NPTEL
NPTEL ONLINE CERTIFICATION COURSE

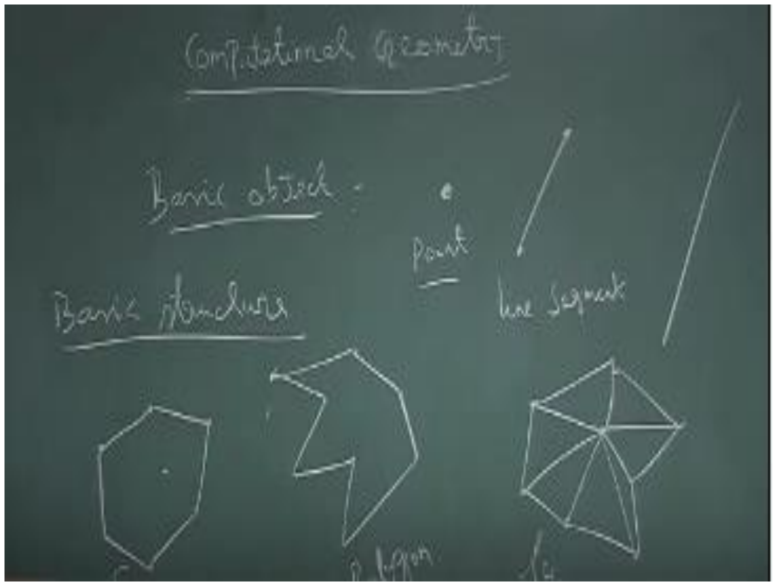
Course Name
Fundamental Algorithms:
Design and Analysis

by
Prof. Sourav Mukhopadhyay
Department of Mathematics
IIT Kharagpur

Lecture 13: Computational Geometry

Okay, so we will talk about computational geometry, it is basically the new idea for its developed algorithm for solving the geometric problem.

(Refer Slide Time: 00:34)



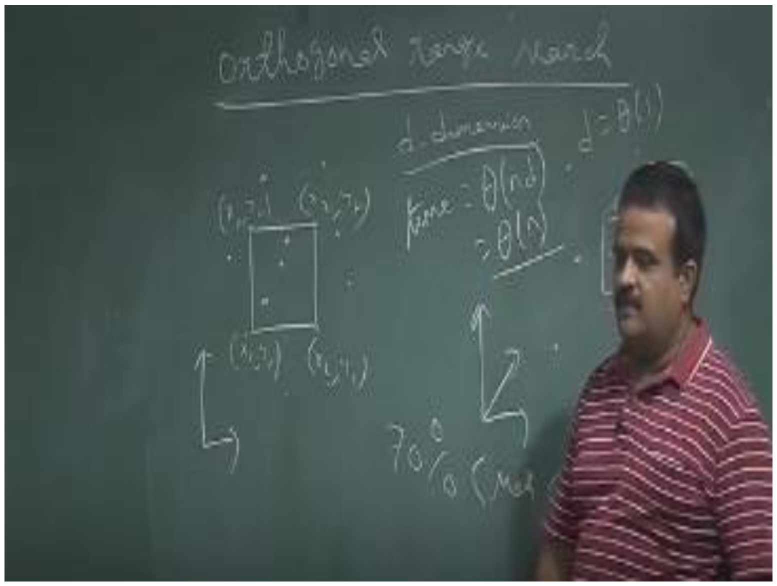
So first of all what are the simple basic geometric object, a point it would be in any dimension here it is in 2D, but it would be having any dimension. A line segment or a line so these are the basic geometric object.

Now based on this suppose we have some basic structure, basic geometric structure basically suppose we have some points, we have some points. Now the problem may be used to finding the polygon, so basically we need to, so this is the polygon, it is the close loop which we will not intersect with each other.

Then the next problem could be triangulation, we have some points and we have to find the triangles. So this is called triangulation, and the second problem could be convex hull we have some point and we have to, so this is the convex hull so we have to close this points in such a vision that if you take any two points it should be inside in that region.

So these are the problem which are used into fluid mechanics or any other area. So now today we will talk about what is called orthogonal range search problem.

(Refer Slide Time: 03:02)



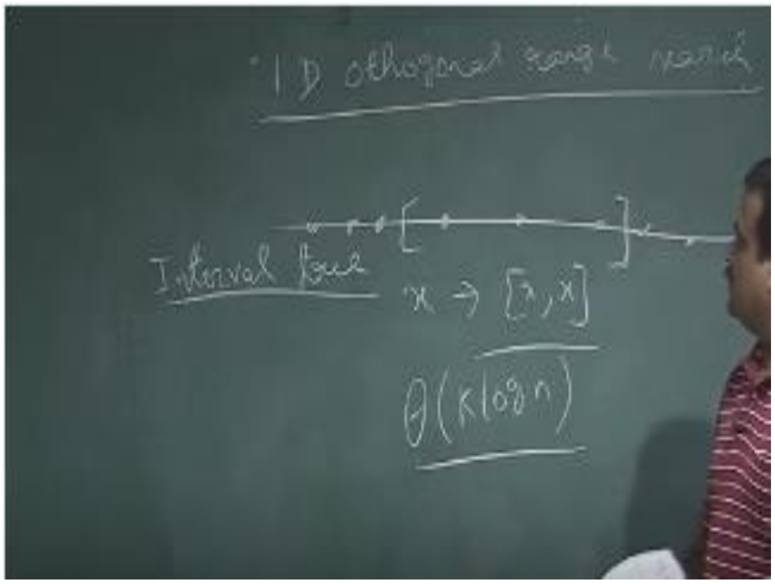
So we will talk about orthogonal range search okay. Now basically the problem is suppose we have some points in any dimension, this could be a database problem this is say, we are in 2D and we could be in 3D also, suppose these are the point in the space, these are the 3D points okay.

Now we have given a orthogonal range for 2D it is a rectangle. So we have to given a rectangle and we have to find out the, whether is there any point in this rectangle or not. And how many are these, so these are the query, is there any points in this rectangle this is on query how many are they, we have to output all the points where in the rectangle.

So this is the query and 3D this is a box like this, so 3D box okay. So this is orthogonal means this is parallel to the axis. Now this could be determines problem, so it could be d dimensional, so like say we have a student record, so each student it having say many attributes like many fails like the marks in the algorithm course, marks in the mathematics, marks in the computer science, the overall marks.

So these are the attributes, so these are the field, so how many subject we have different, different dimension. Now suppose a company came and they are asking for some specific marks they are asking. So they need the mathematics marks to be lies between 70% to 90% maths mark. And they want their age should be like this, so they can fix a box in this, over this database and they can find, try to get this query. So this is on the applications, so now how to solve this problem.

(Refer Slide Time: 06:47)



This orthogonal range search query. So one sort what we can do, suppose we are in 2D so we know this coordinate, so this is the (x_1, y_1), this is (x_2, y_1) and this is (x_1, y_2), this is (x_2, y_2) suppose we have given this rectangle. Now how we can do this, we can just take all the points and we can verify their X Cordiant and Y Cordiant.

If this belongs to this, then we are done. So if what is the time, so that is the name approach or that is the one sort approach. So what is the time complexity for that we have to do it for all the points and if it is in dimension if d dimension this is typically n x d but if d is a constant I mean then this O(n) but what we want to do something better than O(n) so what we can do can use some sorting algorithm for this suppose we are doing this for 1d so 1D orthogonal range okay.

So 1D means we have this line and there are some points lies in the points okay and we have a varying range and we want to report all the points lie between this range so that is 1D range such so how we can use sorting so we can sort the points and then we can do the binary search based on this two pints and however in the middle we can report them, so we can do the binary search, binary search based on this n points.

Two n points okay so how much time it will take it will take basically so sorting will take n log n time because if it is static then it will we have but that is that can be added in the proposing step so 1 line step it is basically the binary search it is basically log n +k we have to report all the points okay but this we cannot extended for higher dimension because we have to sort which coordinate high dimension we have many coordinate x, coordinate, y coordinate so we cannot extend this idea for higher dimension so easily.

So another way is to use the interval tree, we know the interval tree so what we can do, we represent a point by the interval of length 0 okay so we replace all the points by interval then we from the interval tree then we have varying interval and then we search and then we once got a point we delete from the interval tree and we will again make it is a new interval tree so these way so this will take the time basically O (k log n) okay.

So now we are going to discuss a method which is called range tree which will gives us O(log n + k) and that is easily that method can extend for 2D also.

(Refer Slide Time: 09:53)

So this called 1D range tree range search tree or range tree so the idea is to keep the all the elements in the leaf and we will form the BST balance binary search tree so let us take an example suppose this is these are the numbers we the these are the elements we have 12, 14 17 these are the elements we want to put everything in the leaf, so lefts contain the inputs.

So for 12, 42 and then we have 43 59 61 so these are the input now we want to put this input in a leap level and we want to from the BHD so what we do we can do like this do like this and here also like this, like this, like this, like this, like this then in the top the root is okay so these we want a BST so BST means we should take a note all the key value in the left sub tree is less than x all the key value in the right sub tree is less than x , greater than x and this is the key value of the root of that node.

Okays so now so what are the values we can put here we put here any value is between 1 and 6 okay so but we want to put the minimum of the left maximum of the left sub tree so this we will put 6 this we will put 1. So this we will put 12 and this we will put 14 this we will put 8 maximum of what the left sub tree like this we fill this 26, 41 then 35 then this is basically 59, 43 then this is 42 and this is 7 so this called 1D range tree so this is basically a BST we can make it a balance binary search tree okay.

So now suppose we have give this so now how we can put a query suppose we want to query say 17 sorry 7, 41 this is our query and we want to get all the elements in this range this is wanting range search so this is the 1, 9 and 7 orthogonal range so that means this will be basically from here to 41 sop here to here, okay. So that means all the elements from here to here, so if we can report this all the, so all the nodes like this, like this yeah.

So basically instead of reporting this node individual if you can report all the sub tree these are the sub tree whose leaps are basically our answer leaps our basically our outputs, so if we can output this sub trees then we are done then we can go to the corresponding ellipse and we can output those leaps so that is the idea. So we will here try to report this sub tree if we just report 8 that means a sub tree routed at 8.

If we just report 14 sub tree routed at 14 so tree routed at 14 so that means all the leaps of that will be the output so like this, so this is the idea of 1D range query so let us as draw what we are doing.

(Refer Slide Time: 14:59)

So basically the 1D range query so we want to get those sub trees so what we are doing we are starting with the root so this is the root, okay. And then we may see nothing is relevant in the right part of the tree.

Because we are known nobody is telling the right part of the tree, so then we will go the left part of the tree and here if we realize that nothing is relevant in the left part of the tree so we will go to the right part again. So this way we continue and that some point of time we reached to a node where we realize something is relevant in the left part and something is relevant in the right part so this is called split node.

So split node means it is node where something is relevant in the right part something is relevant in the left part all that something is relevant in the left part something is relevant in the right part, so in the split node what we do? Then we continue traversing so we traverse this so maybe we realize that everything here is our in the range then we go to the right like this then we realize everything here is the relevant.

Then we do it for the left part and then we see everything here is relevant we go to the right then like this like this so these are the we report all this sub trees we report the root7 of this sub trees and that will give us the basically nodes and leaps are basically the nodes which are in the range, okay. So this is the general description of the range in, so now how to so let us write the pseudo code for this.

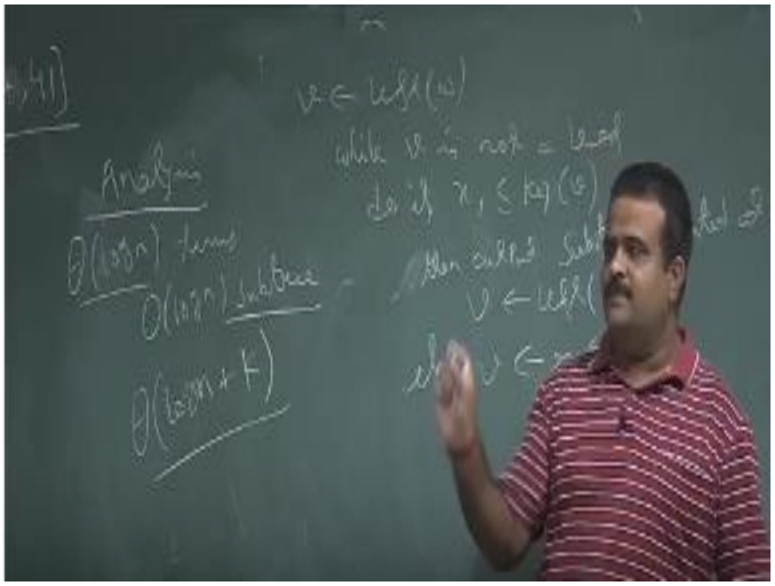
Then it will be more clear, okay. So this is the scenario 1D the range query, so we have given the range okay the range 1D range are range tree and we have the orthogonal is x1 interval x1, x2 so what we do we first take the root and then we continue until we reach to a split node, so how to do that? So while root is while W is not nil not a leap if it is a leap we stop and we check whether this node belongs to that interval or not.

And this is end and x2 is less than key value of W or this is odd key value of W is less than x1, so that ,means we will go either left part or we will go to the right part until we reach to a split node, now do if so this is underlying do if so if x3 < key (w) that means we have to go to the left part then w is the left of w otherwise else w is the right of w, so this will stop either w is the leap node or w is a split node.

So that means we will go either left part if everything is this side or this side we will go to the right if everything is that side or this side and he will go the right with the back side so this will continue until we reached your note which is a basically that split note so we check this is a root like this until we reached to your note where something is relevant over here something is relevant over here so that is the speed note so we reached to a speed not otherwise it is leaf if it is leap, if it is leap we check they are follow then we if it is in that range so that we have to write so this is if w is a leaf.

Then we have output w if the T value of w is life between this range okay otherwise w is a tree note then we have to, to get the point so we have to go to the right and left part and also we have to go to the right part so last first we will check for the left – will be the similar so let us try to write the left our son okay so okay so this is the basically so w is not a leap note.

(Refer Slide Time: 20:44)



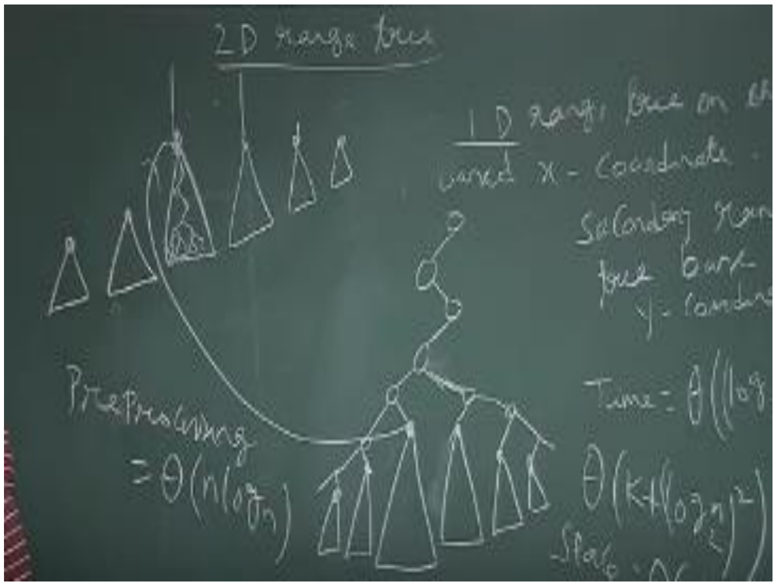
So we are going to the left part so this is basically left part of left of w so now what we are doing if v is not a not a leaf then we will check do if, if x1 is less than = k(p) that means so this is the street note which is v now.

This is basically now we check whether anything is interesting in the left part of v or not if anything is interesting in left part of B then we will output the right sub tree rooted at this so that is the check if x1 is less than this then that means we have to go further left part that means everything is relevant over here in the right part so this is basically T is basically this one this is the speed note B is basically this one so if everything if we have to go to the left part then the everything is right but is the relevant.

So then output sub tree root right of the okay and we then go the left of v again else we got to the right of v else space this is not correct and this we continue until it is leaf note once it is leaf not then we check whether that, that note belongs to this and in the range of that so this is the seedcot for finding the points in that range so let us analyze this okay so how much your spending here so suppose there are we have to report so it is basically balance tree so it is basically log n time so sorry log n log n time required to find the log n sub tree there are maximum log n sub tree is so many sub trees are possible.

Log k actually so this is log n we can write so long n sub tree can be reported and in the log n time and then once we report the sub tree we have to pin the all the outputs so that will take time log n last k okay this will be the this will be the to obtain the leaps of each of this sub tree okay now how much time it will take to for that query that how many sub how many notes are there so these query can be tackle by augmenting this data structure we can store in each store we can store the size, size of the leaps so that way we can augment this and we can have the query in log n time okay so now how we can extend this idea for two dimensional range search such we have seen the 1 inch at the points are on the 1 dimension now if the points have come 2D points.

(Refer Slide Time: 24:49)



So you have some points and we have using the range and this is basically and how we can extend this 1D to 2D so what we do we first make a primary tree based on the x coordinate on the points and so we will just make a primary we call it primary tree primary only range tree on the points based on the x coordinate okay so points are basically 2D so we will make a primary range tree based on the x coordinate that is another primary 1D range tree.

And then for in that tree for each of the point suppose we in that tree this is a point we have a another range tree which is called secondary range tree that is also wanted in tree based on the y coordinate on the points where this, this points are the leave rooted at this node, so for each of this node in a primary range tree we will have a pointer to node of the secondary range tree where with the secondary range tree is made by the y coordinate, so that is the idea. So, so first if we search in this primary range tree so we will get all the points lies between this tree, this is x1 this is x2.

Now this, this basically if we do the run search we are going like this, then this the suppose this is the, this is the straight node so we have something interesting here, something interesting here so from here suppose this we call paint this node then we go this side the go here suppose we all, so go this side here suppose these are tree we are getting in the primary range search then from this side also we are going this side, so this is, okay so like this so these are the basically so these are the basically tree we are outputting after the primary range search.

Now we have a secondary range tree which it made by taking all the points of, from the primary range tree for this point we take all the points over here and we make a secondary range tree based on the x, y coordinate. So this is, this is pointing to a, so we have similar tree in the similar size, okay so these are the tree so, okay so these are the, we have similar type of range tree based on the y coordinate. So now we have a pointer from here to here which is, which is pointing to that, now to that tree now we will do a search in this tree so in this tree all the points are satisfying the x range.

Now we have to search who are the points also satisfying the y range, so this is basically we do the again the range search in this based on this, so we got a speed node then we go left right like this so like this we go here like this, so these are the points is also in the y coordinate and so this, this will output so this is the extension of the one range tree. So in, in this 2D range search tree we have a, we have a secondary range tree based on the y coordinate on all the points, we take the sub tree rooted at this point and we make another range tree based on the y values.

So this is the idea, so now what is the time complexity for this, so we are doing twice this so this si basically $(\log n_2)^2$ now if you have to report k points if they are k points then the time will be basically $k+(\log n_2)^2$ okay, and what is the space, space is basically order n logn because there are logn search tree and for each of this for each of this element we are having a secondary range tree and the height of that is logn so it is n logn the space and the pre processing time is to make this, this also pre processing time is basically also order n logn, okay.

Now we can extend this idea further for are more than two dimensional, so if it is d then three, three dimensional then we will have another range tree one d based on the third coordinate z axis, z coordinate like this. So if you just extended for the d dimensional space then out query time will be, this is in general will be $(k+(\log n2)^d)$ if you are in d, d dimensional point like database and this will be in $(\log(\log n)^{d-1})$ and the pre processing time is order of $(\log n)^{d-1})$ so this is the case we, it is the general form of d dimensional time and space and pre processing. Thank you.