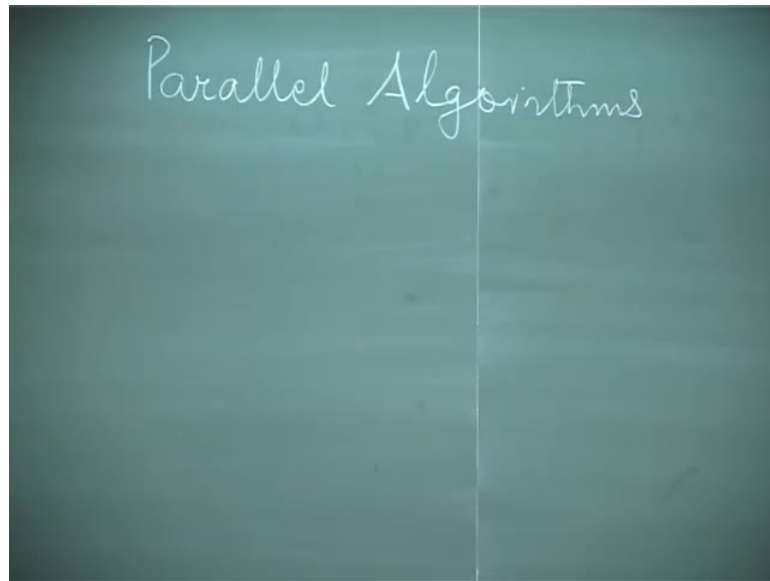


Parallel Algorithms
Prof. Phalguni Gupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 1

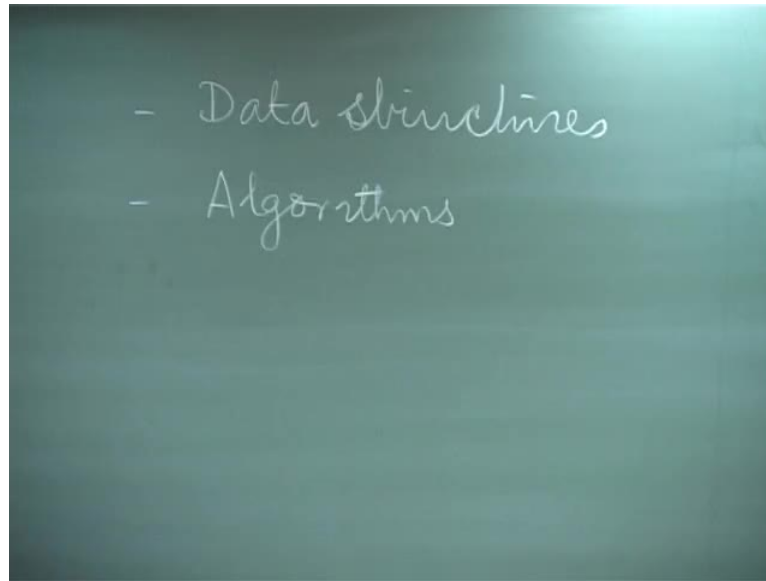
Good morning, this is the course we will be discussing here, that is on parallel algorithms.

(Refer Slide Time: 00:34)



All the parallel algorithms is an algorithm, which will be designing on parallel machines, before we discuss about the parallel machines, I I think then it is first time to tell about that, what is sequential machines, sequential algorithms and the sequential that is structures, what is the other issues involving that, that we will discuss about a parallel algorithms.

(Refer Slide Time: 01:02)



Now, we look at into this course, what do I expect from you is that you have the preliminary knowledge on the data structures and algorithms. Well this data structure and algorithm, these two words completely link to each other and they are very small component on that software engineering life cycle. Now what is algorithm? Algorithm is a step by step procedure to solve a problem by an automation. What is automation? Automated is nothing but I can give you one example suppose, your mother is told you to bring some items from the market say, brinjal and you have gone to the market and you have found that brinjal is not available in the market.

What are you going to do, one way could be possibly will not be purchasing anything and you will be coming back without thinking much and this is, that you have to acted as an automated, because you did not use your brain and to think that, what would happen if brinjal is not available at home. And then, the possibly you will not able to get your food so, idea is that you are not using your brain to solve this problem. So, you have that is, that an automation is an machine, which does not have any brain to think.

So, that is the definition of algorithm now, what happens, in the algorithm we have few things in your mind that, it must terminate after sometime. Otherwise, what that will be an infinite loop and will not be able to decide, whether algorithm gives you the correct thing after certain amount of time. The next program is that, algorithm must be that should not be any long time error or something undefined material existing in the steps

and third factor one should keep it in the mind in the case of algorithm that, it is a must be an effective.

What does it mean, the define I am not able to solve every step of the algorithm but, if it so happen, it may take huge amount of time to solve the problem that is, that may be that may be a possible scenario. Now, now another factor you should keep it in mind in the case of algorithm, it must increase some output. Otherwise, whatever you are designing the algorithm, if you are not getting any error output, you will not using any output then, this algorithm has no meaning.

Now, there should be some input, maybe input is internal one are external one now, based on that, you can design the algorithms. Now, the thing is, the algorithm depended on data now, it may so happen that, you have different sets of data and how you are arranging them that is known as data structure. Now, an algorithm can be suitable for one kind of data structure but, it cannot be appropriate for the another set of our another type of data structure.

So, this is the highly, so it is highly link between algorithm and data structures so, basically, the meaning of data structures is that, the way you are arranging the data is data structure. Now, there are certain operations we will perform on data structures, one is anyone can read the elements, another one is that you want to modify certain element, another one is that you may like to insert some elements or delete some elements from the data set or you may reverse or you may perform certain operations or modifications on some set of some elements on this set of data right.

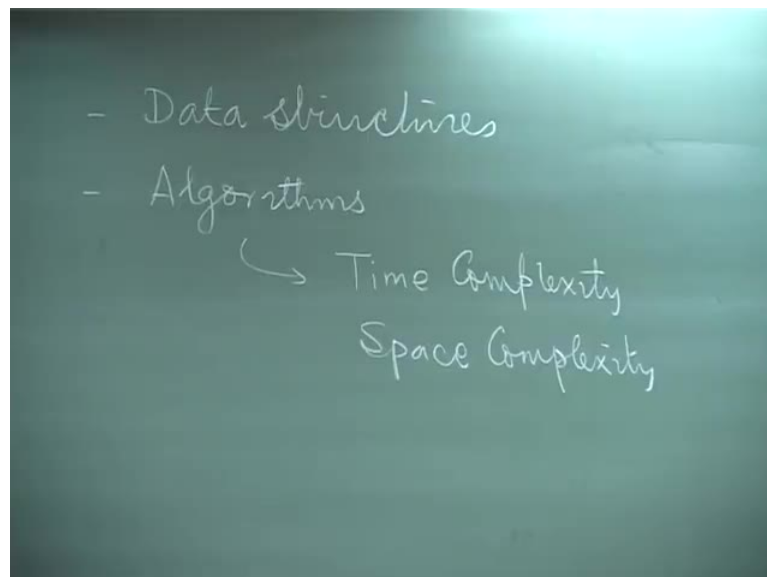
So, these are the few operations, we let perform on data structures now, and a data structure, various type of data structures may be suitable or some type operations may not be suitable for another type of operations. So, why would define the data structure is, the programmer or designer must think about that the type of operations you like to perform the data, depending on that, we design one should design the data structure. So, what is the data structure is defined then only, you can think about to design the algorithms, once you design the algorithm, what are the factors you should keep it in mind.

One, the algorithm must be correct, until and unless the algorithm is correct, you cannot justify that, this will produce the result. Now, once you know the algorithm is correct,

there the question is coming that, the algorithm must be efficient. So, what is meant by efficient? Efficient means that, it must be taking must be able to take less amount of time to produce the result. The another possibility is that, you just taking less amount of space so that, if it is not loaded on the computer however, the space problem is not that difficult here.

Because, because that is easy or the cost wise, the getting memory is not very costly so, we do not consider the space factor much for designing the algorithm. But, of course, the complexity type plays a major role to define or to design the efficient algorithm.

(Refer Slide Time: 07:47)



So, there are two types of complexities, one is known as time complexity, the other one is known as space complexity. Now, how to make that the time for an algorithms, what are the possible way, the possible way is that, you design an algorithm and then, implement on it on some machine and then, you tell how much time it takes right. So, but, this is completely machine dependent and in order to, compare one algorithm with another algorithms, there you need to get the algorithm and you have to run on the same machine.

And then, you will be able to tell which one is better better with respect to time, well another factor is that, why we use to determine that x amount of time on the then, you have to specify the type of machine you have to use, which is changing day by day. So, that is not a good way of measuring the time for an algorithm so, next factor could be

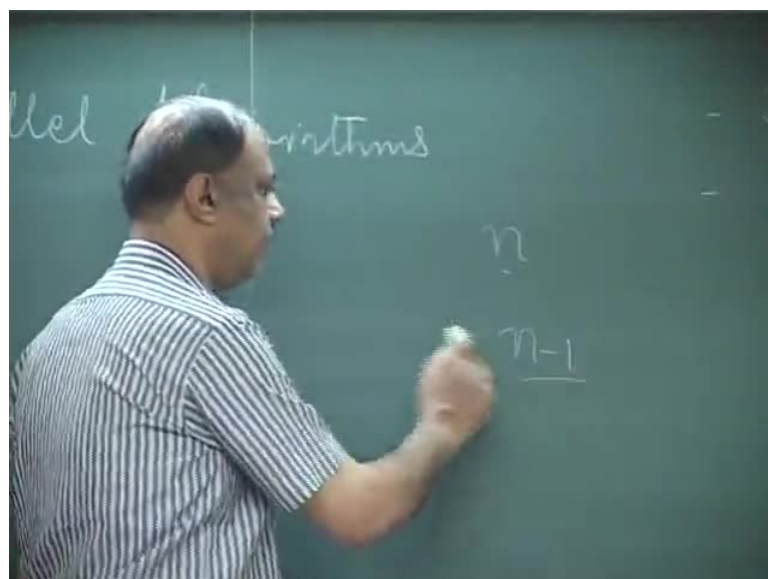
that you can tell the number of operations you are performing on algorithms. And for each category of operations, the number of time that operation operator number of number of times that operations you have performed right.

So, number of operators you have used and for each operator, number of times you have used that you can compute and then, you can estimate the time because, you can tell that for multiplication x amount of time, for edition y amount of time and so on. So, you can compute the total amount of time needed for algorithms or for these algorithms. But, again this is also not an easy task to estimate, how to find out the number of operations, you are performing in an algorithms.

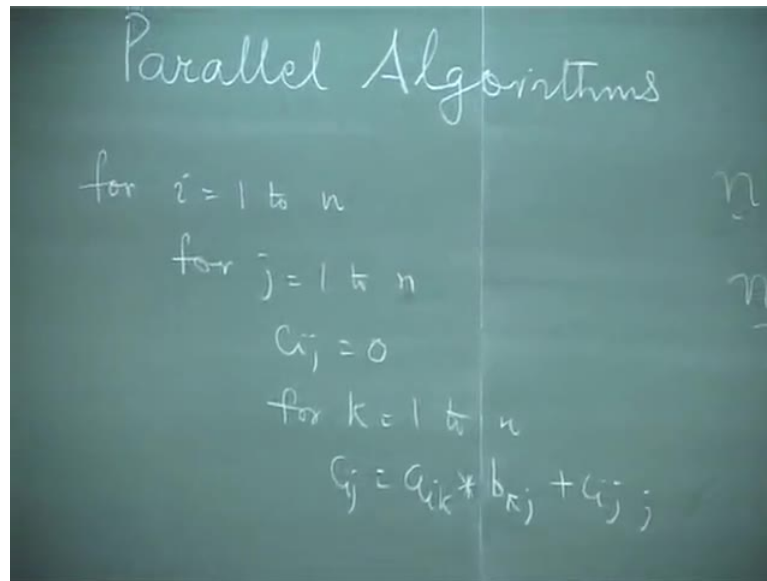
So, what is the default way of estimating the time for an algorithms, one possible thing is that the number of, you parallel algorithm we can find out the what which are the operations are critical. Say for example, in the case of sum of n numbers, addition is the critical operations, in the case of matrix multiplication, multiplication is the critical operations, in case of sorting is the critical operation.

Now, the number of operations for that problems can be defined in terms of input output parameters that is, the number of inputs or number of outputs in generating that can be used to estimates the time complexity. For example, that suppose, I have n elements and you want to find out the sum of n elements then, the operations number of operations or the number of additions will be using is a function of n .

(Refer Slide Time: 11:23)

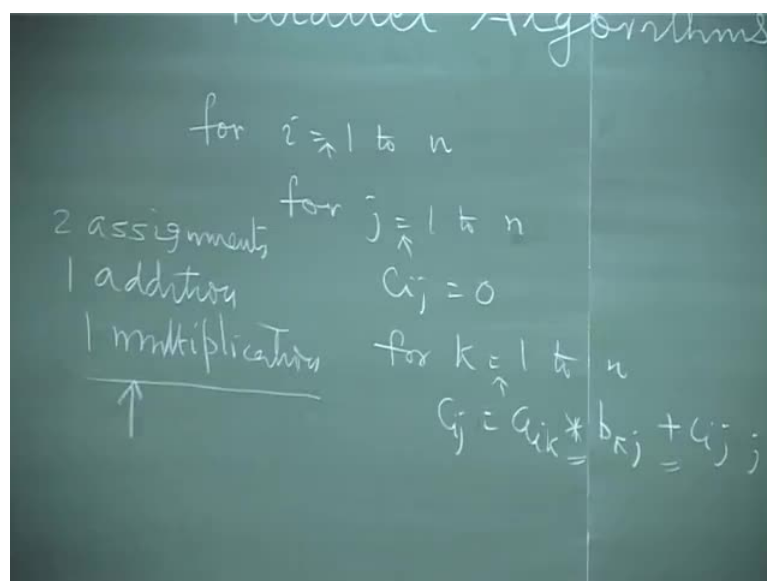


(Refer Slide Time: 11:44)



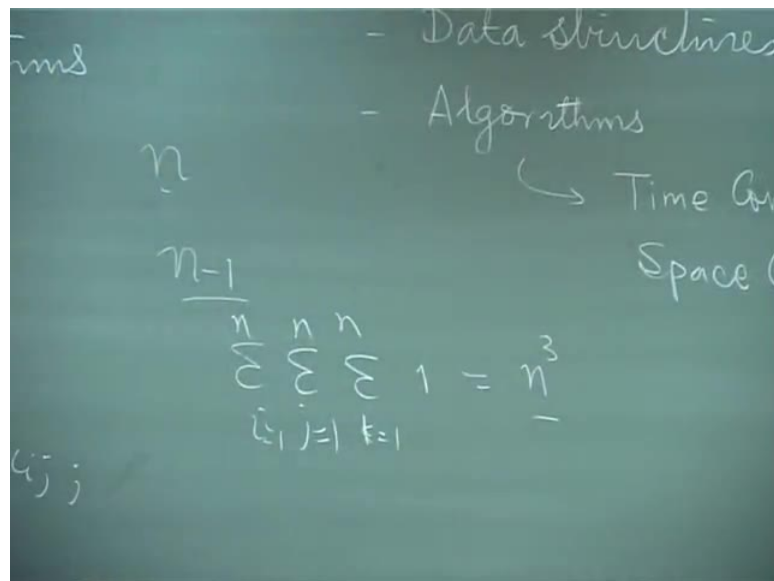
Can you estimate the number of number of additions you need, to find the sum of n elements, yes the number is n minus 1 number is n minus 1, the additions you need to find the sum of n numbers. So, there is the number of additions is known as n minus 1. so, this is the dependent on the input parameter n similarly, in the case with multiplication so, or matrix multiplication, here could be for i equal to 1 to n, for j equal to 1 to n, c_{ij} equals to 0.

(Refer Slide Time: 12:47)



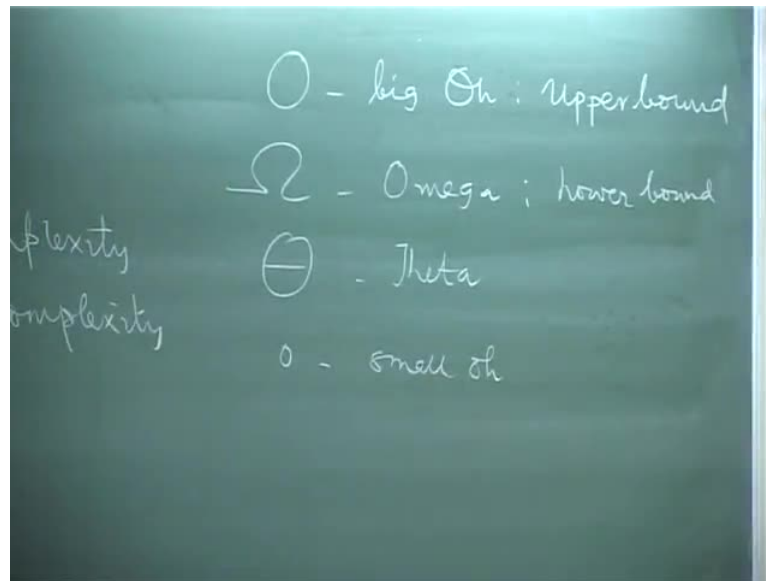
For k equals to 1 to n , c_{ij} equal to $c_{ik} + b_{kj}$ plus c_{ij} right, this is the simple classical algorithm format for multiplications and you observe then, here the two types of operations here, viewed one is plus and another one is multiplication. Now, you know the multiplication is the costlier, multiplication needs more amount of time than additions. So, the critical operations c is here, is multiplication you observe that, if I see whole algorithms then, two assignments two assignments or assignment statements, one addition and one multiplication.

(Refer Slide Time: 13:23)



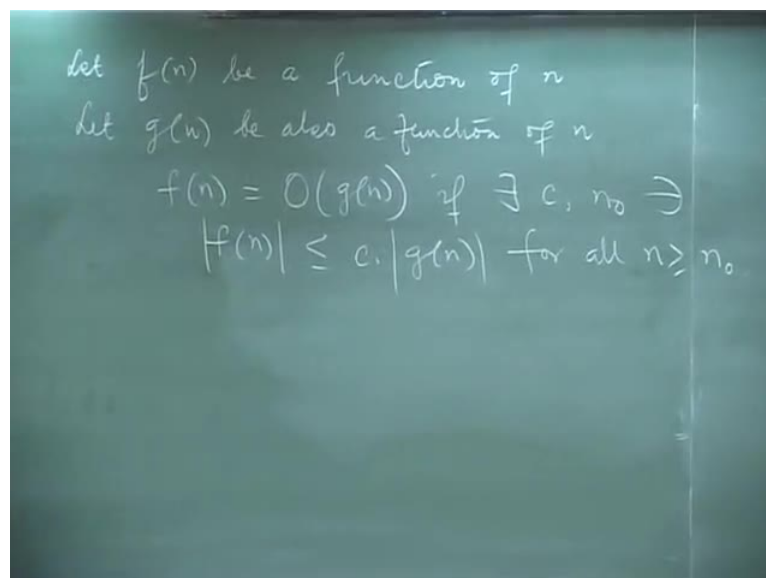
Besides this, note the of this assignments for for this for loops and here, this multiplication is a representation and the number of number of multiplications you need number of multiplications you need to find or to matrix multiplication to compute the matrix multiplication is this one, which is n^3 which is n^3 . So, again you observe there time complexity so, when is the time complexity will be addition using that equal to output parameters.

(Refer Slide Time: 14:24)



Now here, there will you find that the worst case possible and the best case possible so, for example, to find to sort n elements, you will find that sometimes the worst case environment is this and sometimes you may get the best case environment. So, in the case, you want to estimate the worst case and the best case, use these two notations. This known as big o and this is omega notation, for best possible that is, the lower bound and this is known as upper bound.

(Refer Slide Time: 15:29)



Then, we have there are known as theta notation and small o notation. so, what is the theta notation, which we use for worst case time complex f n be a function of n. Let g n be also a function of n then, you get you tell f n is o g n the request to o of g n, if there exists constant c and n are such that, f n is less than equal to c times g n for all, n greater than or equal to n naught. So, for example, you have a polynomial of the n set f n is a n x power n plus a n minus 1 x to the power n minus 1 a one x plus a 0.

(Refer Slide Time: 16:56)

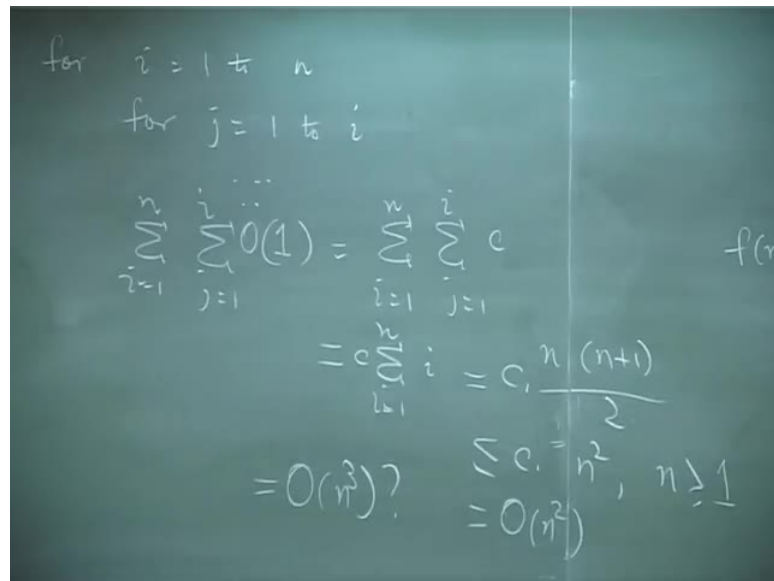
The image shows a chalkboard with the following handwritten text:

- Data Structures
- Algorithms
- ↳ Time Complex
- Space Comp
- $f(n) = \underline{a_n} x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$
- $\leq (a_n + a_{n-1} + \dots + a_1 + a_0) x^{n+1}$ (with a note $a_n \neq 0$)
- $= c x^{n+1} = O(x^{n+1})$

This is polynomial of the degree n now, if it is polynomial of degree n, what does it mean that a n for which, this is the leading coefficient which for all, not equals to zero. Suppose, i want to find out the worst case complex, the worst case complexity of the time and find out by this, this is less than equals to a n plus a n minus 1 a 1 a 0 x power n.

Now, a n a n minus 1, a 1 is 0 they are constant so, I can write c times x to the power x to the power n, one thing you have to keep it in mind, if I can I can write only this is less than equals to this, provided they are all positive. Now, this is twelve digits number otherwise, I should write this is n plus 1 and then, I have to write the n plus 1, get it so, if this is the case so, I can write this is order x to the power n plus 1.

(Refer Slide Time: 18:26)


$$\begin{aligned} &\text{for } i = 1 \text{ to } n \\ &\quad \text{for } j = 1 \text{ to } i \\ &\sum_{i=1}^n \sum_{j=1}^i O(1) = \sum_{i=1}^n \sum_{j=1}^i c \quad f(n) \\ &= c \sum_{i=1}^n i = c \frac{n(n+1)}{2} \\ &= O(n^2) \quad \sum_{i=1}^n i = n^2, n \geq 1 \\ &= O(n^2) \end{aligned}$$

I have for i equals to 1 to n, for j equals to 1 to i and so on now, suppose, I want to compute the complexity of this element so, you will be writing that, summation over i equals to 1 to n, summation over j equals to 1 to i and will be writing say order 1. So, this can be written as summation over i equals to 1 to n and this is becoming summation over j equals to 1 to i, c, c is constant.

So, this is the terms summation over i equals to 1 to n c into i c into i, that will give you c into n into n plus 1 divided by 2, agreed. Now, if it is the case similarly, write this equal to c into n square, c equal to c into n square now, for what is the value of for what value of n this will be true so that, n equals to 1 and n is equal to 2, it is on case 3 and so on. So, i can write this is true for n greater than equal to one so, this i can write this is order n square order n square.

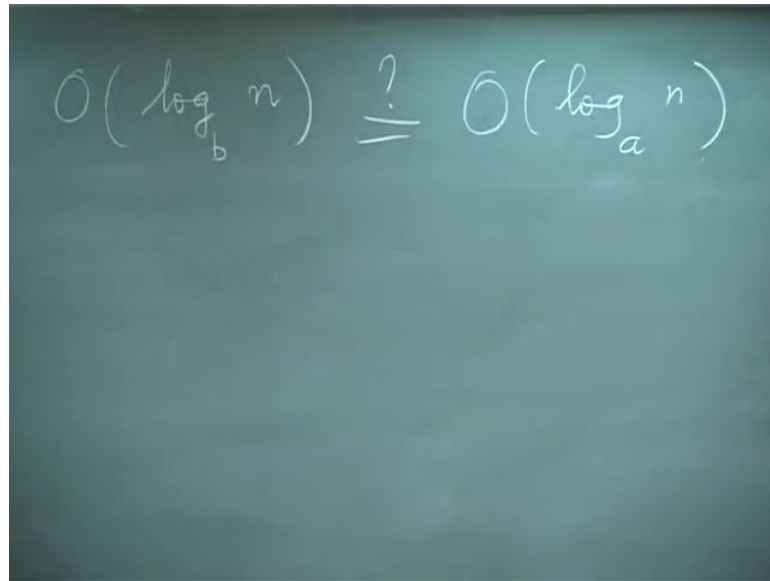
(Refer Slide Time: 20:50)

$$\begin{aligned} & \sum_{i=1}^n \sum_{j=1}^n O(1) && \text{for } i=1 \text{ to } n \\ & \leq c n^2 && \text{for } j=1 \text{ to } n \\ & = d n^3 + c n^2 && \sum_{i=1}^n \sum_{j=1}^n O(1) \\ & \leq (c+d) n^3 && \text{where } d=0 \\ & = e n^3 && n \geq 1 \\ & = O(n^3) \end{aligned}$$

So, can you tell me, can I write can I write instead of, order n square can I write order n cube? Yes, answer is yes, you can write. Because, I can always write I can always write summation summation 1 i is equal to 1 to n, j is equal to 1 to i, this is less than equal to c into n square. This I can write d into n cube plus c into n square where, d is 0. Now, if I assume that the c plus d c plus d n cube and this is always true for n greater than equal to 1 and this is equal to e into n cube where, e is constant.

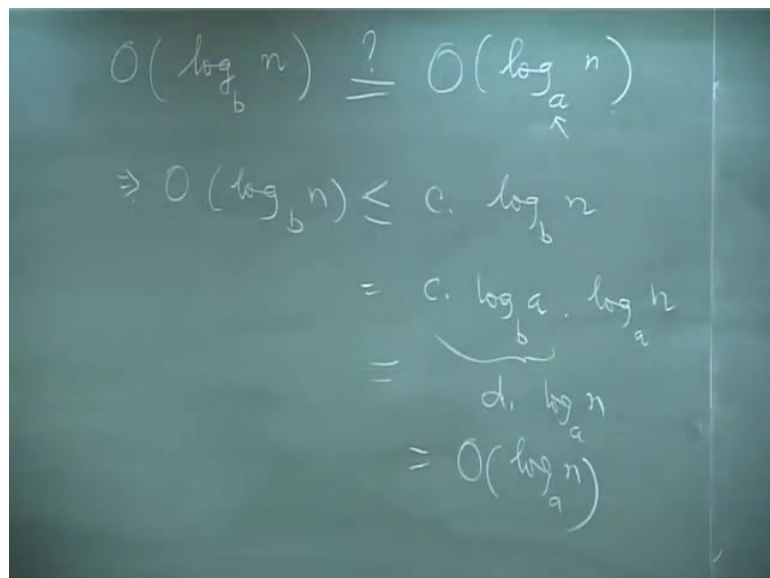
So, it is order of n cube now see you observe that from this I can explain so, I can write o n cube but, problem is somewhere else. Because, I have to save my algorithm and my algorithm is efficient. So, in order to that, I have to see that this is as close as to the lower positive values.

(Refer Slide Time: 22:30)


$$O(\log_b n) \stackrel{?}{=} O(\log_a n)$$

So, which is n square now, I have a small question for you, in this thing where, b and a are the constant order $\log_b n$ with this equals to order $\log_a n$ where, a and b are the two constants.

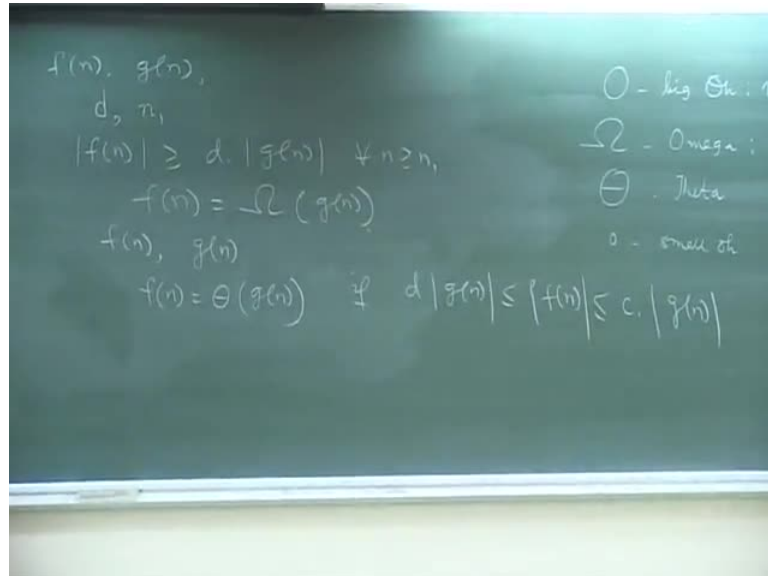
(Refer Slide Time: 23:12)


$$\begin{aligned} O(\log_b n) &\stackrel{?}{=} O(\log_a n) \\ \Rightarrow O(\log_b n) &\leq c \cdot \log_b n \\ &= c \cdot \log_b a \cdot \log_a n \\ &= \underbrace{c \cdot \log_b a}_{d} \cdot \log_a n \\ &= O(\log_a n) \end{aligned}$$

Suppose, I want prove that they are constant whether you write this one, order $\log_b n$ is equal to less than equal to c times $\log_b n$ that is equal to c times $\log_b a \log_a n$. So, this is nothing but, d times $\log_a n$ where, d is constant and you can write order $\log_a n$. So,

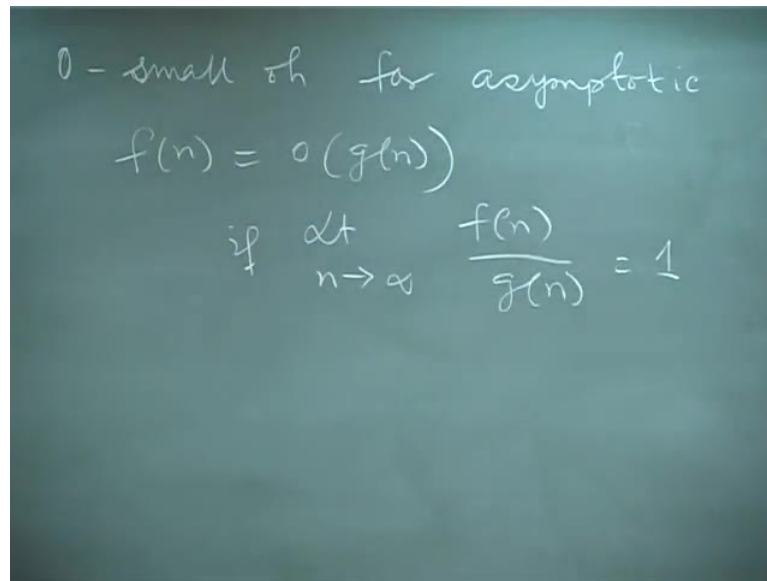
when we can tell about the complexity, we do mention about this base of logarithm base of logarithm because, they have much meaning to define worse case time complexity.

(Refer Slide Time: 24:17).



Now, think about the another symbol, which is known as b omega symbols here, it is talking about lower bound that the p is, you have a f n and g n are the two functions of n. And you have the two positions d and n 1 and here, it is f n is greater than equal to d times g n for all n greater than equal to n 1. That case if it is that case, f n is omega g n that is the case, if i tell f n is omega of g n that means, that it may change the lower bound, which is the constant factor which is the constant factor. Now, theta is another notation we use where, you can have the two functions f n g n and h n then, f n is equal to theta g n, if d times g n.

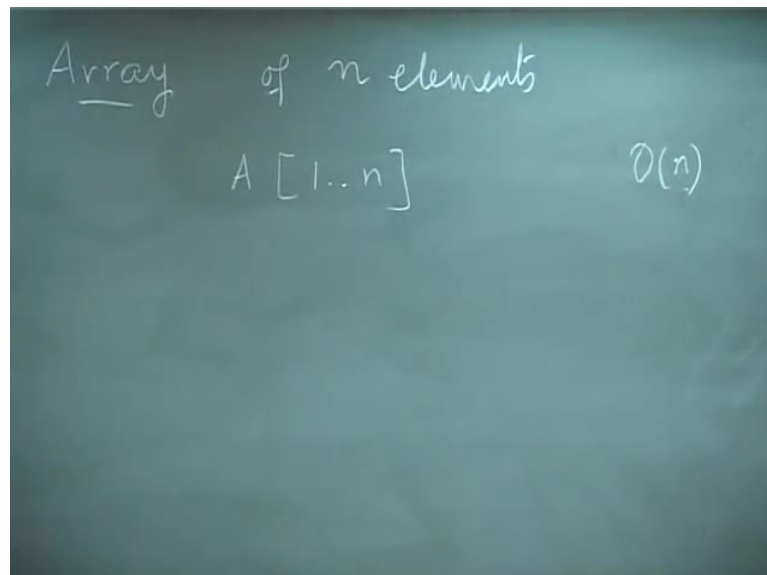
(Refer Slide Time: 26:37)



0 - small oh for asymptotic
 $f(n) = o(g(n))$
if $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 1$

That is, $f(n)$ is achieved both lower bound and upper bound of the algorithms right in the constant factors. So, we introduced the notation theta for this purpose, this is another notation we use asymptotic notation o , small o small o is used for asymptotic that is, $f(n)$ is small o $g(n)$ that is asymptotic to that is, asymptotic to $g(n)$.

(Refer Slide Time: 27:47)



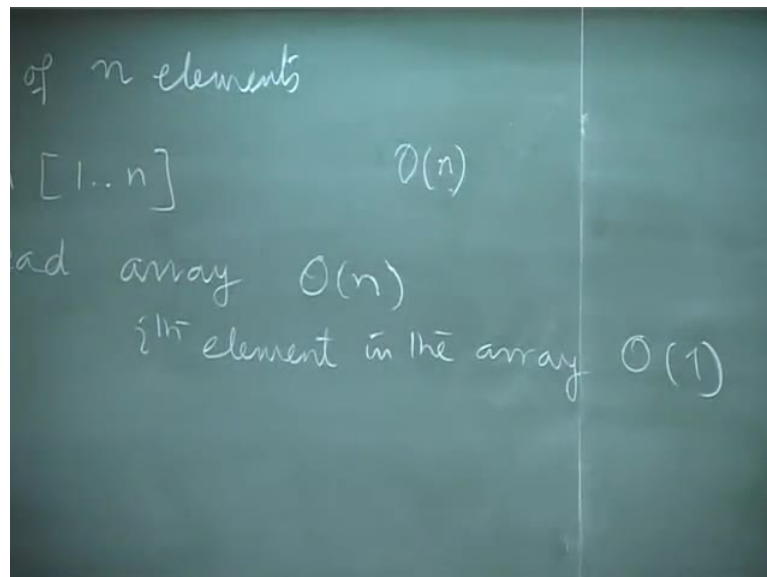
Array of n elements
 $A[1..n]$ $O(n)$

If limit n tends to infinity, $f(n)/g(n)$ is equal to 1 right so, we can write $f(n)$ with assumption to j , if limit n tends to infinity $f(n)/g(n)$ is equals to 1. So, this on the times we use for sequential algorithms for evolving the sequential algorithms now, let us discuss about the

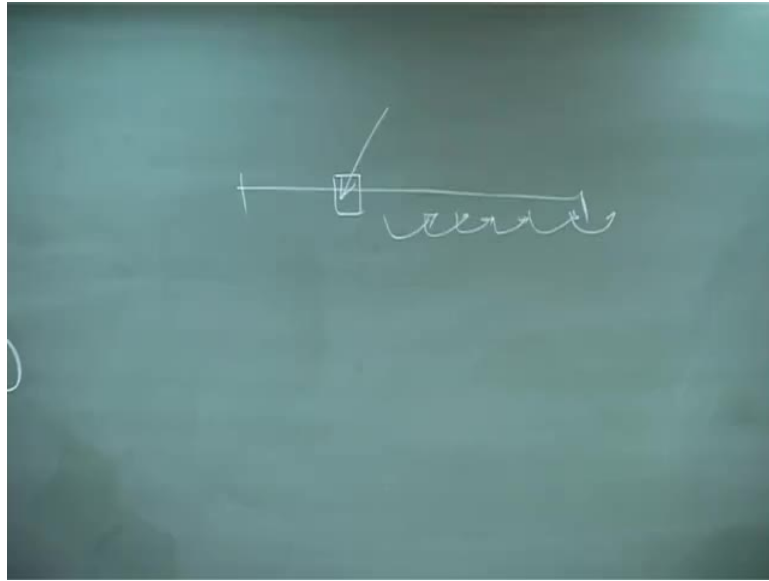
data structures, you have done something in your book. And again of course, and first thing is that, you have done the simple data structure, which is known as array. Now, once we have an array you tell there, what is the number of elements in the array right.

So, array of n elements and so, the first part is that, which is known to you, the number of clear up by which the number of elements in the array that is, the major disadvantage on array. Now, the second thing is that, let us try to perform certain operation on this array suppose, I want to read the elements in the array. So, you need to read all the elements one by one and that takes order n amount order n time now, once this array elements are there then, at any movement you can fetch the i th element in order n time.

(Refer Slide Time: 29:05)

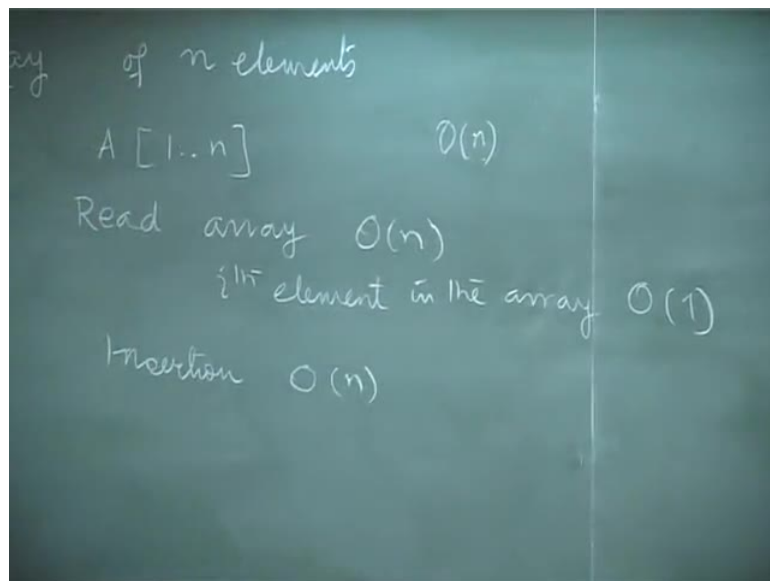


(Refer Slide Time: 29:35)

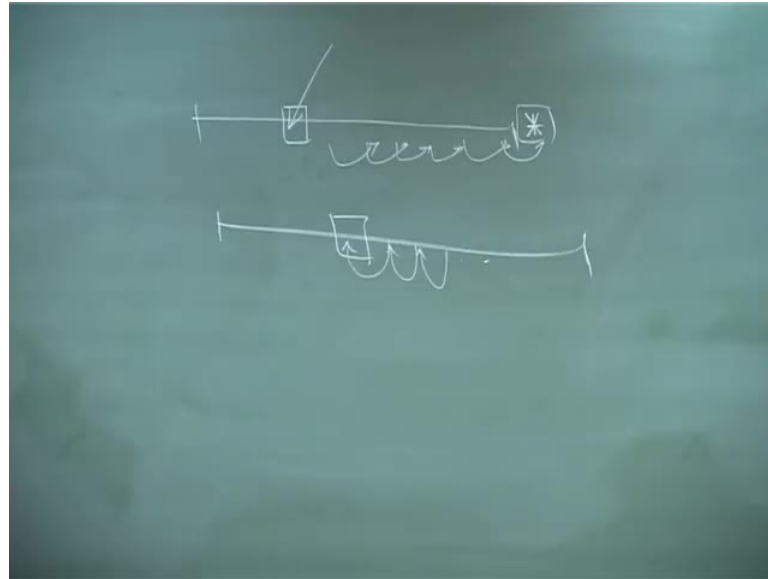


So, reading array element reading array element say, for order n time, once it the reading i th element in the array order of one times, the next one is suppose, I want to insert an element in the array. So, this insertion is not an easy task in the array suppose, I want to insert an element here, what you have to do, you have to get a space for this element. So, you have to get that all the elements to be shifted by one towards.

(Refer Slide Time: 30:07)



(Refer Slide Time: 30:18)

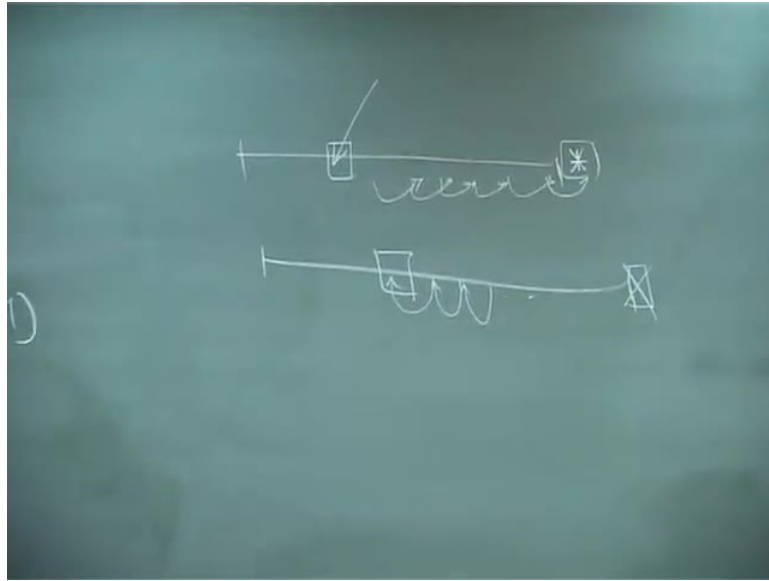


(Refer Slide Time: 30:53)

$A[1..n]$ $O(n)$
Read array $O(n)$
 i^{th} element in the array $O(1)$
Insertion $O(n)$
Deletion $O(n)$

So, in the worst case, the insertion is order of n that is, in the last case but, suppose I want to insert an element here, nothing rule the to addition of, which takes constant of the time, which is the best task. Now, if you think about the deletion what happen, suppose, I want to delete this element suppose, I want to delete this element then, we observe you want to delete. Basically, you cannot just delete it, here to bring this element here, this element here, this element here and so on. So, again deletion takes, deletion of take order n times right but, if the best possible case suppose, i want to delete the last element then, it takes constant amount of time.

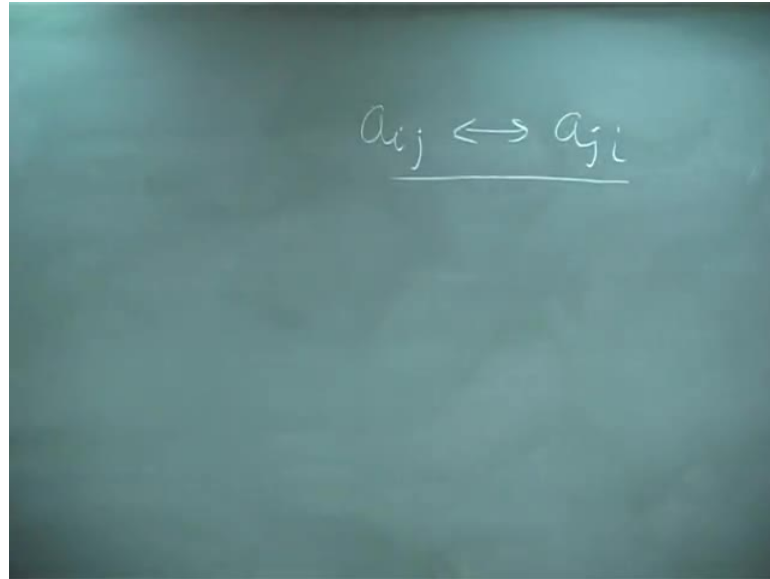
(Refer Slide Time: 30:57)



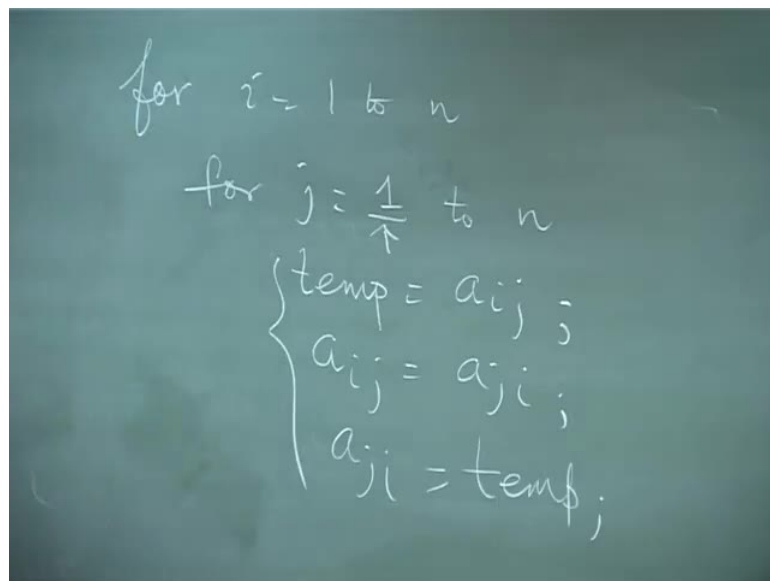
Now, suppose I want to modify the i th element of the array, how much time will it take, it takes order one time. Because, I will just go to the i th element and the excess thing so, I can easily replace the content of the i th element and it takes order one time. So, these are the major steps or major operations. Now, a simple question for you, you know what is the transpose of the matrix, the transpose of the matrix is a very simple operation to do.

But, it is not while I kept writing an aptitude of the transpose of a matrix is nothing but, that is you have a_{ij} has to interchange a_{ji} . So, set is the operation similar going to perform it looks n is the simple just you in touch in between these two elements in getting the other.

(Refer Slide Time: 31:51)

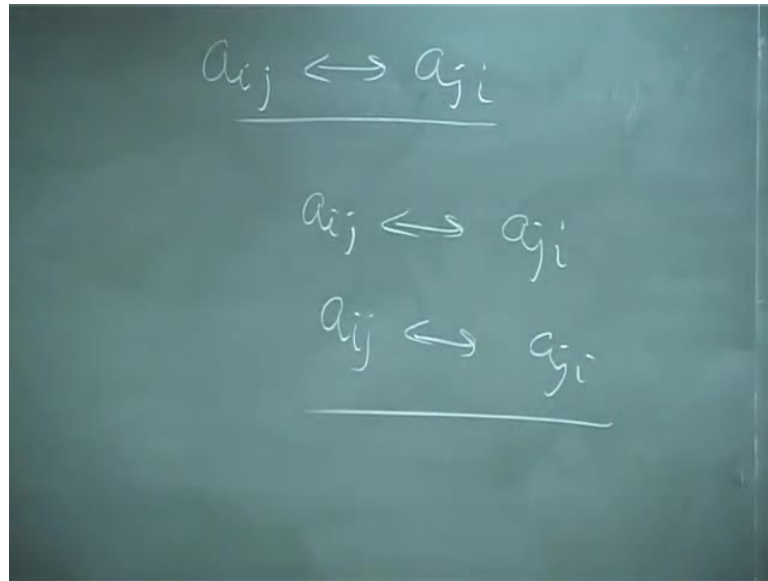


(Refer Slide Time: 32:20)



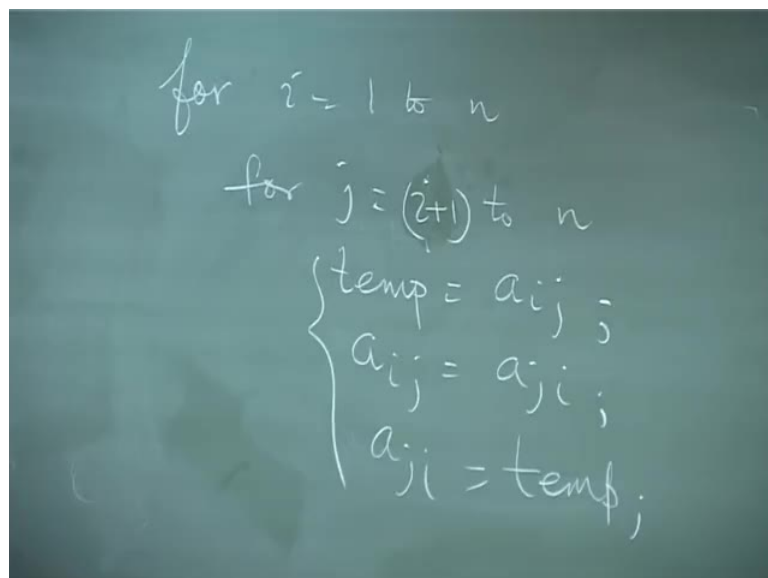
If we write an algorithm point of transpose of a matrix, we will observe that it has it is not that much easy. Let us try, can you tell me what I could write, yes for i equals to 1 to n , for j equal to, temp equal to a_{ij} , a_{ij} equal to a_{ji} and then, you will write to a_{ji} equal to temp. So, if it is the case, then can you tell me, what i could write here, 1.

(Refer Slide Time: 33:19)

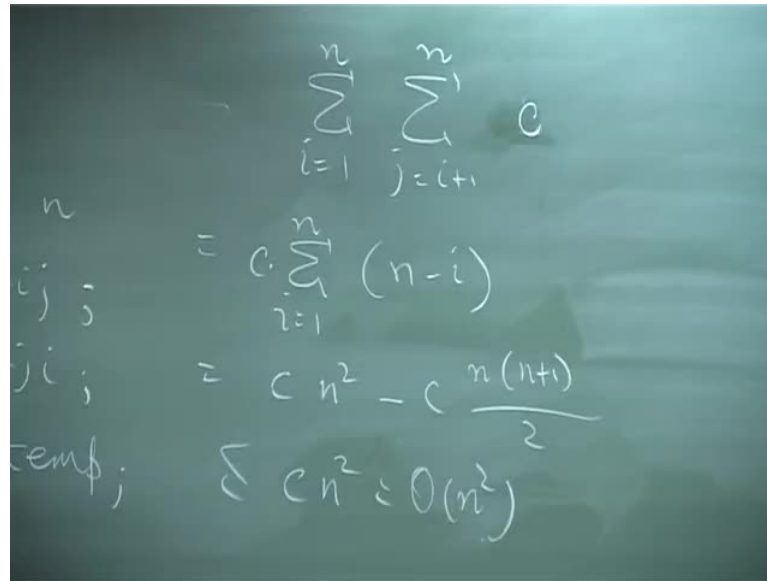


Let us see what happens if I write 1, would you get the transpose matrix, answer is no, it will give you the same matrix again. This is because of the fact that, just for a while will be first signs, you will interchange a $i j$ into a $j i$. Again, when i equal to j and j equal to i , you find that a $i j$ is interchanged into a $j i$. Try at home and see these two times interchange and another. So, what I could write here, one will be I can write i to n in that case, yes you will be getting one time interchange and you get the transpose of the matrix.

(Refer Slide Time: 34:15)

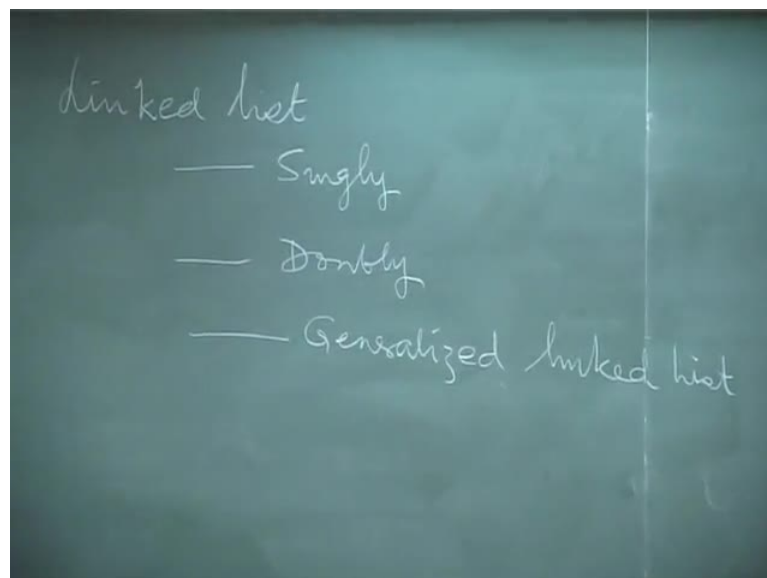


(Refer Slide Time: 34:34)


$$\begin{aligned} & \sum_{i=1}^n \sum_{j=i+1}^n c \\ &= c \sum_{i=1}^n (n-i) \\ &= c n^2 - c \frac{n(n+1)}{2} \\ & \sum c n^2 = O(n^2) \end{aligned}$$

Here, this is there is one problem here, in observe that if I had i to n the initial is the diagonal element would be interchange again and again. But, this is not question of what is no need of interchanging diagonal element of the matrix so, 1 plus I, i plus 1 to n right. Now, suppose I want to obtain the complexity of this algorithm then, you can find out delta i equals to 1 to n, j equals to i plus 1 to n and this is constant time. Because, constant time, so it is c and which, I can write summation over i equal to one to n c times n minus i.

(Refer Slide Time: 35:37)

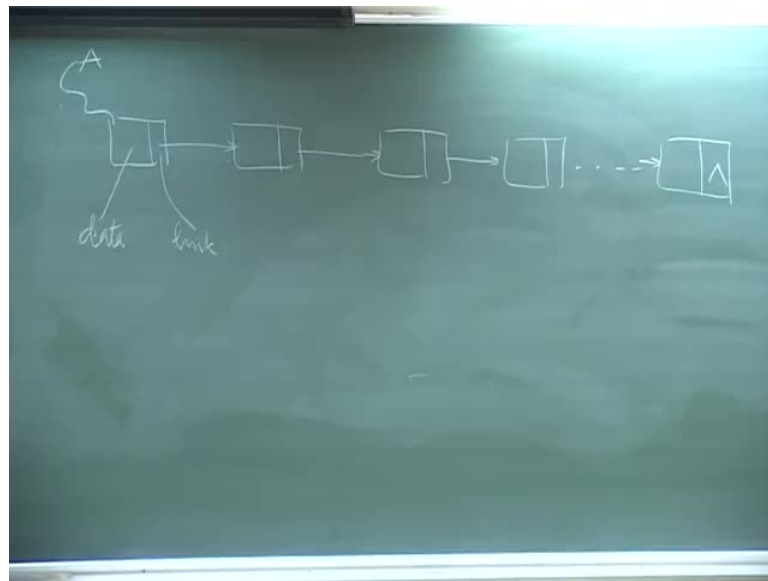


linked list

- Singly
- Doubly
- Generalized linked list

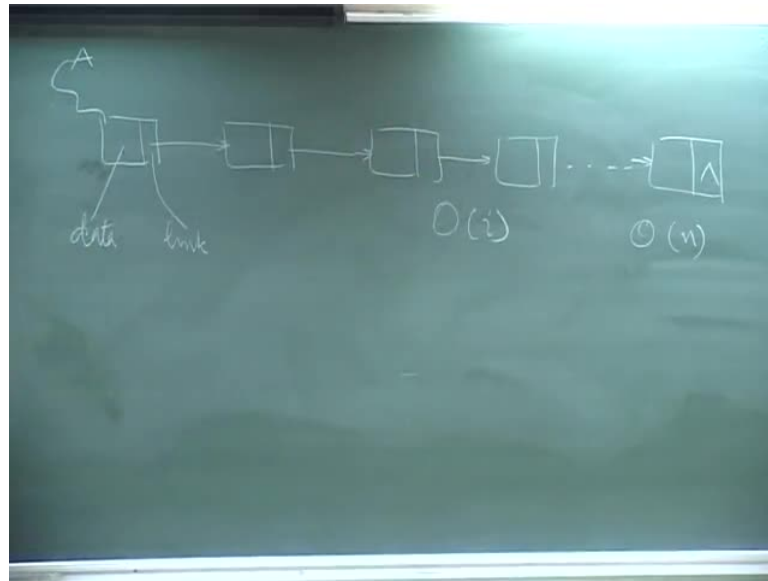
n minus i which I can write $c n$ square minus c times n into n plus 1 divided by 2 and which is $c n$ square which is order n square. So, one thing you remember, the transpose of a matrix their solve n square times that is the minimum requirement and minimum time this is also called. No, there need data structure here occupied linked list, the in linked list, you know one is single linked list, next one is double linked list next then, generalized linked list.

(Refer Slide Time: 36:18)



Now, going to discuss about the single linked list so, we have the top node, this node is having the at least two field one is known as data field, another one is called as link. Data field linked list and so, on link points to the next node, there is the starting node is to the linked list.

(Refer Slide Time: 37:26)

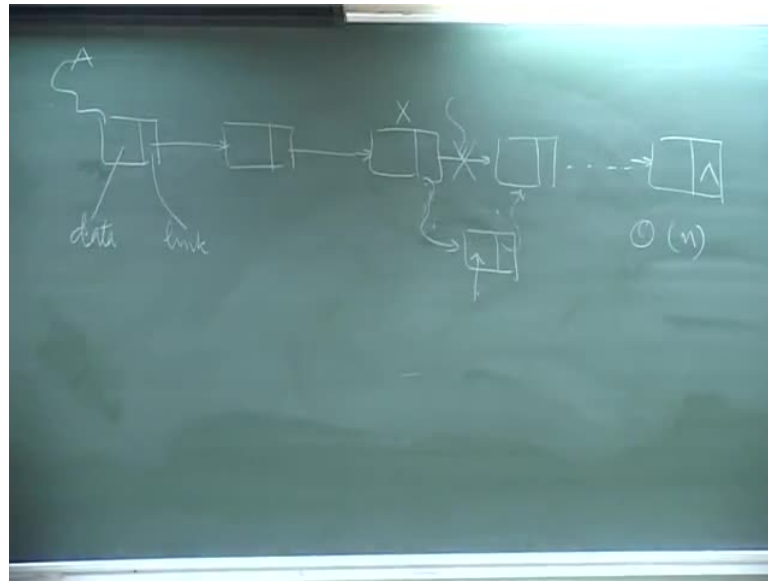


Here, if I am to traverse this linked list let us see what happens, first of all, read all the data. So, if read one by one, it takes order n , order n among of time now, suppose, I want to get the i th element of this linked list then, it travels from the beginning to the higher position, it takes order i times. So, basically you need order n times in the worst case, to move the n th element of the link. Suppose, I want to modify I want to modify the n th element or i th element then, I have to go up to highest position and then, modify the mistakes or the i taps.

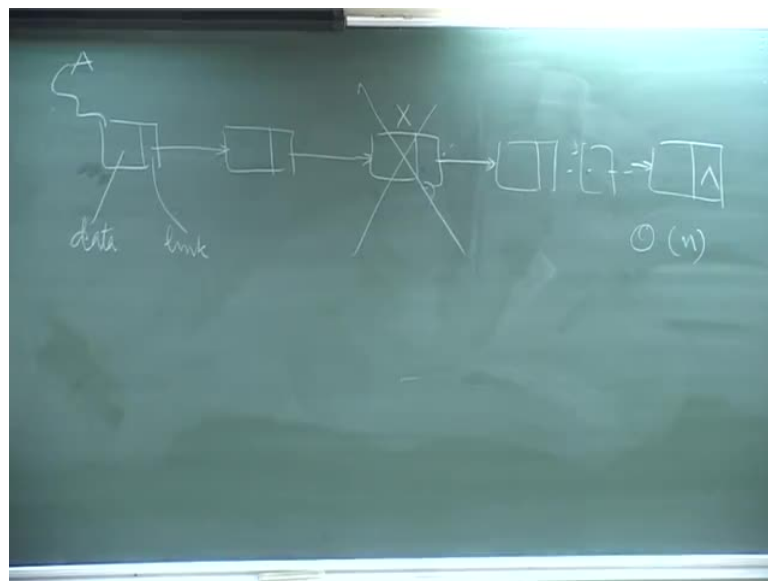
Now, what happens that insertion and deletion, if I do not know way to, which one to be deleted or I know that, it is the highest one to be deleted, I am going to the highest position and then, perform the deletion operation. So, it takes what are the i time to reach them, n th of the your perform the operation similarly, if I have to insert some node here so, the area to reach up to this one and then only, we can able to insert right.

So, let us assure that, this order right can still here plus the insertion or deletion time to be consider right. Now, let us see suppose, I want to insert in node after x , the node x after x but, I go I align, after that by some means is or sometime and then, I will get a node get a node, fill the data.

(Refer Slide Time: 38:57)



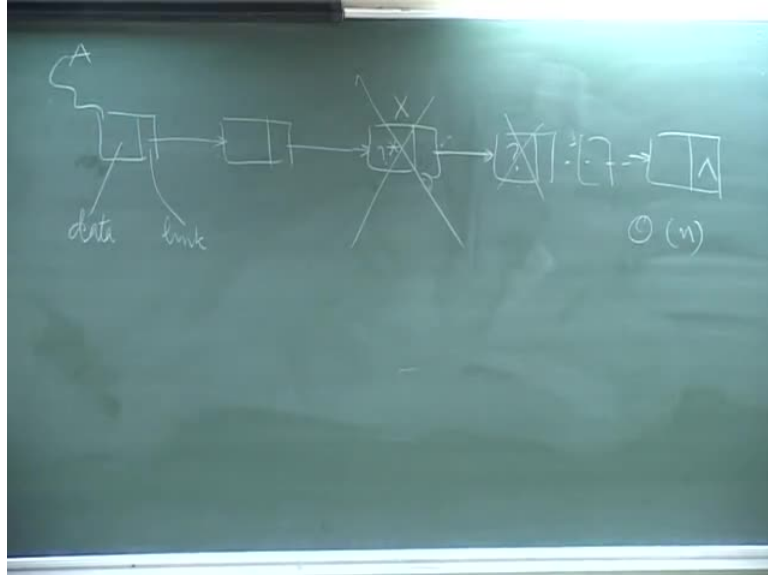
(Refer Slide Time: 39:45)



And then, if I will link this part, if I link this part then, I give the information about this one so, what I have to do, I have to link for this one and then, you will be linking this one. So, link will go, this will be out so, if this will be interchange so, which takes constant amount of time. Now, next one is so, this takes constant time now, suppose, I want to delete the node occurrence, how can I delete the node after x. You have to delete the node after x, what I can do, I will put just the link operate to the link of the next node of the link right. Link of x, we will finding to the link of x of link right, that you know,

which takes constant amount of time now, what i want to ask you the question is differ because, upto this you had done in your under graduate course.

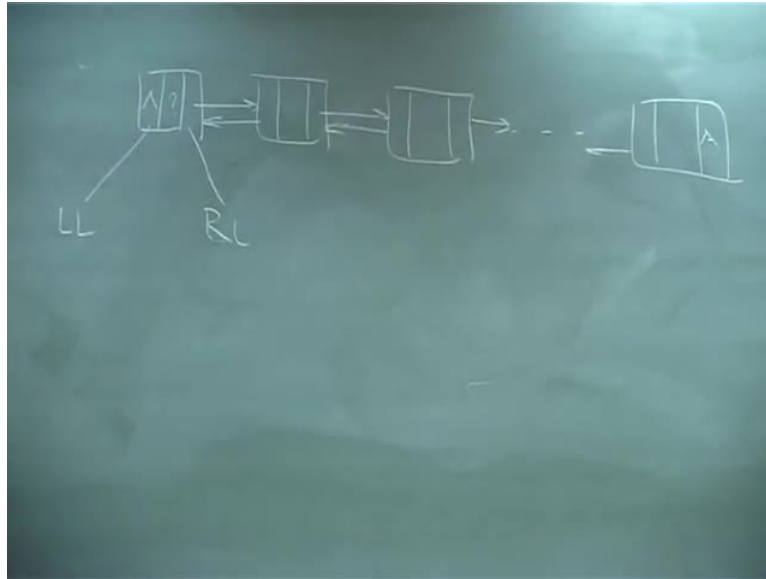
(Refer Slide Time: 40:37)



Suppose, I want to delete this node, how can I delete this node, can I delete it because, if I can do that . So, can i delete this node in constant amount of time, yes, is it difficult. You will understand how we can do it say, is not believing this node because, I can I can go back. So, one could be one we could be then, I copy this data here, I copy this data here and then, delete this node here.

In that case, this takes copy takes constant time and deleting these two takes constant time. So, we can delete this node, the constant amount of time, now, think about the the problem you face in the serial link is of going back, what is very difficult.

(Refer Slide Time: 41:10)

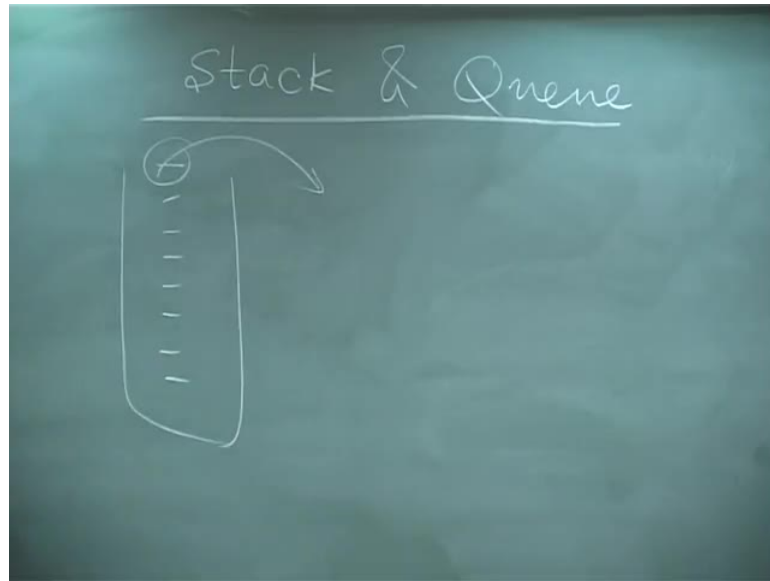


Going backward is very difficult because, you have to start from the beginning so, you know instead of that instead of that, then we did point the toply linked list. Now, doubly linked list is that, get the one is left link, another one is the right link and this is the data field. What is that going backward or forward, is not the problem in the case of the doubly linked list. Now, it is a structure you have to done, which is known as stack and queue right.

See one example could be the, in the library we have seen the books are bind upon the number then, the bookkeeper brings some books again in the cupboard. whenever you have to read a book, what you are going to do so, from that stack of books you will be removing one by one but, that from the top right. See for example, in the dining area coffee or so, when you entered the dining hall, you will find that the plates are stand on the one on the top of another one and so on.

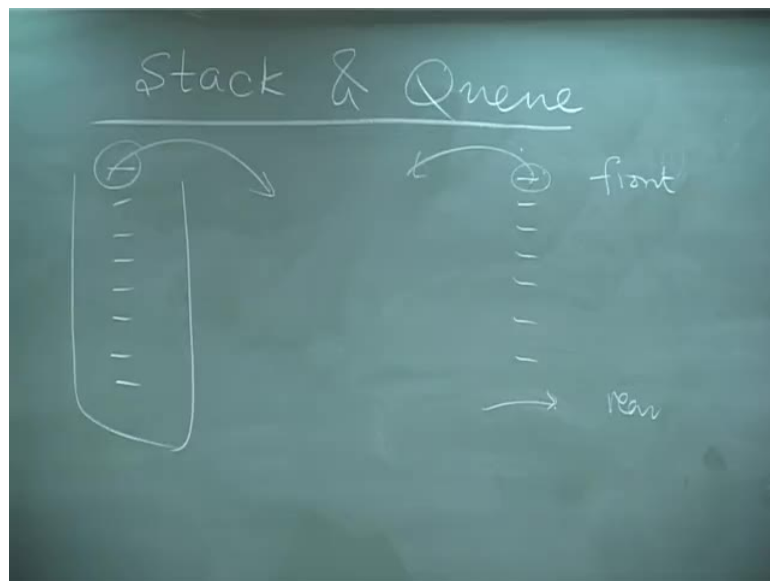
And that, you have, you want to take your food so, you will be taking the plate from the top, again of the dining area. If all sets of plates and you put it on the top of so, that is known as, that is an example of stack.

(Refer Slide Time: 43:36)

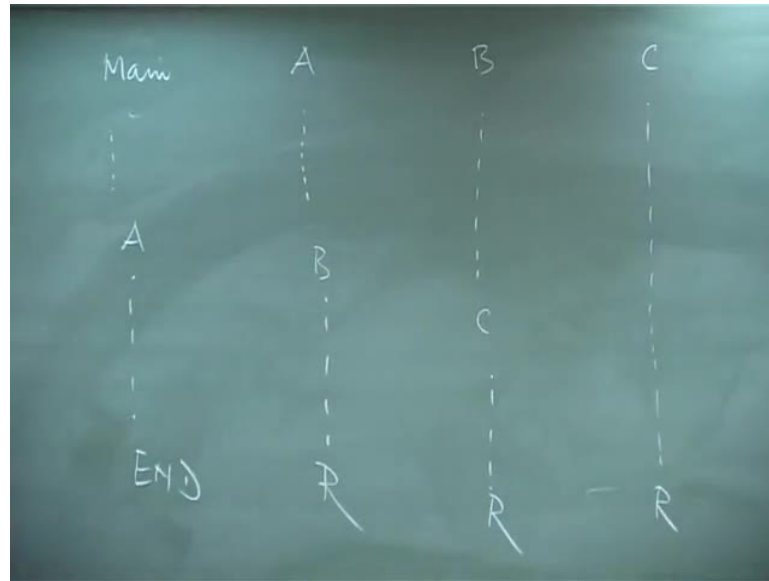


But, the example of queue is very simple, you stand in the movie to get the tickets, the first person gets the ticket in this and the person join to get the ticket, he stands at the end of the queue. Now, we have understand, what happened the stand, the arrangements are like this now, suppose, I want to insert certain element, in the case of stack, we substitute on the top. Suppose, we want to delete this element, the deletion also from the top.

(Refer Slide Time: 44:04)



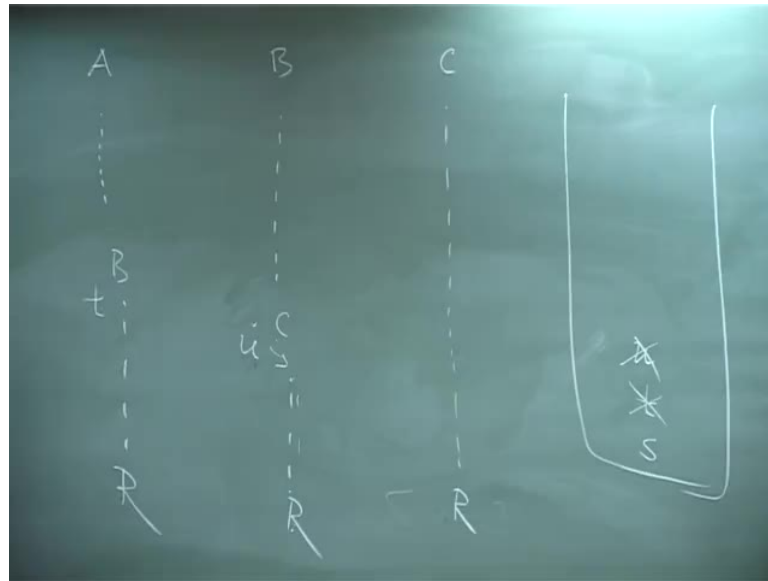
(Refer Slide Time: 44:41)



And if you can take the constant amount of time so in the case of stack, insertion and deletion from one end. In the case of queue, this is if you standing so, another person comes he will be standing at the end and the person coming out is from the front top. So, insertion will be at one end and deletion will be from another end, this is the definition of queue.

See, an example for this stack would be that sub-routine volume so, we have a for main program and we have a routine, here you have calling sub-routine A then, you have sub-routine A, you have calling a routine B, here we have calling C and C is calling here.

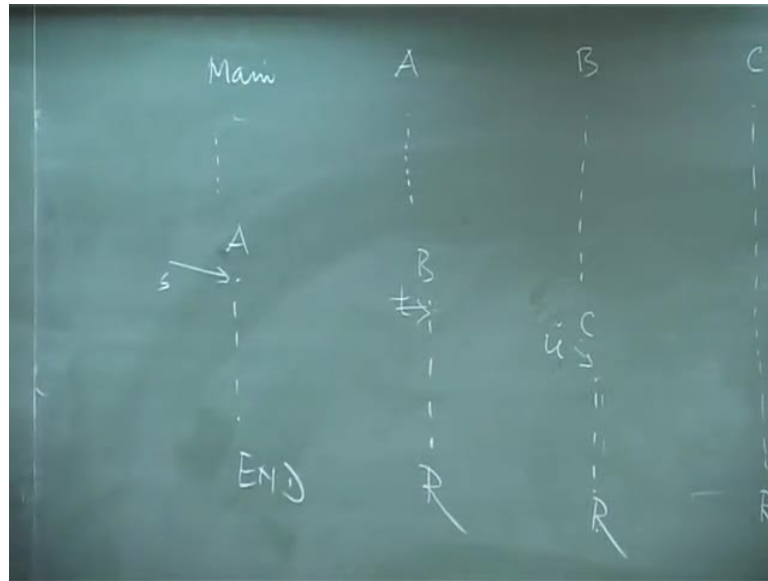
(Refer Slide Time: 45:23)



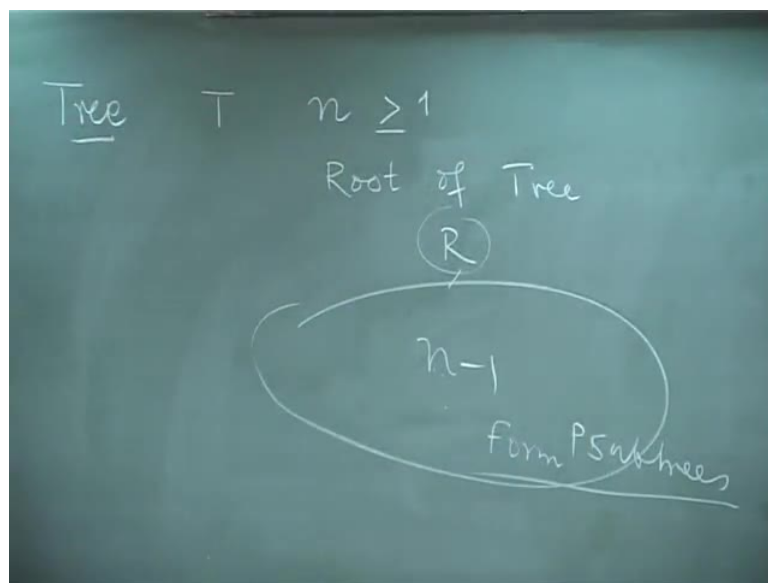
So, here you have the return statement so, return statement here, return statement and here, return end. So, what happens, why here is called A so, the system deserves the next statement, adjust the next statement. So, say, next statement say s, then this is s now, it goes to this one, end of completing and get the address of the next statement of B, it is to be t so, it stores t. So, it is called b and we get c and can be u in the next statement of c. so, here it is stores u.

So, it perform the c now, it assure it has return statement, just comes back to the next statement of C. That is, the adjacent available in the stack, it is u so, it swaps off that u and he comes here, it gets to return statement. Now, they should return to the procedure A and it gets t from the stack, it comes here then, it gets the return statement, they should come back to the main routine to the next statement to execute with this so, that is the one example where stack is used.

(Refer Slide Time: 46:42)

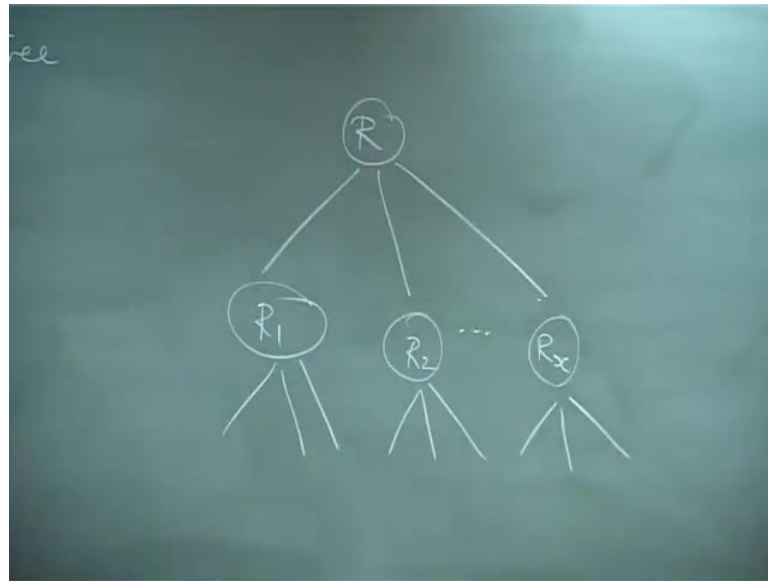


(Refer Slide Time: 46:57)



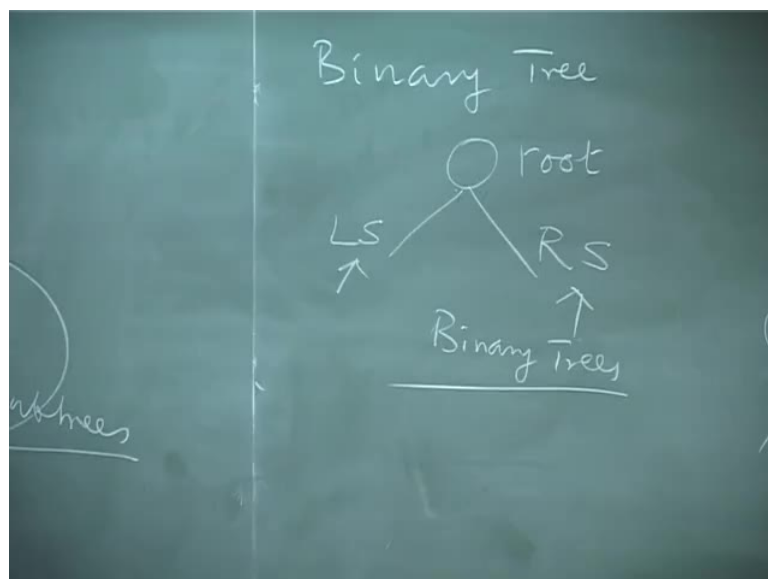
Now, we have or we know the term tree, what is tree, the next terminology we will be discussing the binary tree. Tree tree for takes n nodes, n is atleast 1 node and 1 node is the node is the root of the tree root of the tree and this is specially, dedicated node which is called as root of the tree. And we have the remaining n minus 1 then, connected they are connected to root r , they are connected to the root node r and most of this n nodes form a tree, form a sub-trees, this n nodes form a p sub-trees.

(Refer Slide Time: 48:23)



P is basically, we have a root node r root node r under that, there this some other root node. So, if we this then, we get x disjoint sub-trees. If I this again we get some number of sub-trees and so on. Only condition is that there is atleast 1 node in the tree So, in the case of binary tree, empty empty empty binary empty binary tree is also a binary tree.

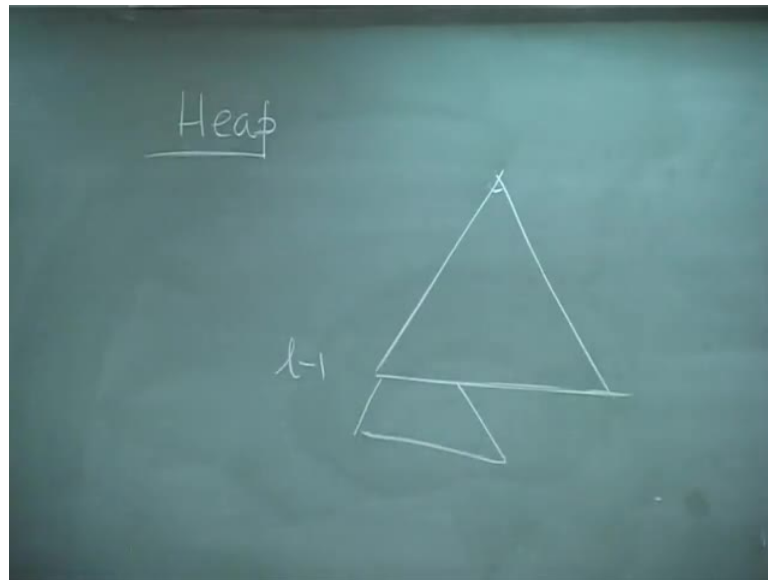
(Refer Slide Time: 49:18)



So that, you can define what is binary tree otherwise then, is specially node will be getting is root node. root node and it is utmost two sub-trees, one is known as left sub-

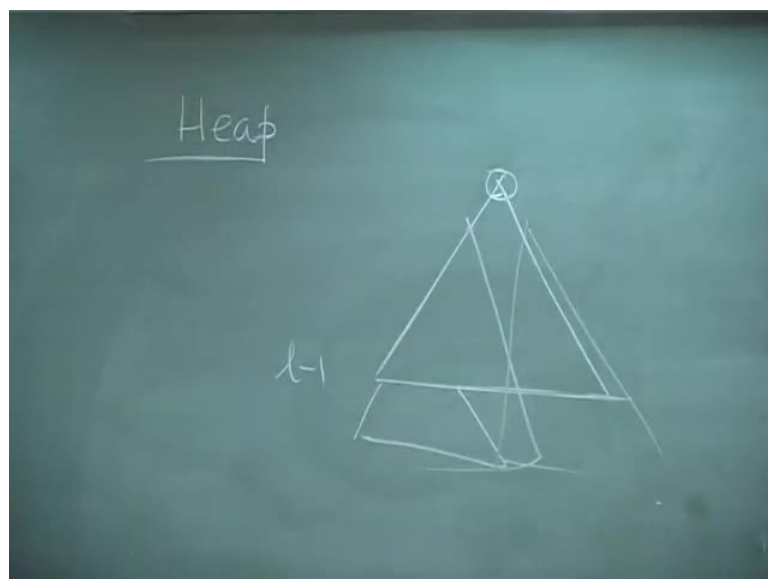
tree, another one is right subtree, left sub-tree and right sub-tree. And left sub-tree and right sub-tree both are binary tree. We have the root node, left sub-tree, root node it has utmost two children. So, the left child if exist, is the root of the left sub-tree and if the right child if exist then, it is the root of the right sub-tree such that, it is a binary tree.

(Refer Slide Time: 50:18)



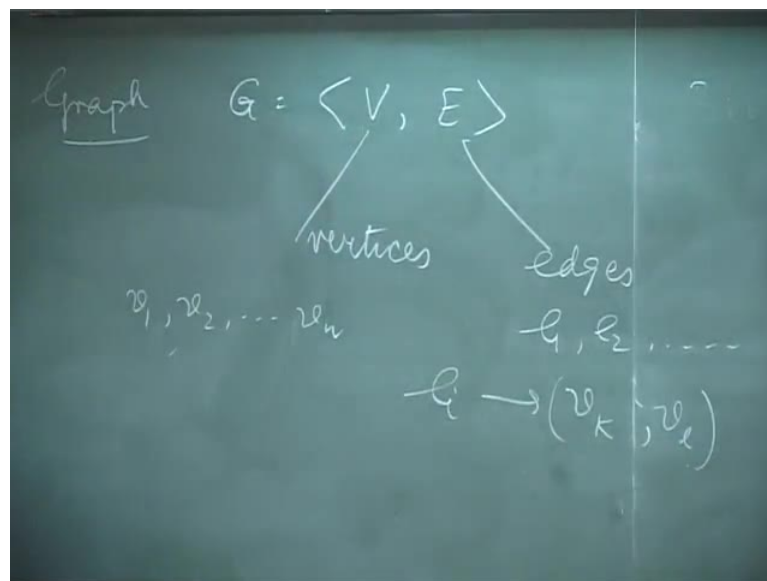
Do you recollect what is heap, heap is a complete binary tree now, what is complete binary tree, complete binary tree means that, it is full that last for one level and in the last level in the last level it is full from from left to right.

(Refer Slide Time: 51:15)



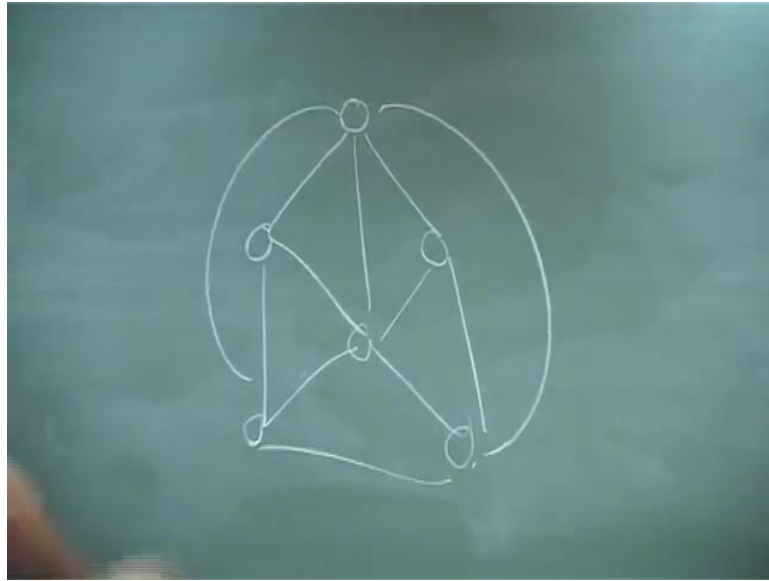
So, heap is a complete binary tree of level and it satisfies certain properties, one property is in case if min heap, this is the minimum element minimum element or this able to scroll of the both element of left and right child and also, this is left sub-tree in a tree and right sub-tree also . So, this is a complete binary tree and this slope contain, in case of min heap it contains of minimum element compare to this tree, compare to right child and left child and left sub-tree is obedient, right sub-tree is also obedient.

(Refer Slide Time: 52:05)

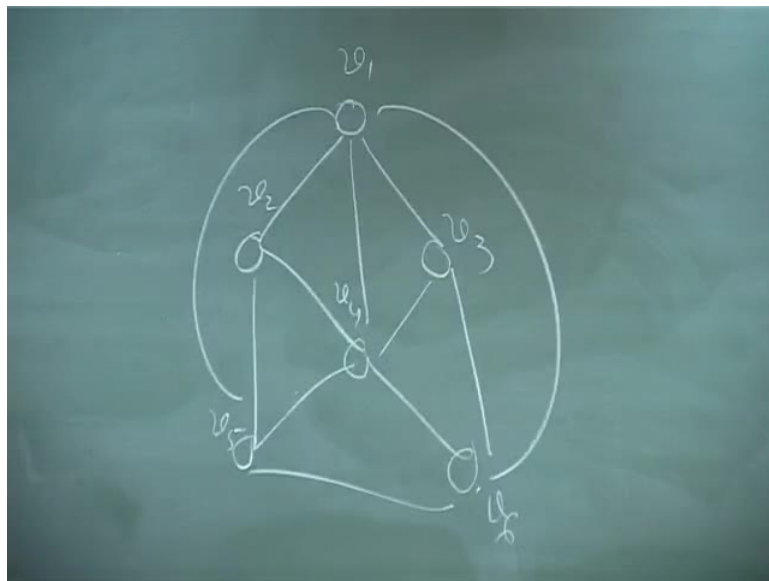


Graph, we define G by V and E where, V is the set of vertices and E is the set of edges. V is set of vertices suppose, you have n vertices, they are defined as the v_1, v_2, \dots, v_n and you have edges e_1, e_2, e_3 and so on from the edges. And e_r is the edge between v_k and v_l right.

(Refer Slide Time: 52:58)

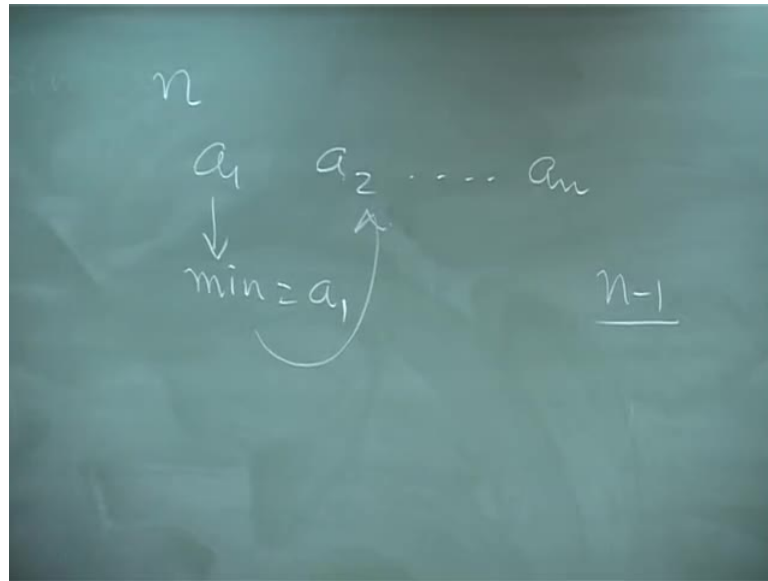


(Refer Slide Time: 53:19)



This is a structure and this a possible here, you can have $v_1, v_2, v_3, v_4, v_5, v_6$ and so on and we can take e , e is also a graph between. Now, can you tell me, what type of graph between, e is a simple graph, if e is a cyclic graph, can you find out can you tell me the number of edges you have there.

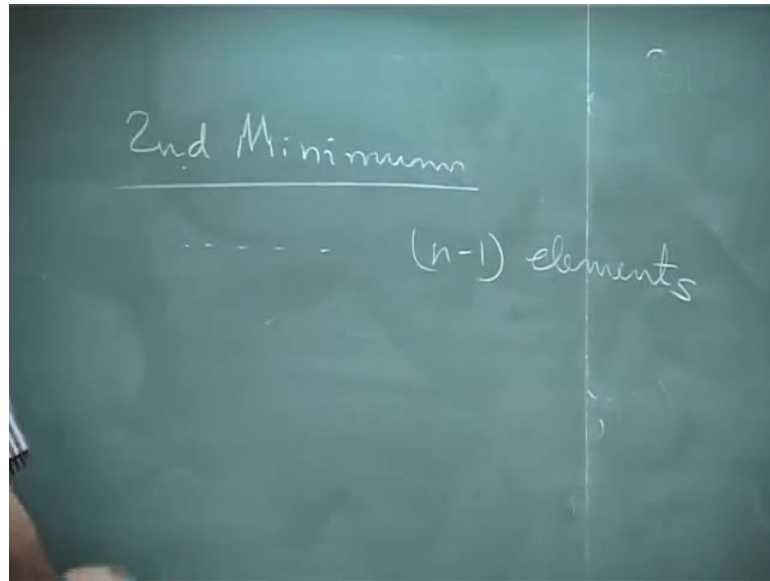
(Refer Slide Time: 54:40)



Then, if you have n nodes then, number of edges will be n minus 1 in the field. Because, if these are connected acyclic graph, which takes, which needs n minus 1, to maintain the connectivity. And if i add 1 is then, it will become the cyclic graph so, that is the this always graph v and e just we defining. So, that why this curving of another now, let us understand, a simple program and see, how you can solve it.

Then can you, do you remember the how to find out the minimum of n elements or suppose, you have n element and you have to find out the minimum of this n element, one possible way is that, you have a 1, a 2 and a n . Initially, you defined minimum is a 1 then, minimum is compared with a 2, if you find a 2 is smaller than min then, min is different by a 2.

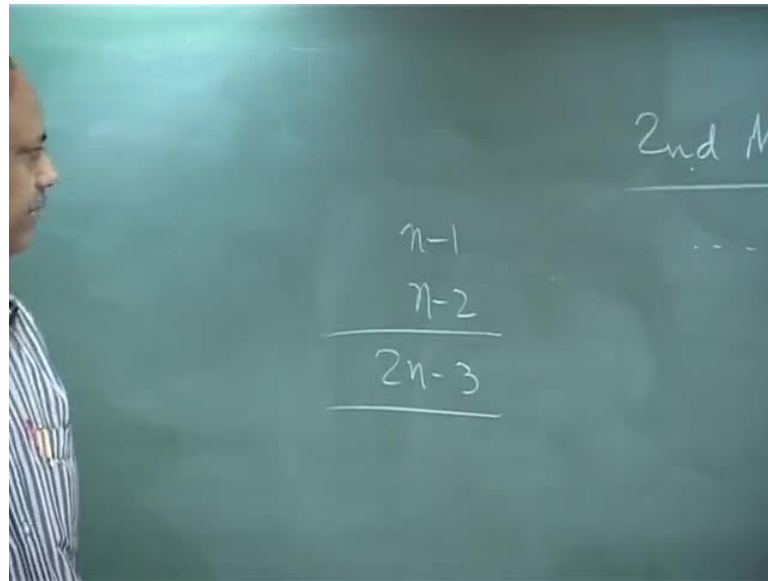
(Refer Slide Time: 55:33)



And then, you compare again with a 3 and if you find a 3 is greater than min, do not do anything, just go for the next one and so on. So, in that process, you need $n - 1$ comparator right, to find out the minimum of n elements. Now, the programs is here suppose, we should find out the minimum of n elements, I am interested to find out the 2nd minimum second minimum.

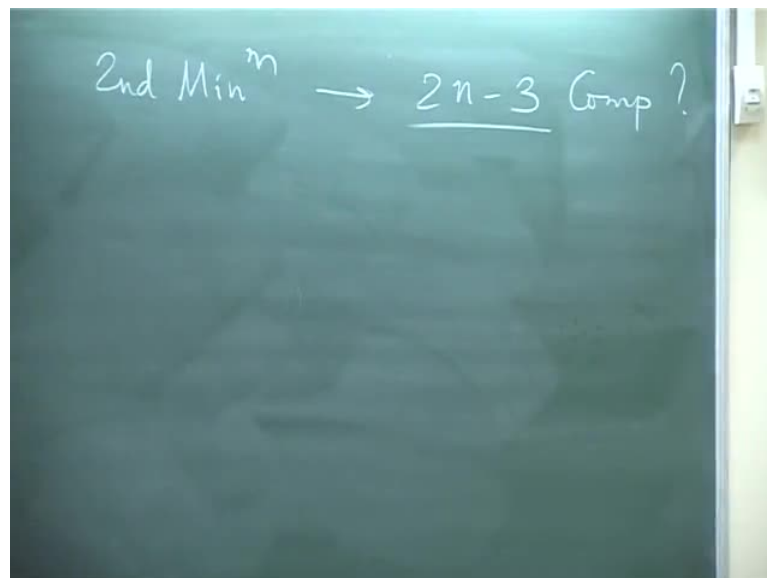
Now, can you tell me, how can I find out the second minimum of this n so, one possible way is that, you first complete find the minimum. Now, you have the remaining $n - 1$ elements and again you find the minimum, that will be your 2nd minimum. So, in the for the first case you give $n - 1$ competitor.

(Refer Slide Time: 56:01)

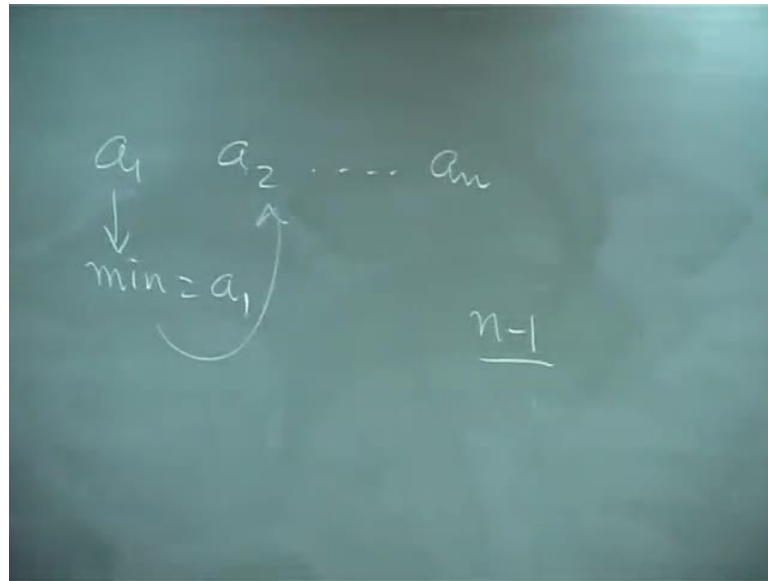


Second time you give n minus 2 competitors so, you get $2n$ minus 3 competitors to find the 2nd minimum. But, is this the best way to do it so, let us conjugate this problem that finding the 2nd minimum.

(Refer Slide Time: 56:15)

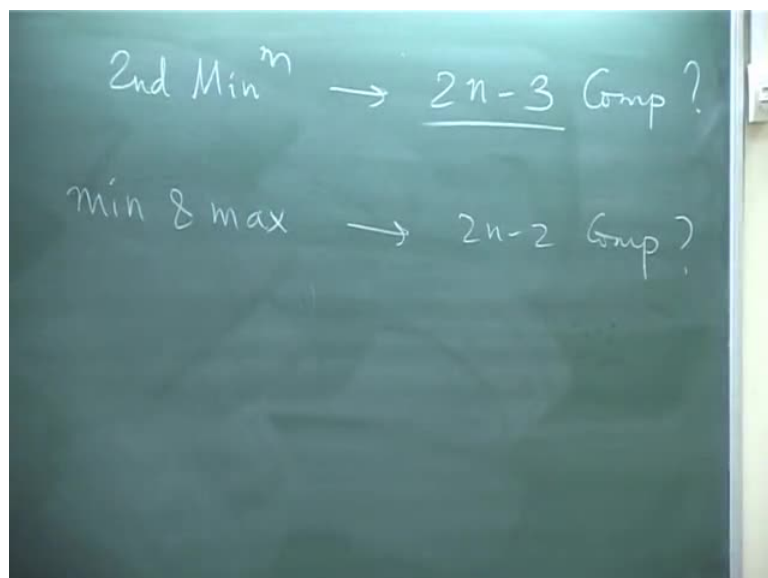


(Refer Slide Time: 56:48)



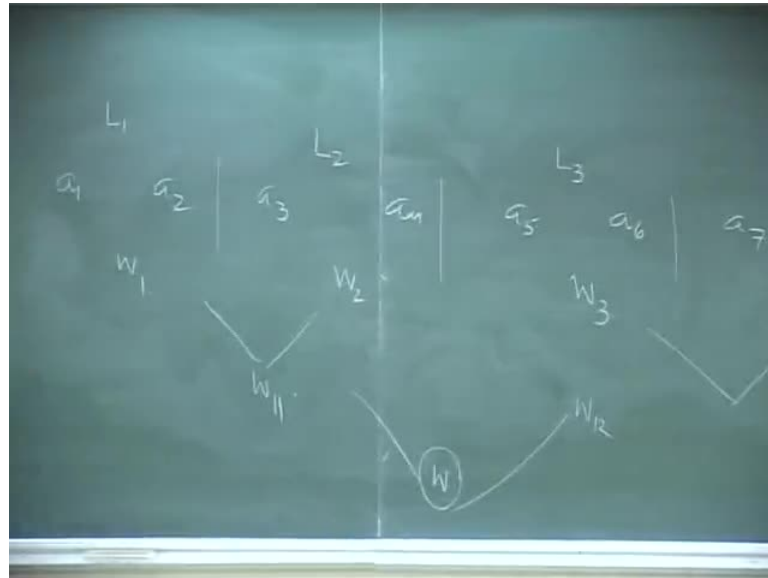
And this moment we discuss that you have seen that this many comparisons you need and our time it is to find out, whether this is the best possible way for finding the second minimum. Now, let us consider the another form suppose, I am interested to find out minimum and maximum of n elements. Now, one way possible that, you find the minimum of this, you check n minus 1 and again, you find the maximum, you get again n minus 1.

(Refer Slide Time: 57:04)



So, to find the minimum and maximum minimum and maximum of n elements, you need $2n - 2$ comparisons right but, is this the best way of doing? Answer is no.

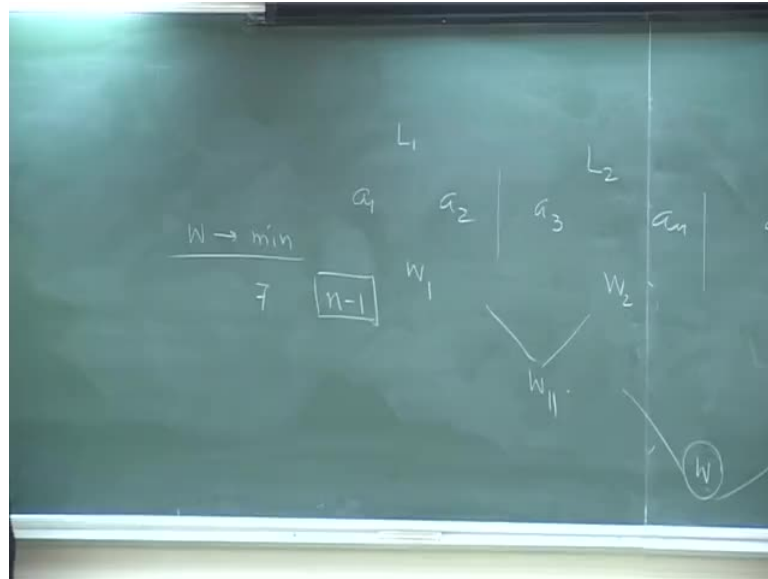
(Refer Slide Time: 59:16)



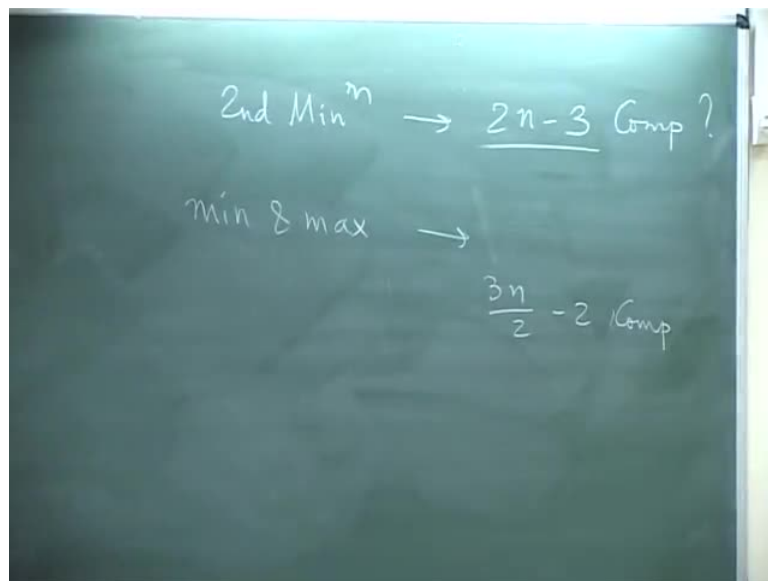
In order to do that, another that you have a 1, a 2, a 3, a 4, a 5, a 6, a 7 and a 8, let us assume that you have this 8 elements, for for simplicity for simplicity let us assume that they were 8 teams are participating in one game. And you want to introduce a knock out tournament among this to find out the who is the winner right. So, in order do that, what we do, we divided into 4 groups, each groups consists of two teams.

And these two teams of each group play each other and one of them will be declared as winner and one of them will be declared as looser, get it. Now, in order to see the winner of this so, this four will be divided into two teams, two groups and the winner is, a winner of these two be here and these two teams will play each other and to be here, to be the winner, that is the procedure.

(Refer Slide Time: 59:30)



(Refer Slide Time: 60:21)

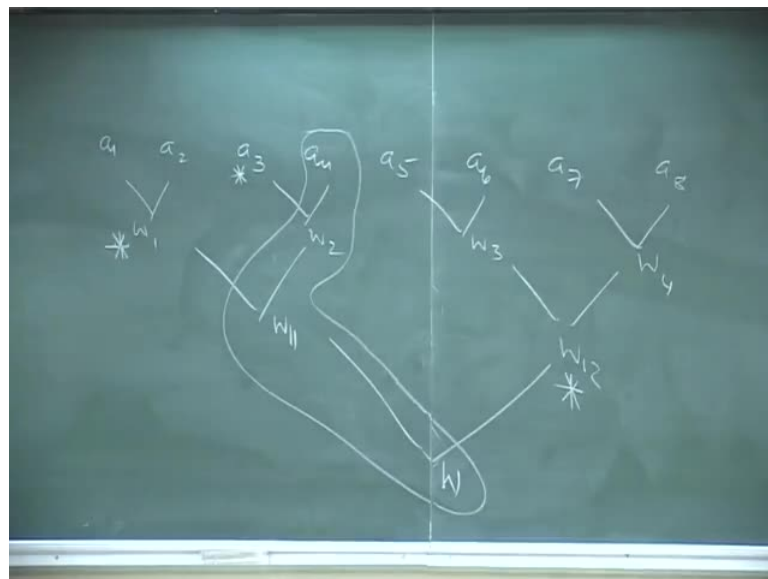


To find out the winner of the same teams by knock out method now, it plays winner by replace w by \min , what happened. So, this itself is the minimum of these 2 elements, minimum of these 2 elements, minimum of these 2 elements and minimum of these 2 elements. So, minimum of these 2 elements is here and minimum of these 2 elements will be here and the minimum of these 2 elements is the minimum of all these 8 elements agreed.

So, you observed the number of comparisons you have used, here 1 2 3 4 5 6 7. So, 7 comparisons you have used that is, n minus 1 comparison to find the minimum, agreed. Now, in order to find out the maximum of these, you observe the maximum cannot be any one of them because, they are the minimum. So, maximum will be one of these 4 elements now, one of these 4 elements means that, you have to find out the maximum of these 4 elements.

That 4 elements are nothing but, n by 2 elements and finding the maximum of these 4 elements is n by 2 minus 1. So, the number of comparisons help me to find the maximum and minimum of n elements is $3n$ by 2 minus 2 so, it is much better than much better than $2n$ minus 2.

(Refer Slide Time: 61:10)

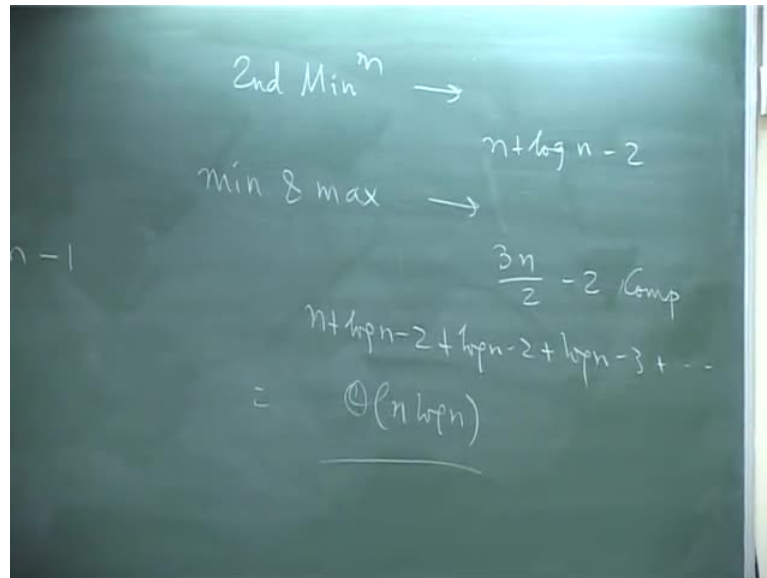


Now, what about what about the another one that finding the, so I loop into this problem on one over the another, you get a 1, a 2, a 3, a 4, a 5, a 6, a 7 and a 8. And you have for winner 1, winner 2, winner 3, winner 4 then, winner winner of these two, winner of these two and then, W is referred by minimum.

Then, suppose, this is your minimum movement minimum movement so, to find the 2nd minimum the the problem will get, this is the problem you can detect and this is the another problem you can detect and this is another problem you can detect right. Because, this is minimum of minimum of these 4 elements, this is one case and this is the

minimum of these 2 elements so, you have that many possible candidates, among that you have to find out the minimum.

(Refer Slide Time: 62:16)



So, this you observe that every height gives 1 element, every level you get 1 element for this level this one, for this level this one and this level this one. The height of the state is $\log n$ minus 1 height of the state is $\log n$ minus 1 so, to find the minimum you get n minus 1 then, height of the tree is $\log n$ minus 1 and to find the minimum, you get another minus 1. Height of the tree is $\log n$ plus 1 plus 1 so, you get $\log n$ minus 1 comparison.

So, 2nd minimum takes n plus $\log n$ minus 2 as many comparisons so, this is the but, again to find the 3rd minimum you have to go for you have to do one by one again. So, it will become $\log n$ minus 2 then, $\log n$ minus 3 and so on so, n plus $\log n$ minus 2 plus $\log n$ minus 2 plus $\log n$ minus 3 and so on. So, you will find that suppose, I have to find out the 1st minimum, 2nd minimum, 3rd minimum upto n minimums so, this becomes order of $n \log n$, which is basically nothing but, 1st minimum, 2nd minimum, 3rd minimum and so on.

It is, we are basically, making the sorting of n elements and which is lower bound of this order $n \log n$, which is matching with that. Let us stop here, and in next class we will be discussing about different paradigms, we used for sequential algorithms. And then, in the

3rd class onwards, we will be discussing starting about the parallel algorithms and what are all the things we drop are in the notes.

Thanks.