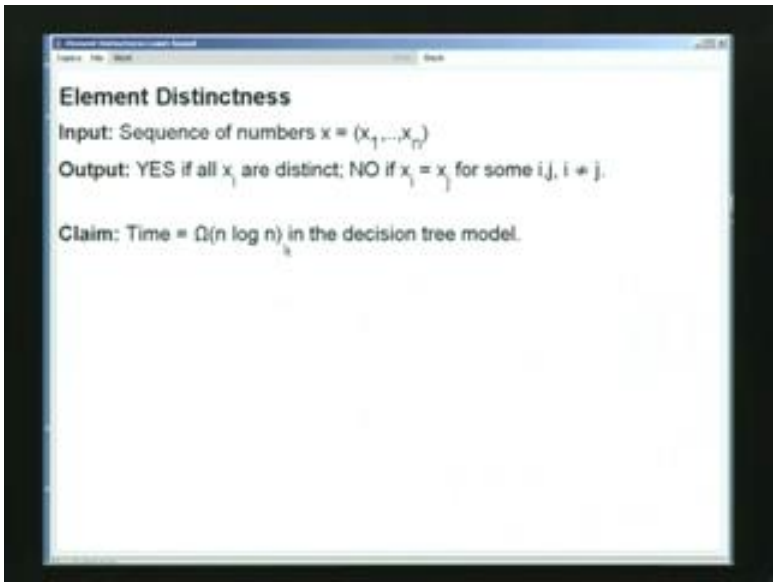
Design and Analysis of Algorithms
Prof. Abhiram Ranade
Department of Computer Science Engineering
Indian Institute of Technology, Bombay

Lecture - 25
Element Distinctness Lower Bounds

Welcome to the course on design and analysis of algorithms. Our topic today is Element Distinctness Lower Bound. This will be the continuation of the previous lecture, which was on sorting lower bounds. Let me start by defining the problem. So, the problem is as follows.

(Refer Slide Time: 01:11)



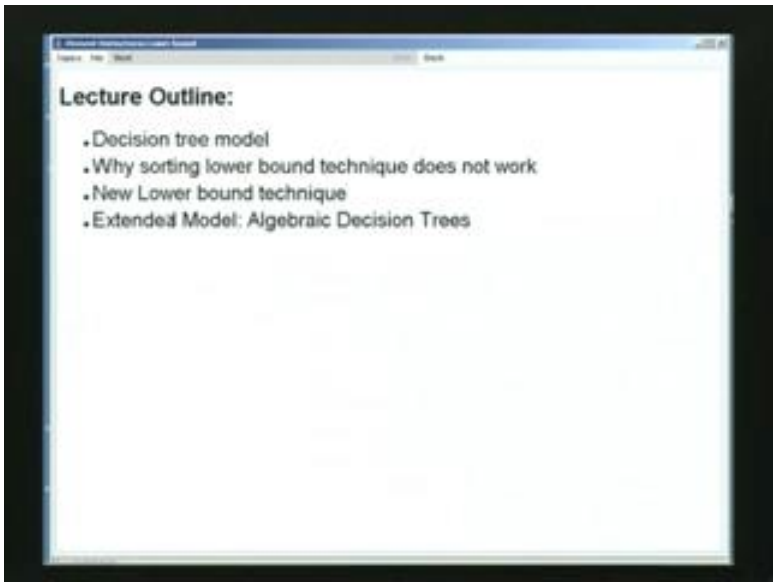
You are given as a input, a sequence of numbers. Let me call the entire sequence x and the individual numbers in the sequence are x 1, x 2, x n. We are supposed to output a yes answer, if all the x are distinct. And otherwise, if some x i equal to x j some two numbers x i and x j, distinct two numbers x i and x j are identical. Then, we are supposed to output and no answer. A no answer means, that no all the elements are not distinct.

What we are going to prove today is that, in the decision tree model, which we talked about last time. And which I will quickly define this time also just for continuity. We will prove that in this decision tree model, the time is going to be n log n. So, this is a non trivial lower bound, in the sense that it says. That you need to do a little bit more, than just examine the numbers, which would just take omega of n time.

It should be quite obvious to you, that if you are allowed to use n log n time. Then, element distinctness can easily be solved. How? Well, in n log n time we can sort all the numbers. And now, if two numbers are identical, they are guaranteed to come next to each other. In which case, we simply have to compare adjacent numbers after sorting and that will enable us to find out, whether all the numbers are distinct are not.

So, in summary in n log n time, we will be able to actually compute, whether the numbers are distinct. However, the subject of today's lecture is not that. The subject of today's lecture is to prove, that at least n log n time is needed. So, we want to prove a lower bound.

(Refer Slide Time: 03:03)

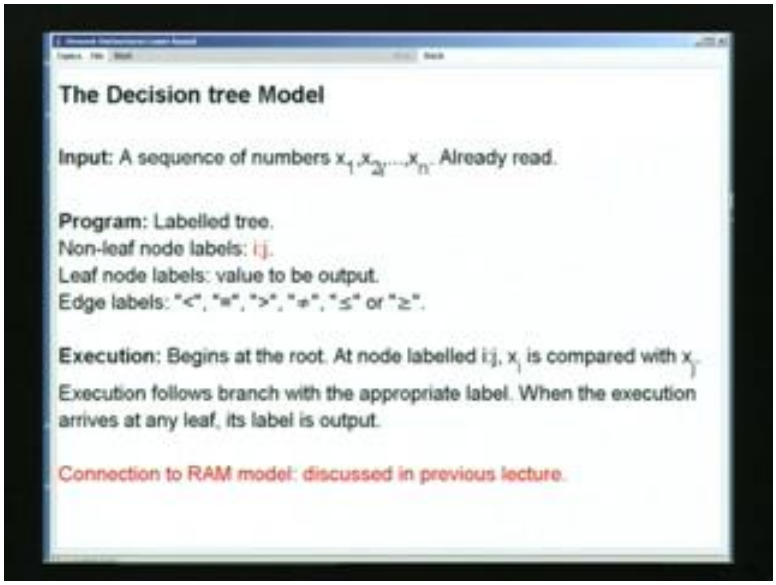


Here is what I am going to do today? So, today I am going to talk about, the decision tree model. I am going to go through this rather quickly. Because, it is going to be very similar to what we did last time. Last time we looked at lower bounds on sorting and we introduced a lower bound technique. So, I am going to explain, why that lower bounding technique does not work, that seem like a very nice technique. But, it will turn out that, that does not really work for the problem, which we are looking at today.

Then, I will talk about a new lower bound technique, which works for this problem of element distinctness. And then, we will prove the lower bound that I mentioned. Finally, I will extend the model. So, instead of the decision tree model, I will define a more

powerful model I will call the algebraic decision tree model. And I will talk a little bit about it.

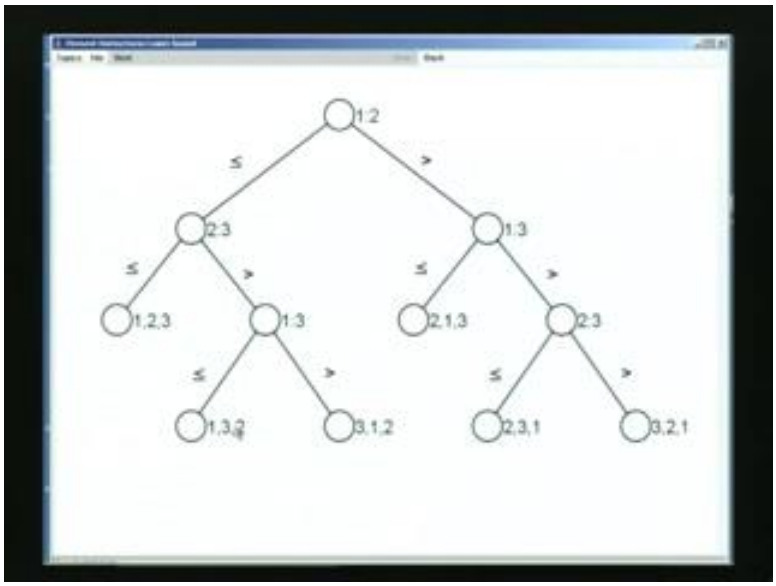
(Refer Slide Time 04:04)



So, let me start with the decision tree model. So, in the decision tree model, the input is always going to be a sequence of numbers. Say, the same sequence x 1 through x n. And we will assume that these numbers have already been read into the model. So, there are no input instructions as such, the inputs are already read. A program in this model is a label tree. And all non leaf nodes are labeled i colon j, where i and j are integers.

And these integers have to be fixed as a part of the program. Leaf node, each leaf node has a label, which just says what is the value to the output? Edge labels are relational operators. So, less than equal to greater than or not equal to less than less than or equal to greater than or equal to. Let me quickly take an example of a decision tree program.

(Refer Slide Time: 05:12)



So, here is a decision tree program for sorting three numbers. As you can see, each non leaf node is labeled i colon j in this case 1 colon 2. And it is each leaf node is labeled by the answer, that is to be output. And furthermore, each edge is labeled with a relational operator ((Refer Time: 05:34)). Execution begins at the root, at each node labeled i colon j, x i from this set of inputs is compared with x j.

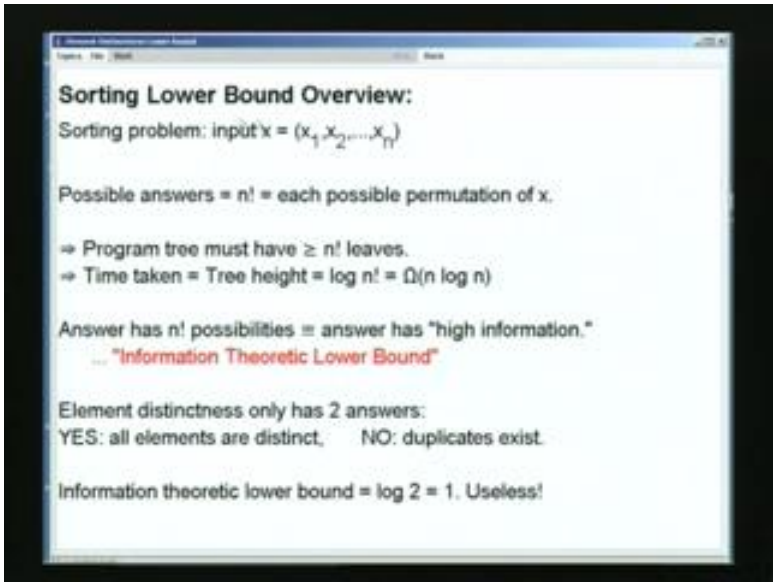
Whatever the outcome of that comparison is, the corresponding branch is found with the corresponding label. The corresponding outgoing edges are found and execution follows that branch, that outgoing edge. So, you go down to the node, which is at the other end point of that edge. And you repeat this whole thing, until you get to a leaf. When you get to a leaf it is label is the output. So, that is the answer that you are going to compute, that you wanted to compute.

So, two things to be noted. So, as we saw in the example, there is going to be a separate program for each input size. So, there is going to be a separate program for n equal to 1, for n equal to 2, for n equal to 3 and so on. And we saw a program for size n equal to 3, not for element distinctness, but for sorting. And then, the other point we noted is, that although we are looking at a decision tree model. And a decision tree model is not exactly like our computer, like any of our computers.

It does in fact, have a connection to the RAM model or the Random Access Machine model which in fact, resembles our computers. And this relevance has been discussed in

the previous lecture. And let me remind you what that relevance was. So, we said in the previous lecture, that if we have a lower bound in the decision tree model, when it applies to the RAM model, not completely. But, it applies to comparison based algorithms in the RAM model. Comparison based algorithms are simply algorithms, which compare the keys, they do not perform arithmetic on the keys. Or they do not use, the keys to induct into the arrays. They simply compare the keys and of course, they make copy.

(Refer Slide Time 07:48)



So, here is a quick overview of the sorting lower bound. The input to the sorting problem was the sequence x. The same sequence which we have mentioned, consisting of components x 1, x 2, x n. So, these are all numbers and we want to sort these numbers. The key thing to observe in this problem or in the lower bound argument is that there are n factorial possible answers.

So, either I could say that, this is the sorted sequence or I could take a permutation of this. And say that is a sorted sequence. I could take every possible permutation of this and that could be my answer. Every possible permutation could be an answer and there are n factorial permutations. And therefore, there are n factorial possible answers. This is a very important point, in this argument.

Now, the answer is going to be printed at the leaf of these trees of the program tree, which means that, if you want to print out n factorial different answers. Then, you must

have n factorial leaves, you have no choice. Because, a single leaf can just print a single answer, it will print out that entire permutation. But, that entire permutation constitutes a single answer. So, if execution arrives to that leaf, then that is the only answer it can print out.

So, if your program tree is going to be even capable of printing n factorial different answers. Then, it had better have n factorial leaves. But, if it does have n factorial leaves. Then, the height of the tree has to be at least log of n factorial, which is n log n. And in fact, the height of the tree is the worst case time. And therefore, the worst case time taken is at least n log n. So, this is a rough sketch of the argument, that we saw last time.

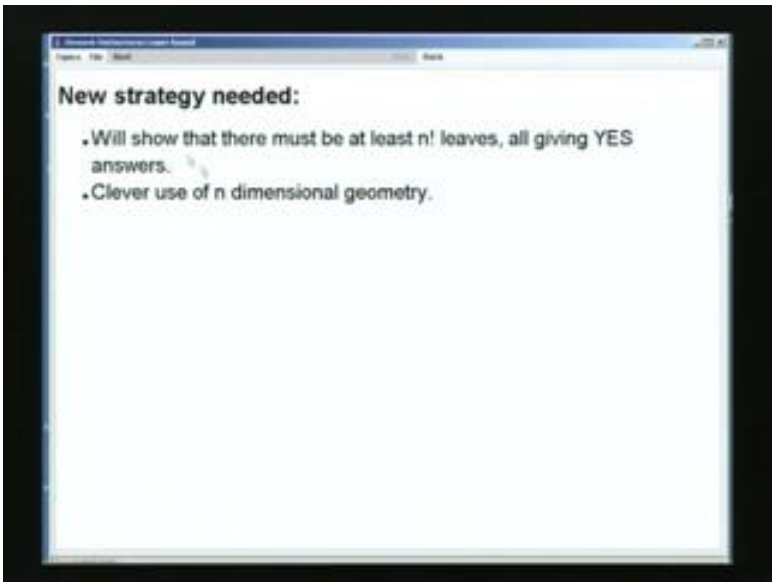
This idea, that if the answer is n factorial possibilities. Or if there are n factorial different answers, in the language of information theory, can be expressed as saying that this answer has high information. So, whatever we are going to print out as the answer, if you think of it as a variable, it has a very high information content. The answer can take n factorial different values. So, it has the information quantity in it is high is something like n factorial.

In fact, information theory measures information as the number of ways, the log of the number of ways. In fact, it is going to be log of n factorial, very roughly. In any case, bounds of this kind, where we say that there are so many ways in which this variable can take a value are therefore, called information theoretical lower bounds. So in fact, you will see the sorting lower bound often refer to as the information theoretic lower bound.

Let us now turn to the element distinctness problem. How many answers do we have? Well, the answers are two really, the answer could be yes, which means all elements are distinct or the answer could be no. No means some duplicate exist. So, we only have two answers, so our answer has only two possibilities. So, log of that is not going to be very large it is going to be in fact, exactly one.

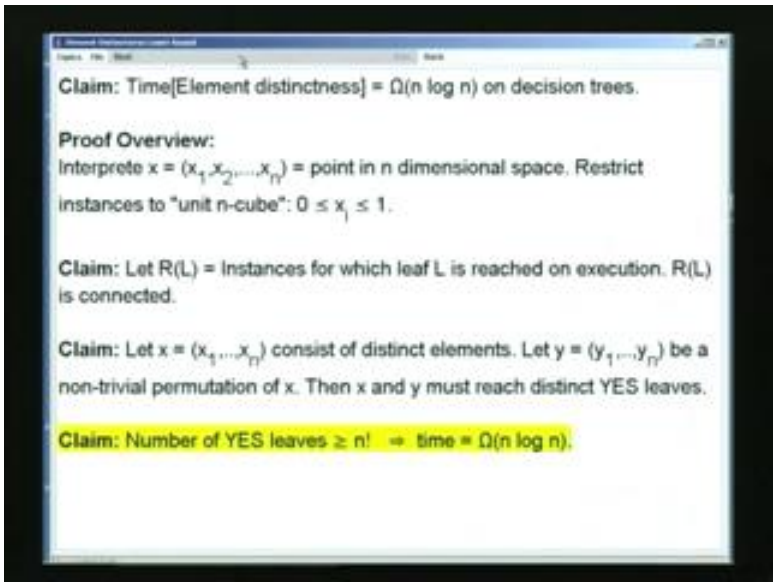
So, if we just say that there are two possibilities. And therefore, there are two leaves that does not help us much. Because, log of 2 is 1 and it just says that, the tree must have height of value, which is really a silly bound. So, we need to do something better. So, we need to have a new strategy.

(Refer Slide Time 12:01)



So, here is a rough sketch of the strategy. So, we will show that there must be at least n factorial leaves. Again the n factorial is going to turn out be somewhat significant. But, we will prove that in fact, there must be n factorial leaves all of them giving yes answers. So, this is going to be a interesting argument. And we will be making some rather clever use of n dimensional geometry. Do not be worried about n dimensional geometry, most of the time for the purposes of getting intuition. You can visualize, what is going on in two or three dimensions. And that usually tends to be enough which is in fact, going to be the case in our proof. However, if we want it algebraically write down things, the arguments can get a little bit complicated. But, fortunately even the algebraic argument that I am going to show you is going to be rather simple.

(Refer Slide Time 13:12)



So, here is our main claim again, we are going to prove that the time required for the element distinctness problem is going to be at least n log n on the decision tree model. Let me remind you what this claim means. So, this claim asserts a problem lower bound, it says that the time required for this problem, irrespective of any algorithm is this. So, it is not a bound on an algorithm. Well, if you want to think about algorithms, it is a bound on all possible algorithms, in this model of course.

Here is a quick overview of the proof, the proof is a little bit long, but not terribly so. The first idea is going to be to interpret our input sequence. So, we have our input consists of n numbers x 1, x 2 all the way till x n. So, we are going to think of this n tuple as representing, the coordinates of a n dimensional point.

So, x 1 represents the first coordinate, x 2 represents the second coordinate, x n represents the nth coordinate. And the entire thing represents a point and will call that point x. We are going to restrict these instances to the unit n cube. What do I mean by that? Well, we are going to insist that all the x i's lie between 0 and 1. We will prove the lower bound under this restriction.

So, that should not be a cause of any worry. If we prove the lower bound under this restriction of course, it works if we do not have this restriction. So in fact, if we do not have this restriction, who knows things can get even much worse. But, we are not

worried about that, we just want to argue that things can certainly get at least as bad as this. And so it is to have this restriction.

Here is the first claim, that we are going to make. We will use the notation R of L, which stands for the region for L. So, the region of L, where L is a leaf is a set of all input instances for which leaf L is reached at the end of execution. So, this is the definition of what R of L means, what we are asserting is that this R of L is connected. I will explain to you what connected means in just a minute. But, it is the usual notion.

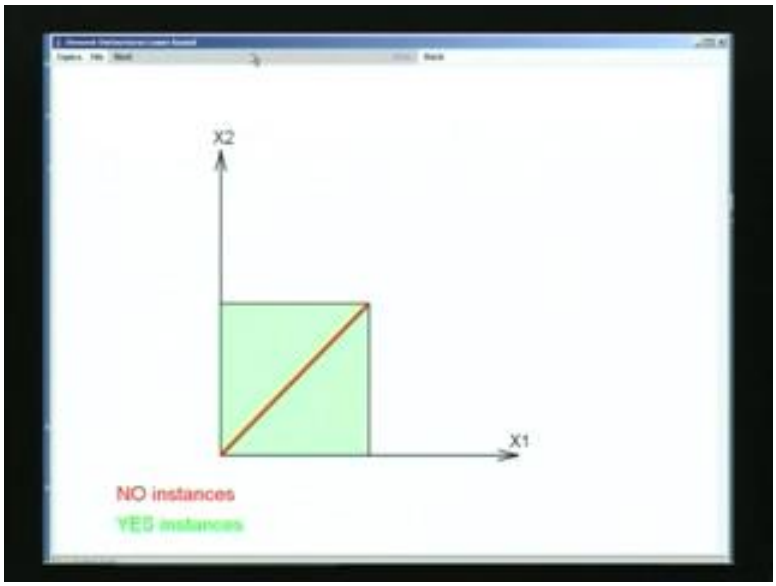
So, a region is connected, if it looks together or if there are two points in it. And they can be joined by a path, within that region. We will do this a little bit more formally and we will write it in a minute. The second claim goes something like this. Suppose we have a point x whose coordinates are distinct. So, the sequence consist of distinct elements or alternately these coordinates are distinct.

Now, we take y which is the non trivial permutation of these coordinates. So, here is this coordinates, pre reorder than. Such that this sequence and this sequence look different. That is what x and y are. So, the main claim is and this is the key claim in this entire proof x and y must reach distinct yes leaves. So, the point is this, that this will allow us to argue, that there will be at least two distinct leaves or who knows more.

And then, we will argue in fact, based on this essentially, that the number of yes leaves is bigger than n factorial. The time is simply going to be the height of such a tree. And it is going to be log of n factorial or n log n. So, this will immediately follow from this, we will see that later. So, let me go over each of these items. And let me explain each of these items.

So, let me start with this, we are going to interpret, this x as a point in n dimensional space. And of course, we are restricting the points to unit n cube. As I said, it is hard to visualize n dimensional space. So, we are just going to leave it two, we will visualize two dimensional space and we will say what happens? And you must note that, even if we work with two dimensions, we will get enough of the insight.

(Refer Slide Time: 18:16)



So, here is our visualization. So, our instance is this x that first coordinate is x 1, second coordinate x 2. So, this is our x 1 axis, this is our x 2 axis. Our instances come from the unit cube. So, here is the unit cube in two dimension, this entire interior is a unit cube. So, this is one, this is one, this is where our instances come from, what I mean by that is I pick a point over here. It is first coordinate is the x 1 value, it is second coordinate is the y 1 value.

So, let me continue the analogy of little bit further to show you, what we make sure we understand at least this case of two dimensions very well. I claim that all the instances which lie on this diagonal are no instances for this problem of size two. Well, what are the instances which lie on this diagonal. So, the instances on the diagonal simply are those points whose x 1 coordinate is exactly equal to the x 2 coordinate.

But, if the x 1 coordinate is equal to the x 2 coordinate, then we know that they may n dimension distinct. And therefore, these points on the diagonal are in fact, the no instances. The interior of this triangle and the interior of this triangle are the yes instances. Why, because if I pick a point in it, I know that the x and y coordinates cannot be equal, the x and y well the x 1 and x 2 coordinates cannot be equal.

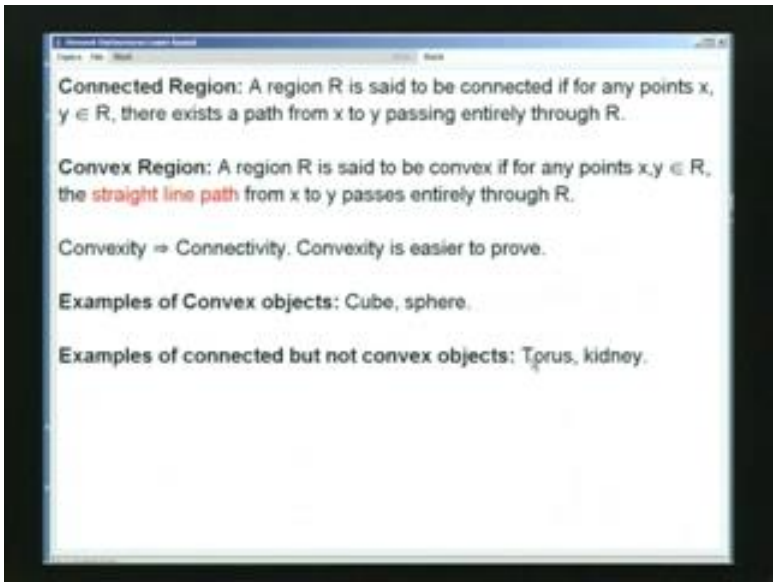
Similarly, over here this is the line which divides the square into two parts. And this is the line on which the x 1 and x 2 coordinates are in fact, equal. So, what we have done over here is, we have taken our problem. And we are viewing it geometrically and you

will see, that this geometric view point gives some interesting insights ((Refer Time: 20:25)). So, we have finished interpreting these instances in a two dimensional space or in particular in an n dimensional space.

So, we have finished this part. We interpreted our input instance, as a point in n dimensional space. Well, in this case it was just two dimensional space. But, you should get the idea and we also restricted the instance to be in n cube, which in this case was just the unit square. Now, you want to prove this claim. So, this and this are the two main claims.

This claim says, that if we look at the region of this cube from which instances will reach L. Then, this region is going to be a connected means. So, let me start by defining what connected means.

(Refer Slide Time 21:21)



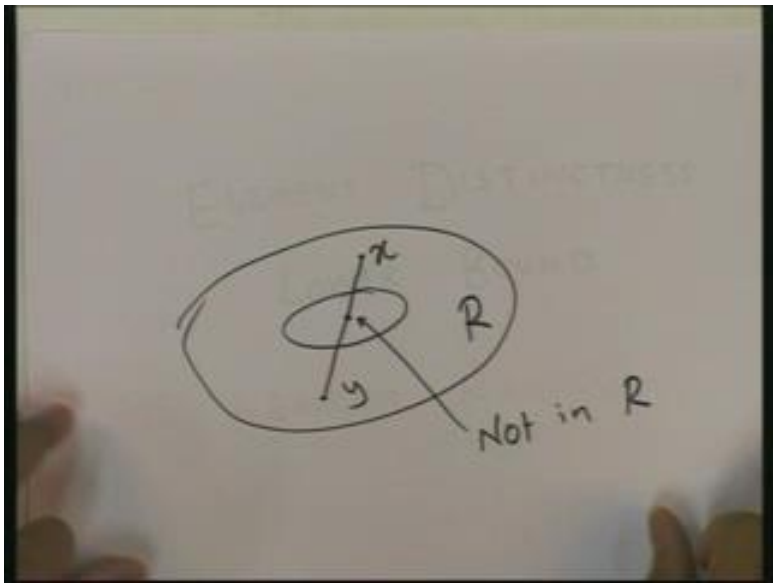
A connected region in n dimensional space. So is as follows A region R is said to be connected. If for any points x or y in R, there exists a path from x to y passing entirely through R. So, we have any points x and y in R and there exists a path from x to y, which goes only through the points to R. So, what is a connected region? So, the cube that we mentioned for example, is a connected region the interior of it. Well the surface of it is also connected region.

So, let me now define a convex region. So, this is going to be needed in the proof. A region R is said to be convex, if for any points x and y in that region. The straight line path from x to y passes entirely through R. So, here we just said, that the any path that some path passing from x to y, must pass entirely through R. Now, we are making the stronger requirement. So, we are now saying, that in particular the straight line path from x to y must pass through R.

Notice, that convexity is only a special case of connectivity. So, in other words, if we know that a certain region is convex, then it has to be connected of course. So, the reason why we worry about convexity is that, if we look at only straight line paths, they are very easy to reason with. And therefore, its often much easier to argue, that is a certain region is convex. So, convexity of a region is easier to prove. And therefore, we are worried about convex regions.

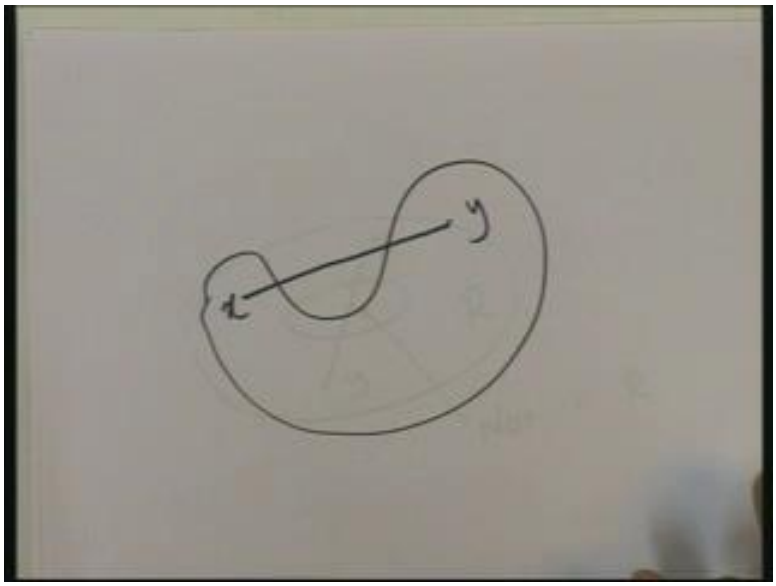
But, notice that if we prove that something is convex, we are also proving that it is connected. So, some examples of convex objects are the cube, the fill the whole cube or the whole sphere. Examples of objects which are connected, but not convex say something like a torus. So, let me draw a picture.

(Refer Slide Time 23:53)



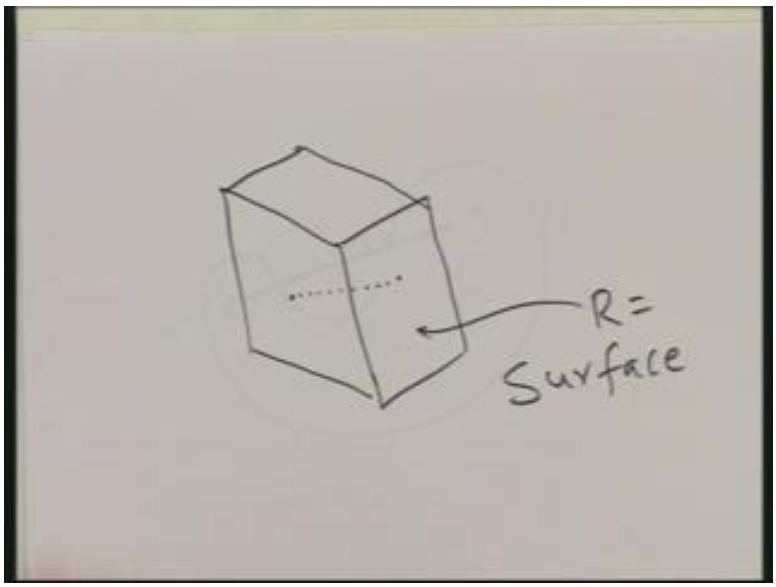
So, if you have a torus, so it has a hole in it. So, if I pick a point x over here and a point y over here. Then, the line joining the straight line joining them would pass through this region, which is not in R. So, this torus is itself in R, but this region is not is R.

(Refer Slide Time: 24:17)



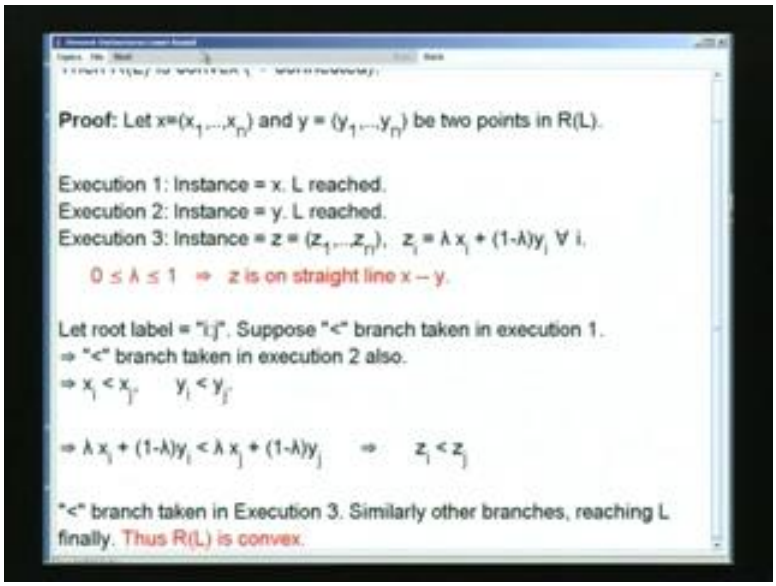
A kidney shaped region or a cashew shaped region is also not convex. Because, I can take a point x over here, a point y over here and this line passes outside the region.

(Refer Slide Time: 24:33)



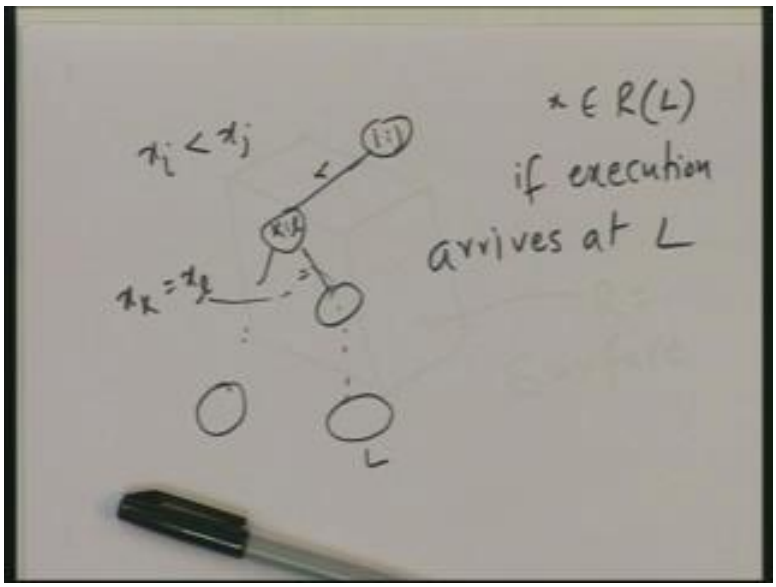
If I look at a cube and if I just look at it is surface not the interior. But, if I just look at it is surface, if I take a point over here and a point over here and this phase. Then, the line joining them has the straight line joining them, will pass through the interior, which is not in this region R. And therefore, this shell is not convex either. So, that describes what connected means ((Refer Time: 25:14)), formally and also what convex means.

(Refer Slide Time: 25:17)



So, here is our claim. So, we claim that R of L is the set of instances for which leaf L is reached on execution. And then, we want to argue that R of L is convex. Well, actually we wanted to argue, that it is connected. But, we will in fact, argue that R of L is convex, which will assure that it is also connected. Let me pictorially remind you, what this R of L is...

(Refer Slide Time: 25:53)



So, here is our decision tree and this is leaf L. So, what is this region that I am talking about? Well, if I start with any instance x and suppose I follow this path and I reach L.

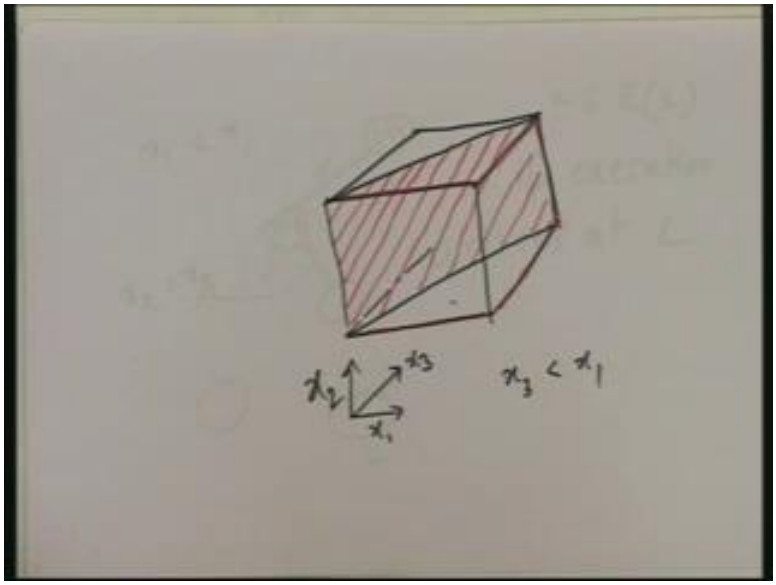
Then, I will say that this instance belongs to R, if execution arrives at L. So, then it belongs to R of L in the region of L. So, let me first begin by giving some intuition. So, I said that x is a set of points, such that if I start from here and I get eventually I get to L.

So, what do I know about x? Well, each node that is visited has some condition associated with it. So, may be here the comparison is between i and j. And say this is the less than path, then these two together say that, this x must satisfy x i less than x j. May be this label over here is k l and say this is the equal to path. Then, this says that the condition satisfied must be x k equal to x l. So, if any instance gets to this level L, I know that all of these conditions along this path must be satisfied.

So, this is one way to characterize the region, the set of points which reach L during execution. But, notice that this characterization is geometric. So, this just says that the ith coordinate is smaller than the jth coordinate. So, this is naturally putting our region R of L into some parts in our unit cube. So, let us start with this root itself. So, which are the instances, which can visit the root, well at the root any instance will arrive.

So in fact, any x in the entire unit cube, will arrive at the root, will start at the root. So, instances visiting the root constitute the entire cube. What about instances visiting this node? So, this node is visited by those instances, for which x i is less than x j. Now, here is the key insight.

(Refer Slide Time: 28:44)

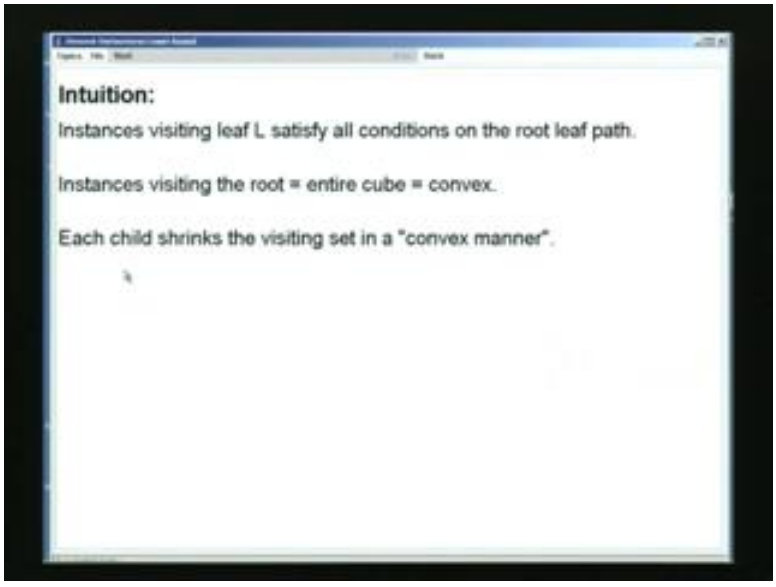


So, asserting that x i is less than x j is equivalent to saying that, if this is our unit cube. Now, I am looking at three dimensions and may be this is some x 1, this is x 2, this is x 3. And if I say, that x 3 is say less than x 1 what do I do? Well, I will look at this x 3 and I look at this x 1. And then I look at first, that portion where x 3 is equal to x 1. So, it is this, it is this plane let me just shade it.

So, it is this slice through the centre of that cub. That is, where x 3 is equal to x 1 and if I want x 3 to be smaller. Then, which side should I take then x 1. So, I want x 1 to be larger and x 3 top be smaller. So in fact, this is the entire region. So in fact, it is this region, the wedge shaped region that is facing us. So, the idea is this, the moment I assert a condition, I am going to slice my current set.

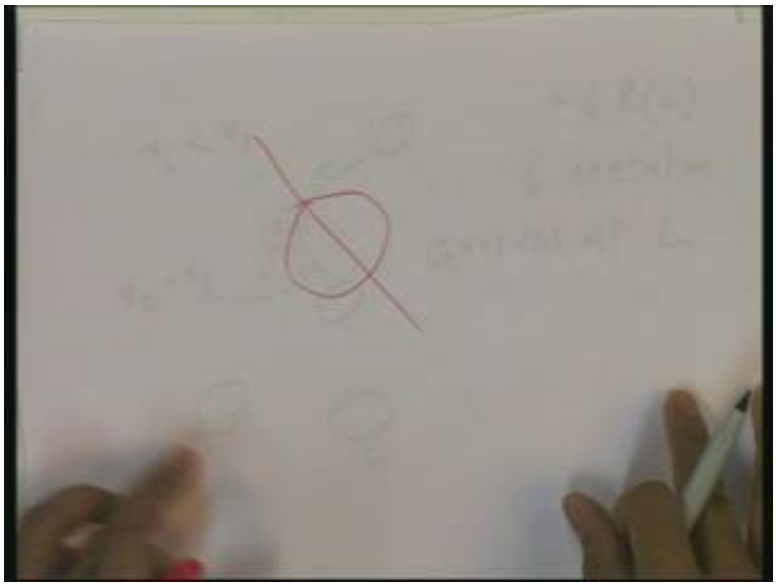
And I am going to take one part of it, if my assertion was something like x 3 equal to x 1 then I won not take one part, but I will take that slicing region itself. So, notice that I started off with the entire cube which is convex.

(Refer Slide Time: 30:18)



And the important point is that, whenever I go to a child as in this. I am going to shrink the set of instances, which visit this ((Refer Time: 30:28)). And when I shrink them, I will be shrinking them in a convex manner.

(Refer Slide Time: 30:36)



So, it is sort of I will take my region I will take a region, which is convex then I will slice of a part of it. But, this slicing operation maintains convexity. So, that is roughly the idea. So, even if I do it several times, the region that I am left with at the end is going to be a convex region. And therefore, also connected region, that is roughly the argument. So, now we are going to see it more formally ((Refer Time: 31:05)).

So, here is the proof, suppose x and y are two points in R in this region. So, I am going to consider three execution. In the first execution the instance is going to be x, what do I know about this? I know that L is reached by definition, by our assumption that x and y are points in R of L. So, when I finish this execution I know that L is reached.

In execution 2 I am going to start with y. What do I know about this? Well, I know that for this point as well L is reached. So, even for this point L is reached right, again that is because I said that y belongs to R of L which is nothing but saying if I do an execution with instance my instance equal to y I will reach that same leaf.

My third execution is the interesting execution. Here, I am going to start off with an instance which I will call z. Z let me remind you has n coordinates, just like x and y. So, I will call those z 1, z 2 all the way till z n. And I am going to set z i in a curious looking manner. Z i is going to be lambda times x i, where lambda is some positive number between 0 and 1. I will tell you more I will write that down in a minute, and lambda times x i plus 1 minus lambda times y i.

So, this is how each z i is going to be set. So, it will be some kind of an average of x i and y i, where the weights for x i and lambda. And the remaining weights come from y. So, if I take the case lambda equal to 0 what does it mean. So, if I take lambda is equal to 0, then this part goes away and I get y. If I get lambda equal to 1, this part goes away and I get x.

And if lambda is somewhere in between, what do I get? If lambda is equal to half, then I get half of this and half of this. And in fact, I get the midpoint of line segment x, y. If I take other values of lambda I will likewise get, points on the line segment joining x and y, the straight line segment line joining x and y. So, this is the key behind, this is the key part of the definition set. I am going to I have defined z, so that it happens to be on this straight line x, y.

And so long as I restrict lambda between 0 and 1, it will be in the interior of this line segment x, y. So, what do I have to prove in order to I give that, this R of L is convex. Well, the definition says that straight line path must lie inside of R of L. If I prove that then I am done, that is exactly what I am going to do. So, I am going to analyze this execution 3 and figure out what happens during execution.

So, let us start with root, the root label is i colon j. And suppose, the less than branch is taken in execution 1. I am just taking this as an example, the argument will really work for every possible branch. So, the less than branch is taken in this execution. In this execution, what do I know about this execution? Well, clearly the same branch will be taken in execution 2.

Why? Well, in execution 2 we reach final finally, the same leaf. And there is only one way to get at that leaf. And therefore, it had better be along the same path. So, even in this second execution, we are going to follow the same branch. What can we conclude from that? So, from the fact that in the first execution, this branch was taken. It clearly means, that x i and x j got compared and x i turned out to be less than x j. In the second execution y i and y j got compared. And y i turned out to be less than y j.

So, this is what we know, if we assume that the less than branch was taken in this execution. Now, I am going to multiply this by lambda and this by 1 minus lambda and I am going to add these two things. So, let us see what happens. So, I claim that I get this inequality, let us check that out. From the left hand side I am going to get from this

inequality lambda times x i, which I have got over here. From this I will get 1 minus lambda times y i, this is what I have got over here.

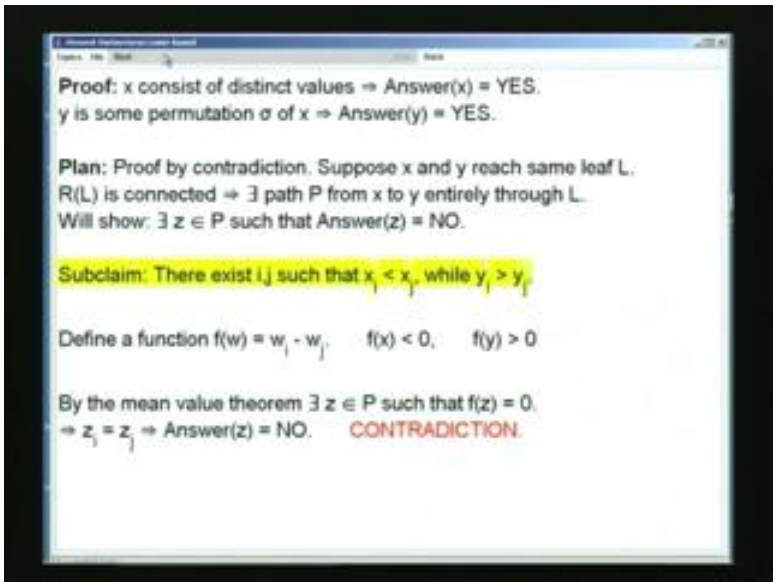
On the right side I got lambda times x j from this inequality and from this inequality I got 1 minus lambda times y j. So, what we have now is this inequality. So, this is the inequality that we got, let me just complete this argument. So, to complete this argument what we have is, that this part is simply z i and this part is simply z j, so this is z i this is z j. So, we have concluded, that z i must be less than z j.

But, that is what we wanted, why? Because, if z i is less than z j, we know that the less than branch will also be taken in execution 3. So, for this first root we have proved that execution 3 will follow the same path, as that followed by execution 1 and execution 2. So, this argument can be made if instead of less ((Refer Time: 37:38)) than we took some of the other.

And we can make it at every node along that path. And so we can argue that finally, L is going to be reached. So, what we have argued is that, z will reach L z is any point on this line segment. And so if I start with any point on this line segment I reach L. And therefore, I have concluded that R of L is convex. So, I have proved this claim. So, let us now turn to the next claim.

So, this claim says that ((Refer Time: 38:20)) if I have a point or an instance x, whose all coordinates are distinct. And y is another non is a permutation on this, which is not exactly equal. Then, x and y must reach distinct leaves. Of course, they must reach yes leaves, because their coordinates are all different. But in fact the claim asserts that they must reach distinct yes leaves.

(Refer Slide Time 38:50)



So, you are going to prove this. So, x consists of distinct values, means the answer of x must be yes, y is some permutation sigma. So, the answer to y is also yes. We are going to prove this result by contradiction. So, we are going to assume that say x and y reach the same leaf L. Then, we know that every the region corresponding to every leaf L is a connected region.

If it is a connected region, then there has to exist a path p from x to y, which passes entirely through L. This is what we know from the previous claim. So, what we will show, that if you tell me that it is this path I will show, that there exists a point z on this path p, such that the answer to z is no. Now, this will be a contradiction, because z lies on P, P supposed to lie inside R of L.

And what we have argued is that, the answer is no whereas, the answer for L is supposed to be yes. So, this would be the contradiction. So, this is what we are going to do? Let me start with the sub claim. The sub claim says that there have to exist i and j. Such that, the ith coordinate of x is strictly less than the jth coordinate of x, whereas the ith coordinate of y is bigger than the jth coordinate of y. I will prove this in a minute. But, let me just examine it is implications.

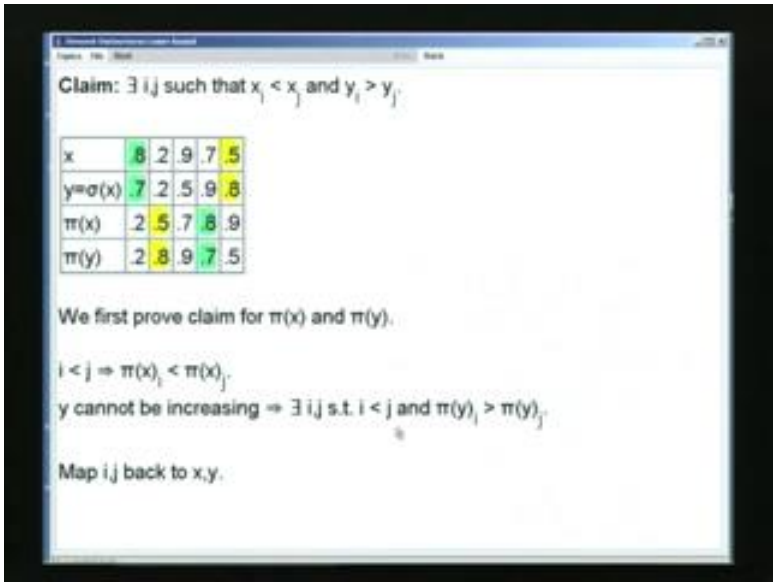
So in fact, this is going to give us the proof almost immediately. So, let me define a function f in this space, where f for a point w is simply the difference between the ith and the jth coordinates. So, what is f of x, f of x is x i minus x j, but x i is smaller than x j. So,

f of x is less than 0, what is f of y? Well, y i is bigger than y j. So, y i minus y j is bigger than 0 and f y is bigger than 0.

So, f of x is less than 0, f of y is bigger than 0, they are joined by this continuous path p. So, what happens is we move along this path from x to y. The path is continuous, this is a continuous function. And so, by the mean value theorem, there has to exist a point z on p such that f of z is 0. If f of z is equal to 0, what does it mean? This means that z i equal to z j, but z i equal to z j, then two coordinates are equal. That means, the answer to z is no.

The coordinates are not distinct. So, the answer is a no, but this supposed to be a point inside R of L. So, the answer had better been yes, so there is a contradiction. So, we have proved our basic claim. So, all that remains now is to prove this sub claim. So, let us prove that.

(Refer Slide Time 42:32)



So, the claim is that there exist i, j such that x i is less than x j and y i is less than y j. So, I am going to do this with an example to help you understand what is going on. So, here is my example. So, I am looking at say five dimensional space, this is my x, the coordinates are all points in the unit cube. This is why remember that y is just a permutation of this permutation sigma. And y is such that, it is not identical. So, the same numbers as x is repeated over here.

But, it is not repeated they are not repeated identically. Well, they can be identical at one place that is. Now, here is the key step, I am going to define a permutation pi which sorts x. So, pi of x then is going to simply take these and rearrange them in increasing order. So, it is going to 0.2, 0.5, 0.7 and so on. It may not be clear to you, why I am not defining this permutation pi. But, please bear with me, the answer will hit you in a minute.

So, I know what pi is I know how to take x and generate pi of x from it. I apply the same permutation on y as well. So, what happens well I look at the column, the column of x which moved over here. And I take the corresponding value for y and move it down over here. So, it is both are 0.2 over here, so, it is both are 0.2 over here. Then, 0.5 came from here. So, the value over here must also come over here 0.7 came from here. So, below it is 0.9, so over here also there is 0.9, then we have 0.8, 0.7 from this and 0.8, 0.9, 0.5 from here.

So, we have rearranged x and we have rearranged y. Now, be patient with me just for a minute, I am going to prove this claim. But, I am going to prove it for pi x and pi y, where it is easy to see. And you will see that, they can just trace it backwards to x and y. So, what is it mean to prove the claim for pi x and pi y? Well, we are supposed to find i j of this property.

So, because we sorted and here is now the reason for sorting it, if i is less than j we know that pi of x i has to be less than pi of x j, because this is sorted order. So, pi of x sub i the ith component of pi of x must be smaller than the jth component of pi of x. So, this is smaller than this, this is smaller than this and so on. So, long as I choose i smaller than j, this first property is guaranteed to me. What do I know about y? So, here is one sequence I change that and then permuted.

So, I know now that somewhere, this new sequence has to be non increasing. This was in increasing sequence, this is a permutation of it. So, somewhere this sequence has to be non increasing. So, let us say that there exist i j, such that i less than j and if it is non increasing. Then; that means, pi y must be greater than pi y j. But, notice that then we have found these two.

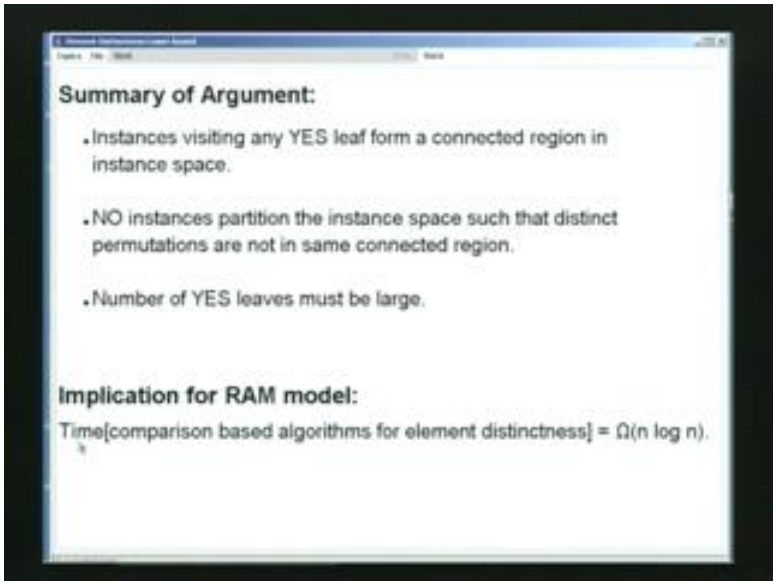
So, we have found i j such that, pi y is bigger than pi y j, whereas pi x i is bigger than pi x j. So, just to illustrate in this example. Here is the case, where this is smaller than this,

but this is larger than this. Now, we have proved it for this pi x and pi y. How do we go back? Well, the idea is simply we take we figure out, where these columns came from our original example.

This column came from here and this column came from here. And sure enough, this number is less than this whereas, this number is greater than this. So, I will skip the algebra. But, this is exactly what is happening? We just have to follow back and then this property will hold nevertheless ((Refer Time: 46:40)). So, what have we proved, well we have proved this claim. And this was all that was needed to prove our original claim, which was this ((Refer Time: 46:48)).

So, once we prove this claim what do we know that if we have two distinct permutations of this. If we distinct permutation of this, then that must reach a different region. But, now I know that the number of distinct permutation can be n factorial. And therefore, the number of yes leaves has to be bigger than n factorial. And the time has to be bigger than the height of the tree, which is log of n factorial or at least n log n. So, this finishes the claim that the time for element distinctness is at least n log n on decision trees.

(Refer Slide Time 47:30)

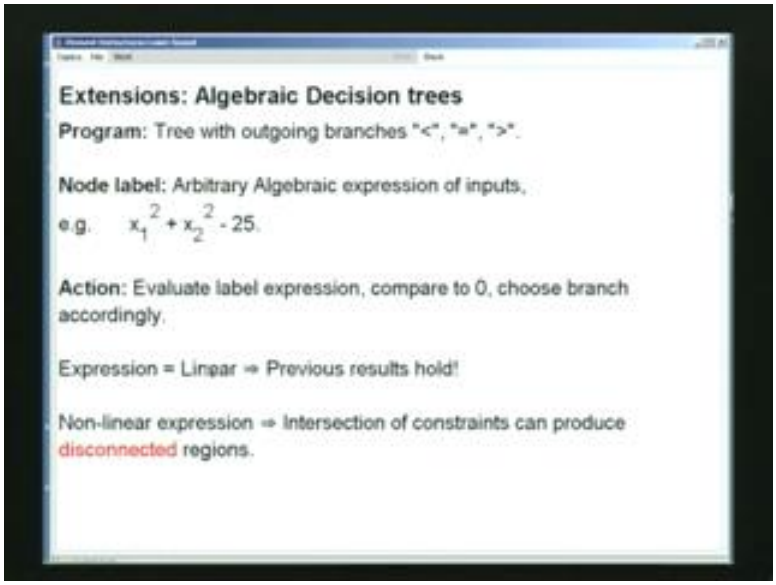


Here is the quick summary of the argument. So, what we did here was, that we said that the instances visiting any yes leaf, former connected region in the instance space. No instances partition the instance space, such that distinct permutations are not in the same

connected region. And therefore, we conclude that the number of yes leaves must be large.

What is the implication for the RAM model? Well, for the RAM model we can conclude exactly using ideas similar to last time, that the comparison based algorithms for element distinctness, must take time n log n.

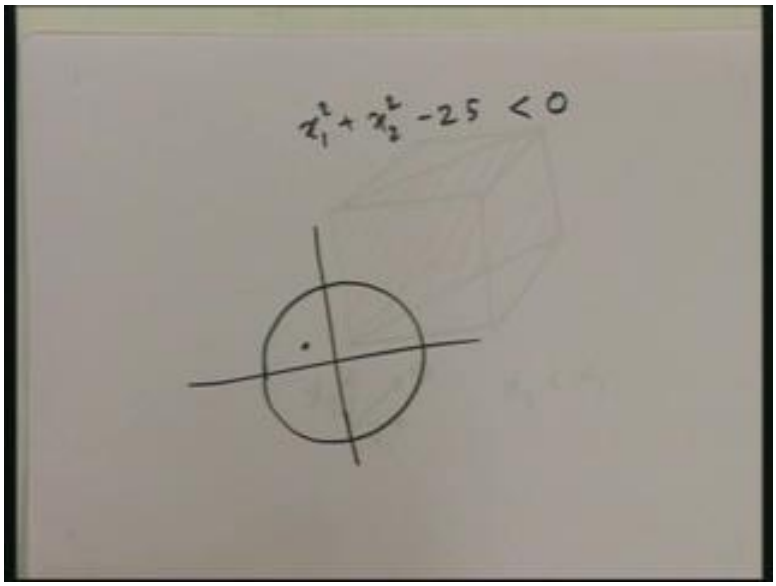
(Refer Slide Time 48:13)



I want to quickly extend this to the case of algebraic decision trees. So, here is a quick definition. So, again the program in this model consists of trees with outgoing branches, less than equal to or greater than. But, this time we will have these three relational operators, just for simplicity, although you can have other operators too. The node labels are no longer i colon j, but they are algebraic expressions, over the input the components of the input. So, x 1 square plus x 2 square minus 25.

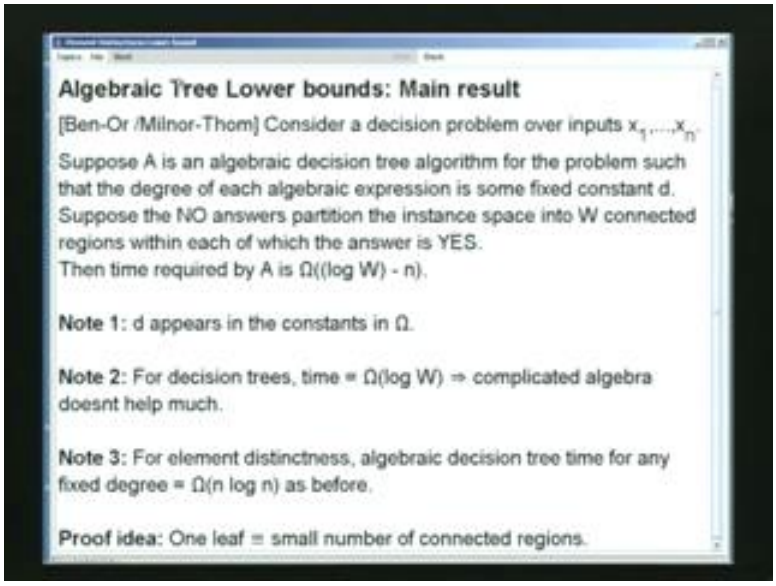
The action is we are going to evaluate the label expression. And we are going to compare it to 0. So, this expression is equal to 0, then we will choose the equal to branch. If this is less than 0, then we will choose the less than branch. So, let me just take a quick example. So, if our expression is x 1 square plus x 2 square is 25 or minus 25.

(Refer Slide Time: 49:18)



Then, the condition x 1 square plus x 2 square minus 25 is less than 0. Simply means, that our point lies inside this. So, we are restricting our point to be inside this region. If the expression is linear instead of this, which is a quadratic. Then, our previous results actually hold. Unfortunately, if the expression is non-linear ((Refer Time: 49:50)), then the intersection of constraints can produce disconnected regions, which cannot happen if things are linear.

(Refer Slide Time: 49:57)



The main result is something like this, this is a deep result actually from algebraic geometry. So, we have a decision problem over inputs x 1 to x n, by a decision problem we simply mean, that we mean a problem whose answer is yes or no, just like element distinctness. Suppose A is a algebraic decision tree algorithm for the problem. Such that, the degree of each algebraic expression is some fixed constant d. So, x 1 square plus x 2 square would be degree 2.

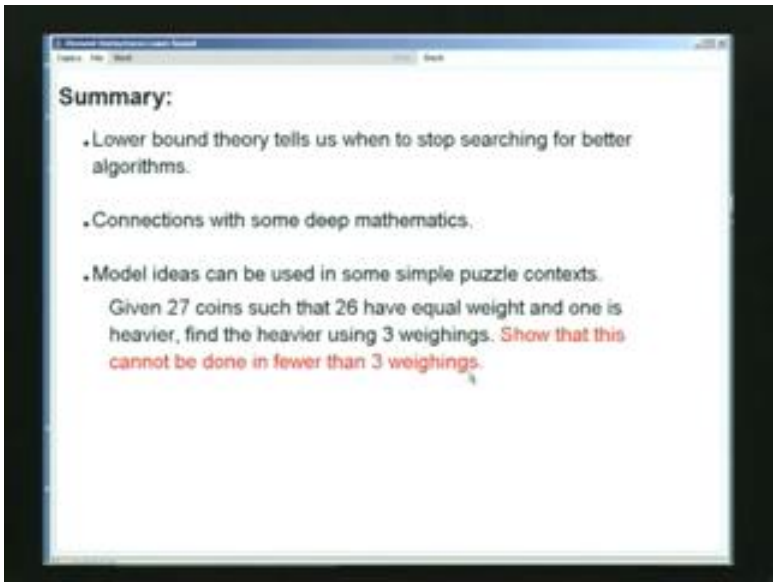
Suppose, the no answers partition the instance space into w connected regions, within each of which the answer is yes. So, in our case for simple decision trees, this w for the element distinctness was n log n. But, in general the time required is going to be by this algorithm is going to be omega of log of W minus n. Now, you might wonder what happened to that d. So, this d actually appears inside this omega. So, there is a constant of proportionality, which depends upon d.

For decision trees the time is log of W, not even omega it is actually just log of W, a log of W to the base 2. So, what this theorem says, that the complicated algebraic model does not really help all that much. So, the lower bound that we get is almost as good. Well, it is a little bit smaller, because of this minus n and may be the proportionality constant is a little bit different, but it is essentially the same lower bound.

For element distinctness in fact, there is no change, the algebraic tree for any fixed degree will give us n log n as before. The proof idea is actually pretty difficult. Now, as we said a single leaf can correspond to a small number of connected regions, not just exactly one connected region.

So, now we have to get some heavy duty machinery from algebraic geometry to count the number of connected regions. When that you get when you take intersection of several constraints. If the constraints are linear, then it is very simple. If the constraints are high degree algebraic expressions, then this becomes rather complicated.

(Refer Slide Time 52:30)



So, quickly summarize lower bound theory, that we have been looking at in the last two lectures, tells us when to start searching for better algorithms. This is very good, because it is good to know that you are done. Another interesting point of this theory is that it has some connections, with some really deep mathematics. Algebraic geometry is supposed to be rather deep area of mathematics. And it has connections to many other fields also.

Here is a very simple context, in which this idea can be used. So, I will leave it as a problem for you. Suppose, you have 27 coins, such that 26 have equal weight. And one is heavier, find the heavier using 3 weighings. This is probably we have puzzle that we have solved. Now, I want you to use the ideas expressed in this lecture to formulate a decision tree model using which, you should be able to argue, that you cannot do this in fewer than 3 weighings.

Thank you.