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: you are given an input—a sequence of numbers. Let me call the entire sequence \(x\) and the individual numbers in the sequence are \(x_1, x_2, \ldots, x_n\).

We are supposed to output a yes answer if all the \(x_i\) are distinct and otherwise, if some \(x_i\) is equal to \(x_j\) for some two numbers \(x_i\) and \(x_j\) are identical, then we are supposed to output 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 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 (Refer Slide Time 3:00).
how well $n \log n$ 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 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 its going to be very similar to what we did in the 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 doesn’t 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 okay 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 4:01)

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 one 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 these inputs are already read a program in this model is a label tree and all non leaf nodes are labeled i colon j 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

let me quickly take an example of decision tree program 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 colon two and its each leaf node is labeled by the answer that is to be output and furthermore each edge is labeled with a relational operator
execution begins at the root okay at each node labeled i colon j x i from this set of inputs is compared with x j what ever the outcome of that comparison is the corresponding branch is found with the corresponding label (Refer Slide Time 6:01)

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 its label is the output so that’s 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 one for n equal to two n equal to three and so on and we saw a program for size n equal to three 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 okay
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 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 \( x_1, x_2, \ldots, x_n \) 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 \( n! \) 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 \( n! \) permutations and therefore there are \( n! \) possible answers this is a very important point in this argument now the answer is going to be printed at the leaf of the tree of the program tree which means that if you want to print out \( n! \) different answers then you must have \( n! \) 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 the leaf then that’s the only answer it can print out so if your program is going to be even capable of printing \( n! \) different answers then it had better have \( n! \) leaves but if it does have \( n! \) leaves then the height of the tree has to be at least \( \log n! \) which is \( n \log n \) and in fact the height of the tree is the worst case time and therefore the worst case time is at least \( n \log n \) so this is a rough sketch of the argument that we saw last time
this idea 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 (Refer Slide Time 11:01)

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 information 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 its going to be log of \( n! \) factorial okay very roughly in any case bounds of this case where we say that there are so many ways in which this variable can take this value are therefore called information theoretical 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’s going to be in fact exactly one okay
so if we just say that there are two possibilities and therefore there are two leaves that
doesn’t help us much
because log of two is one and it just says that the tree must have height one which is
really a silly bound
so we need to do something better (Refer Slide Time 12:00)

so we need to have a new strategy 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
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 an interesting argument and we will be making some rather
clever use of n dimensional geometry
okay don’t 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
so here is our main claim again we are going to prove that the time required for the element distinctness problem is going to 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 its not a bound on this algorithm well if you want to think about algorithms it’s 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
okay the first idea is going to be to interpret our input sequence
okay so we have our input consists of $n$ numbers $x_1$ one $x_2$ two 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 $n$th 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 $x_i$ lie between zero and one
we will prove lower bound under this restriction
so that should be any cause of worry if we prove the lower bound under this restriction of course it works
if we don’t have this restriction so in fact if we don’t have this restriction who knows things can get even worse but we are not worried about that
we just want to argue that things can certainly as bad as this and so its okay to have this restriction
here is the first claim that we are going to make we will use the notation $r(l)$ 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 on execution at the end of execution
okay 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’s 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
this suppose we have a point x whose coordinates are distinct okay 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 we reorder them such that this sequence and this sequence look different
that’s 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 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 in fact bigger
the time is simply going to be the height of such a tree okay and it’s going to be log of n factorial or n log n (Refer Slide Time 17:40)
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 is but we are going to interpret this as a point in n dimensional space and of course we are restricting the points to unit n cube
its 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
so here is our visualization so our instance is this x that first coordinate is x one second coordinate x two
so this is our x one axis this is our x two 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 okay this is where our instances come from what i mean by that is i pick a point over here it’s first coordinate is the x one value second coordinate is the y 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 are simply those points whose x one coordinate is exactly equal to the x two coordinate
but if the x one coordinate is equal to the x two coordinate then we know they may not be 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 sorry x and y well the x one and x two 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 one and x two 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 gives some interesting insights
okay so we have finished interpreting these instances in a two dimensional space or in particular in a n dimensional space

okay so we have finished this part

we interpreted our input instance as 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 in n cube which in this case was the unit square

now you want to prove this claim okay 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 region

so let me start by defining what connected means

a connected region in n dimensional space okay is as follows a region r is said to be connected if for any points x or y in r

okay 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 (Refer Slide Time 23:51)
so let me now define a convex region

okay 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 in a straight line path from $x$ to $y$ passes entirely through $r$

so here we just said that any path that some path passing from $x$ to $y$ must pass entirely through $r$

now we are making the stronger requirement so now we are 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 it's often much easier to argue that 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 something is convex we are also proving that it is connected

so some examples examples of convex objects are the cube fill the whole cube or the whole sphere examples of objects which are connected but not convex say something like a torus (Refer Slide Time 25:07)
so let me draw a picture 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 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 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.
if i look at a cube and if i just look at its surface okay not the interior but if i look at its surface if i take a point over here and a point over here and then this phase 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.
okay so that that describes what connected means formally and also what convex means.
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 argue r of l is convex.
well actually we wanted to argue that it is connected 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 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 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 with intuition so i said that x is a set of points such that i start t 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 I know that all of these conditions along this path must be satisfied (Refer Slide Time 28:21)

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 i th coordinate is smaller than the j th coordinate

so this is naturally putting our region 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 okay so in fact any x in the entire unit cube will arrive at the root will start at 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 so asserting that x i is less than x j is equivalent to saying that this is a unit cube

okay now I am looking at three dimensions and may be this is some e x one this is x two this is x three and if I say that x three is say less than x one what do I do well I will look at this x three and I look at this x one and then I look at first that portion where x three is equal to x one
so it is this so it is this plane let me just shade it so it is this slice through the centre of the cube 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 okay so in fact it is this region the wedge shaped region that is facing us okay 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 wont take one part but i will take that slicing region itself so notice that i started with the entire cube which is convex and the important point is that when ever i go to a child i am going to shrink the set of instances which visit this and when i shrink them i will be shrinking in a convex manner so it is sort of i will take my region i will take a region which is convex and then i will slice of a part of it but this slicing operation maintains convexity so that’s roughly the idea so even if i do it several times the region that i left with at the end is going to be a convex region and therefore also connected region that’s roughly the argument so now we are going to see it more formally 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 \( x \) okay 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 \( l \) is reached in execution two i am going to start with \( y \) what do i know about this well i know that for this point as well i know that for this point as well \( l \) is reached so even for this point \( l \) is reached right again that’s because i said that \( y \) belongs to \( r \) of \( l \) which is nothing but say 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 of with an instance which i will call \( z \) 
\( z \) let me remind you has \( m \) 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 \( z_1 \) in a curious looking manner
$z_i$ is equal to $\lambda \times x_i$ where $\lambda$ is some positive number between zero and one.

I will tell you more I will write that down in a minute and $\lambda \times x_i + 1 - \lambda \times y_i$

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 $y_i$ okay where the weights for $x_i$ and $\lambda$ and the remaining weights come from $y_i$ (Refer Slide Time 33:05)

So if I take the case $\lambda$ equal to zero what does it mean so if I take $\lambda$ is equal to zero then this part goes away and I get $y_i$

If I get $\lambda$ equal to one this part goes away and I get $x_i$ 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 $x_i$ and $y_i$

If I take other values of $\lambda$ I will likewise get points on the line segment joining $x_i$ and $y_i$ the straight line segment line joining $x_i$ and $y_i$

So this is the key behind this is the key part of the definition set I am going to I have defined $z$ that happens to be in this straight line $x_i$ and $y_i$

Okay and so long as I restrict $\lambda$ between zero and one it will be in the interior of this line segment $x_i$ and $y_i$
so what do i have to prove in order to prove that this r of l is convex well the definition says that straight line path must lie in r of l
if i prove that then i am done that’s exactly what i am going to do so i am going to analyze this execution three and figure out what happens during execution so lets start with root the root label is i colon j and suppose the less than branch is taken in execution one
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 okay in this execution what do i know about this execution
well clearly the same branch will be taken in execution two why well in execution two 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 one minus lambda and i am going to add these (Refer Slide Time 37:02)
so let's see what happens so i claim that i get this inequality lets 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 one 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 this inequality i got 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 okay 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 three
so for this first root we have proved that execution three will follow the same path as that
followed by execution one and execution two
so this argument can be made if instead of less than we took some of the other and we
can make it at every node along the 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's now turn to the next claim so this claim says that if i have a point or an instance x whose all coordinates are distinct and y is another 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 38:47)

so you are going to prove this okay so x consists of distinct values means the answer of x must be yes y is some permutation sigma
okay 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 okay 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 \( p \) lies \( z \) lies on \( p \) supposed to lie inside \( l \) inside of \( l \)

okay 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 \( i \) th coordinate of \( x \) is strictly less than the \( j \) th coordinate of \( x \) whereas the \( i \) th coordinate of \( y \) is bigger than the \( j \) th coordinate of \( y \)

i will prove this in a minute but let me just examine its 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 \( w \) is simply the difference between the \( i \) th and the \( j \) th coordinates

so what is \( f \) of \( x \)? \( f \) of \( x \) is \( x_i - x_j \) but \( x_i \) is smaller than \( x_j \) so \( f \) of \( x \) is less than zero

what is \( f \) of \( y \)

well \( y_i \) is bigger than \( y_j \) so \( y_i - y_j \) is bigger than zero and \( f \) of \( y \) is bigger than zero

so \( f \) of \( x \) is less than zero \( f \) of \( y \) is bigger than zero (Refer Slide Time 41:38)
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 so by the mean value theorem there has to exist a point \( z \) on \( p \) such that \( f(z) = 0 \)

if \( f(z) = 0 \) what does it mean this means that \( z_i = z_j \) but \( z_i \neq z_j \) 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 \( l \)

okay 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 so the claim is there exist \( i, j \) such that \( x_i < x_j \) and \( y_i < y_j \)

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 \( y \) is just a permutation of this permutation \( \sigma \) and \( y \) is such that it’s not identical so the same numbers as \( x \) is repeated over here
but it’s not repeated they are not repeated identically well they can be identical at one place that is okay

now here is the key step i am going to define a permutation \( \pi \) which sorts \( x \)
so \( \pi(x) \) is then going to simply take these and rearrange them in increasing order
so it’s going to “point two” “point five” “point seven” 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 \( \pi(x) \) from it

i apply the same permutation on \( y \) itself 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 both are “point two” over here then “point five” came from here so the value over here must also come over here
“point seven” came from here below it is “point nine” so over here also there is “point nine” then we have “point eight” “point seven” from this and “point eight” “point nine” “point five” here
okay 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 trace it backwards to x and y
so what is it mean to prove the claim for pi x and pi y okay well we are supposed to find i j of this property
okay 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 (Refer Slide Time 44:57)
right because this is sorted order so pi of x i x sub i the i th component of pi of x must be smaller than the j th component of pi of x
so this is smaller than 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 permuted so i know now that some where 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's say there exist i j such that i
less than j if it is non increasing then that means \( p_i \) y must be greater than \( p_i \) y j
but notice that then we have found these two
so we have found i j such that \( p_i \) y is bigger than \( p_i \) y j whereas \( p_i \) x i is bigger than \( p_i \) 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 \( p_i \) x and \( p_i \) y
how do we go back (Refer Slide Time 46:14)

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 that this number is less than this whereas this number is greater than this
okay so i will skip the algebra but this is exactly what is happening we just have to follow
that and this property will hold nevertheless
so what have we proved well we have proved this claim and this was all that we needed
to prove our original claim which was this
so once we prove this claim what do we know 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 permutations 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 one factorial or at least $n \log n$.

so this finishes the claim (Refer Slide Time 47:19)

so this finishes the claim that the time for element distinctness is at least $n \log n$ on decision trees.

here is the quick summary of the argument so what we did here was that the instances visiting any yes leaf former connected region in the instance space (Refer Slide Time 48:01)
no instances partition the instance space such that distinct permutations are not in the same connected region and therefore we conclude that 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 $n \log n$
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 okay over the components of the input so $x$ one square plus $x$ two square minus twenty five the action we are going to evaluate the label expression and we are going to compare it to zero so this expression is equal to zero then we will choose the equal to branch if this is less than zero then we will choose the less than branch so let me just take a quick example
so if our expression is \( x_1^2 + x_2^2 \) twenty five or minus twenty five then the condition \( x_1^2 + x_2^2 - 25 \) is less than zero simply means that our point lies inside this (Refer Slide Time 49:54)

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 then the intersection of constraints produce disconnected regions which cannot happen if things are linear

the main result is something like 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 \( x_1^2 + x_2^2 \) would be degree two suppose the no answers partition the instance space into connected regions within each of which the answer is yes

so in our case for simple decision trees this was \( n \log n \)
but in general the time required is going to be by this algorithm is going to be \( \Omega(\log w - n) \)

now you might wonder what happened to that \( d \) so this \( d \) actually appears inside the \( \Omega \)

so there is a constant of proportionality which depends upon \( d \) so decision time for decision trees the time is \( \log w \) not even \( \Omega \) it is actually just \( \log w \) to the base two

so what this theorem says that the complicated algebraic model doesn’t really help all that much okay so the lower bound that we get is almost as good it is because of this minus \( n \) and may be the proportionality proportionality constant is a little bit different but its 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 (Refer Slide Time 52:02)

the proof idea is 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 a heavy duty machinery from algebraic geometry to count the number of connected regions when that you get when you take intersection of several constraints are linear then its very simple
if the constraints are high degree algebraic expressions then this becomes rather complicated
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 its good to know that you are done
another interesting point of this theory is that it has some connections with some 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 twenty seven coins such that twenty six have equal weight and one is heavier find the heavier using three weighings
probably the puzzle 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 three weighings thank you (Refer Slide Time 53:28)