertions. So, it actually introduces another variable in the whole process. Now, how do we know which ordering should we choose, and when we do not know anything about these things what we do? Choose random order and hope that, you know, it will take care of itself, seems to work in many cases, rather magically.

(Refer Slide Time: 48:03)



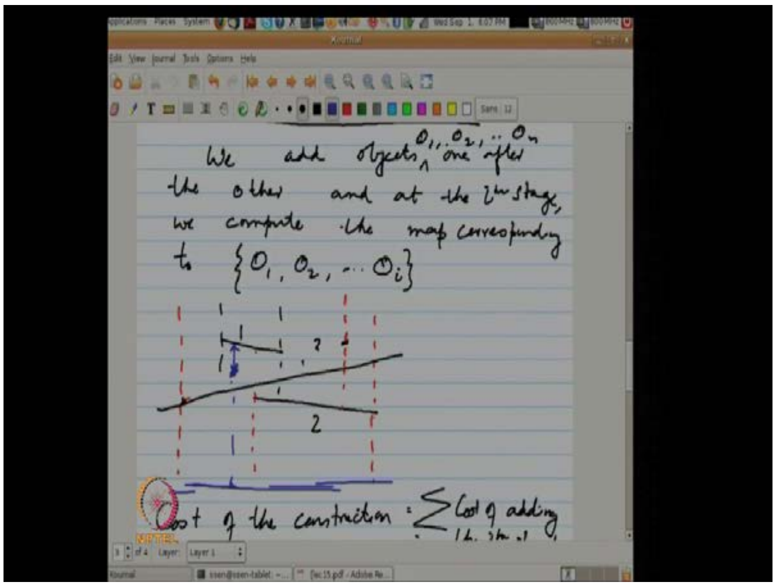
Randomized Incremental Construction (RIC)

- Generate a random permutation π of the objects $\sigma_1, \sigma_2, \dots \sigma_n$
- Insert according to π
- Expected running time is $\frac{1}{n!} \sum_{\pi} T_{\pi}$

So, choose, we call this procedure randomized incremental construction, RIC for short. And what we do? The first step is that generate a random permutation pi of the objects. So, let us call these objects sigma 1, sigma 2, sigma n and insert according to pi. So, then, how do we measure the running time of this or the cost of this? So, we will basically, take the average over all possible permutations. So, the expected running time is the average. So, let me just write it mathematically. Summation over all pi, 1 over n factorial is a possible number of permutations, and the time corresponding to pi, still actually, just some equations, you know, nothing really, there is no meat in it at this point. So, why should this be good? We have to prove it that this is good.

And this is something that I will not be able to do today, or may even in the next few lectures, but when we address this problem, you know, we will develop a whole, you know, a whole mechanism of analysis which will also be applicable to this trapezoidal map. So, just for today just be aware that, you know, the we are going to, the way we are going to construct the trapezoidal map is do a random permutation of segments and build the map incrementally by inserting one segment after the other and update the map, that is the way we are going to do it. And finally, you know, what we will be looking at is expected running time of this over all possible permutations, it does not depend on the line segments it only depends the permutations, whatever be the line segments the analysis that we do will only just do an averaging over the permutations. But it should, it will not matter what the input line segments are, so it is worst case over any set of line segments, that is a way we will do the analysis later on.

(Refer Slide Time: 51:23)



We add objects $O_1, O_2, \ldots O_n$ one after the other and at the i^{th} stage, we compute the map corresponding to $\{O_1, O_2, \ldots O_i\}$

Cost of the construction : $\sum$ Cost of adding O_i

The other thing that I would just draw your attention to quickly is drawing, sorry, constructing the trapezoidal map also has a byproduct in the form of a point location data structure location data structure, that may not be obvious, how, or maybe it is to some. Why am I claiming that? Let us go back to this figure again where I inserted these line segments one after the other. So, there is some point, you know, which we want, you know, we want to find out in the final trapezoidal map, you know, where that point is located, or in other words what is the vertical, what is the answer to the vertical ray shooting.

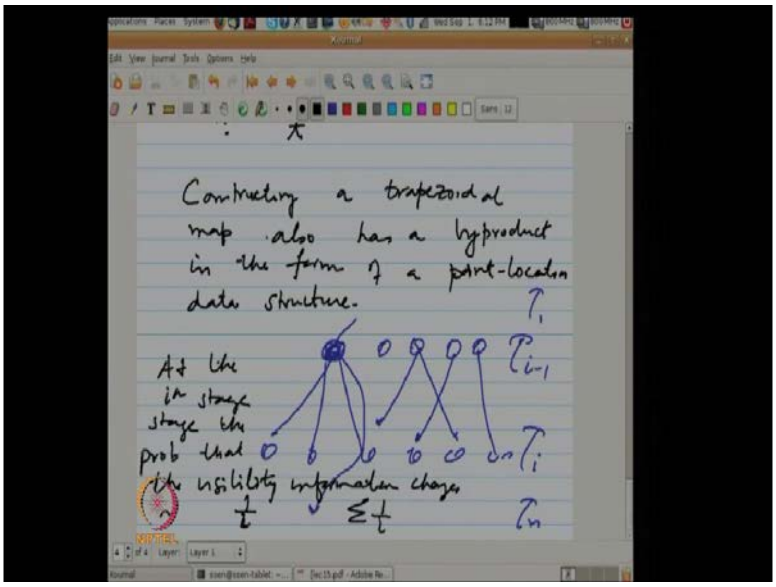
Remember that these are equivalent problems, the point location, which sub division it falls in the same as, you know, when we shoot a vertical ray, which line segment it hits. So, what I am claiming is that so, before we added this line segment consider a point, may be it was this point, this blue point. So, when these two line segments are there the vertical visibility, sorry, the vertically visible line segment above this was essentially 1. So, 3 prime, let us forget about the 3 prime, let us not confuse, plus 1, and the below one, you know, it is in a box, may be it is, you know, just the one that is, that is below.

Now, when we actually added the next line, next line segment things changed, you know, because it actually went through this, and for this particular point the visible, the downward visibility line segment changed and it could be so for many other points too depending on what point you are looking at. But then what is happening is that as we keep on adding the line segment, you know, I am actually developing a superimposition of trapezoidal maps, there was initial trapezoidal maps of i segments when I added the i plus first line segment I obtained another trapezoidal map which is superimposed and I can think about superimposing these two things.

So, if I again knew my point, my visibility information with respect to the trapezoidal map i minus 1, I can if I keep, if I know about the interaction of these two t i and t i minus 1, I can refine my, I can refine that visibility information- exactly the way that we did in the even in the case of (( )) decomposition, you know, we thinned down the triangulation and then we keep on adding triangles and we made sure that we kept track of which triangles interacts with another triangle similarly, here also we are going to keep track of which trapezoid interacts with another trapezoid as we keep on adding line segments?

So, this is sometimes also called the history. So, if you maintain the history of this trapezoidal map there by basically maintaining the interactions of the trapezoids at level i, i plus 1, i plus 2, etcetera, that whole thing is also a data structure, it is a simple data structure for the point location, or vertical visibility map, or ray shooting, or whatever you wanted. So, this is again a generic method that people use for building data structures; when you do the incremental construction as the byproduct if you store them actually, I do not need to store them if I am only looking the trapezoidal map, I only look at tau of n and I am done, but if I keep track of this tau i, tau i plus 1 and how they interact, how the trapezoids of these two maps interact, then I also get a data structure, a point location data structure right out of that.

(Refer Slide Time: 56:00)



Contructing a trapezoidal map also has a byproduct in the form of a point-location data structure.

At the i^{th} stage the prob that the visibility information changes $\frac{1}{i}$ $\sum \frac{1}{i}$

Now, the question is of course, again, that the data structure that we get, you know, can again be defined by some kind of a level graph, you know, here are these trapezoids of, you know, level this is for c i minus 1and then, I have these trapezoids of tau i and how they actually interact, that will basically give us this, give us the point location data structure, same thing. Again you know, it is going to interact in a certain way and you know, eventually we are going to chalk a path through this, though this a level graph. However, unlike when we did (( )) decomposition, where we proved that the degree of each node is constant etcetera, here we have no such bound.

So, again, when we do that full analysis of the randomized incremental construction we will show you that, you know, it actually leads to a very efficient data structure where we cannot prove that, you know, every node has a has a constant degree, but we will be able to prove something like the expected degree will be constant. So, that eventually, again, you know, but node that, you know, here we have, we are not talking about, you know, throwing out half the vertices, here we are adding one segment after the other, so, we are going from tau i minus 1, tau i all the way up to tau n, so, they are not log n levels, so, they are tau 1, tau 2 all the way up to tau n levels.

So, we are not going to copy these nodes at every level, what we will show is that for any given path that path will not visit more than about roughly log n vertices even if we are adding one segment after the other; because what is happening is that this segments are going to only affect things locally, so, this point of course, got affected by this line segment, but it may not get affected by, you know, this line segment, it really depends on the order in which we inserted the line segment; we can have another line segment here, now, if this line segment 1 was not inserted before the let us say, this is number 5, then it will get, if we insert 5 after 1, it will not get affected by 5, but we inserted 5 before 1, then it will get effected.

So, this, you know, I will actually, when we do the analysis later on I will demonstrate this with the help of actually insertion sort, because that again has a similar analysis I can do. And then, we will see that actually at any point the chances that it will get affected; I will just give just an idea- at the i eth stage, stage the probability, so, you know, right now it will all sound, you know, very wage. But let me just (( )) the probability that a, you know, the visibility information will change is proportional to something like 1 over i, so that when you some up over 1 over i it is log n. So, this will be the generic, general sort of analysis that we able to, bounds that we will be able to prove. And but how this probability came and how etcetera, these are, you know, not that complicated, but you know very, very cleaver, the person who came up with, you know.... So, I will stop here today.