Computational Geometry

Prof. Pankaj Aggarwal

Department Computer Science and Engineering

Indian Institute of Technology, Delhi

Module No. # 11

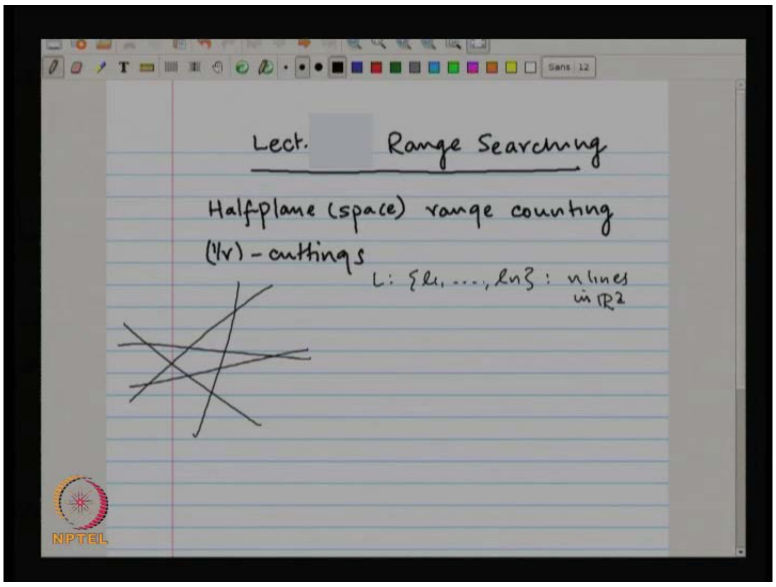
Range Searching

Lecture No. # 05

Range Searching Half Plane (Space) Range Counting

So, I will continue with the half plane range counting. In the last class we talked about half plane range reporting, and then I we discussed if we allow quadratic space how one can answer half plane range counting in logarithmic time. Today, I want to give you somewhat different solution which also extends to higher dimensions and also the technique that I will use here will be useful for the other extreme when we have only linear space and you want to answer queries efficiently. So, half planes is called half space in high dimensions, the notion that I want to introduce today is called cuttings.

(Refer Slide Time: 01:10)



Lect. Range Searching

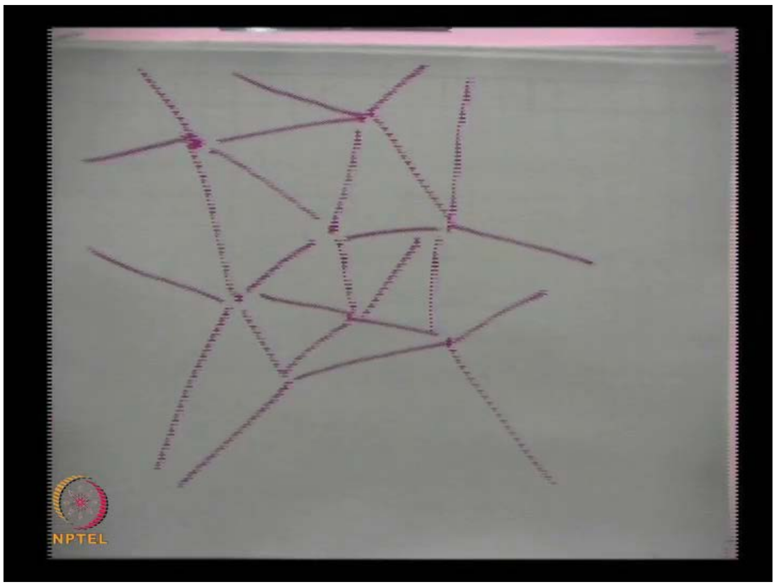
Halfplane (space) range counting

(1/r) - cuttings

$L: \{l_1, \ldots, l_n\}$: n lines in $\mathbb{R}^2$

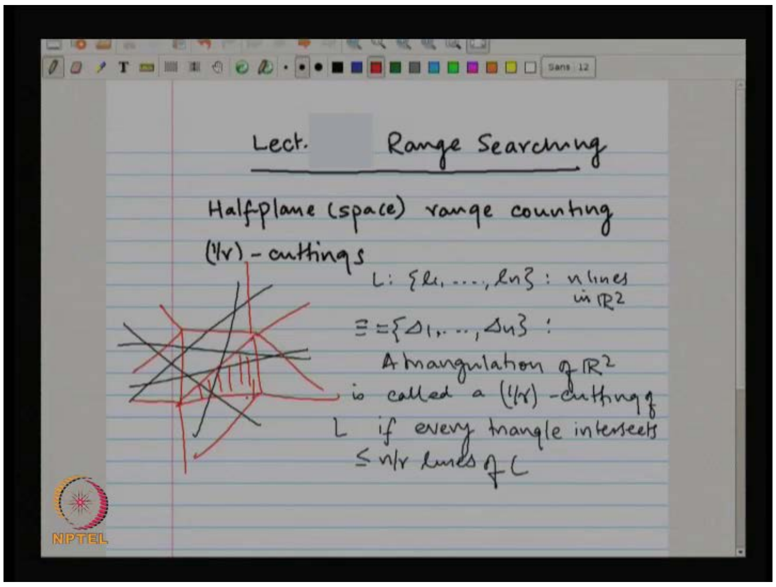
So, I will describe it in 2 d and then you can think about it higher dimensions as well as the same concept. So, suppose you are given a set of lines.

(Refer Slide Time: 01:51)



So, L is a triangulation of plane is a planer sub division. So, so triangulation is a planer sub division in the plane if I talk about 2 D triangulation. Now, there is one technicality that we are talking about the entire plane. So, some of the triangles will be unbounded because plane is infinite. So, we define the triangle as an intersection of 3 half space half planes. So, this way basically that is what I was trying to draw that, but that is what you think about triangulation.

(Refer Slide Time: 02:48)



Lect. Range Searching

Halfplane (space) range counting

(1/r) - cuttings

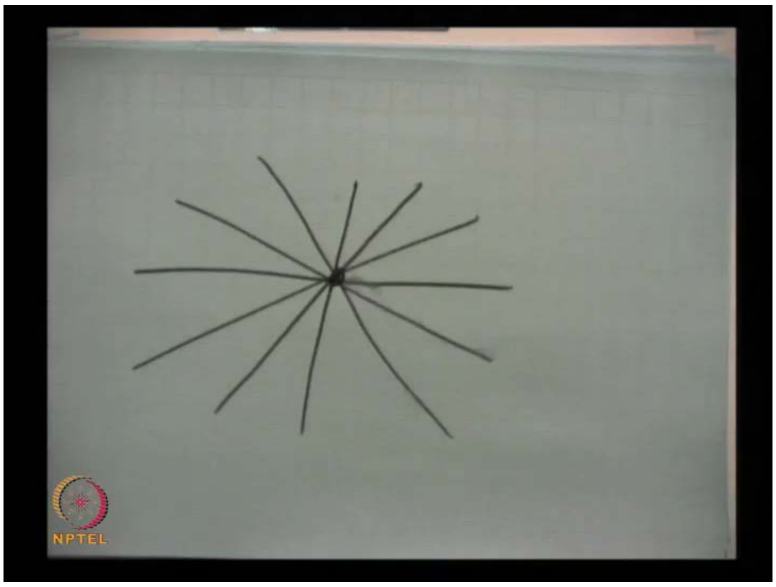
$L: \{\ell_1, \dots, \ell_n\}$: n lines in $\mathbb{R}^2$

$\Xi = \{\Delta_1, \dots, \Delta_n\}$:

A triangulation of $\mathbb{R}^2$ is called a (1/r) - cutting of L if every triangle intersects $\leq n/r$ lines of L

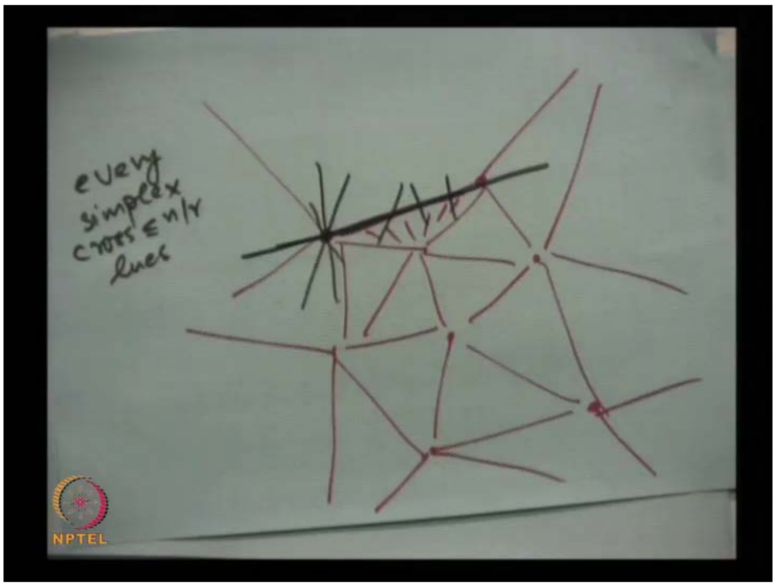
So, 1 over arc cutting we have a triangulation. A triangulation is called off 1 over r cutting of L if every triangle intersects at most n over r lines of L. So, it means that what you want to do is have a triangulation, something like this red lines, this vertices of the triangulation have nothing to do with arrangement, they can lie anywhere. What you need is that property that you should take any triangle and you count the number of lines that intersect the triangle that are at most n over r. So, every triangle intersects a few lines. Now, there is one technicality that I should point out and that is of following.

(Refer Slide Time: 04:35)



So, suppose you have situation that all lines pass through a single point. If that is a case then what the select of triangulation you use then this point will lie in some triangle and that triangle if the this point and the n line passes through this point (( )) we cannot have such a condition because this will there is no way that you have a cutting 1 over arc cutting because all L lines passes through a single point. So, if that is the case we need to come up with the more technical definition. So, what happens is when you go back to triangulation then what is called a complex some special complex.

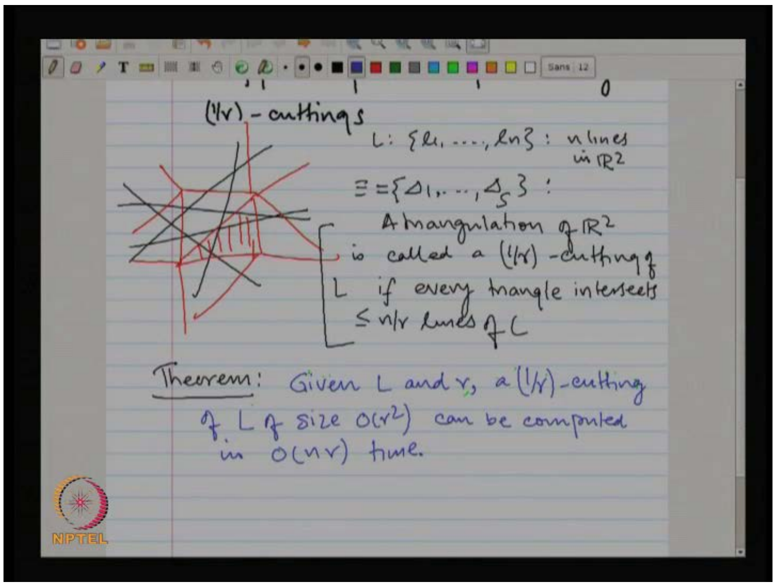
(Refer Slide Time: 05:19)



If you remember that a vertex is a 0 degree simplex and an edge is a 1 dimensional simplex and triangle is a 2 dimensional simplex. So, what you sort of say is that the you define the cutting that you have a triangulation such that if you take any simplex and now when I talk about the triangle or any simplex these are opened because I think about this triangle two-dimensional triangle you do not count its boundary. You just take the interior. Similarly, when you talk about this edge you look at only the interior of this edge you do not count its end point, end points are counted only at the vertices. So, you say then you say sort of say is triangulation is a cutting if every interior, if every triangle or every simplex crosses n over at most n over r lines. So, every simplex crosses what this means is a you say a line crosses a simplex if line intersects the simplex, but line does not contain the simplex.

So, for example, if you have lot of this lines passing through a single vertex you do not care about those lines that is. So, do not count because for vertex crossing does not make sense. Now, in the case of edges you look at the things that only passes through the interior that crosses is a you do not consider the line that contains this edge. So, that is the technicality which becomes even more important in high dimensions, but I will be sloppy and I will use the definition that I wrote on the on the slide you sorry.

(Refer Slide Time: 07:23)



(1/r) - cuttings

$L : \{\ell_1, \dots, \ell_n\}$: n lines in $\mathbb{R}^2$

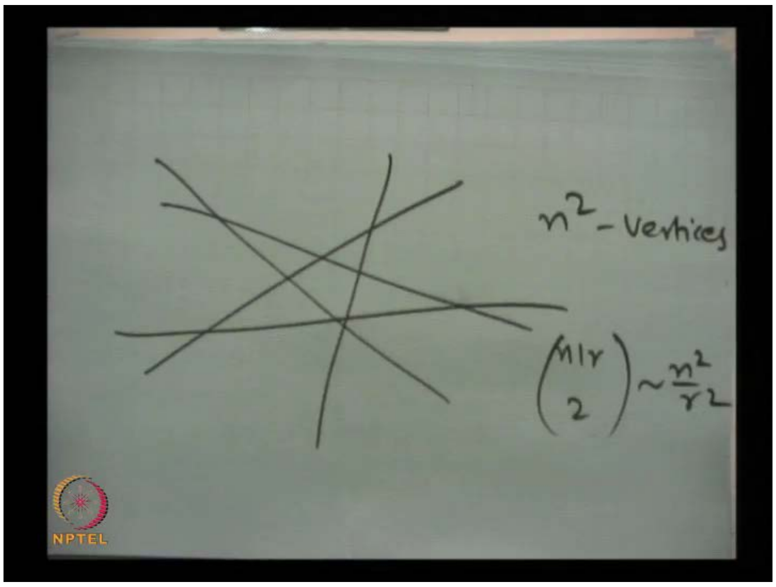
$\Xi = \{\Delta_1, \dots, \Delta_s\}$: A triangulation of $\mathbb{R}^2$ is called a (1/r) - cutting of L if every triangle intersects $\leq n/r$ lines of L

Theorem: Given L and r, a (1/r)-cutting of L of size $O(r^2)$ can be computed in $O(nr)$ time.

So, this I should write it, Maybe I should use s that is all. So, so question is what is a size of 1 over r cutting. If you want to have this condition, if you want like to have this condition then how many triangles do you need and here is a theorem given L and r a 1 over r cutting can you read it or it is not good color.

Let me try this 1 from to. So, that is a theorem. What it says is there is a cutting of a size order of r square. So, one thing is that first thing observe is the size of cutting depends on r, but not on n its independent on n. So, does not matter how many lines you have that always you are cutting of size r square and it can be constructed in time order of an r. Now, first I should tell you that this is a right answer because we cannot hope to do better than r square.

(Refer Slide Time: 09:37)



So, here is a simple argument. Now, if you let take L lines, you take n lines how many vertices are there in the arrangement and choose 2 all right, where n square I will ignore the constant then n square vertices. If a triangle contains only n over r lines how many vertices it can contain?

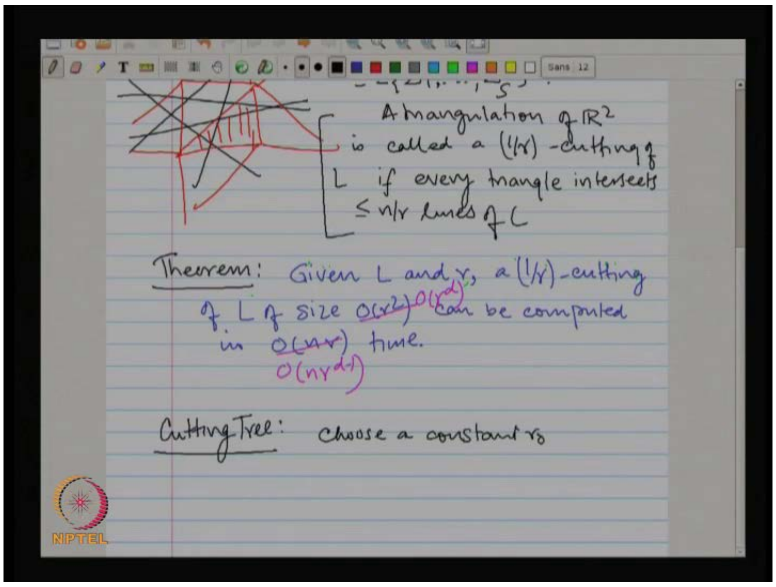
(( ))

Well, because I am not sure where (( )) shows up.

Even say n by r choose 2 or. So, where this. Did you mean n denominator r choose to or you n by r in the. So, you mean that such you mean. So, this is the. So, this is saying n square over r square, alright. Its n over r lets say quadratic it is such what you should think about is the have k lines then you have a k square vertices. If you only n over r lines intersecting, it will not have more than n square r square vertices there. So, if a triangle can contain only n square over r square vertices and this is triangulation of the all plane each vertex are lying 1 of the triangles.

So, you need at least r square triangles to cover all the vertices. So, that says that r square is the best you can hope for. So, this is synthetically the best one can do. Now, I will not give you the proof and I will defer it to my extreme colleague Sandeep, he will do it which is next Friday or next week and it is a very fundamental resulting geometry and It is used to develop many geometry data structure that you will see here, but also for many divided conquer algorithms it is a. So, that is a theorem. Now, in it is also too in high dimensions. In high dimensions triangle becomes simplistic and you have hyper planes. So, instead of lines your hyper planes in three dimensions and triangles become simplices.

(Refer Slide Time: 11:54)



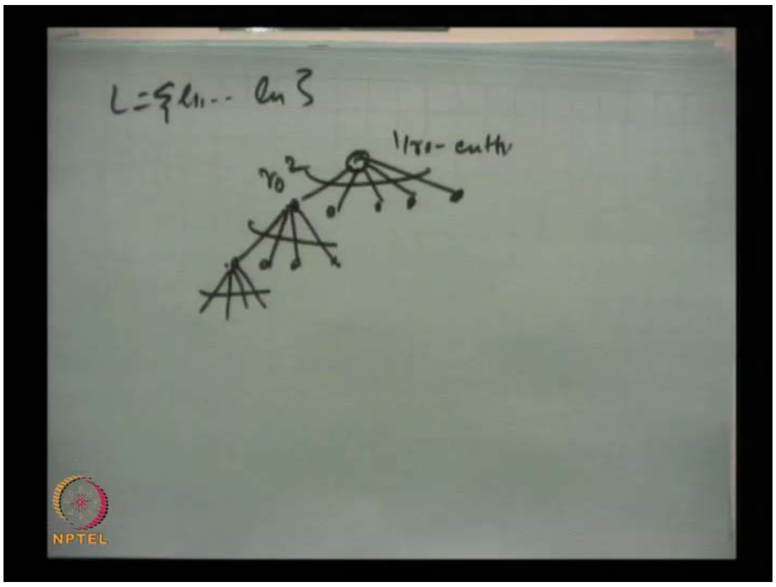
A triangulation of $\mathbb{R}^2$ is called a $(1/r)$-cutting of L if every triangle intersects $\leq n/r$ lines of L

Theorem: Given L and r, a $(1/r)$-cutting of L of size ~~$O(r^2)$~~ $O(r^d)$ can be computed in ~~$O(nr)$~~ time.
$O(nr^{d-1})$

Cutting Tree: Choose a constant r_0

And then what you know is the following. The size becomes r to power d and this becomes r d minus 1. So, you pay factor of r for each dimension that is what that is what you should think about. So, now, why is this result interesting in the context of half plane range counting? For that I need one more concept and that I will call cutting tree data structure. We think about. So, cutting tree is a computing the cuttings recursively. So, choose a constant r 0 and what you going to do is you are going to compute cuttings recursively. So, in the beginning you have all lines. Lets given a sort of L lines.

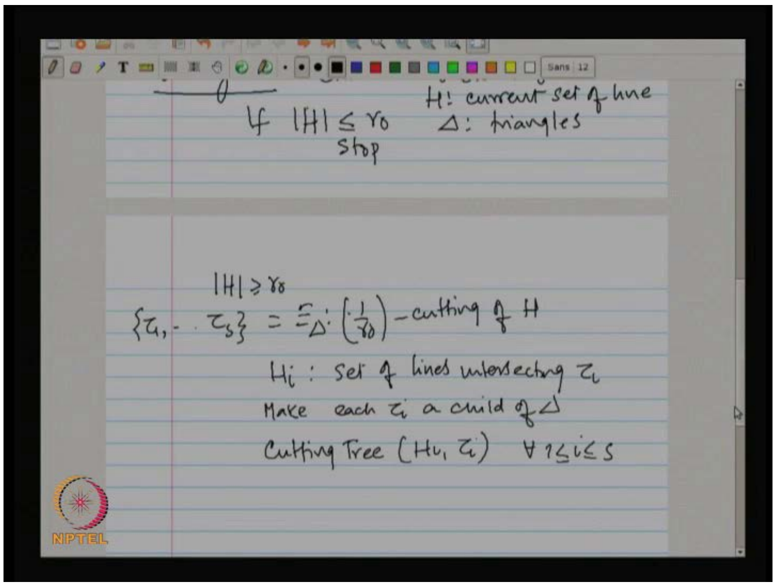
(Refer Slide Time: 13:0)



So, you have set of lines. What you do is choose the constant r 0 you compute 1 over r 0 cutting. So, at the root you compute a 1 over r 0 cutting. So, it will have r 0s and ignore the constant you to have r 0 square triangles for each triangles intersect fully n over r 0 lines and you do the cutting for those lines again. So, you get each triangle, each channel corresponds to 1 of triangles of the cutting. So, I will write it more formally on the slide, but this one is to give you the idea. So, compute the cutting, take a triangle for the cutting, look at the lines that intersected, you compute the cutting again for those lines and do it again.

Now, you get another set of triangles. For each triangle now, you look at the which lines get intersects and do the cutting again and you keep on doing it until you are left with the few lines.

(Refer Slide Time: 14:10)



If $|H| \leq r_0$ stop

H: current set of line

Δ: triangles

$|H| \geq r_0$

$\{\tau_1, \ldots \tau_s\} = \Xi_\Delta \left(\frac{1}{r_0}\right)$ - cutting of H

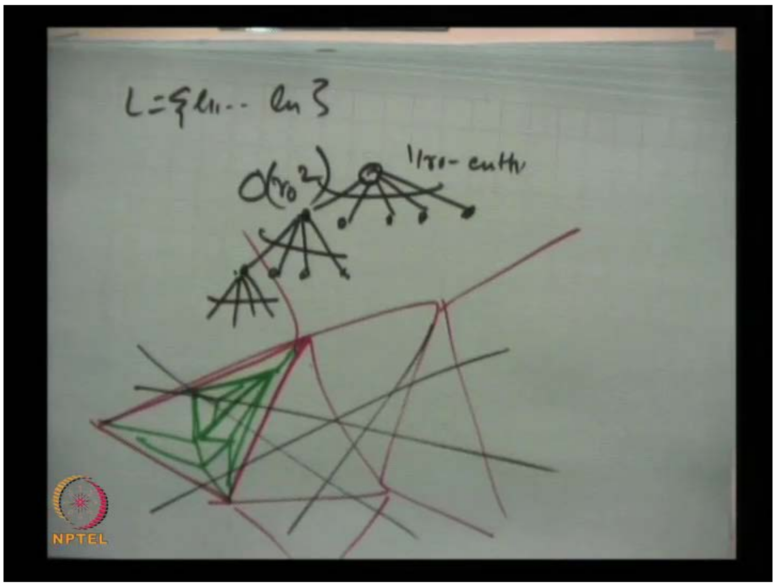
H_i : set of lines intersecting τ_i

Make each τ_i a child of Δ

Cutting Tree (H_i, τ_i) $\forall\, 1 \leq i \leq s$

Now, let me write it formally. Let me. Since, I am defining a recursive procedure, that means do the following. The recursively lat say instead of H is the current set of lines and delta is a triangle that these lines intersect. Initially delta is a dark plate because at the root. So, let us call this one as tau 1 tau. So, we complete the cutting of these lines. That H i be the set of lines intersecting tau i. Make each tau i a child of delta and then you can say compute the cutting tree recursively on the H i and tau i that is all.

(Refer Slide Time: 16:11)

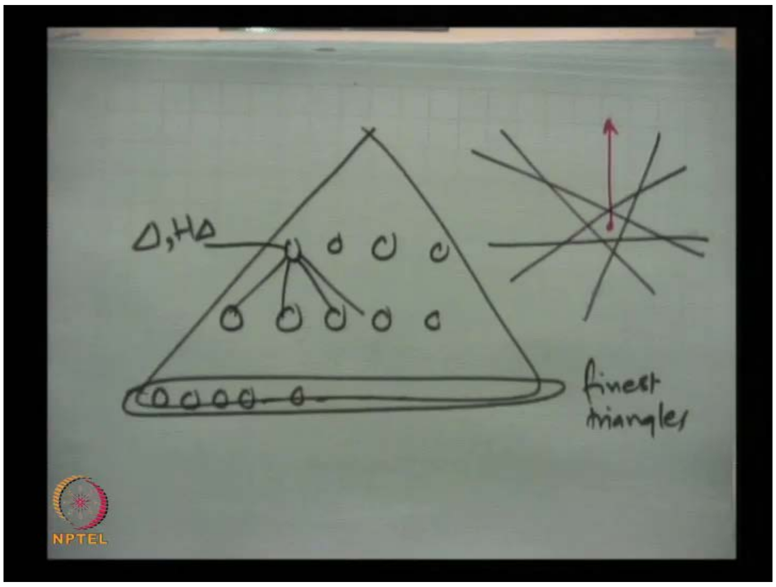


So, what happens is that if you think about initially you have a set of lines. I compute a cutting something like this. Now, I pick up a triangle in this cutting. Look at the lines that it intersected and I compute a another cutting here. Something, like this. Now, you have another triangle recursively look at the here is the only 2 lines may be you will stop it because r 0 is x more than 2 otherwise you recursively compute, keep on do it again. So, this is all.

So, this way you get a tree because look at the unravel all the recursion what you are getting a tree and the fan out of tree is large its r 0 square because the size of the cutting is r 0 square or some constant times r 0 square. Does this makes sense or should I repeat this cutting tree? Good question. You will see why they are important because.

So, what you think about is that you are cutting getting a tree, am I right. Each node of the tree is associated with the triangle before which you constructed. Now final stuff that you have is corresponds to a leaf.

(Refer Slide Time: 18:06)



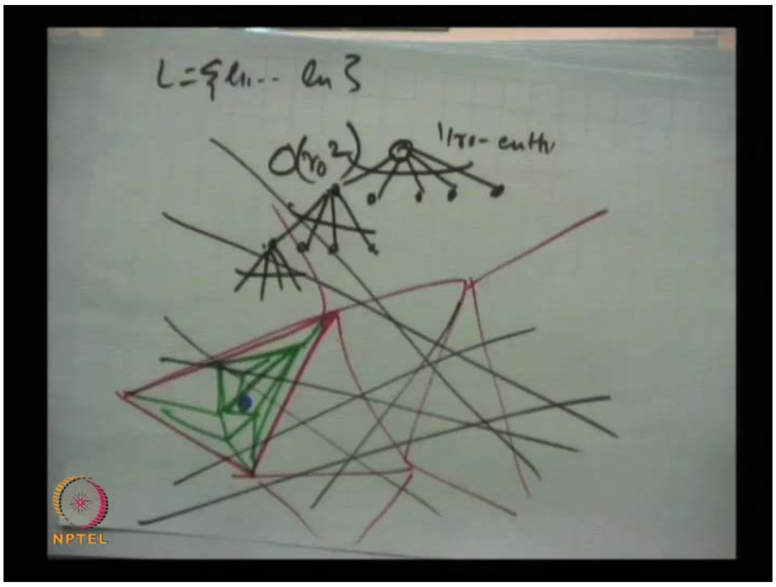
So, let us suppose you build a tree. So, the leaves the nodes the leave nodes you have here you have a triangles, they are the finest triangles and then the nodes here interior nodes they will cause possible a bigger triangle that you constructed and you will see that these triangles will help us to guide us a search because otherwise if you land up here then it will be a problem. You will see and refer when I talk about that.

Now, the question is why I talked about cutting tree? What I want to show you is that you can use a cutting tree to answer half plane, half plane range counting query and this works in higher dimensions also. So, driven by that what is a half plane range counting in the dual, you are have a set of lines and when I give you a query point tell me how many lines lie above this point right. That (( )) the problem.

Now, in 2 D we will lucky we could complete the arrangement and we could compute this. We had that just readymade data structure, point location data structure. You all know Sandeep. Now, you can. There is a point location in 3 D as well the running time is query time is not log on its longer square r, but in high dimension it is not clear now you do point location in generally sub divisions.

So, this technique does not extend to higher dimensions and the cutting tree you will see works in higher dimensions as well. So, for now if I use a how do I use cutting tree using how do you use a cutting tree to solve this problem, can someone think about it. So, just to remind you that each node here has a triangle associated with it and let us say sort of lines 3 H delta, which are lines intersecting there and then you compute a cutting and then you got a refinement of this triangle. What I should do here. So, if you look at this data structure, look at the structure how you will answer you sort of additional information at each node, but then how will you answer half plane range counting.

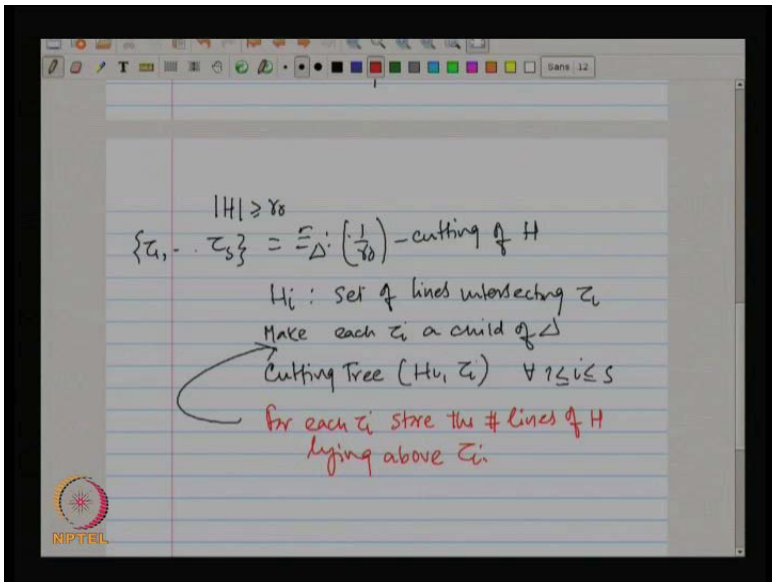
(Refer Slide Time: 20:41)



So, if you look at a cutting, Not all lines will intersect a triangle. So, some. So, if you think about it. So, let us say let me draw of some more lines. Now, I focus on this triangle 2 lines or 3 lines they are intersecting this triangle. So, if I know that as what he said suppose if I know the query point lies here suppose it lies here. Now, the lines that intersected this triangle I have no clue whether they lie above or below, but lines that do not lie intersect this triangle, I know, that any point beside this triangle the answer will be the same.

So, if a line lies below the triangle that does not matter where the query point lies inside this triangle, this line will lie below this point. In if a line intersects line at the triangle, does not matter that the query point lies inside the triangle I know that this line will lie above this point. So, what I can do is since I care about the lines flying above the point for this triangle I will what I do for each triangle I stored the number of lines that lie above this triangle.

(Refer Slide Time: 21:54)



$|H| \geq r_0$

$\{\tau_1, \ldots \tau_s\} = \Xi_\Delta \left(\frac{1}{r_0}\right)$ - cutting of H

H_i : set of lines intersecting τ_i

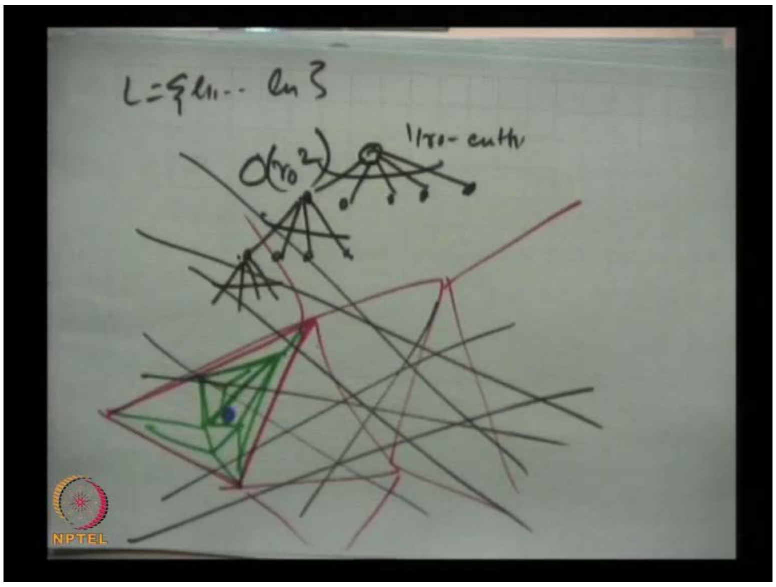
Make each τ_i a child of Δ

Cutting Tree (H_i, τ_i) $\forall\, 1 \leq i \leq s$

for each τ_i store the # lines of H lying above τ_i.

So, what you do is you modify this construction for each tau (( )) the number of lines that. So, I stored this. Now, I do it at every node. So, now, what I know is when I once I know query point lies at this red triangle I know that these 2 lines lie above it, I have counted them, this I do not care. Now, I have to worry about the lines that intersect this point. Now, there is a second level cutting, green cutting for these lines. Now, again I know that lies inside here well in this picture there is no other like suppose

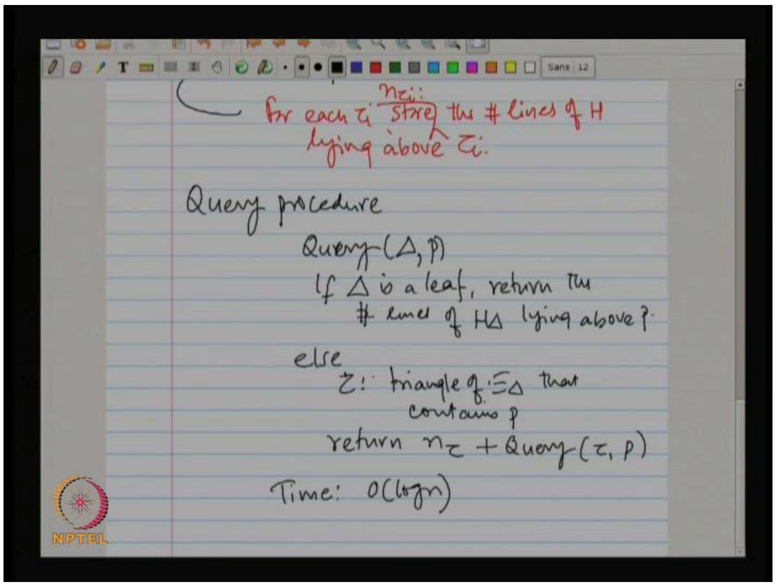
(Refer Slide Time: 22:48)



If you have you also have like this line. Now, I know that if I know that the point I just trying at this line lies above. So, I can now I can adjust this point. Now, I need to figure out only about these 2 point. If I go back this go down to this point. No, because they have already counted them. So, you only worry about the lines that intersect the red triangle because the lines that do not intersect the red triangle.

So, I only need to worry about the next scale I do not need to worry about the lines that intersects this triangle. So, I count them.

(Refer Slide Time: 23:32)



for each τ_i $\overset{n_{\tau_i}:}{\text{store}}$ the # lines of H lying above τ_i.

Query procedure

```
Query($\Delta$, p)
    If $\Delta$ is a leaf, return the
        # lines of $H_\Delta$ lying above p.
  else
    $\tau$: triangle of $\Xi_\Delta$ that
            contains p
    return $n_\tau$ + Query($\tau$, p)
```

Time: $O(\log n)$

So, the query procedures becomes the following. Let us call this quantity as n tau i. . So, query lets set a note w. If delta is a leaf within the number of lines of H delta lying (( )). So, if you think about the tree what is happening the recursively you are at a node, some delta on this tree and you have a set of if it is a leaf then there is nothing there, see just count because there are only r 0 lines. If you remember that the condition for the cut for the a node is a leaf is only if there are r 0 lines there and r 0 is a constant. So, just look at the constant of lines (( )).

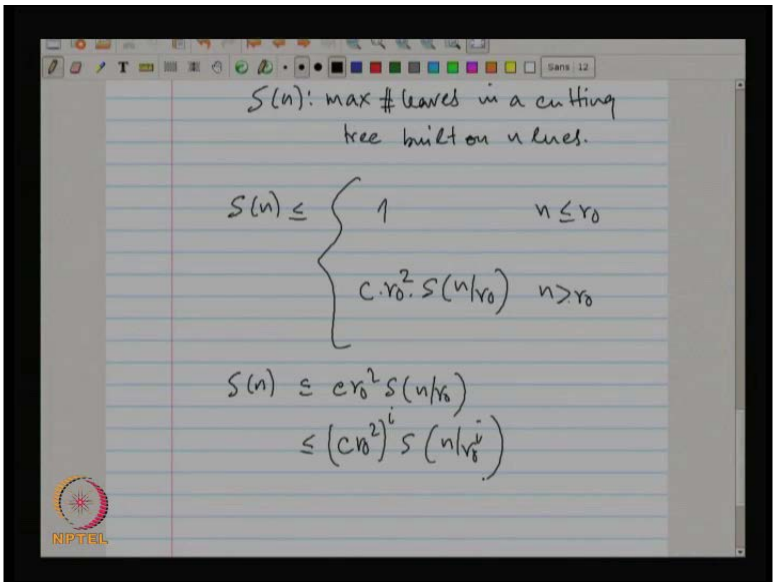
Let tau be the triangle of cutting scope there that contains P. Otherwise, you find the triangle for example, in this case you found a triangle in the red triangle that contains a query point and then what we do is we return the value and tau.

So, how many triangle lie above this rectangle plus recursively count the things here plus query. So, that is all. So, what it sort of says is at let us say look at this node if it is a leaf just count the number of lines explicitly that lie above it because the only constant for the size. Otherwise, find the triangle that contains query point. You have already scored, how many lines lie above this triangle. You have that quantity take that quantity and that recursively among the lines that intersect the triangle and count the number of lines that lie above the query point recursively and add that quantity. That is it

So, what is a query time? So, what is a query time? So, you are following a path. Say it again. So, that is a good question, but I will answer your question a little later because that will be more of the size of the data structure because in terms of query, what is a query will be (( )) pardon. So, it then r 0 is a constant, log of n right because it will log into the base r 0 because what happens in n line because then you get n over r 0 fraction that decreases the size of that fraction. So, is log n ignore the constant r 0 for now and you spend constant time at each nodes in this query procedure. So, query time is log n.

Now, let us ask answer this question. How many leaves are there because since it is a tree the total size of tree will be proportional to the number of leaves.

(Refer Slide Time: 28:16)



S(n): max # leaves in a cutting tree built on n lines.

$$S(n) \le \begin{cases} 1 & n \le r_0 \\ c \cdot r_0^2 \cdot S(n/r_0) & n > r_0 \end{cases}$$

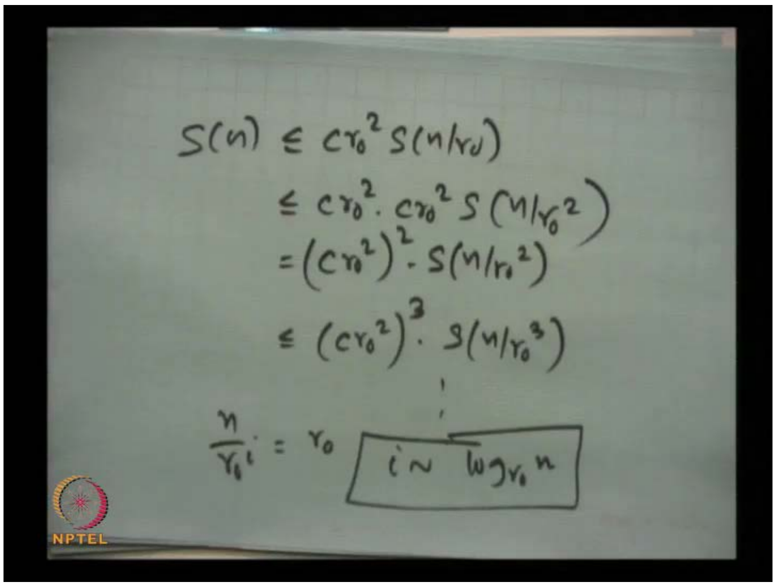
$$S(n) \le c r_0^2 S(n/r_0)$$
$$\le (c r_0^2)^i S(n/r_0^i)$$

So, let us say S n is a size of data structure say let say selection number of leaves in the cutting tree built on n lines So, maximum number of leaves in the cutting tree built on n lines. So, what is a size if. So, if n is less than r 0 then it is a this 1 it is a leaf. So, it is only 1. So, it is a there is only 1 leaf. So, interesting cases n is bigger than r 0 then what happens how many leave then what happens. So, what is the recursive formulae you will have.

So, think about that is n is your counting number place in this root is sub tree here. So, what will be the recursive formula you will write. So, because if the number of leaves is the number of leaves in each of these sub trees 2 tree at each of the children, there are r 0 square the number of things here, number of children and each of them has only at most n over r 0 lines. Now, you have to be little careful because this order of r 0 square that will give us some headache because this is not precisely r 0 square this is a constant so.

What let us write it is C times r 0 squares square and S 0 that is the number of leaves. So, let us look at this sequence. What happens to this (( )). So, if you start expanding it what will happen is if you expand it i times what will you get ? You will get C times r 0 square power i times S because every time you size becomes 2(( )), if you want to go little slowly.

(Refer Slide Time: 31:07)

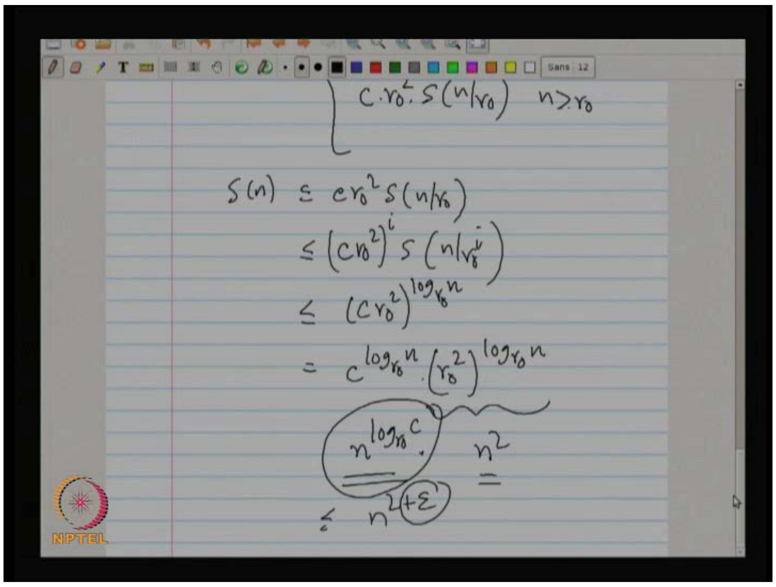


$$S(n) \le c r_0^2 S(n/r_0)$$
$$\le c r_0^2 \cdot c r_0^2 S(n/r_0^2)$$
$$= (c r_0^2)^2 \cdot S(n/r_0^2)$$
$$\le (c r_0^2)^3 \cdot S(n/r_0^3)$$
$$\vdots$$
$$\frac{n}{r_0^i} = r_0 \quad \boxed{i \sim \log_{r_0} n}$$

So, S n I wrote as C time's r 0 square s. So, next time when you do it because you lose another r 0 factor, which is C r 0. So, when you do it third time you will get. So, that is what you get the recourse. Now, when a references stop then n does becomes r 0. So, this is. So, when n over r 0 i becomes r 0 or if you want to sort of say it may be 1 becomes like that. So, so let say i becomes roughly log to base r 0 n.

So, now, you if you look like it here and then when it becomes you wish here then you get S of r 0 its basically 1.

(Refer Slide Time: 32:34)



$$\begin{cases} c \cdot r_0^2 \cdot S(n/r_0) & n > r_0 \end{cases}$$

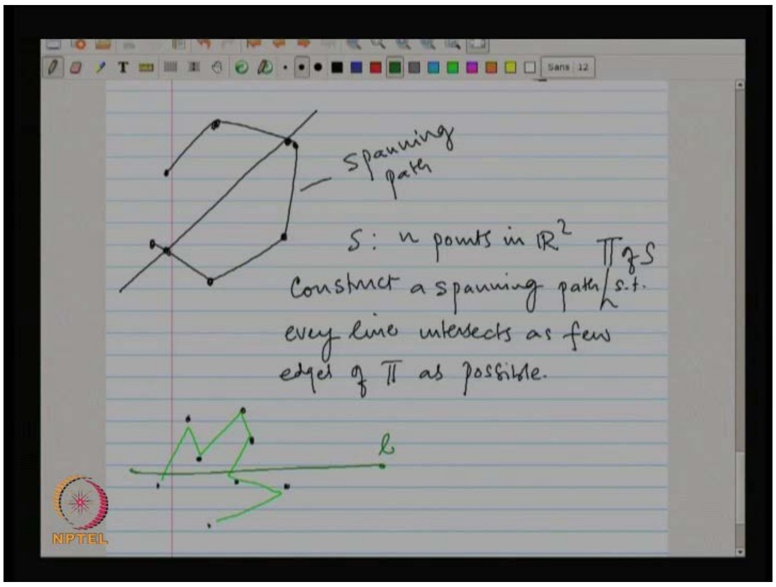
$$S(n) \le c r_0^2 S(n/r_0)$$
$$\le (c r_0^2)^i S(n/r_0^i)$$
$$\le (c r_0^2)^{\log_{r_0} n}$$
$$= c^{\log_{r_0} n} \cdot (r_0^2)^{\log_{r_0} n}$$
$$n^{\log_{r_0} c} \cdot \quad n^2$$
$$\le n^{2+\varepsilon}$$

So, so this size is C r 0 square log of r 0 n. Now, this looks little nasty. So, let us say handle it carefully. So, this is C log r 0 n times r 0 square. Let us talk about this 1, what is this term and this n square. But you have this. This also you can write in a different form. So, this you can write if I will not you do manipulations a log.

So, this is saying its n log r 0 C. So, the size of data could have been n square except this nasty little (( )). Now r 0 is a constant that I choose and C is the constant that is coming from the size of the cutting. So, I can choose r 0 as large as I want and by making r 0 larger I can reduce this term S as small as I want to edit sort of constant, but what I pay is I did some cheating here in the log n is actually the log n here is r 0 times square log n because there is a tree here. At each (( )) of the tree contains the query point. So, you have to spend r 0 square time here. So, he spends r 0.

So, what happens is that you can make that r if you make r 0 large what is happening on increasing of fans, fan out tree is becoming shrinking, depth is shrinking, but fan out is increasing spend more time at each node. So, you that sort of says you cannot make it arbitrary small too. So, normally what you write it is basically I give you some parameter epsilon say and then you can say the running time is at most n to power epsilon and then given a epsilon you can choose r 0 to be sufficiently large constant, that will refer epsilon. Any questions about this?

(Refer Slide Time: 34:45)



S : n points in $\mathbb{R}^2$

Construct a spanning path Π of S s.t. every line intersects as few edges of Π as possible.

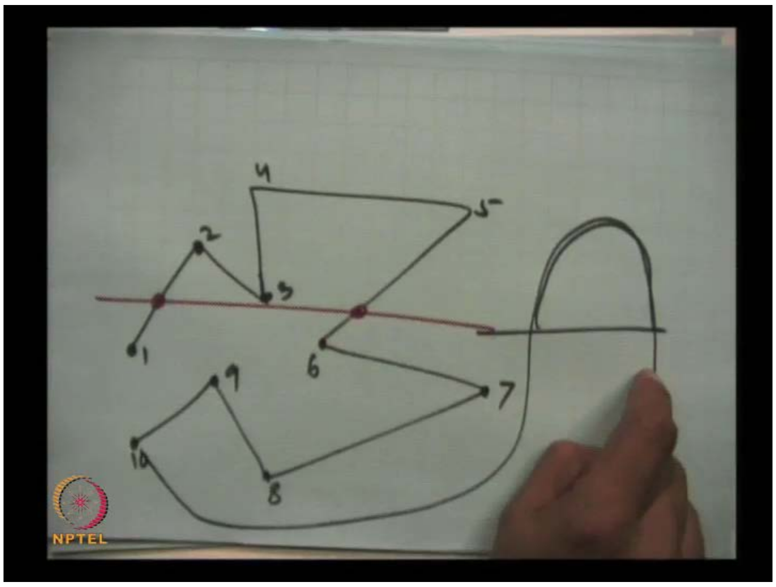
Now, let me move to linear size data structure. Now, remember that the points and contact position when you talk about the range reporting what I said was the if the points in conjoint position then I knew that the query line into a sects lets polygon at only 2 points. So, you can think about that if you raise 1 edge of the polygon and here is the set of points you can think about the remaining polygon chain. This is what is called a spanning path like spanning tree, because a spanning tree is a tree that whose vertices are the precisely vertices the input the point. So, graph and this is a path. So, it is a spanning path because it contains its each vertex is one of the input point, it recovers all the input points.

So, if the points are in conjoint position. We have a (( )) property that if you take any line, it intersects the point the path at only 2 points. Now, if the points are not in conjoint position then what happens. So, suppose then you cannot be so lucky. That is because if you look at the spanning path it is going to intersect the line more than 2 points.

So, question you ask is the following. If you are given a set of points, construct this spanning path such that every line. Let us call it Pi of S as few edges of Pi as possible. What I want to do is give you a set of points to be let me. Now, I want to construct a path with the percept every line should intersect this path at only you should intersect few edges of the path. Now, before you say what we can do let me tell you why this why we want this. Now, suppose if I have a line, this is line I have, look at the intersection part. Suppose, it intersects here only this 2 end points then what you know is in between these 2 intersection points which was of same property here that either all of the points lie above the line or all of them lie below the line.

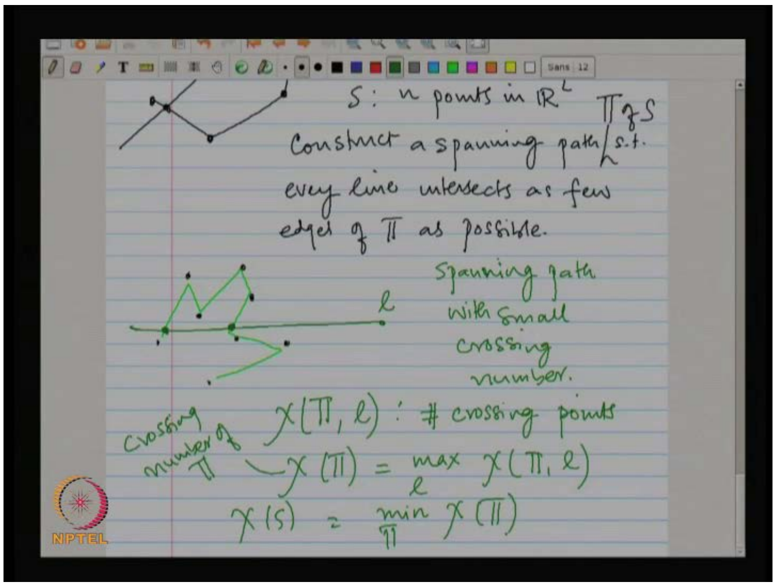
So, if I can if I tell you that this line intersects this spanning path at only few places and I can also find those intersection points quickly, then I can answer the query because then what I need to know is that how many points lie below it and since this path is in linear order if I know the index of the vertices of the intersection points I can subtract and I can tell you that how many points lie there.

(Refer Slide Time: 38:51)



So, suppose you have a path and here is a line. Well this is too good to be true, but let us it is a. Then suppose I can find each intersection point quickly. Let us not worry about finding the intersection points, then what I do is let us say this is the I store the index 1, 2, 3, 5, 6 when I if I tell you that this is edge and you know this index then you know that how many points lie here then you can find similarly well rest of the lie below, but suppose if the path went something like this up again you can do this and the time you will spend, the number of things that you have to add will be proportional to the number of words intersection points. So, if I can tell you that there are very few intersection points then that is good and you can find them. Now, for the convex case you are very lucky because there are only 2 intersection points, but you would not be so lucky for non convex case. So, this motivates this problem what it is called finding the spanning path with the small crossing number.

(Refer Slide Time: 40:12)



S : n points in $\mathbb{R}^2$ $\quad \Pi$ of S

Construct a spanning path Π s.t. every line intersects as few edges of Π as possible.

Spanning path with small crossing number.

$\chi(\Pi, l)$: # crossing points

crossing number of Π $\quad \chi(\Pi) = \max_{l} \chi(\Pi, l)$

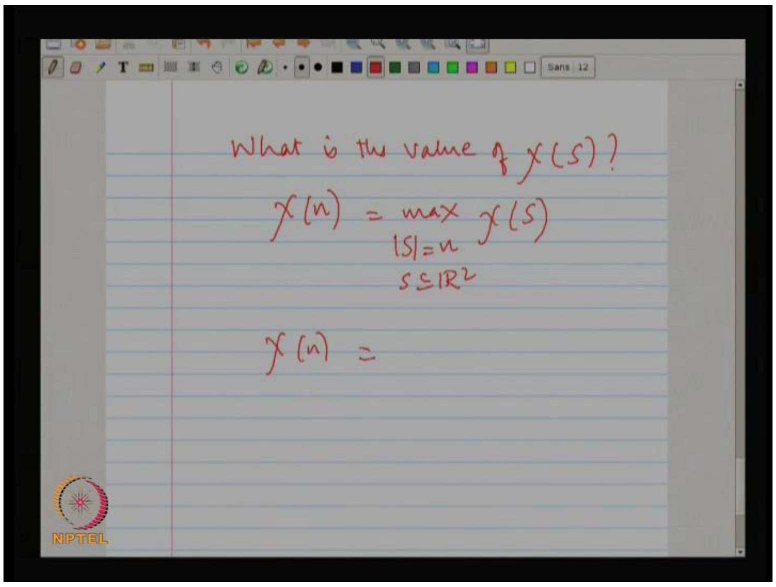
$\chi(S) = \min_{\Pi} \chi(\Pi)$

So, this is called spanning path. Question, go ahead. Well I have not told you yet that sort of tell you the answer and I have not told you, but the answer you should expect that may some deception you are just 1 step ahead of me if you. So, the lets ask the following let us just answer his question, what he question he raised, that is what I was going to say do it. So, let us introduce some notation. So, let us.

So, for a path pie and a line L lets say the number of crossing points, number of crossing points is the number of this edges it crosses. This is a for a fix line for a fix path you say look at this then I define the quantity that is a crossing number for pi is a maximum over all lines and that a for a given path you look at the what is a maximum for what is a what is this can have.

Now, what you want to do is for a given set of points you want to construct a path that minimizes quantity, so the maximum number as you do it. So, question you ask is how large the sky of x of x will be.

(Refer Slide Time: 42:24)



What is the value of $\chi(S)$?

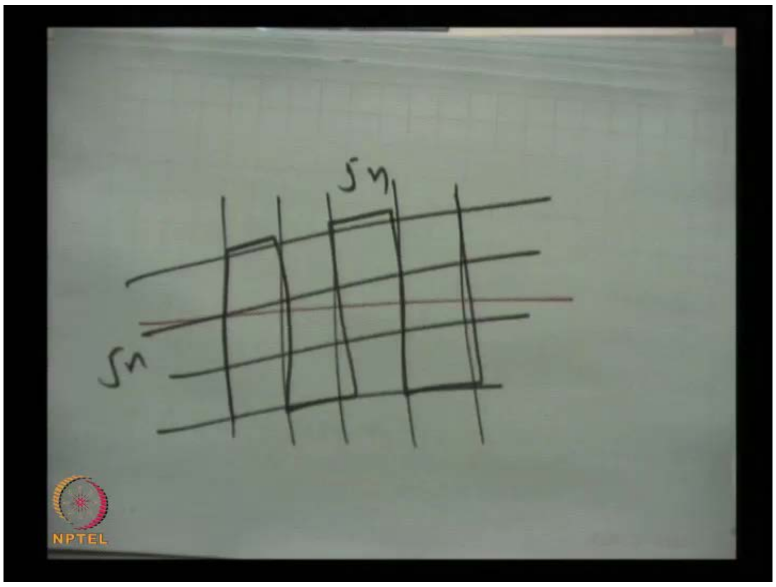
$$\chi(n) = \max_{\substack{|S|=n \\ S \subseteq \mathbb{R}^2}} \chi(S)$$

$\chi(n) =$

So, question you want to ask is what the value of x s. So, does anyone to make a guess. So, I was going to come to that line basically. So, for what is of worst case bound. So, if you like Sandeep, what Sandeep said maybe I should introduce another thing is a if you will take a set of n points what is a worst you will get there.

So, the question is what is the value of k i of n what does your intuition say. So, what about the points on a grade. Suppose, you take a grade points you take a root n by root n grade. What do you think the answer will be for that? Pardon sorry say it again.

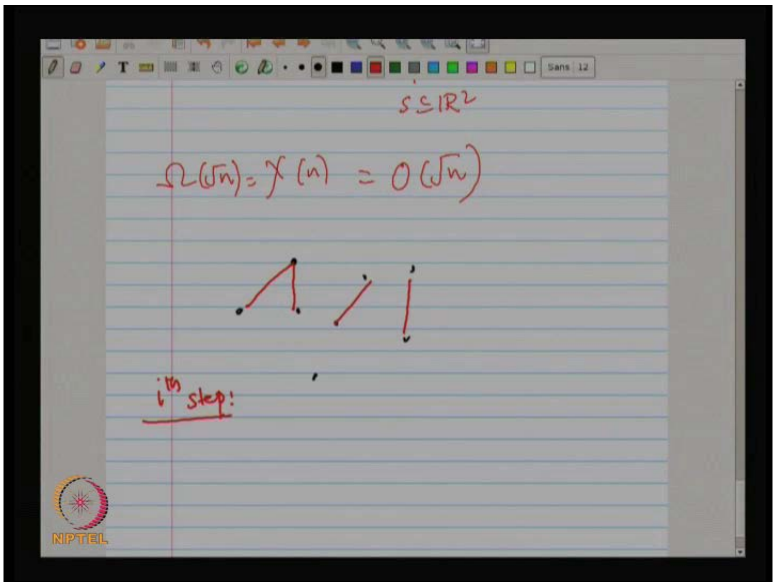
(Refer Slide Time: 43:32)



Why do not you order n because you are taking root n by root n grade. 2 times am I right it because if you take if you will let us say just take, let us say this is this is just a spanning path. Now, if you take any line it will intersect each column only once each vertical line only once, am I right. So, it will intersect root and edges. So, for grade you can do root n and you can also convince which is this will (( )) some more work that you cannot do better than root n for the grade, which does not matter this is 1 you have constructed, does not matter how you construct a spanning path for this grade points you will stuck with root n. So, root n is a lower bound for k i n.

So, this is you know that it is a root n is a lower bound. The question is I know that for grade points I can do square root of n, but when I say square root of n I mean order of square root of n and, but can I do it for any points site and in terms of this is it leads the case, this is it cannot do it and what I will do is I will give you very quick.

(Refer Slide Time: 44:53)

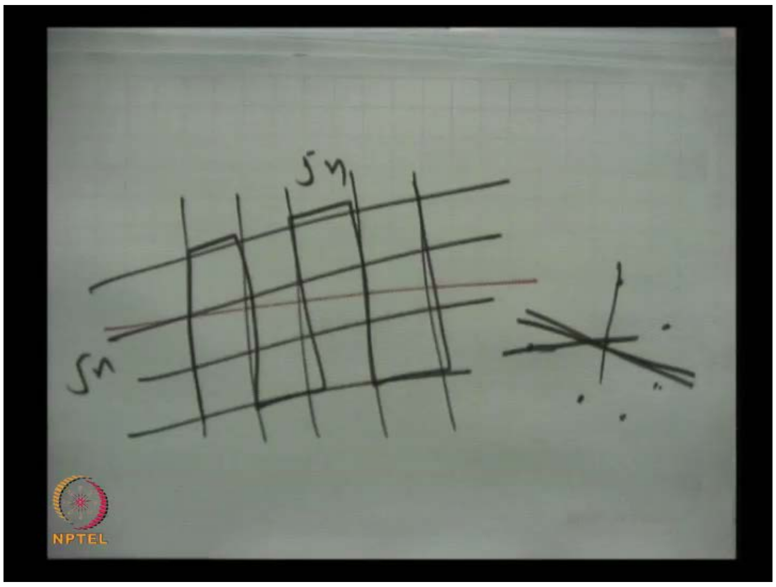

$S \subseteq \mathbb{R}^2$
$\Omega(\sqrt{n}) = \chi(n) = O(\sqrt{n})$
i^{th} step:

So, can I afford to go a little overtime? So, it might not be recording, but you can hear me the tape might not be this. So, let me first describe the construction and then I will do analysis. The reason I want to do this analysis is very cute and it has it is the similar idea used I many other different places too.

So, now and the proof is constructive. So, I will give you an algorithm how I construct the path. So, remember that. So, what we have this here, we have a set of n points. You have a set of points and I want to construct a path. So, that every line intersects at most again I will forget the constant, exclude of n edges what I will do is I will it is like a algorithm you have seen for computing minimum spanning tree. So, I will add 1 edge at a time.

So, the algorithm you will work in n minus 1 phases, in each phase I will add 1 edge to it. So, what will happen is that I will be maintaining a collection of paths at any given time. So, in the after in the r i th phase after I finish i phases I would add it i edges and you will have a collection of those path basically. So, you will have something like this, something like this you will have. So, let me describe you what I do it, how I choose a new edge at each phase and that is a construction.

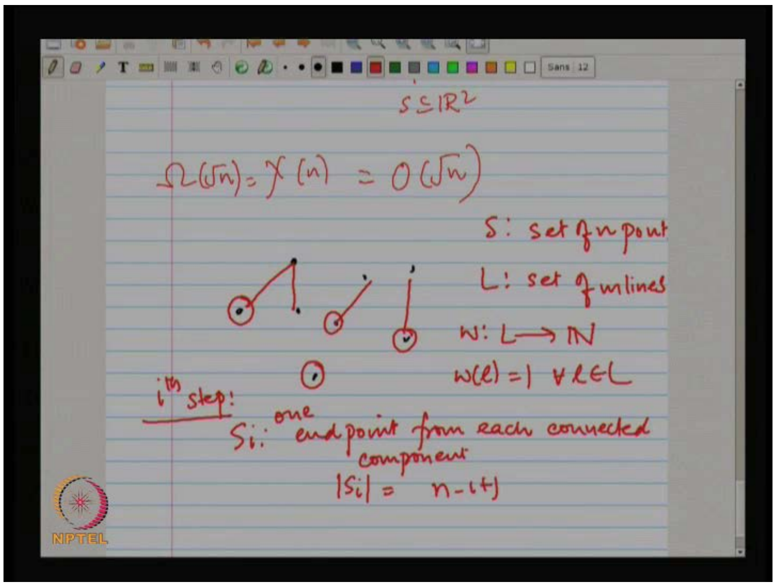
(Refer Slide Time: 46:38)



So, now, let me do the following. Now, other thing you observe is the following. Although the number of lines are infinite, but if you have set of points you only have to worry about the lines that pass through pair of points because if suppose if you have 2 lines. So, that the same set of points lie above it and same subset of points lie below it then there will these 2 lines will intersect the same set of edges for spanning path. So, I do not need to worry about I can think of these 2 lines as the same. So, what I can do is I can just choose lines which through passes the pair of points and that can be sort of set of representative lines with respect to which I need to construct the path.

So, what you have is your set of points and you have a set of representative lines and in the worst case you choose n square lines, but actually what it turns out that you have to choose a very small set of line, but I will not go into that.

(Refer Slide Time: 47:33)



$S \subseteq \mathbb{R}^2$

$\Omega(\sqrt{n}) = \chi(n) = O(\sqrt{n})$

S: set of n point

L: set of m lines

$w: L \longrightarrow \mathbb{N}$

$w(\ell) = 1 \ \forall \ell \in L$

i^{th} step:

S_i: one end point from each connected component

$|S_i| = n - i + 1$

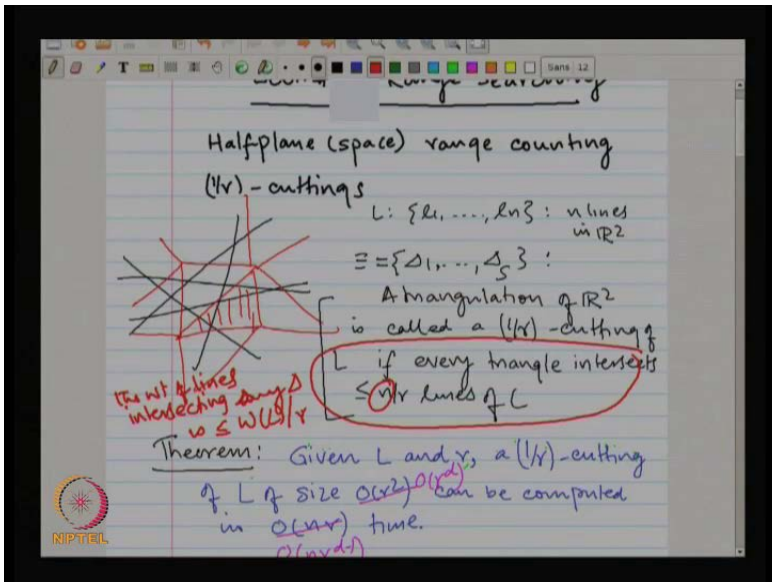
So, you have a set S set of n points and you have a set L the set of m lines. Now, what I will do is lines will become weighted. So, there will be a weight function. Initially the weight of each line will be 1, but weights will increase because what will happen is when I find then I find them an edge and if I find the line intersects this edge the line will increase its weight because the hey this line has already intersected some edges. So, be careful with the future they do not add the edges that this lines intersect the analyze it. So, that is intuition. So, in the (( )) step I do the following.

So, you have some paths. You. So, set S i is the following. For each of them you pick 1 of the n points. In the beginning you have S i will the same as S (( )). So, S i is the n point 1 n point, each connected component. What will be the size S i will be?

This after i stuff this is because you have added i minus 1 edges. Because in the after i steps I have done the (( )) because you have finish up to the i th step. So, I finish the i minus 1 step. So, I added i minus 1 edges. So, the number of points you will be choosing will be n minus i plus 1 because in the first 1 you will have all the stuff and then you added 1 edge and 1 point disappears and so on. And let us call this as n i. Just to make the my notation little simpler.

Now, I should say 1 more thing. When I introduce a cutting, this cutting was introduced for the unweighted set of lines. Now, you have weight weighted set of lines.

(Refer Slide Time: 50:17)



Halfplane (space) range counting

(1/r) - cuttings

L: $\{\ell_1, \ldots, \ell_n\}$: n lines in $\mathbb{R}^2$

$\Xi = \{\Delta_1, \ldots, \Delta_s\}$:

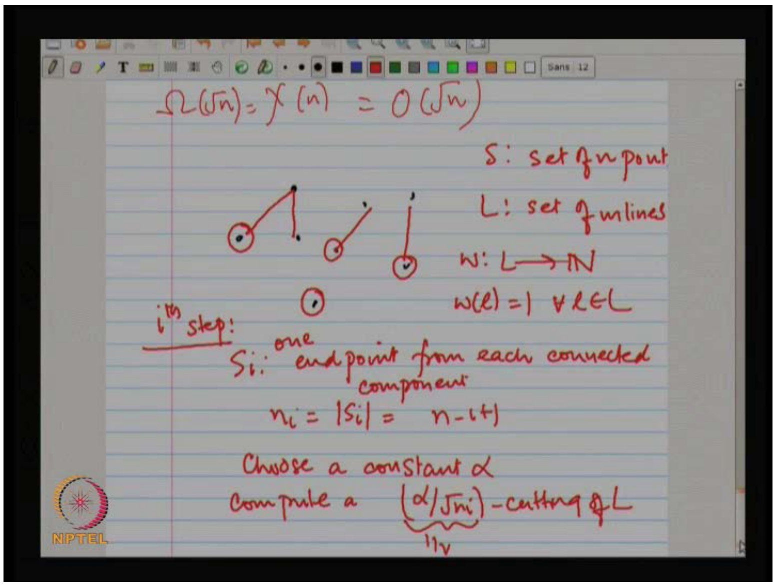
A triangulation of $\mathbb{R}^2$ is called a (1/r)-cutting of L if every triangle intersects $\leq$ n/r lines of L

The wt of lines intersecting any Δ is $\leq W(L)/r$

Theorem: Given L and r, a (1/r)-cutting of L of size ~~$O(r^2)$~~ $O(r^d)$ can be computed in ~~$O(nr)$~~ $O(nr^{d-1})$ time.

So, here you think about this is special case when the weight of each line is 1 and that set total weight was n. Now, if you have weighted lines then you say that you do not care about n what you want to say is that that now if every triangle. So, this statement will change to following. The weight of lines intersecting any triangle is at most total weight over r. So, that was the generalization because n was the total weight of lines when the weight of each line is 1. If you look at the total weighted lines look the total weight you want to say that each none of the triangle is too heavy, the weight of lines intersecting triangles utmost total weight over r and this theorem it still holds.

(Refer Slide Time: 51:18)



$\Omega(\sqrt{n}) = \chi(n) = O(\sqrt{n})$

S: set of n point

L: set of n lines

$w: L \rightarrow \mathbb{N}$

$w(\ell) = 1 \ \forall \ell \in L$

i^{th} step:

S_i: one end point from each connected component

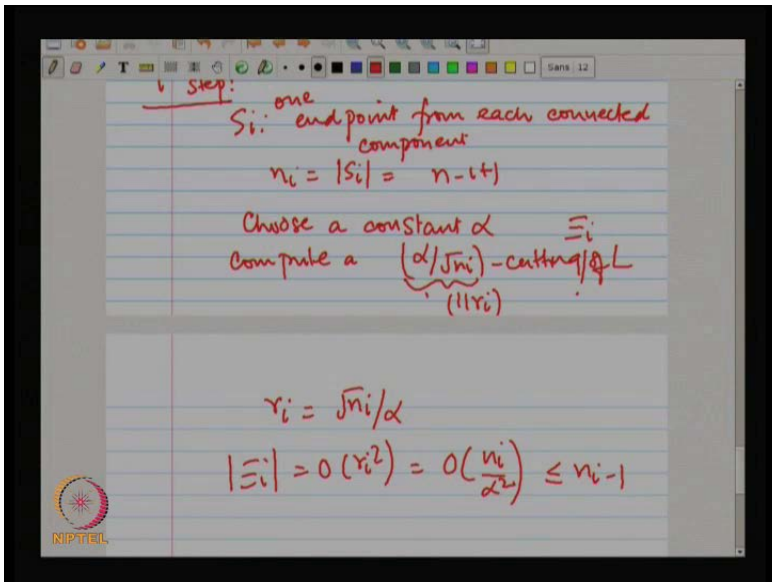
$n_i = |S_i| = n-i+1$

Choose a constant α

Compute a $\underbrace{(\alpha/\sqrt{n_i})}_{1/r}$-cutting of L

(( )) what I do is, choose a constant alpha which I will tell you in a minute and compute a. So, r is this is the value, this is 1 over r. r basically is.

(Refer Slide Time: 52:05)



i step:

S_i: one end point from each connected component

$n_i = |S_i| = n-i+1$

Choose a constant α Ξ_i

Compute a $\underbrace{(\alpha/\sqrt{n_i})}_{(1/r_i)}$-cutting of L

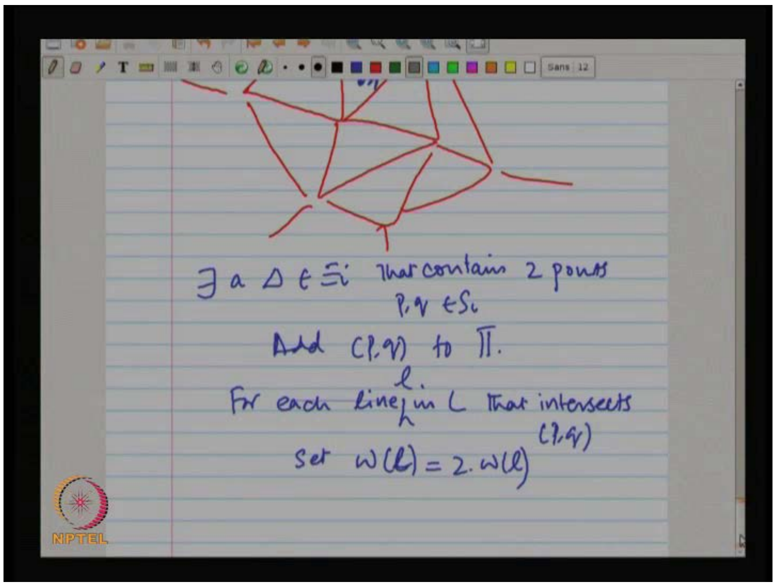
$r_i = \sqrt{n_i}/\alpha$

$|\Xi_i| = O(r_i^2) = O\left(\frac{n_i}{\alpha^2}\right) \leq n_i - 1$

So, if you want to think about it then r I have chosen is this is the value of. So, you think about this 1 over r i cutting I have computing. Now, the size of the cutting. So, computed cutting lets called it k i. So, size of k i is order of r i square is order of n i sorry which is n i square over alpha. If I choose alpha to be sufficiently large this is I can make this point it to be n i minus 1 because or not by choosing parameter because you have constants if I choose alpha to be a bigger than the constant C in the cutting then this will become less than 1.

Does this make sense or I lost you? So, size of the 1 over i cutting is (( )) of r i square, r i was root n i over alpha. So, you get sorry n i over alpha square. So, you get by choosing if you choose alpha 2 sufficiently large this point it will be enormous. Now, what I know is number of points is n I, number of triangles is at most n i minus 1 of the cutting.

(Refer Slide Time: 52:05)



$\exists$ a $\Delta \in \Xi_i$ that contain 2 points $p, q \in S_i$

Add (p,q) to Π.

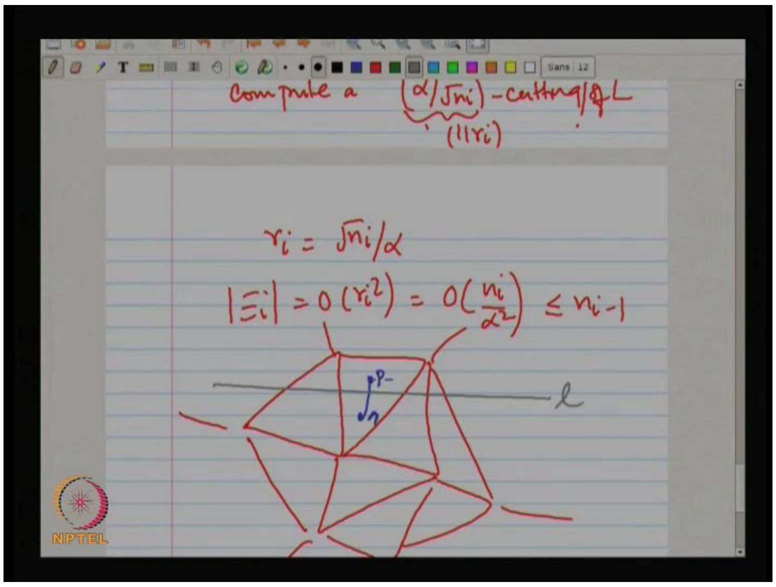
For each line ℓ in L that intersects (p,q)

set $w(\ell) = 2 \cdot w(\ell)$

So, here is a cutting. Suppose and then suppose you have an unbounded triangles. Now, number of triangles is utmost n i minus at most n i 1 and number of points is n i. It means that there is a 1 triangle that contains at least 2 points. So, it means you have a situation that you have a triangle that contains 2 points. By generating the whole principle because n i points are lying inside n i minus 1 triangles then there is to be at least 1 triangle that contains at least 2 points.

So, you can add this edge to your. So, it means there exists a triangle that contains 2 points. Let us call them p q sorry add p q to k i. So, that is a construction. Not yet, one more step and then for each line in L that intersects p q set double its weight.

(Refer Slide Time: 52:05)



Compute a $\underbrace{(\alpha/\sqrt{n_i})}_{(1/r_i)}$-cutting of L

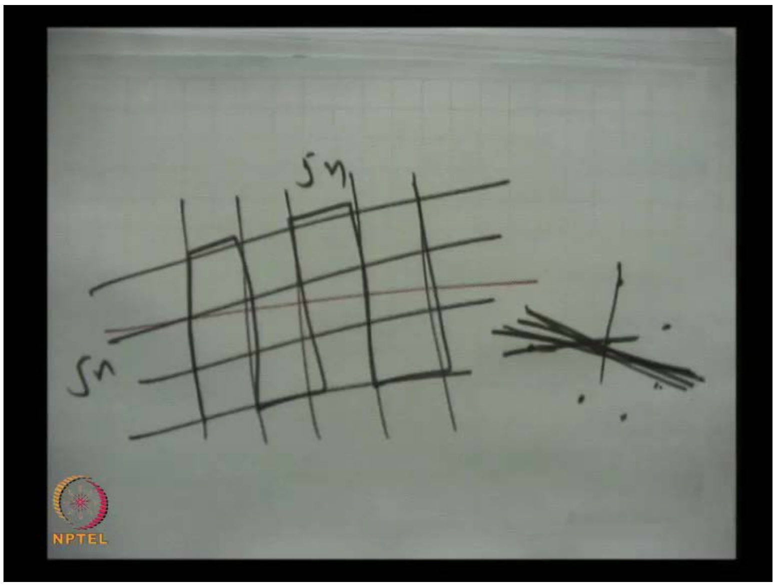
$r_i = \sqrt{n_i}/\alpha$

$|\Xi_i| = O(r_i^2) = O\left(\frac{n_i}{\alpha^2}\right) \le n_i - 1$

So, this is a point p q and suppose this was a line L that intersects this edge that you just add it, you double its weight because you want to paralyze it because you know now you know this line intersects this edge. So, you want to be careful because you do not want to choose another edge that intersects this line or you want to choose, but you want to decrease the probability of choosing that edge, that is a intuition. Any questions about algorithm.

No, there is no query. This is construction the path pardon. So, these are the set of lines you had. So, if the construction I had you start with set of points and set of lines. I said the lines that all the take the what I argue that it is enough to look at the lines that pass through pair of points. Because this is that I have said.

(Refer Slide Time: 56:19)

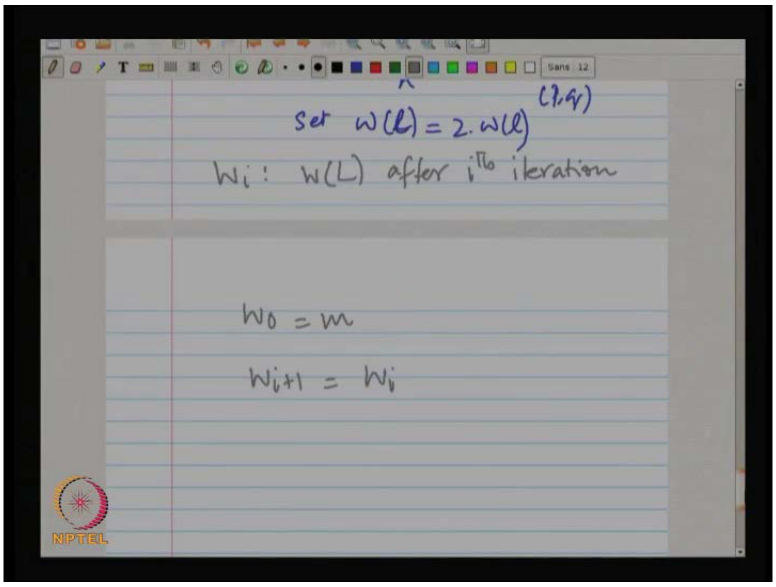


That I said that if you have 2 lines, so that is all same sub set of points lie above it and below it then it will intersect same set of edges of the spanning, n is spanning path. So, what you can do is for each of the such thing choose 1 line and there will be n choose to such lines.

So, those are lines and initially I said that the weight of each line to be 1 and then I cap and then double it. So, this is a construction. Now, I need to argue that this construction guarantees that no line intersects, no line intersects more than square root of n edges. Yes, I have not ensured that I will, but I have not meanwhile, but I will, but I will, but I am not saying, but I will show you that this quantity is true, this part I want to prove. Now, I have not done that yet. So, this is a algorithm.

Now, I have to do analysis that path I constructed, the crossing number is square root of n and I need 5 more minutes for this. Now, it is a double counting argument.

(Refer Slide Time: 57:39)



set $w(\ell) = 2 \cdot w(\ell)$ $(1,4)$

W_i : $W(L)$ after i^{th} iteration

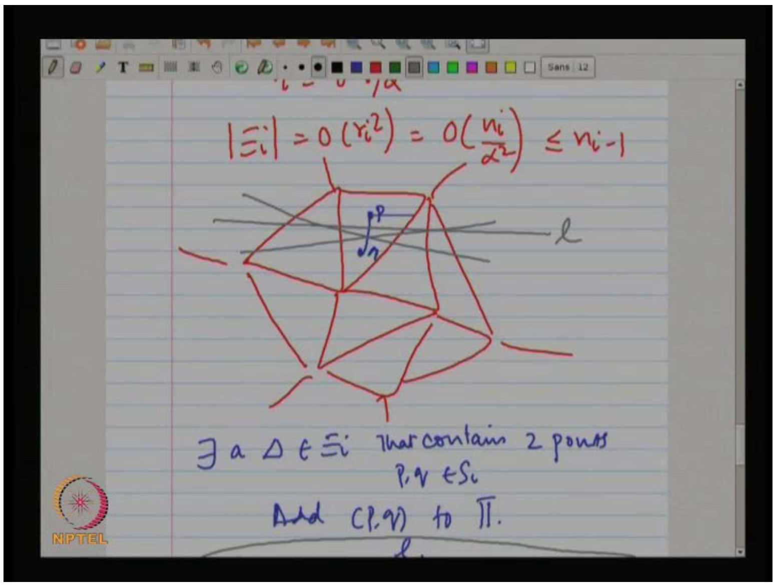
$W_0 = m$

$W_{i+1} = W_i$

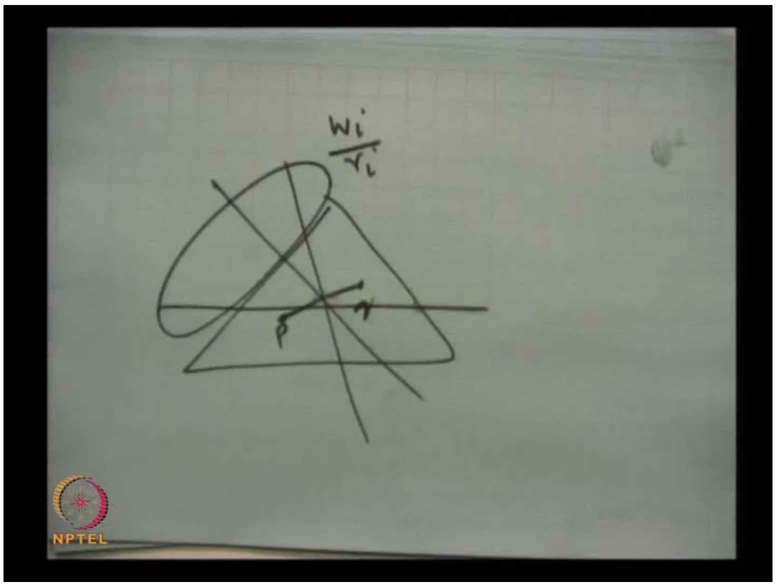
So, let us look at the say w i is a total weight of lines after i th equation. So, what is a w 0. So, w 0 was initially the lines you know it is m because the m lines, m is n choose 2, but let us not worry about it. So, it is n. So, weight of each line is 1. So, total weight of lines is m. So, that is one thing.

Now, everything you notice is another important relation is what is if I know w I, you need to write w i in terms of w i plus in terms of w i. So, now, one thing you notice is when does a weight increase? The weight only increases here. So, this only place where (( )) increase the weight. Now, and here is the fact I will use this is the cutting because what notice is edge lies inside this triangle.

(Refer Slide Time: 58:59)

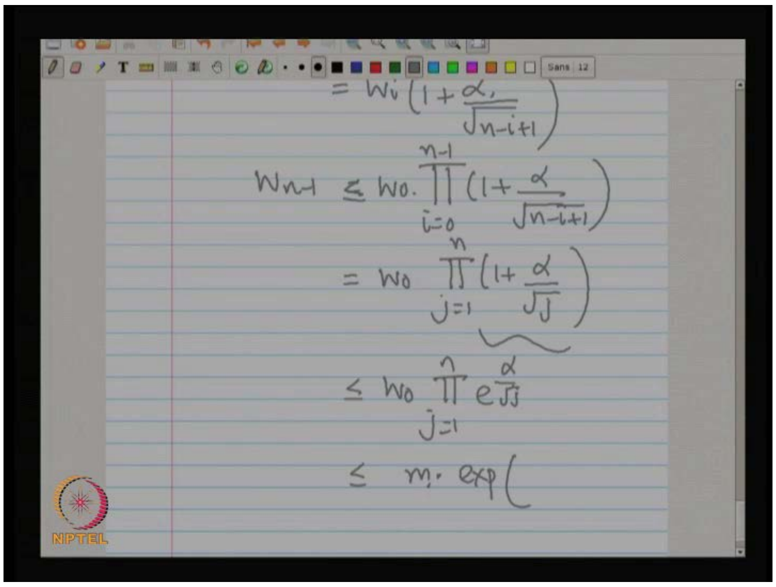

$|\Xi_i| = O(r_i^2) = O\left(\frac{n_i}{\alpha^2}\right) \le n_i - 1$
p
q
l
$\exists$ a $\Delta \in \Xi_i$ that contain 2 points p, q $\in S_i$
Add (p, q) to Π.

So, if you look at the weight of this lines since I doubled it, weight of these lines, the lines that intersect this edge p q also intersects that triangle. (Refer Slide Time: 59:14)



So, here is a triangle p q sorry here is a edge p q. So, any line that intersects this edge also intersects this triangle. Now what I know is that since this is the cutting the total weight of these lines is what? It is w i over r i.

(Refer Slide Time: 59:38)



$$= W_i \left(1 + \frac{\alpha_i}{\sqrt{n-i+1}}\right)$$

$$W_{n-1} \leq W_0 \cdot \prod_{i=0}^{n-1} \left(1 + \frac{\alpha}{\sqrt{n-i+1}}\right)$$

$$= W_0 \prod_{j=1}^{n} \left(1 + \underbrace{\frac{\alpha}{\sqrt{j}}}\right)$$

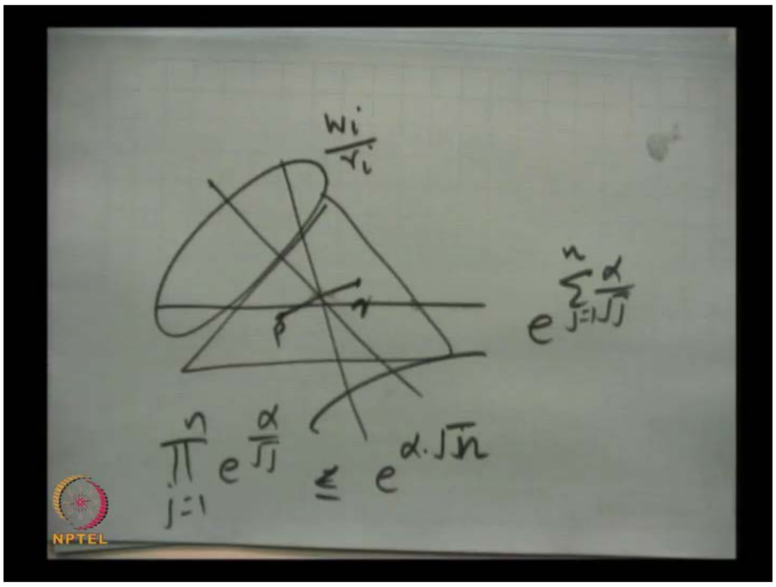
$$\leq W_0 \prod_{j=1}^{n} e^{\frac{\alpha}{\sqrt{j}}}$$

$$\leq m \cdot \exp($$

So, what I know is that this is that most r i and so this is and r i was this quantity. So, this is alpha n, n i was let me write it was n i was this quantity n minus i plus 1. So, what you get is. So, now, if you expand it what you get is w n after you finish the whole thing or w n minus 1 because the n minus 1 phases will be w 0 and 0 2 n minus. I might be messing the, 1 plus alpha plus 1 and this you can rewrite as just reverse in the reverse order because it is easier to write. So, I just writing the series in reverse order that is all I did.

Now, this quantity you can write as, alright 1 plus x (( )). So, then product becomes sum here and so this becomes w, w 0 is m, I should m times.

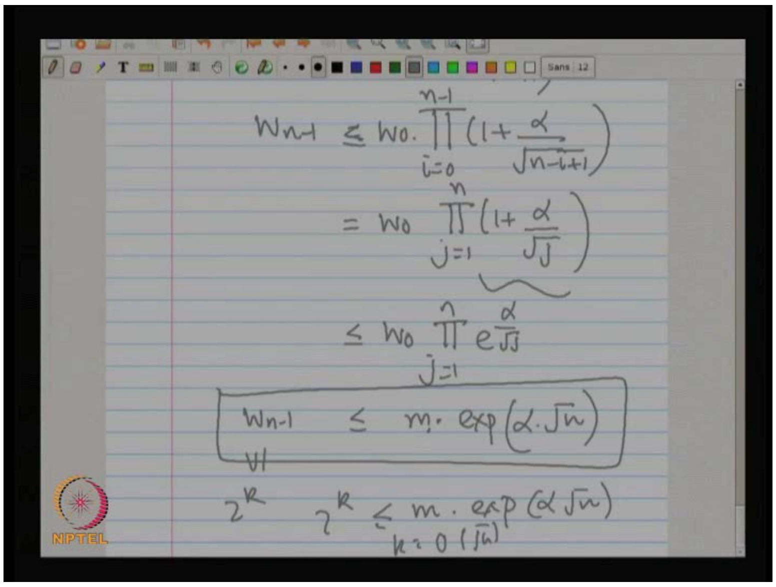
(Refer Slide Time: 1:01:55)



$\frac{w_i}{r_i}$
p
r
$e^{\sum_{j=1}^{n} \frac{\alpha}{\sqrt{j}}}$
$\prod_{j=1}^{n} e^{\frac{\alpha}{\sqrt{j}}} \leq e^{\alpha \cdot \sqrt{n}}$

So, when. So, so pi j is equal to 1 to n into power alpha root j. This becomes e 2 power this product becomes sum in to exponent some think it as an integral and this basically becomes roughly alpha times square root of n. So, because this becomes e to power sigma alpha over root j, j is equal to 1 to n and if you integrate 1 over square root of j i up to n you get square root of n. So, that is what happens.

(Refer Slide Time: 1:02:34)



$$W_{n-1} \leq W_0 \cdot \prod_{i=0}^{n-1} \left(1 + \frac{\alpha}{\sqrt{n-i+1}}\right)$$

$$= W_0 \prod_{j=1}^{n} \underbrace{\left(1 + \frac{\alpha}{\sqrt{j}}\right)}$$

$$\leq W_0 \prod_{j=1}^{n} e^{\frac{\alpha}{\sqrt{j}}}$$

$$\boxed{W_{n-1} \leq m \cdot \exp(\alpha \cdot \sqrt{n})}$$

$$\overset{\mathrel{\rotatebox{90}{$\leq$}}}{2^k} \qquad 2^k \leq m \cdot \exp(\alpha \sqrt{n})$$

$$k = O(\sqrt{n})$$

So, this is becomes a alpha times. So, this is 1 thing you get. So, this the final product. Now, on the other hand suppose if a line intersects k edges, if a line intersects k edges what will be its weight? 2 to the power k.. So, if a line intersects k edges it weight will become every times it intersects an edge you double its weight. So, if you double it k times it will 2 to the power k. So, it means if you have a crossing line at least k times the total weight of lines will be at least 2 to the power k.

So, this is on the other hand 2 to power k. So, what you know is the 2 to power k is m times exponent n. Now, if you take the log on both sides you are going to get I will k to be square root of n.