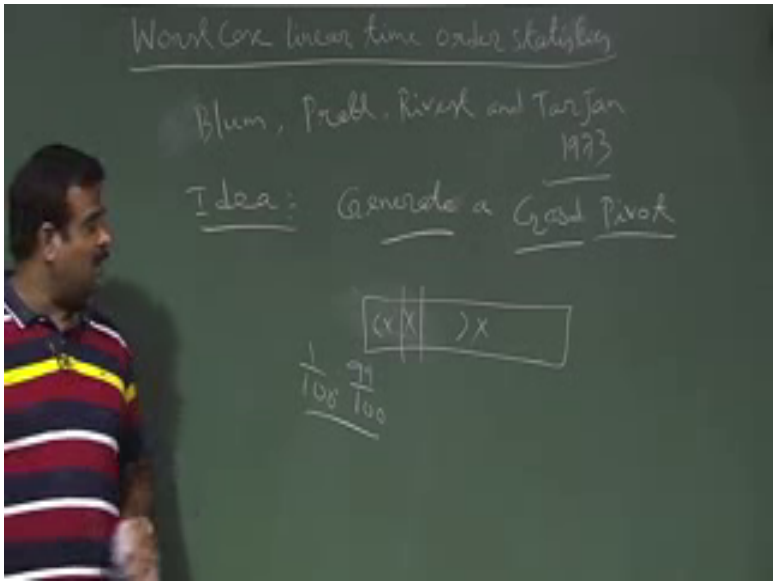
An Introduction to Algorithms
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture – 20
Worst Case Linear Time Order

So we have seen the select algorithm and randomized select algorithm where basically the problem is to find the i th smallest element. Now we have seen that depending on because basically we are using the quick sort partition algorithm. So, basically depending on the partition the pivot element we have the we it partition the array in to two parts. So, depending on this it will give us the time (Refer Time: 00:50) see, but in the worst case always it is 0 is to n minus 1 in the worst case. So, far we have seen.

So, now we in this lecture we want to talk about a guaranteed I mean the worst case linear time; time complexity for finding the i th smallest element and this is guaranteed worst case. So, this was invented by this people Blum Pratt I think Rivest 4 scientists and Trajan.

(Refer Slide Time: 01:20)



So, in this in the work we have. So, that was in I think 1973. So, this algorithm is proposed by this 4 scientists in 1997. So, idea is to basically their idea is to generate a good pivot. So, idea is to generate a good pivot recursively.

So, what do you mean by good pivot good pivot. So, that pivot means it is always minimum or maximum so that means, it will be always partitioning all this 0 is to n minus 1. Now good pivot means if we can ensure that some portion of the element will be less than x, some portion of the element will be greater than x. Maybe it is very little portion 1 by 10 is to 9 by 10, then also we have seen we are lucky then also we have seen we have linear time. So, if we can ensure the pivot is good in the sense that if you can ensure there are certain portion may be very small amount, even it could be work for 99 by 100.

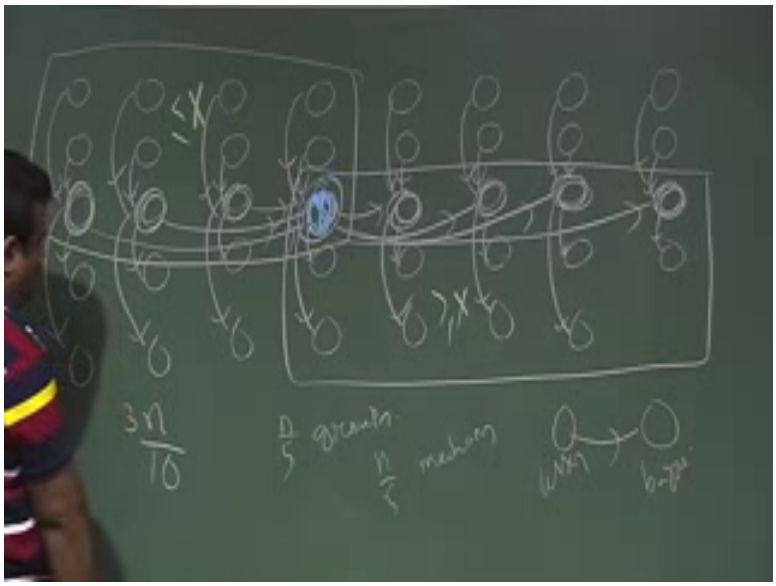
So, if we can ensure that a certain portion is less than x, and certain portion is greater than x then that choice of pivot is good pivot. So, the idea is to get such a pivot. If we can get such a pivot from this array, so we will use that pivot and then we call the partition and then obviously, if we know that there are some portion guaranteed, there are some portion will be left side and some portion will be right side. If this thing is guaranteed then if we call the partition then obviously, it will be array will be like this. So, it is guaranteed that some portion will be here some portion will be here.

So; that means, when we call again in the recursive call, they this then will go for here again in this subsequent recursively we are calling this divide and conquer technique, even in this call, even if we can choose again a good pivot. For this sub call then also again it will partition into certain portion over here should not. So, we want to avoid the 0 is to n minus 1 case, and that can be avoid only by choosing a good pivot that will guarantee that there are some portion over left side some element less than x and some element greater than x.

So, the idea is to choose how one can choose a good pivot. So, that is the their idea. So, if we can choose a good pivot which ensure that and that choice has to be recursively, that ensure that that guaranteed there will be some elements which are less than x, there will be few elements which are greater than x. Then we choose that x as our pivot and then we call partition, and then that will be that partition will be divide the array into like this portion not that 0 is to n minus 1. So, how we one can choose a good pivot. So, for that what they did.

So, we have given array. So, what we are doing? We are dividing these array into 5 element groups.

(Refer Slide Time: 05:03)



So, we have given the array of n elements. So, we are dividing into 5 element groups like this. So, we have we are grouping the elements into 5 element groups like this, may be in the last one we have only 3 4 element because this n may not be multiple of 5, but anyway. So, this is the this is our we have given a array A array is form say we have a array say 1 to n or p to q. So, we take first 5 up to 5 as a first group, second group like this. So, we are grouping this array. So, we divide the array into 5 element groups.

Now, now what we are doing? Now we are finding the median of this group. So, this is only 5 elements. So, we can sort it up, we can just sort the element and we can get the median. Suppose this is the median of this group and this is the median, this is the median, this is the median this is the median. So, we sort this and got the middle point of the group. So, these are the medians of this individual group now we find. So, we have how many groups are there. So, there are n elements. So, roughly n by 5 groups are there so that means, n by 5 medians are there ok.

So, among this, now, we have n by 5 medians. Now we find the medians of the medians and that we will find recursively by the same function call we will come to the code. So, now, we want to find the medians of medians. So, suppose this is the medians of median and this is the medians of medians and these we are going to choose as a our x. So, this is basically our pivot element we are going to choose, this is our x. Now if we can choose this as a pivot then why it is a good pivot so that means, what is the guarantee.

So, good pivot means it will guarantee that some fraction of the element will be less than x, some fraction will of some portion of the element, some portion will be greater than x. So, this we have to argue. So, this is the medians of medians. So, now, we will follow this notation. So, this is median of this. So, this is less than this, this is less than this. So, we will use this symbol as less than. So, this means this is lesser and this is bigger, we can use this symbol lesser by this. So, this is the medians of medians. So, this is lesser now there this is lesser than this, like this, like this ok.

Now this is the median of this group, so that means, this will be lesser than this, this will be lesser than this and this is like this. So, this is lesser this is lesser. So, like this we have. So, this is the median of this group. So, this is the relation we have less than relation. So, this is the median of this group. So, this is lesser this is lesser like this. So, we continue with this like this like this like this. So, we can leave the last one like this. So, now, if we look at all the elements over here say this element, now this element is less than this because this is the median of this group this element is less than this.

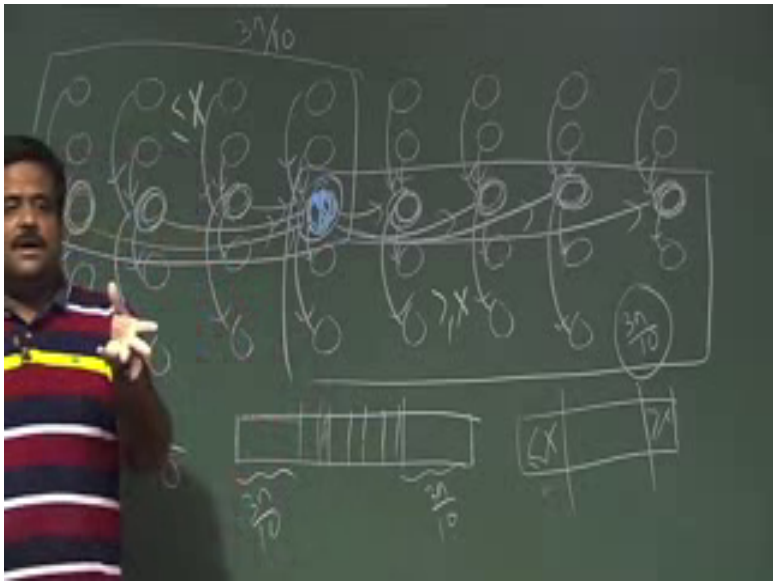
Now, this is the median of then now this element is less than this so; that means, this element is less than this. So, if you take any element over here they are basically less than equal to x. So, all the elements over here are basically less than or equal to x, and that is guaranteed. Because if you take any element this element this element is less than this, this is the median of that group and this element is less than this because this is the median of the medians so; that means, this element is less than this. So, this is true for any element less than equal to. So, if you take this element this is the median of this group. So, this element has to be less than this. So, this is the less than symbol we have use.

So, this is the portion where all the elements are less than or equal to x, and similarly this is the portion. So, this is the portion if you take this element say, this element is greater than this and this is greater than this. So, these element is greater than this. So, this is the portion where all the elements are greater than or equal to x, because by this relation if you take this element, this element is the this element is less than from this group median and this median is less than from the medians of the medians. So, this element is less than this.

So, we have a portion guaranteed we have a portion which is less than x, we have a portion which is greater than x. Now what is the size of this portion? Now size of this portion is basically. So, there are n by 10 medians, n by 10 groups now this is almost half of their. So, size of this is basically sorry n by 5 size of this is n by 10. So, among this n by 10 there are 3 elements each from each group. So, there are n by 10 such groups are there, which are basically i mean have a portion of which is basically less than x and how many of them? 3 of them.

So, basically 3 n by 10 is the size of this 3 n by 10 is the size of this portion, where elements from that that group that portion is less than equal to x, and similarly here also the size of this portion is 3 n by 10. So, basically x will be sitting somewhere here.

(Refer Slide Time: 12:35)



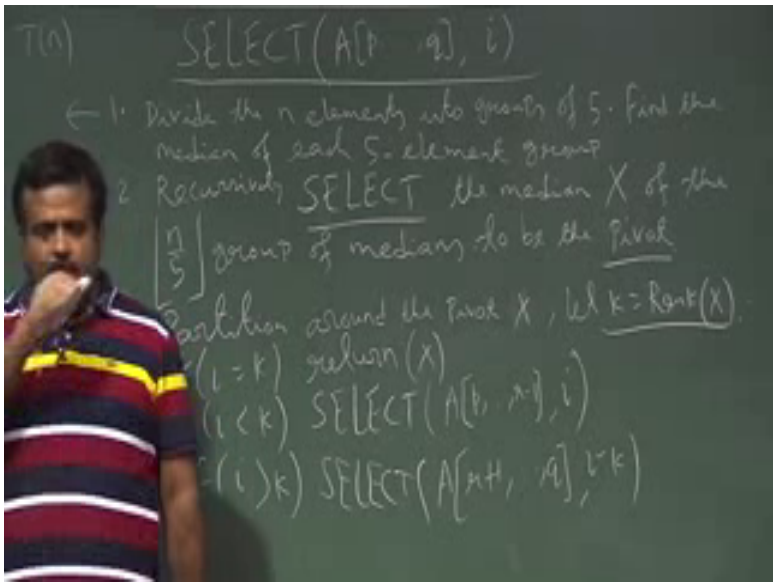
So, this is the. So, this is 3, so 3 n by 10, and this is also 3 n by 10, and if we. So, x will be somewhere in this here I mean we do not know where will be x, but this is guaranteed. So, this is the good pivot because we are guaranteeing that there will be some portion which are less than x, 3 n by 10 that portion size is 3 n by 10 and there will be some portion which is greater than x, 3 n by 10 and x will be sitting somewhere here. So, now, if we take that x as a pivot and then if we call the partition, then it will be like this. So, x will be sitting somewhere here and this. So, this is the guaranteed that the partition would not be 0 is to n minus 1. So, we are ensuring that there will be some portion which is less than x, there will be some portion which is greater than x.

So, that if we can ensure then if we call the partition algorithm by choosing that x, then it will not give us the worst case scenario like 0 is to n minus 1. So, it will give us some fraction left side some fraction right side. So, that will give us the recurrence. So, we will talk about that, but that the this is the idea. So, this is the idea to choose a good pivot. So, this way we choose the good pivot and this. So, here also we are involving some time. So, this should be taking consider into the algorithm. So, that will write the pseudo code for this, but let us just recap this. So, this is what we are doing we have a n element.

We are partitioning we are dividing the element into 5 element groups, and then we choose the median of each groups maybe there are 5 elements. So, we can just simply sort this and we can find the median. And then we find the medians of the medians. So, we will write the pseudo code for this and then that we are going to choose as a pivot element and then we call the partition by taking this x as a pivot and then this will partition into it will avoid the worst case scenario and then we have the same select.

So, we will find the rank of this and then if the k is less than let us write the code. So, this is the idea behind to choose a good pivot recursively. So, let us write the pseudo code for this algorithm.

(Refer Slide Time: 15:31)



So, this algorithm we referred as say this is also select, but capital select. So, we have n elements say 1 to n. So, what we are doing let us write this in English description. So, we

divide the array divide the n element into group of 5 element; groups of 5 and then we find the median of each group of each 5 element group then.

So, we got the median of the each group then we find the medians of the medians by using the same select algorithm. So, that is the recursive call. So, use the same select to find the medians. So, we have how many elements how many medians. So, we have n by 5 groups so. So, now, we have n by 5 medians. So, now, we want to find the median of the medians. So, we call the same select to find that. So, we recursively call the select same function to find the median and that is our pivot x of the n by 5 group of median.

So, we have medians of medians; I mean here we can use the lower ceiling because n may not be a multiple of 5 median to be the pivot. So, x is this medians of medians is our pivot and that we are finding using the recursive call of this same select function. So, we will talk about time complexity. So, now, this is our pivot element X. Now we use this pivot and then we call the partition algorithm and so, we partition the array around the pivot sorry we partition around the pivot X and then let k be the rank of X. So, if we call the partition it will return the position. So, suppose it is we call the partition.

So, suppose this is this is 1 to n, but in general it is p to q. So, it will return the position of x and then r minus p plus 1 is basically our k. So, k is the this is the i th r th this is the index of the pivot element. So, r minus p plus one is basically this is the rank of x, our x is sitting in the sorted array. So, now,. So, this is the r will be the r is coming from this partition algorithm and now. So, we know the rank of x now the code is same, if i is equal to k then we are lucky we return x, because we got the i th smallest element otherwise if i is less than x then we have to call the select again.

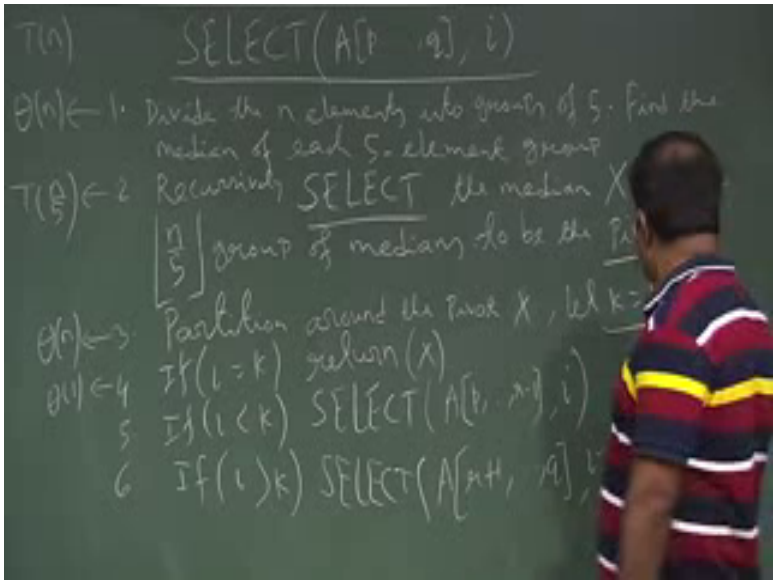
So; that means, r is less than x means we have to look at the. So, it is dividing a array into two part, we have to look at the left part of the array. So, A recursively call select from a. So, if it is p to q. So, here it is say p to q and if the partition is returning r. So, this is basically p to r minus 1 comma i else, if i is greater than k then we call the select. So, we have to look at the right part of the array. So, this portion of the code is similar to the select algorithm. So, only thing here difference is the choice of this pivot element. So, this is the. So, this portion is similar.

But only thing this is the way we choose a good pivot. So, that is the idea. So, we have to choose a good pivot in order to avoid the that partition 0 is to n minus 1. So, this is the

way we choose the good pivot. So, let us write the time complexity of this. So, how much time we are spending here. So, for that let us just write the T n be the time complexity for this run time. So, we divide the element into 5 element groups and find the median. So, how much time we are spending there. So, basically, we have n elements.

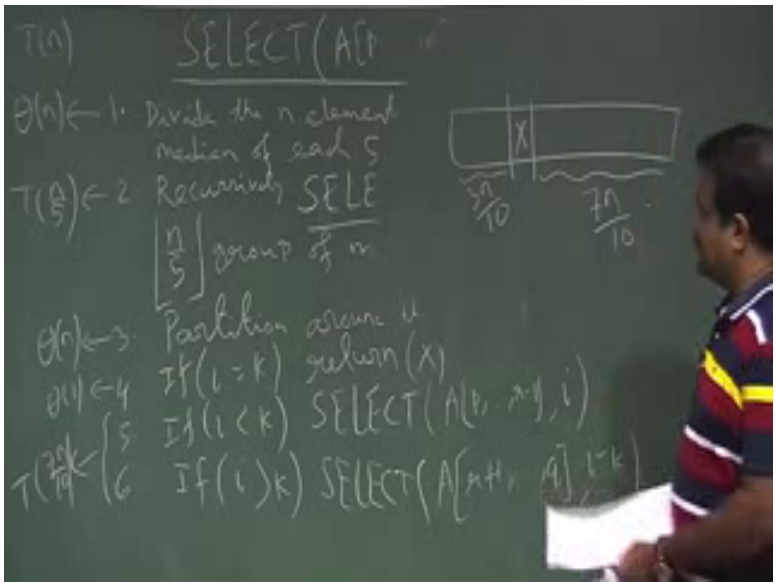
So, we are divided into 5 element groups sorry. So, how many groups? So, n by 5 and each group we are finding the median. So, we can just sort this 5 elements excuse me, only 5 elements.

(Refer Slide Time: 21:51)



So, this will take constant time. So, there are n by 5 such groups. So, this is a theta of n. So, now, recursively we call. So, now, we have we need to find the medians of this medians median of the medians. So, that for that we are again calling the same select. So, that will take us theta of n by 5 a T of n by 5 because we are calling the same function recursively T of n by 5. So, then this step will take the partition, partition will take the linear time theta of n, now if you are lucky enough we will got this as x, but otherwise we have to go for either left part of the array or right part of the array this is the conquer step. So, we have to call either left part of the array right part of the array. So, for that what is the size of that maximum size? Maximum size is basically.

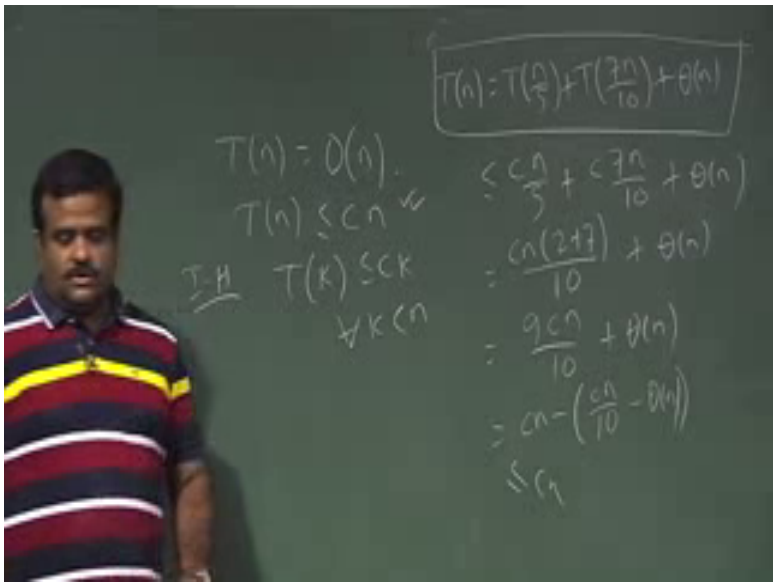
(Refer Slide Time: 22:57)



So, it is basically dividing the array into two part, we know that this part is 3 n by 10 and in the worst case suppose x is sitting here, and we are going to call the right part of the array.

So, this is roughly 7 n by 10. So, this is basically in the worst case this is basically T of 7 n by 10, because in the worst case we have to go to the right part which is the biggest one this is the worst case analysis.

(Refer Slide Time: 23:38)



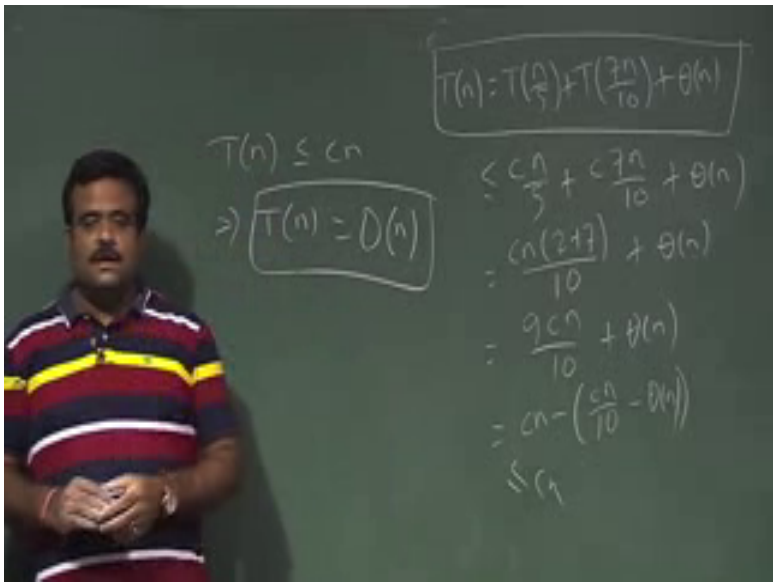
So, the T n is basically sum of this. So, T n is basically. So, T n by 5 plus T 7 n by 10 plus this theta of n theta of 1 theta of n all come. So, this is the recurrence we got for this select algorithm. So, now, we have to get the time complexity for this, we need to solve this.

So, how to solve this? So, we can try for recursive tree to get the solution otherwise we can try for substitution method to get the solution. So, let us try for substitution method. So, we are thinking that T n will be big o of n. So, this is our assumption so that means, we are thinking that T n must be less than equal to some c n some c constant. So, now, we need to take a substitution method. So, we need to take a induction hypothesis. So, we assume T of k must be less than c of k, for all k less than n and then we need to prove that T of n is less than this.

Now, we have this recurrence now this is basically less than. So, this is n by 5 which is less than n. So, we use this induction hypothesis this is c n by 5 plus we have again 7 n by 10, which is also less than c n by which is also less than n. So, we have this n by (Refer Time: 25:25) I put this we can just write c 7 n by 10 plus theta of n. So, this is basically what this is basically 10 into. So, c n we take common. So, this is 2 plus 7 plus theta of n. So, this is basically what? So, its nine c n by 10 plus theta of n. So, this is basically you can write is at.

So, we want to write this we want to show this less than c n. So, to show this we have to write this to be c n minus of some quantity and that quantity has to be positive. So, what is that quantity? So, basically c n by 10 minus theta of n. So, now, this. So, we choose c in such a way that this will be positive. So, c is in the our hand we can choose c such a way this will be positive and then if once this is positive this can be written as c of n so; that means, this is established. So, by the help of so this is the substitution method.

(Refer Slide Time: 26:48)



So, we can just write c of n is T of T of n is less than c of n. So, this imply by the substitution method theta of n. So, that is the worst case runtime for our select algorithm and this is not a randomized algorithm, this is just a this is to get the pivot as a good pivot, we choose the pivot good pivot in that way. So, this is the guaranteed. So, guaranteed worst case run time is linear by choosing that even we can establish this by using the recursive tree. So, one can draw the recursive tree and can see that the run time will be also linear.

So, this is the worst case linear time order statistics finding the i th smallest element, and this is the guarantee this is not the expected this is not the average case this the worst case linear time algorithm for finding the i th smallest element.

Thank you.