Computational Geometry

Prof. Pankaj Aggarwal

Department of Computer Science and Engineering

Indian Institute of Technology, Delhi

Module No. # 11

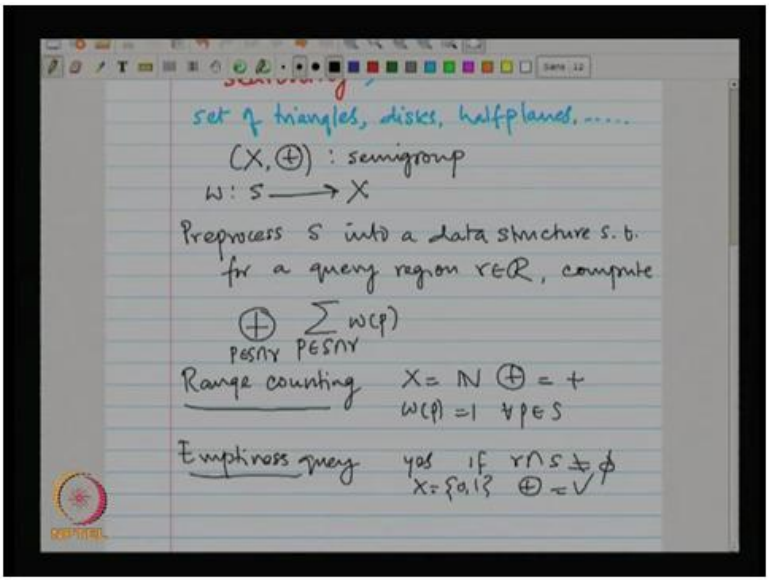
Range Searching

Lecture No. # 04

Range Searching

So today, I will talk about non orthogonal range searching, and let me just sort of to remind you that will describe the range searching problem in abstract sorting, and then we will instantiate for some specific problems

(Refer Slide Time: 00:47)



So just to remind you that in abstract sorting, you are given a set of points; and you are given a set of ranges, so here you have a set of points S, and ranges or regions, earlier in the in the class, when you learnt about orthogonal range searching, the regions were rectangles, so that is for orthogonal range searching. So, if you have R is a set of rectangles, this is orthogonal range - set of orthogonal rectangles, for x is a line rectangles range. So, that was what you saw? You saw the k d trees, and you saw range

trees and I think you also saw some other data structures, and you saw how you answer the query sufficiently by preprocessing the point's sets into a data structure.

Now, you will go wilder, and will not restrict to ourselves to orthogonal rectangles, but we will consider other regions. So, for example, we might consider triangles that is a one possibility, that regions are triangles or we may consider disks that the set of points inside this disc or we may think about, what we call half planes. And you can think about other regions, you can there are something called semi algebraic regions, you can think also ellipsis or parabolas or any other shapes, you can think about defined by some polynomials or some other functions when you talk about. So, in general basically let me long sort of say here, so what you have this, so R could be a set of triangles, disks, half planes etcetera.

Mostly, I will talk about two D, but tomorrow to the end of the class I will talk about some, I will make some remarks about higher dimensions. So, within in in formally speaking, what you want to do is, we want to have a some semi group. So, you have let us say (no audio from 03:20 to 03:27) so you think about you are given a semi group. So where semi group you have some sat some set, and you have a some plus operation that difference between group and semi group is unlike groups. In semi group you do not have negation negation, you do not have negatives inverse does not accessed. The group is then, you have plus and minus both and semi group we have only plus operation there is no minus operation.

So, for example traditional addition is semi group, but thinks like maths, I give you set of numbers operator plus operation could be maximum, could be minimum, if you think your sets, union, intersection and so on. You can have different operations and this in general is called semi group. So, you are given a weight from sigma s sigma, and so the question the problem will be pre process s into a data structure, such that for a query region for a query range compute (No audio from 04:58 to 05:08) my notation become bad, so let me change my notation little bit, let me call it X w(p) (( )) and to just give you some… So, you have you have in the when it learnt about orthogonal range searching assume you saw both range reporting and range counting at least that basically you want to say, let us say think about range reporting.

Let us say range counting first (No audio from 05:43 to 05:50) will that is I do not say that is a binary operator that plus that is why I wanted to differentiate between plus and sum range. So, this is a binary operator the plus that o plus that I wrote plus inside the plus o sign and this is over a multiple apply multiple times, so that is all. If you do not like this one you can ask Sandeep was complaining one can use, so range counting, if you want to think about, then that is a kind of simplest case, then x is nothing but natural numbers, because you want to count, and plus is nothing but the standard plus and w of s each w of in that see, let me write it simply, because it is not a w of p is equal to 1 for all p and s, so this very simplest case.

Now, suppose if I, so basically what you said of this complicated, we are saying is that count the number of points inside the query region. Now, suppose let me ask the following question, I do not even care about that counting, just tell me that whether there is any point inside the query region emptiness query. So, what is called emptiness (No audio from 07:20 to 07:28) yes written yes, if it is not empty. So, if I want to formulate this in the semi group way, how will I do it?

(( ))

Say it again.

(( ))

So, x becomes a binary zero or one and what what is the operator.

(( ))

What is the dot?

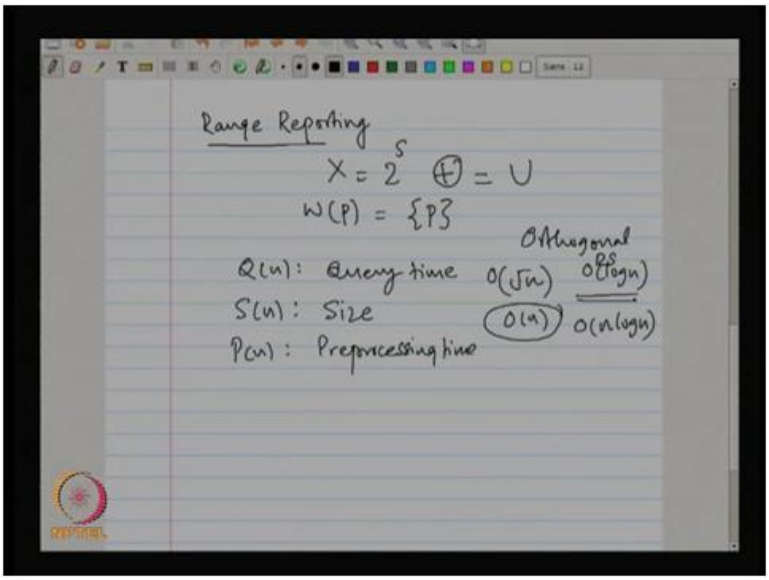
(( ))

Yes you wanted do r operator.

(( ))

Yeah, so ok, so you do r. So, when it is it will be written one if any point inside, so basically, so x becomes binary 0 1 am I right? Now, more interesting is, where the hell how how would (( )) fine yeah.

(Refer Slide Time: 08:28)



So, range reporting, so this is more interesting, so how will I formulate range reporting in this frame (( )) what is the first let us talk about what is X. No not the point itself because we think about, what could be the domain, because you are think talking about the point the weight of a point is a point itself, but what should be the set X no no, so no it should (( )) it is not r square. So, this set X, which is should be define in enumers am I right? So what are the possible elements in your in the output, in some sense you should think about all the subsets am I right?

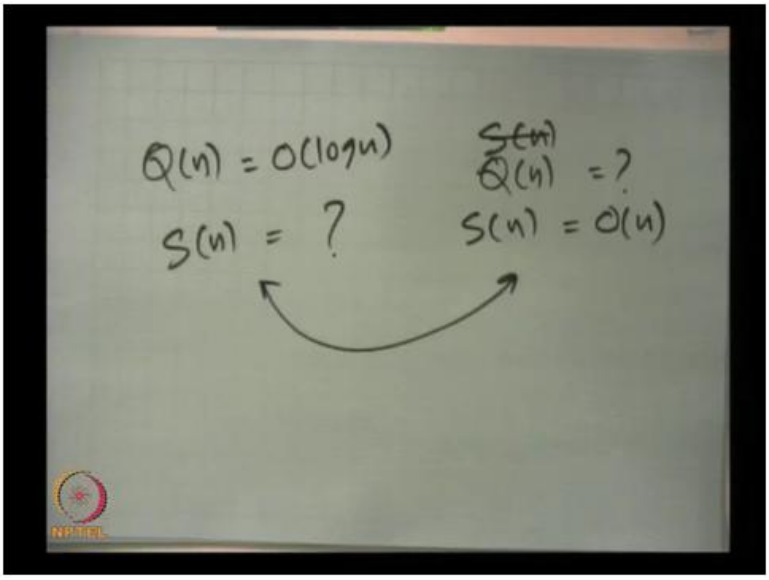
So, this is x will be power set of s. So, it basically then you reporting your reporting a subset of the input points it is all possible sub sets, and plus will be operator will be union and will be the single term, so that is set of s. So, this an abstract frame work that captures all the different version of range range searching problems. (No audio from 09:48 to 09:58) Now, also when you, so this is the basic abstract (( )), it is cycle is what do we care about, we care about three things in the in the order, one is we care about query time; we care about the size of the data structure, and P(n) which is the preprocessing time, as we have seen all this system, but I am just reminding you, so that you know what I am talking about.

Now for the orthogonal range searching, when you saw when you saw the k d trees, then if we use the space to be order of n, the query time was order of root n, and here you allow a log n space, then you can do it log n time range counting. Let us say, so this was

orthogonal range searching, so here it was not too hard that when you allow the linear space, the query times was route n, but if you want to get query time to be roughly log n, you just have to blow up the size of the data structure by log n. Now, I should say that the size of a data structure has to be at least linear, because we have to store each point exact at least once, if you want to answer the queries exactly. And in the binary search or the finding success of an item takes log n time, so the query time will be at least log n. So, this will be these is though, so question will be, can I always achieve log n query time using all most linear space.

Let us I do not care about log factors. So, the case of orthogonal range searching, we were lucky, we could answer the query log n time, all most using linear space and log n space, when we go to non orthogonal range searching, we would not be so lucky. If we use a linear space the query time will be large, of course if you are willing to spend linear time, then it is trivial just scan through all the points. So, goal is you want to answer a query in sub linear time less than linear time, so question is using linear space, how fast you can answer a query that is one extreme. The other x, the other question you ask is, if you want to answer a query time in log n time, how much space I have to use.

(Refer Slide Time: 12:31)



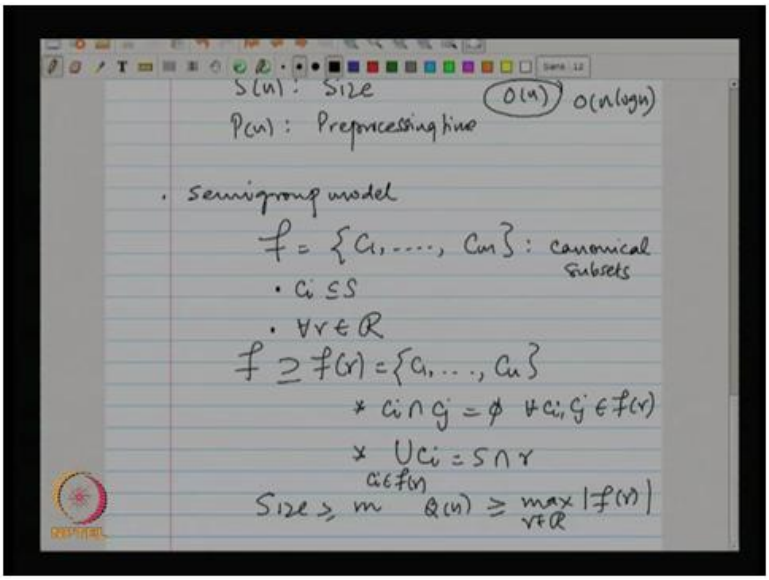
So, the two end of the spectrum says that Q(n) is order of log n, what is the space of first you can do it or if I do let me see, if S(n) is order of n, then how much time you need. So, those are at and you want to have trade off and then you ask ask the question, can I then

combine these two to get a trade off, they say that I have that much resource is available, that you have on let say m space, using m space how much, how fast, I can answer a query. Now, I should say that depending on the application either query time is critical, and you have to really answer query very quickly or sometimes the space is very critical, because if the data is too large, you cannot store spend more than linear size in the data structure.

To just to give you some examples, for example if I look at the network routing, so the question that comes is that, if I when a packet reaches at a node, where where should be set and this can be formulated range searching query. Now, there it is very critical that the query should be answer very quickly in a few nanoseconds right, because you want you do not want network delay, so query time is very critical, do what your space you need to do (( )) and there are some other challenges, I am just over simplifying the problem, but the and there the query so critical that they cannot even delay on software. So, there is a dedicated hardware to answer this range queries there for networking there all this network switching circuits they build to do that.

Now, other extreme is, you have gigantic data bases am I right? And, where you cannot spend more than linear space because just is storing all the data it is a it is so much space like Google of example lot of NASA data, which is in tera bytes and penta bytes of data, so (( )) you can do linear space. So, those are the two hard limits, that we have the extremes and that is of people has studied those question. So with that introduction, now, let me sort of say talk about two general techniques that I used in range searching, and all the most of data structures that you will see they fall in one of those category sometimes may be the both.

(Refer Slide Time: 15:02)



So a typical data structure for range searching and let us called, I will say it is called one technique, this you should think about us, what is called semi group model. So, you give a point right and the data structure if you remove all the technicalities, what it does is the following. It is stores a family of subsets this is these are called canonical subsets these are called canonical subsets such that well so far, I am not said anything meaningful that for any range R, for any size you you represent the points that lie inside the query range as the union of this canonical subsets. So, f(r) which is sub set of f, let us say C u, so you present the points that c i j empty for all c i c j in, so all this things should be disjoined and union of c i precisely (( )).

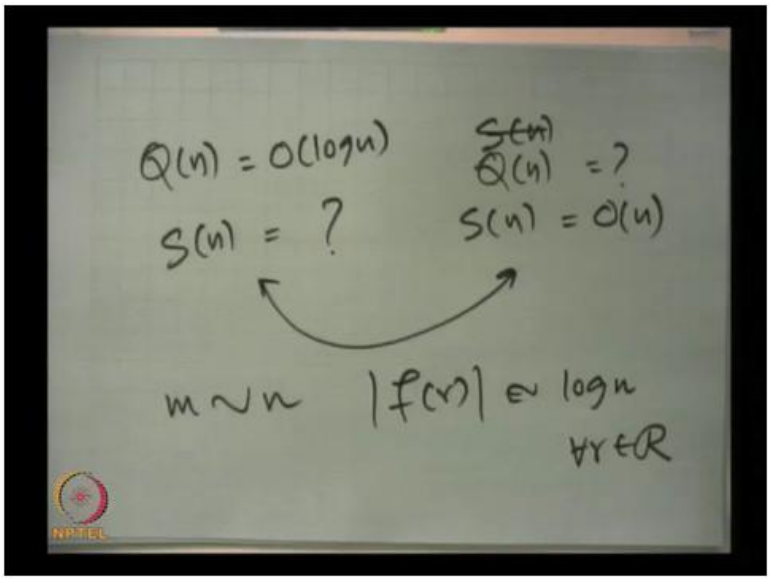
So, what it means is that, you take a set of points, you do some kind of decomposition, it is not a partition, but choose some good sets subsets, so therefore any query range, you can represent the points that lies at the query range as a union of this canonical subset. These are the, this is what I mean by canonical, because these are canonical subset set anything I can represent as a part of this one. So, if you think about the group size this is a kind of the base sides, if you want to think about this way. Now, so far again I am not said anything meaningful in the sense, because you can store all possible subsets am I right? Because you look at all, let us think about the rectangles rectangles are defined by four edges.

So then you can see, you can, if you think a little bit about it, they are only n to power of four different possible outputs. So, I can define n to power four, because I did not say that anything about on m, so you can represent all possible query outputs and say these are my canonical subsets, then for any query range, they will be precisely f(r) would be (( )) over a single one one of that canonical sub sets, so far this is not very meaningful, what we want to do is, you want to do in such a way that both m and u should be as small as possible, that what you want to guarantee is m is, so let me make one more comment.

If I store this canonical subsets, if m is number of canonical subsets, then I need at least m, my size of data structure is at least m. It may be more because how I store it, but if I am storing m canonical subsets, then I need the story, which is at least m. So, the size is is at least m and the query time Q(n) will be max. So, you look at the query, when we look at how many canonical subsets you need, and if I can, what I can do is, if for each canonical subset, I compute it is, if I thinking in this abstract frame work, that I have written here, but I do is for every subset I compute it is weight am I right? And I have stored that weight, so I store that weight, then if I can find this size in the set f(r) then I take those weight, and I add them up, I written the query, then the query is out time.

The time of taking query will depend will be at least the size will be basically the size of this size f(r). So, in the… If you are looking at the worst case query time, you question you ask is for any query, what is the number of canonical subsets that will represent that query output. So, what you like to do is, since you want to keep both space and query time small, you like to keep both of these quantities as small as possible. So, basically minimize (No audio from 19:34 to19:44) so I really what you like to do, you like to m to be as close to liner and keeping m to be close m then you would like to f(r) to do very small, because remember the query time.

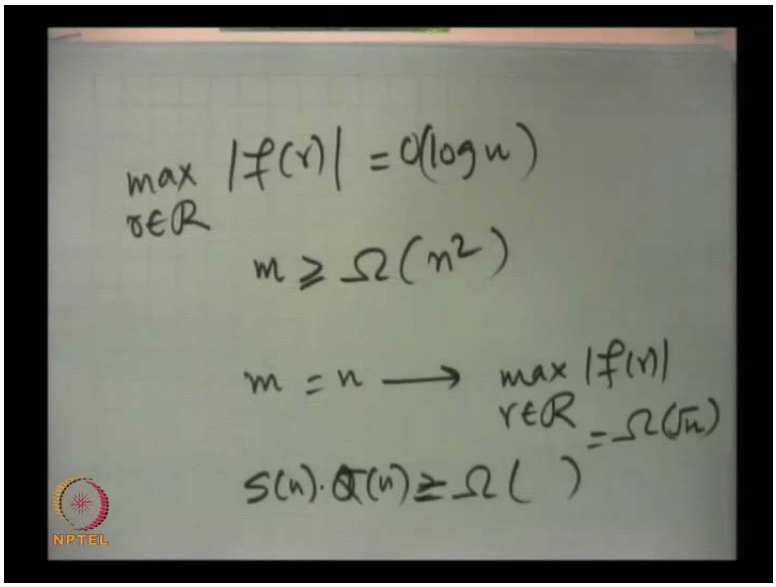
(Refer Slide Time: 20:09)



You want to be roughly log n then but you like to say is that for any query f(r), what you like to say is that m is roughly n and f(r) is roughly log n for all R, so this very strict requirement that you would like, for rectangles we could achieve that, if we look at the range tree or k d tree data structure that you saw, both of them fall in this framework. And for the k d tree m was precisely m, if you ignore the constant and, but the size of f(r) was root n, because you have to visit root n nodes, so it was root n, but by allowing m to be log n, but allow m to be n log n, it could reduce f(r) to log square n or log n depending how exactly do it.

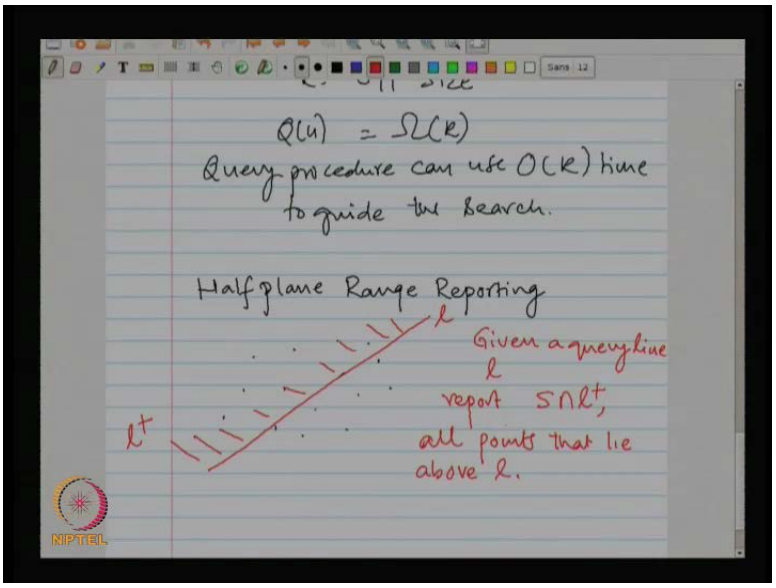
So, that is the game one has to play and question, how much you can succeed, how much you can squeeze? Now, of course the number of inside did not say so far is that one challenge will be, I know that, I can represent a query output with a few canonical subset, but how I find those canonical subsets quickly that will be one question, and also, how precisely I compute this canonical sets, and so that those are that challenges. And when I talk to specific problems you will see, but that is a general frame and this frame work is also very useful to prove lower bounds, that you cannot do better than this and that is how they show that, well if I want to describe, what they want show is, even the lower bounds for orthogonal range searching that is how they were proved.

(Refer Slide Time: 21:45



So, if you want to say that max (no audio from 21:42 to 21:49) is let us say odd of log n, suppose if this what they show, then what you show is that m has to be at least omega something. So, for example, if the ranges are triangles then what you can, what people have shown is that, if you want to get this bound then m has to be at least n square or you can do other way, say that if m is n, then this implies that maximum, something like this and here it will be root n. So that, that is are the lower bounds are proved, typically what people prove is that you prove say S(n) times T(n) sorry Q(n) has to be at least, so, but you say is that the space time, the query time product has to be at least this much, and that is how people prove, but I will not talk, I will not prove the lower bounds in this class, but I just wanted to tell you this frame work, because this is lies under lies most of the (( )).

(Refer Slide Time: 23:12)

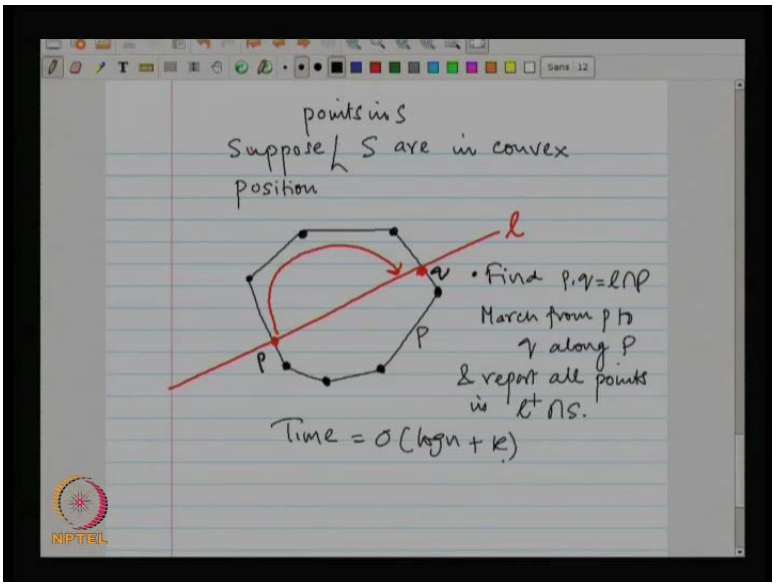


And the second idea that is used in range searching is the following, what is called filtering search, this is meant for range reporting. Remember that if you are range, if you are reporting the points such and if you are going to report k points, if you know that the k is the output size right. If you are going to report at k points, then you are going to spend at least k times, so the query time is actually state. If I know that fact, if I am going to spend k time, might be use that k time to help my search as well, so I can spend additional, because I do not care about constants for an asymptotic analysis, if I know that I am going report k points that might I might spend another order of k time to do help my guide my search. So, then I can use then the query procedure and use k time to guide searching and this will be you see that this will be critical for many some of the data structure that I will talk about, any questions so far? Is this clear or…

So, now that I have made general remarks, I want to now go to a specific problem, and I will start with the very simple problem is a half plane range reporting. First I will talk about reporting, I will show you the filtering search, then I will talk about half plane range counting. So (No audio from 25:04 to 25:21) so now the problem is following, this is specific problem of range searching, you have a set of points, query is a line and a line as I have said line divides a plane into two regions, each of them is called half plane. Let us say and I I am interested in the half plane lie about the line. So, let us call this line l and this side of a line which lies above line, let us call it l plus. And the question will be

given a query and l report report all the points that lie above the line (No audio from 26:10 to 26:26) right so that sort I will do.

(Refer Slide Time: 27:00)



So let us start with the simple case and that will help us for the generally stuff suppose (No audio from 26:41 to 26:51) S (no audio from 26:52 to 27:10) So, first let us assume the case when points and S are convex position, but I mean by if you look at the points they are form the vertices of a convex polygon, so (No audio from 27:19 to 27:38) so let us say this is, these are the point set. Now, if I want have a query line l, and I want to report all the points that lie above the line, so how will you answer them, and how much time will you take. (No audio from 27:57 to 28:19). So, how will you do it, before you tell me the running time, so how will you do it.

(( ))

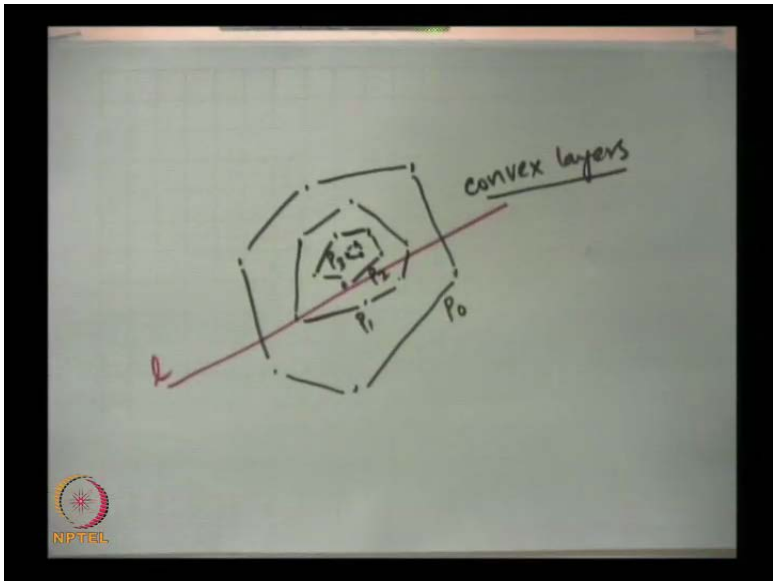
Which point?

(( ))

So you find this, let say these intersection points, we just covers that polygon am I right? So, what you do is first you find the the let us say left intersection point and then you start following this boundary and you merge on the boundary and report all this points, so (No audio from 28:53 to 29:20) So, let us this let us call this polygon as p am I right? Now of course, there is a possibility that line may not intersect in the polygon am I right?

that is also possibility, then what happens? These are all other the line or lesser then the sort that is the simpler case and we do not have to worry about, q then march from p to q along p and report and you said the time is said is log n plus k. So, that is that is was simple, now what happens is the points do not lies at the convex position right then what will you do (No audio from 30:18 to 30: 28) yeah so (No audio from 30:30 to 30:37)

(Refer Slide Time: 30:30)



So, what you do is, you first compute the convex hull at this point. Now, among these points, I know how to answer a query in this points I do it. So, for then I have to vary about the remaining points, but now basically I do the only n points and then I do the same thing. Let me add some more points (No audio from 31:08 to 31:18) So, let us call this layer as p 0, p 1, p 2, p 3, so this is called onion peeling, because it is like peeling layers after layers and they are called convex layers (No audio from 31:33 to 31:39) So, I think you had a problem either in the homework or in the midterm about the maximal layer am I right? So, that is this a different and this are convex layers.

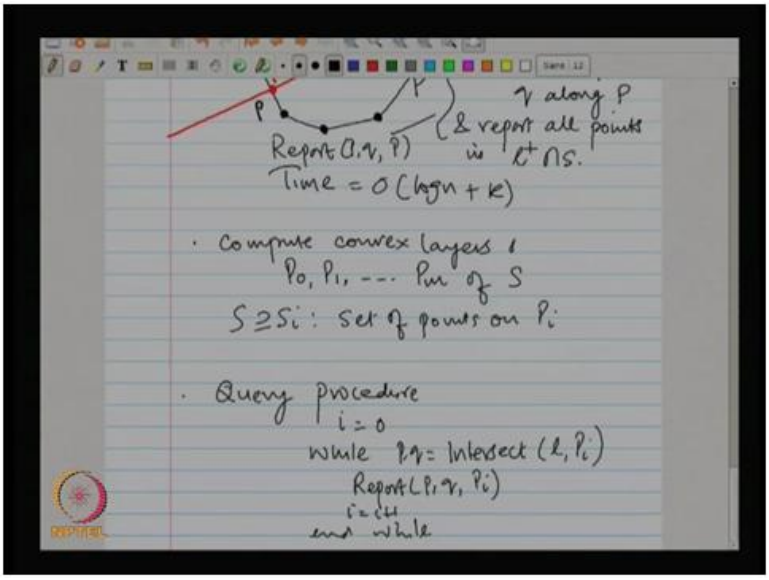
Now, you observe the following, I will worry, we will not talk let us say ignore for now, how I compute this layers, how much time they takes to compute those layers, but suppose you computed these layers. Now, let us look at a line l, now how will you answer the query. So, in wich in in which direction should you go, so which layer you will look at the first, look at, so look at outer most layer first and use the previous

procedure to report all the points am I right? Then you go to the inner most, then you go to the next layer and so on. When do you stop?

(( ))

Below right, because you know that if the if a this layer, once you know i th layer does not intersect the line, then none of this inner layers will intersect the line am I right? Because they are raise inside this convex polygon, so that is why it is important you go from outside to inside.

(Refer Slide Time: 33:02)

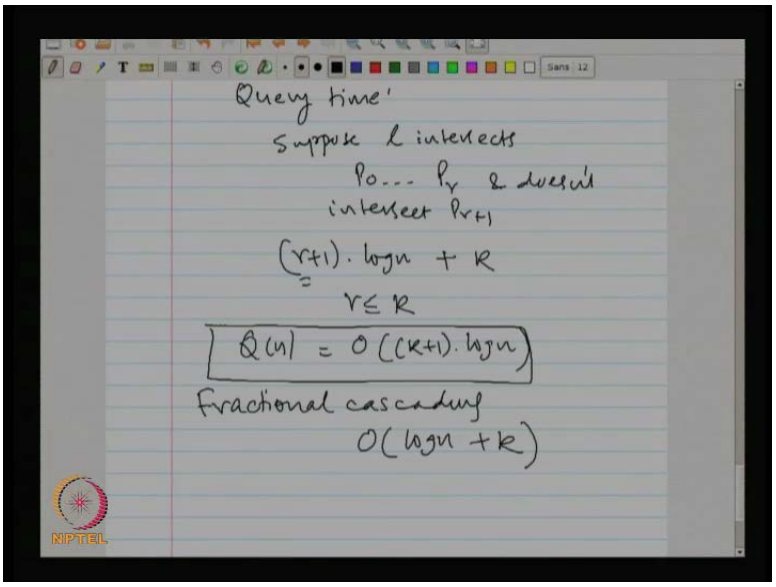


So, the data structure becomes compute convex layers, let us call them p 0, p 1, p m of s. So, s i is let us say the set of points and p i, then query procedure is i is equal to 0, it is start from the outer most layer, while let us call this procedure as intersect p l. So, that will the procedure that computes intersection points and let us assume that this intersect this intersect procedure either returns intersection point or return it returns null. So while, p, q (No audio from 34:14 to 34:20)

So, compute the intersection points of the lines with the layer, so either if you return a point, then you do the while otherwise you go. And then you say report and let us call this procedure has report p, q and p, p i and y am I right? So, that is what we do. Now, first let us talk size or data structure is linear, because each point is stored on one layer y

am I right? So, the size of the data structure size is order of n, we will talk about the preprocessing time later.

(Refer Slide Time: 35:27)



So, what is a query time?

(( ))

Say it again, why order of log n plus k, yeah so, but you are spending log n time for each layer, log n to the number of layers am I right? So suppose so suppose let us look for the following thing, let us write a let me little care full. Now, suppose l intersects p 0 up to p, let us say call p r and does not intersect p r plus 1. So, it intersects r layers then so, then it what happens is that, you do a binary search r plus 1 times, because the first time you detect this not intersect you stop am I right? So, it will be r plus 1 times log n plus the number of points reporting plus k. Now, what is the bound you can give on r, because r is a number of layers, but that is too much, because then the query time because n log n, so why is it k?

(( ))

Said so yes if you look at the… if a line intersects a layer then at least one of vertices of that polygon lies above the line am I right? So, what you know is that r is almost k, so that quick the query time is k plus 1 times log n. Now, you (( )) why wrote k plus 1, if I am using big on notation, why I wrote k plus one times log n, this k could be 0 am I

right? So, then this expression becomes order of 0, which is not right because you spend at least log n time am I right? So, that is why I have to write it k plus one times log n and this this you have to be very careful specially in number of time specially also, when you do the base of log, when you doing it.

So this you have to be careful with this, one this because some time if you are not careful with this k plus 1. You can get wonderful results, which are not true. So, this is (( )) query time. So, have you learnt fractional cascading in this class, have you heard yeah come on either yes or no why everyone is quite, either you have heard or learnt fractional cascade or not done it or you think that you are sleeping in the class, you do not know what happens in the class.

(( )).

Pardon.

(( ))

We talk about it once

(( ))

Heard, but not learnt clear enough. So, let me continue with that tradition you hear it again, but do not learn it. So, if I have time at some point I think need technique and I would like to do it, but if I do that today then I would not be able to finish it. So, let me postpone it and I will try to do it. So, the fractional cascading technique, what it does is because you are spending log n time doing binary search at each layer. Now, what you like to do is, you like to do binary search only once in the outer most layer and may be store some additional information am I right?

And then you spend only constant time, how many of you have heard about escape lists, most of you heard about escape lists, so idea is the kind of very simple should think about it am I right? What does skip list that you do it. You basically you you propagates probabilistically or deterministically some of the points up and up and up, but and that helps you to do the search. And the same idea fractional cascade is very similar idea, what you do is, you remove, you move some of the points from inner layer promote them keep them there, but also promote them up to the next layer.

And similarly, fashion of the points you promote them up and you keep on doing this promotion and then when you do the binary search here. You can not only you have done the binary search here, but implicitly you also got then binary search in for all the layers, because you can go in constant time from here to here, and then here to here in the constant time, which is what happens skip list am I right? Because what you do is, you do a binary sorry and then you have the pointers to the next layer that helps you go in next layer. So, that is a basic idea in this skip list, but it requires more work and I will skip that details, but and if you use that fractional cascading the time becomes log n plus k.

Now, let me go back to the, go to the preprocessing time, if I do naively I compute the convex hull that takes n log n time. Suppose you have done the sorting, you sort the points once then you can do it in linear time, so let us say you spend linear time after n log n preprocessing, you can compute the convex hull linear time. You find the point lie on the on the convex hull, you through them out compute the convex hull again and repeat it, but if you do it naively. You might be in the situation that the number of layers can be linear; because the layers might be just triangles am I right?

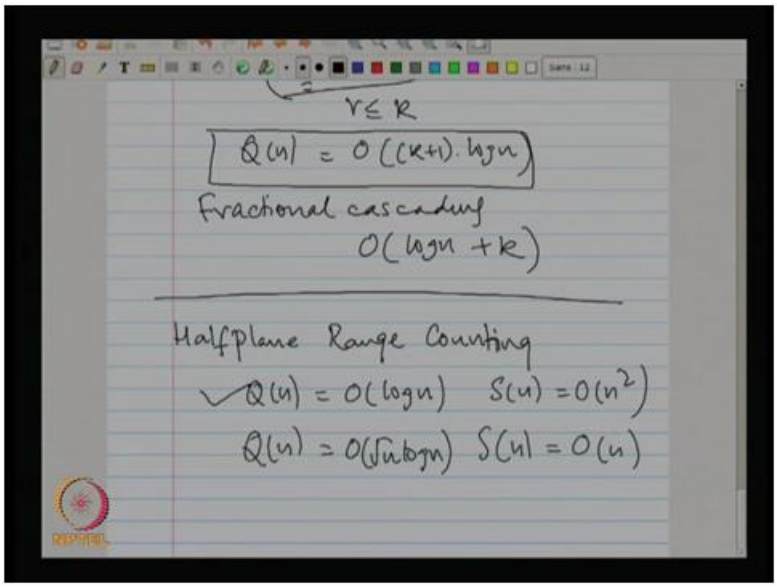
So, you spend linear time at each time it for each layer, but the number of layer is n, so query preprocessing time the total time you spend is quadratic, but that is too much and there are data structures and you can find in your text book, that you can maintain the convex hull under deletions in log square n times. So, you can delete set of points and update the convex hull log square n times. So, what you can do is, you can remove this points, find the convex hull delete this points then now you have the convex hull of the reaming points. And you spend log square n time in deleting a point and each point is deleted only once. So, the total time becomes n log square n.

So, naively preprocessing time is n square, you can bring it down to n log square n using dynamic convex hull, but that is a set of the procedure takes log square n rigorously allow to delete and insert both and you are allow to delete any point, but here you are not inserting any point, we are deleting only deleting points and we are also deleting in very nice manner, we kind of peeling the onion layers. In that case by being more careful, you can do delete and you can delete a point in log n time. So that basically brings it down to n log n. And again you heard it, but you did not learnt it, how will you do it. So, n square is sort of very simple I can do it, but you can do.

So, without sort of telling you how the p and hence the p n is n log n, so that basically what you can hope for, because query size is linear, query time you know, you cannot do better then log n and k you have to spend to report k points, but this a filtering search example of filtering search, because you look at the r plus one log n, basically I use the factor k to charge the time in doing the search am I right? Now, if I want to do counting am I right? If I do not want to report, just want to count how many points lies inside half plane, then this is nasty am I right? Because then I since I am not reporting the points, I do not have k to play with right, then you would like say I should able to count the number of points in log n time, but then this procedure does not work, because r could be k could as large as n, but with the first term it is still there, this term is still there am I right?

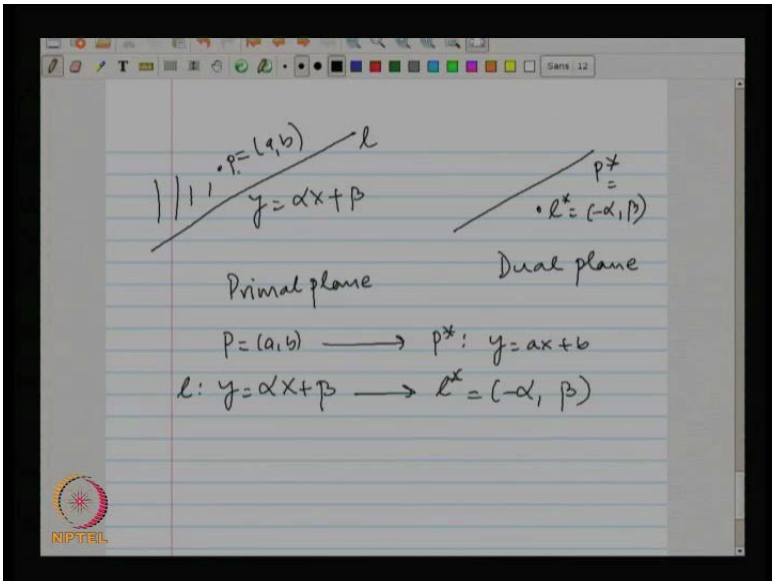
Because by fractional cascading, you get it of log factor, but you, but you have to visit each layer and the number of layers could be n. So you spend linear time, so just say filtering such place, because we either heavily use the fact, that I am going to report spend k time in reporting those points, and I use that very strongly. Now now, what I want to do is, I will start today, and I would not finish it today. Let us talk about half plane range counting, what can you do to count it and it turns how this is a will be the first instance you will see, that if you want to do counting, when you cannot do this cheating or filtering search, then you cannot get this, you cannot answer a query in log n time using linear space.

(Refer Slide Time: 45:20)



So, what let me sort of tell you at outset, what you will get for half plane range counting, if you want Q(n) to be log n, then S(n) will be n square, and if you want Q(n) to be S(n) to be order of n, then Q(n) will become that actually the way all describe it will be square root of log n, you can get rid of the log factor, but this square root of n and what tells out, that there is no hope of doing anything better, you cannot do better than this, so which is surprising. So, let me talk about this one, the first one, because this is the simple and I can finish it today and then the second one I will talk on tomorrow any questions?

(Refer Slide Time: 46:40)

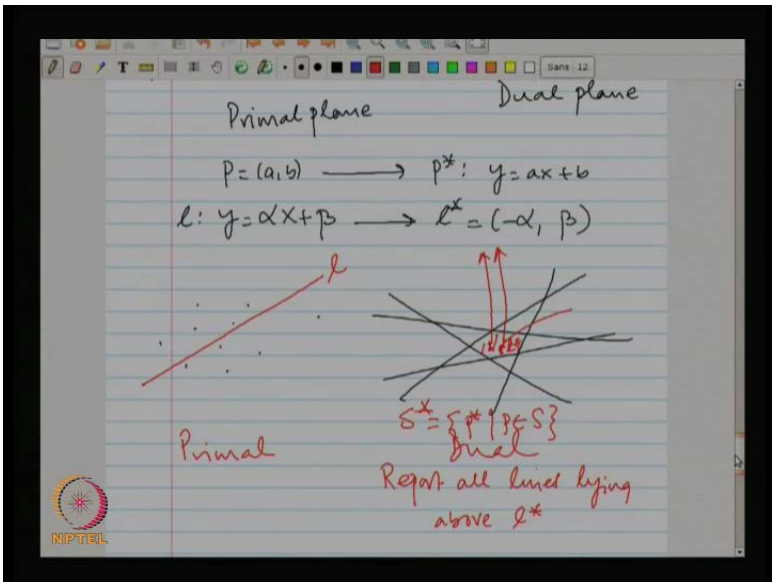


So, I will use duality, which you have seen in the earlier. So, if you have said a point and a line l, so p is let say a, b there are many forms of duality and I will just use what is convenient here, and there are I think you might have seen a different one earlier, but I use this one, I do not when I, what I was talking about, but I did this one here. So, if you have such a thing, then you can, what you can do is, you can map point p to a line p star. So, this is a primary plane and this is the dual plane, so p star becomes… So, let me write it here, so p is equal to a, b that maps to a p star, which is y is equal to a x plus b and align that maps to point l star, which is minus alpha beta.

So, and what you sort of the picture that I have drawn, it will illustrate that in the primal plane, If p lies above l then it is dual line p star lies above the line l, l star. Now, if you play with the co efficient little bit, then you can do difference ways of doing it, but this is a property, I wanted to preserve that the it is the p lies about line l, then p star lies above

the line. So, if you have set of points, if I then I ask the following question, if I take the so half plane range, so range was range searching was that you wanted to let say report or count the number of points lying above the line am I right? So now, if I (( )) it then what happens.

(Refer Slide Time: 48:46)



So, you have a set of points; let us say in the primal. Now, each of them utilize that becomes a line am I right? Now, if you have a line l, this becomes a point here in the primal plane, so this is primal, this is dual am I right? So, in the primal you wanted to report all the points at lie above the line, so what does it mean in the dual.

(( ))

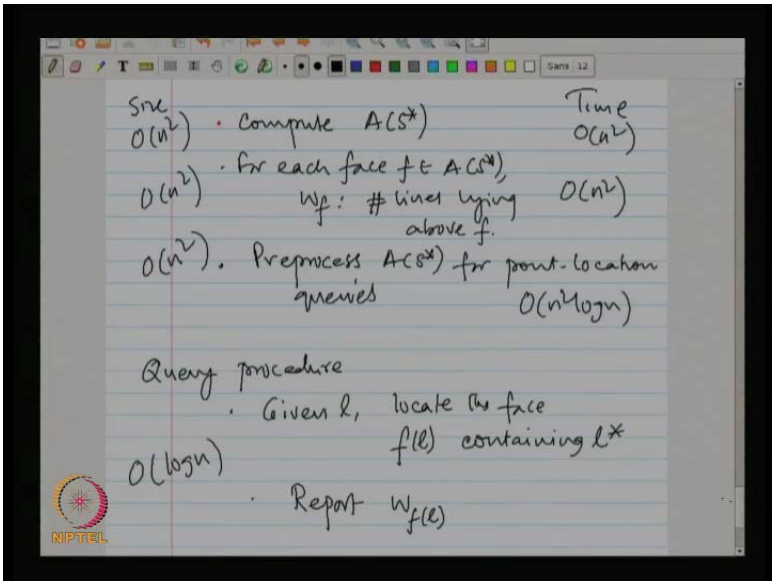
Yeah all the lines that lie above this this point I will start. So, if you think about drawing the vertical ray, report all the lines that intersects the vertical ray starting from l star am I right? So, here report all lines lying right. Now, other than you know to say, if you take two such rays.

(( ))

So, the above am I right? Because you notice this picture, that p above a line l is the line that above of the point in the dual. Now, if you notice that if you take two such points, and look at the lines that lie above it (( )) these both of these vertical rays intersect the same set of lines. So, if you have two points that lie in the same face, here in this

arrangement then they intersect the same the vertical rays insects the same set of lines. Now, if I want to do the count then, what what it means is that for each face, here you store how many lines lie above it, I want to do counting. So now, what you do, so can someone summarize the data structure, so how will you answer, what data structure will you construct. So, I remember that we talked about arrangements am I right? This is what you do is, (( )) we have a set of lines. So, let us say s star is a set of lines, which are the dual to the points, what this is what I have drawn here, is the arrangement of this lines am I right? Which is the planer decomposition?

(Refer Slide Time: 51:29)



So, first is you do, you compute the arrangement, then what you do for the arrangement (No audio from 51:36 to 51:43) so what do you want to store for the arrangement, this one what I told you.

(( ))

For each face, so the lines that lie about it am I right? For each face f, let us say w of f is the number of lines and then finally, what you do, how do you answer a query, what you (( )) what you need to know is, when I get a query line look at the dual point and find out which face it lies. So, is this you have seen, what is that called this data structure, yeah someone said it, point location query. You so now here planar graph preprocess, it appears a submission preprocess for point location queries (No audio from 52:35 to 52:51) and the query procedure (No audio from 52:53 to 53:00)

Given l, locate the face f l containing l star am I right? Find the face that contain, report w f of l report what you store the quantity that you stored there for that face. So, what is the query time, how much time in the point location query take log n time am I right? So this is log n, what is the size of arrangement n square am I right? So, this arrangement is such this is an n size is now, well this is computing quantity for each phase that I am square faces this also takes only n square space, what the size of the point location data structure order why order n? When you say point location is order n, what is n there, number what, how many faces are here, n square am I right?

So, it is a linear in the size of the planer sub division and but the linear size is here quadratic, so it is order of n square. So, you have quadratic as data structure and computing the arrangement that you seen takes n square time am I right? So, this is size; and this takes n square time; and when you compute the arrangement, you can maintain this also, I leave it for you to figure it this out. So, this takes n square time, what is preprocessing time for point location? Come on I am running out of time, I need to finish it. So, if you have planer sub division size and how much time does it take?

(( ))

It is not linear.

(( ))

No.

(( ))

n log n right.

(( ))

Why will you take an n square, why not n square log n?

(( ))

Yeah it is n square log n am I right? So, you can do (( )) you get it has a log factor by being more careful, but let me not get into let me just stay with smallest count. So, what

you get? You get this data structure, whose size is n square, query time is log n and preprocess time is n square log n.