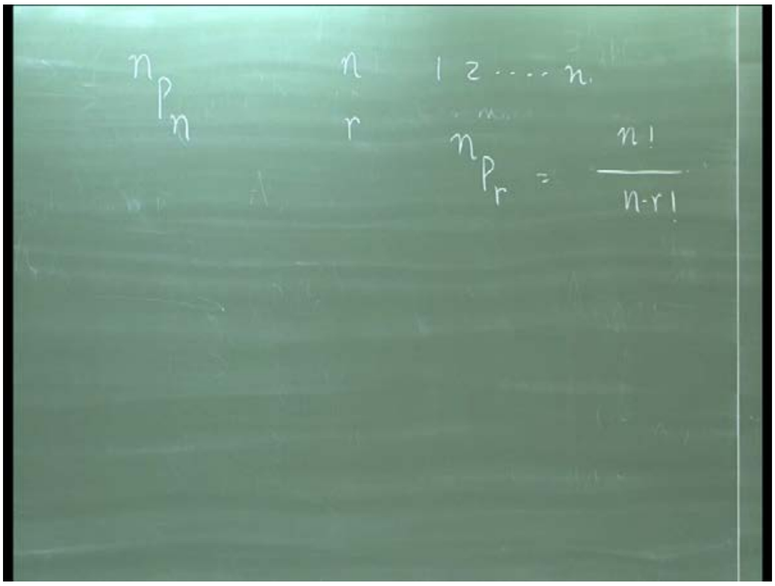
Parallel Algorithms
Prof. Phalguni Gupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 20

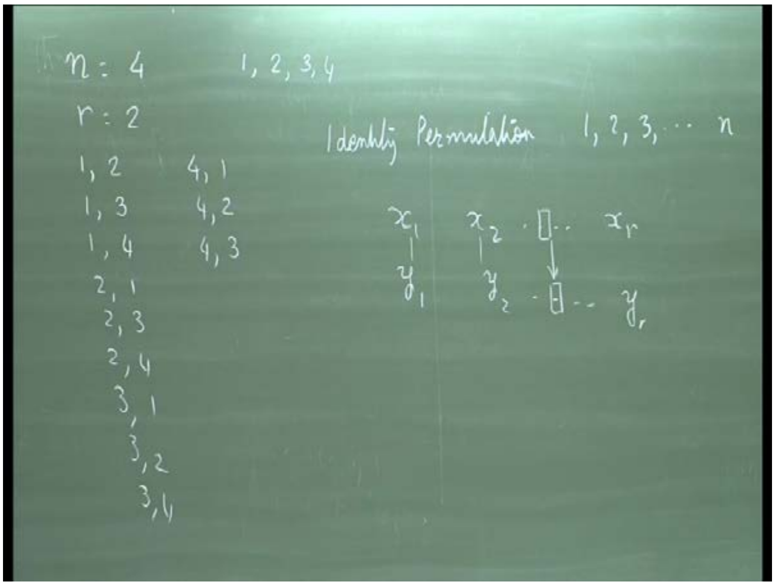
And in this part what you will be discussing is how to generate the sequence of permutations for permutation generations technique, combinational generation techniques, real generation techniques. The thing that they have presented permutation, this should be known to you that permutation is the arrangement of r elements out of n, right.

(Refer Slide Time: 00:50)



So, if you use the inverse then if I have n elements that elements are numbered 1 2 n and if you also arrange this n elements there were n p n possible ways of distinct ways you can arrange them. Now more generic one the given n elements 1 2 3 up to n I want to make the arrangement of r elements, right. So, the possible number of distinct permutation will be n p r, right, and this n p r is basically n factorial by n minus r factorial.

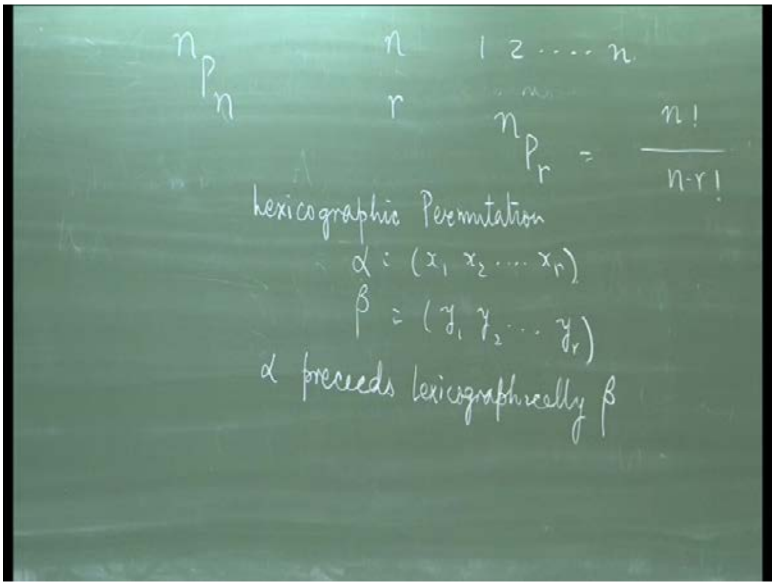
(Refer Slide Time: 01:49)



So, for example suppose I have n equals to 4 and r equals to 2 then possible arrangements will be, say, n equals to 4 the elements are same, 1, 2, 3, 4 then possibility equals to be 1, 2; 1, 3; 1, 4; then 2, 1; 2, 3; 2, 4; 3, 1; 3, 2; 3, 4; 4, 1; 4, 2; 4, 3. So, these are the possible elements you have, right. So, what is the basic definition or definition of permutation is nothing but it is arrangements of r elements out of n; that gives you the permutation, and in the term we will be using identity permutation. This identity permutation of n elements is nothing but combination of 1, 2, 3 up to n. This is the identity permutation. So, the two permutations are distinct; we can guess that the two permutations are distinct.

One that the existence of one element is different from one permutation to another permutation; that means that suppose I have one permutation x 1 x 2 x r and another permutation is y 1 y 2 y r, these two are distinct. One possibility is that all of them may be same except one of them is different, right. The occurrence of all of them is same, right. This is may be one two three four five six up to r, here it is one two three four five six seven eight up to r minus 1 and this is r plus 1. Then you can tell they are distinct or the arrangement of the same elements is different; that means all these elements occurred here but occurrence position is different, right. For this example that 1, 2 is a permutation, 2, 1 is also another permutation, but the occurrence existence of all these elements are here same; only the ordering is different, right. If the ordering is different, then it is a permutation or the elements are different, 1, 2 and 1, 3; these two elements are different, then also it is a permutation. Now this one word will be using at whole is lexicographic permutation.

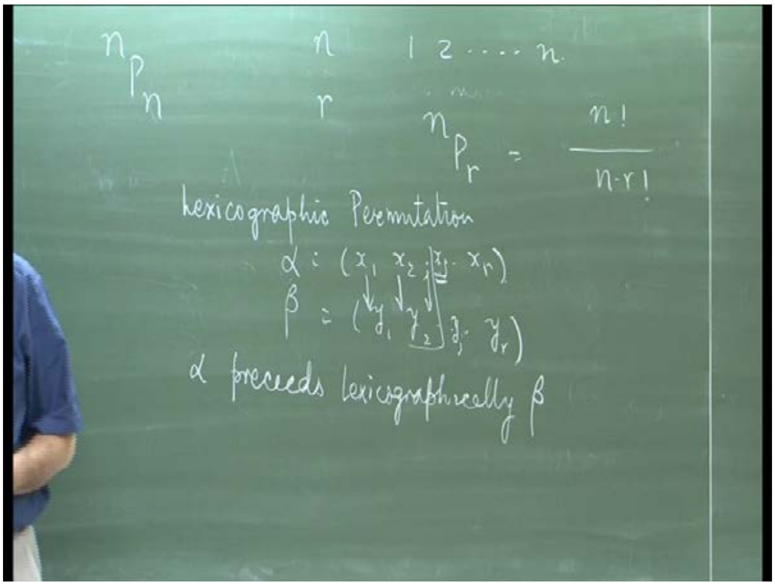
(Refer Slide Time: 05:12)



So with lexicographic, suppose alpha is the permutation of r elements and beta is another permutation of r elements, then we tell alpha precedes lexicographically beta.

(Refer Slide Time: 06:14)

(Refer Slide Time: 06:52)

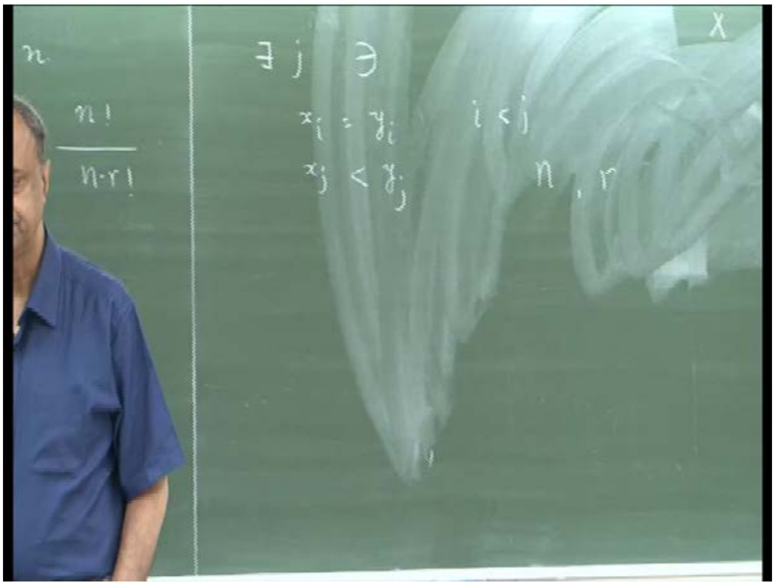


If there exist j such that x i is equals to y i, i less that j and x j is less than y j. So, we tell alpha precedes lexicographically beta if there exist a j such that there exist a j here x j y j such that all of them are same; this is same, but this one is less than this one, okay.

(Refer Slide Time: 07:12)

So, a permutation x we tell it is a gamma x lexicographic permutation, right. The permutation we tend to the gamma x lexicographic permutation when there exist exactly gamma minus 1 permutation precedes lexicographically your x, got it; that x is a permutation we tend to the gamma x is lexicographic permutation then there exist an exactly gamma minus form permutations lexicographically preceding the x, right. That is why it is written as gamma n permutation. So, we will give one example, say, I have n equals to 5 insert r equals to 3. Now if I have to write down all the lexicographic permutation then I can write it is 1 2 3, 1 2 4, 1 2 5, then 1 3 2, 1 3 4, 1 3 5, then 1 4 2, 1 4 3, 1 4 5, right, then I can write 1 5 2, 1 5 3, 1 5 4 and so on. So, this is about your definition of lexicographic permutation.

(Refer Slide Time: 09:12)

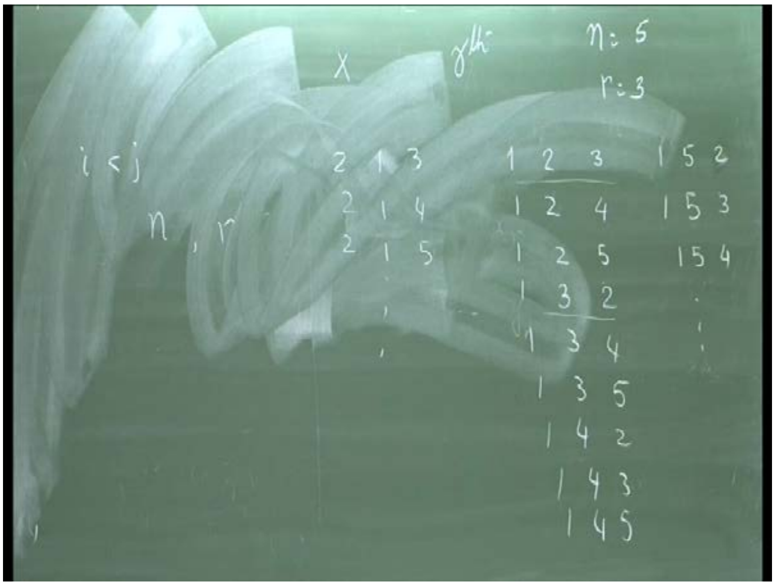


Now what is the problem? Problem is that given n and r I want to generate all permutations, right, this number may be very large, so I need the help of parallel machines. So, can we have parallel algorithms to generate all n p r arrangements? So, the first algorithms actually there are two algorithms which came out together and one of them is order algorithms, and let us discuss first the order algorithm then we will come to all of this. This is the first algorithm whatever we were presenting this is not based on the lexicographic idea; it will give you all set of permutations and at each iteration it will do some reposition, so that after r iterations you will be getting a permutation of r elements.

(Refer Slide Time: 10:37)

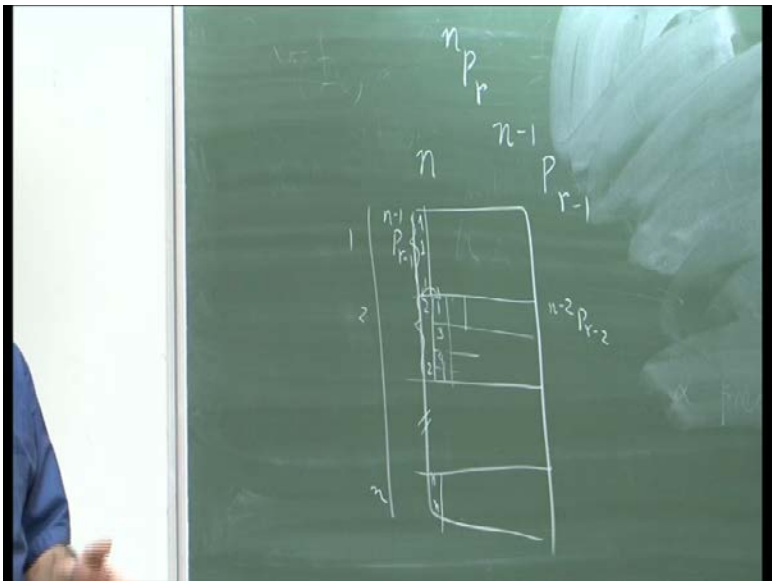
What I am telling is that it is an r iteration factor. At each iteration it interchange one element with the another one. So, after r iteration he gets all the permutations, right; that is the idea. Now one thing you observe that if I have n p r permutation, this permutation makes use of the arrangement of r elements, right, but it is not like combinations. You can think that in this case it obeys certain distributions all 1's will be there irrespective of the same occurrence of elements.

(Refer Slide Time: 11:36)



Say, 1 2 3 and 1 3 2, the occurrence of elements are same but only ordering is different. So, if I observe one two three four five six seven eight nine ten eleven twelve; 12 of them starting with one’s then you will be getting 12 of them starting with two’s, 12 of them starting with three’s and so on, right. So, for example this one will be 2 1 3, 2 1 4, 2 1 5 and so on. There were 12 such cases with two, right.

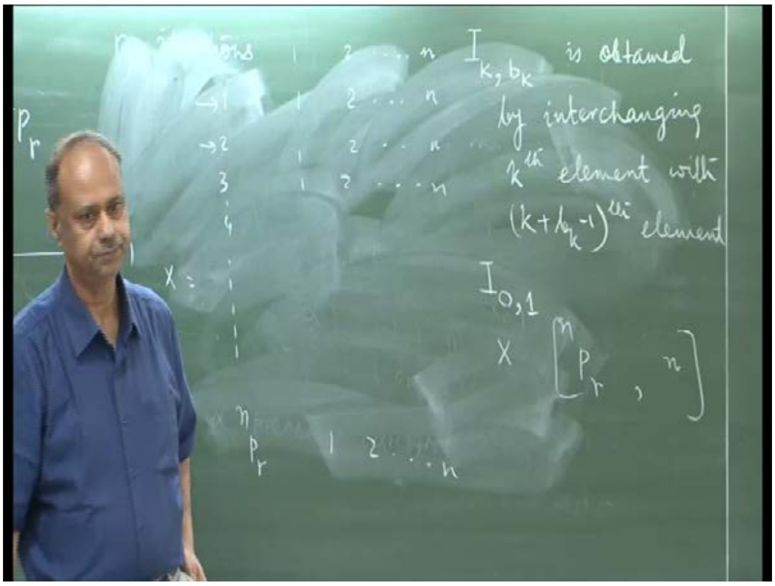
(Refer Slide Time: 12:23)



Basically that if I have n p r arrangements then there are these n p r elements can be divided into n blocks. Each block is having sign n minus 1 p r minus 1, okay. Now this what I did? You have n p r; I can divide into then n minus 1 p r minus 1 size blocks. How many size blocks? n blocks, n blocks each of size n minus 1 p r minus 1. Now the first block one block contains the element one, first element is one. Now second block contains the first element 2 and n th block contains the first element n, is it okay. So, there are n blocks; first block contains one, second block first contains two, and nth block contains first element n. Now I have to make some arrangement such that this two come as first element, n come as the first element and so on, right.

Now that if I know that first element is r or two now here that whatever I will be putting all one’s here just I am interchanging these two elements, I have put all ones, right, and two comes with first element, is it okay. So, how many elements are having one’s; that is n minus 1 p r minus 1. Now among these n minus 1 p r minus 1 whose first element is two, second element all of them will not be one; some of them will be one, some of them will be three, some of them will be four but not two, because two has already occurred. So, what I am doing I will divide it again this one into n minus 2 p r minus 2 size group; that means n minus 1 groups each of size n minus 2, right. Now among this these elements will be having one, these elements will be having three, these elements will be having four and so on, okay. So, that is the idea we want to introduce. So, you proceed it with r iterations up to r iterations you will be getting all the desired elements.

(Refer Slide Time: 15:43)

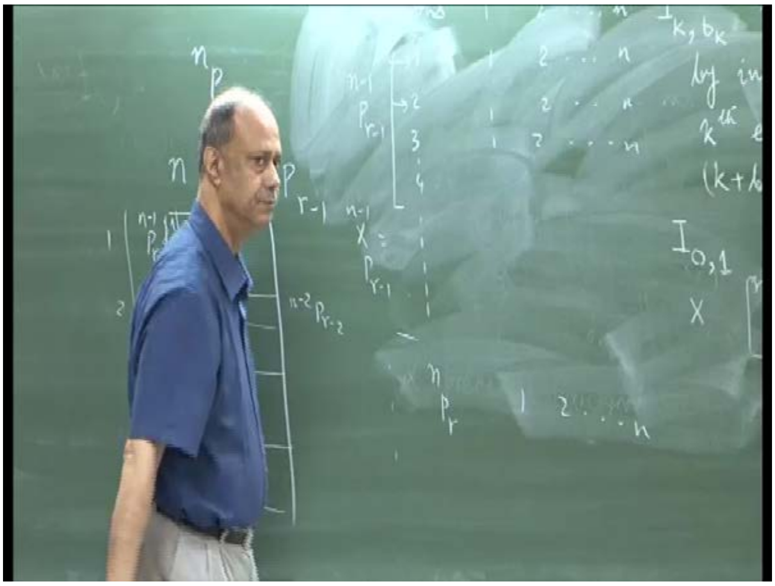


So, let us define first a tower I k b k; I k b k is obtained by interchanging the k th element we kept as b k minus 1. At this moment we will assume that b k is one number one integer for this b k we have integer. So, I k b k is obtained by interchanging the k th element with k plus b k minus 1 th element. Now initially I 0 1 means there it is nothing; it is not doing anything because k is zero and b k is one this is also zero. So, zero th element is interchanged with zero element, nothing is done, so that is a stationary one. And I also assume x, what it gives you is x is the old array of size n p r, n. So, it is a two dimensional array of size n p r and n; that means you have 1 2 3 4 n p r and here this side you have 1 2 3 up to n.

So, this is a large matrix two dimensional array, that is your x, and initially you assume that each row contains n th identity permutations. Each row contains the identity permutations, what it means? That it takes 1 2 3 up to n, 1 2 3 up to n, 1 2 3 up to n, initially you assume the identity array. So, this is I 0 1, this is I 0 1, this is I 0 1. What is I

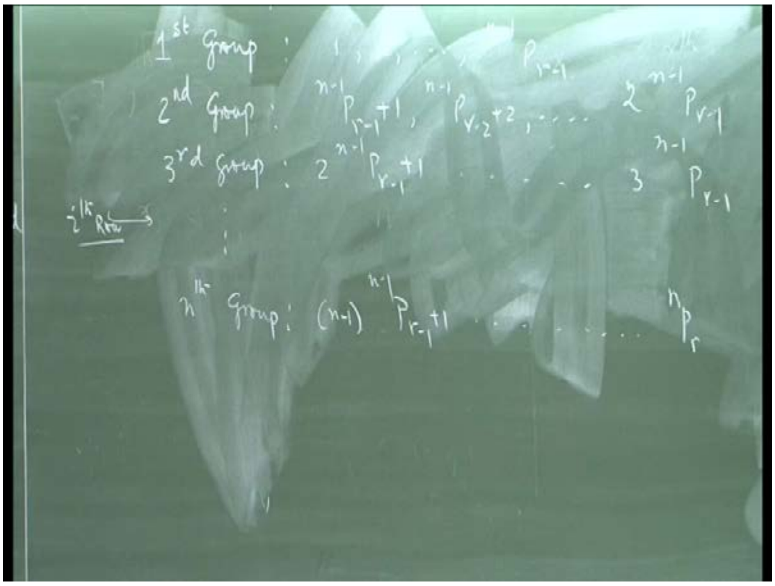
0 1? I 0 is the identity permutation and what is x? x is a two dimensional array each of size, size of the array is n p r cross n, right. So, as I told you that it is r iteration algorithm. At the first iteration I divide these n p r rows into n groups, and this division is equally done.

(Refer Slide Time: 19:41)



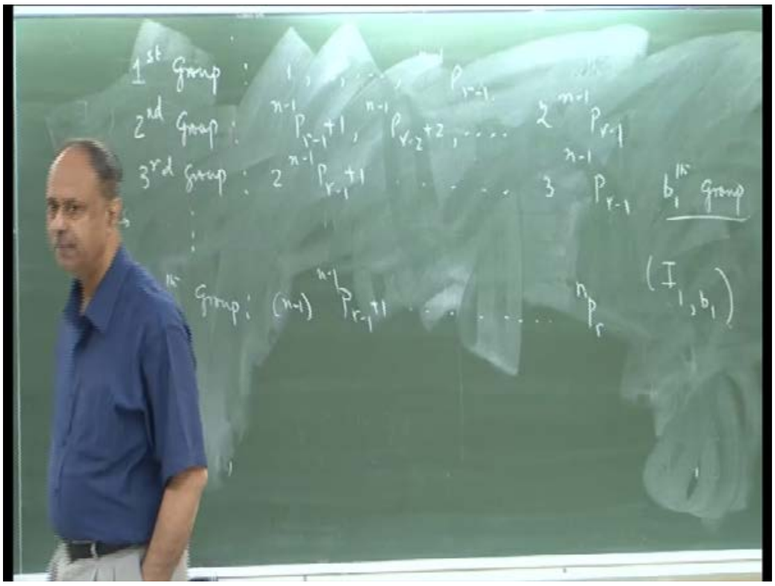
So, that means here first group of size n minus 1 p r minus 1, second group is also size n minus 1 p r minus 1 and so on.

(Refer Slide Time: 19:58)



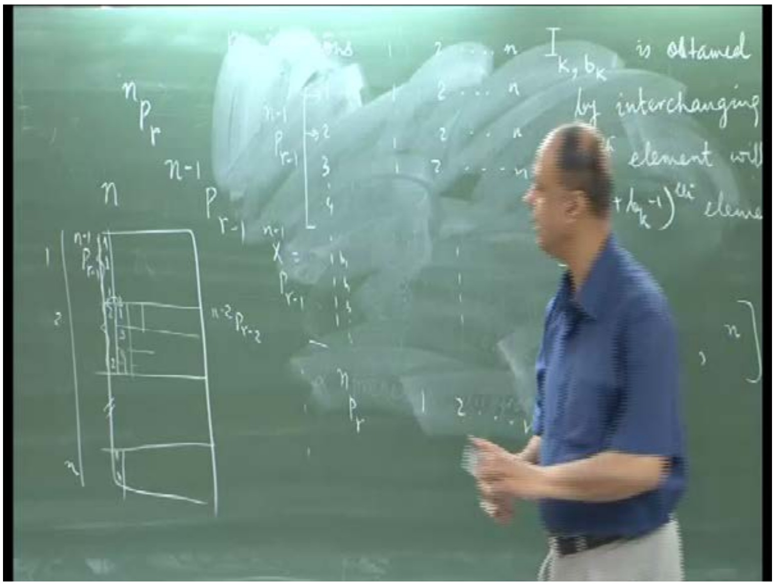
So, first group contains the elements 1 2 up to n minus 1 p r minus 1; second group contains n minus 1 p r minus 1 plus 1 n minus 1 p r minus 2 plus 2. Similarly, you have the third group 2 n minus 1 p r minus 1 plus 1 p n minus 1 p r minus 1 and you have nth group, right. So, then what I did? I divided this r n p r rows into n groups and the first group contains the rows 1 2 3 n minus 1 p r minus 1. So, think about the i th row, so i th row let us assume that is in the b 1 th group, i th row is in b 1 th group.

(Refer Slide Time: 21:46)



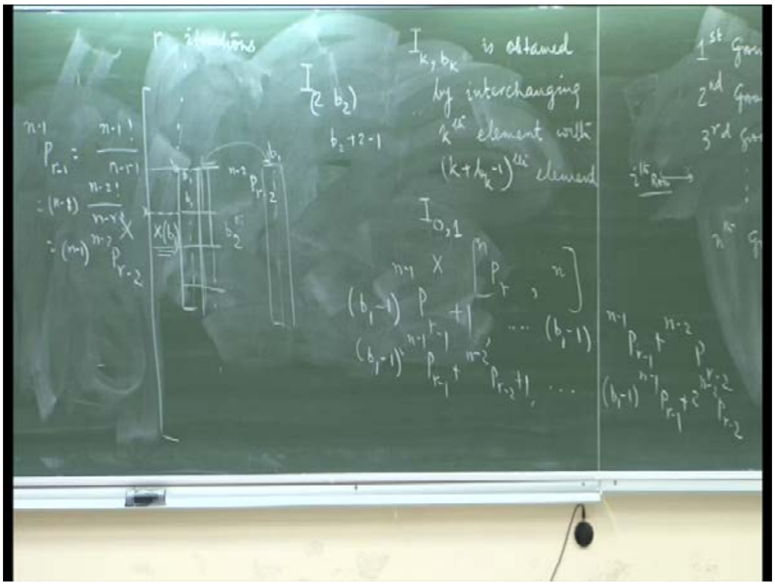
So, you can easily find out what is the index of i th row in which group it is b one th row, so it is b 1 minus 1 n minus 1 p r minus 1 plus 1 to b 1 into n minus 1 p r minus 1, that i must be lying between these two numbers, is it okay. So, it lies in the b th group and you remember the first element as I discussed this is first n minus 1 p r minus 1 element first element will be one, first element will be two and b one th group the first element is b 1. So, what I will be doing if it lies in the b 1 th group then I perform the operations I 1 b 1; that means the first element is interchanged with b 1 th element.

(Refer Slide Time: 23:00)



So, b one th group first element will be b 1 b 1 b 1 and so on, and here all of them are one, then you just have to interchange that is from the definition of this, agreed. So, let these elements or these rows because I told that what is x? x is the complete set of rows.

(Refer Slide Time: 23:27)



So, you have x this is your x. Now I got these many elements they are belonging to your first element is b 1, these rows having first element b 1. Let us name this as x b 1; x b 1 is nothing but the set of rows, right, whose first element is b 1 and of course one has come here because I have just interchanged. How many such rows are there? n minus 1 p r minus 1. Now these n minus 1 p r minus 1 rows are divided into n minus 1 groups, each of size n minus 2 p r minus 2 because these number of elements is n minus 1 p r minus 1 and this I can write n minus 1 factorial n minus r factorial, is that okay. So, I take out n minus 1 n minus 2 factorial by n minus r factorial which I can write n minus 1 n minus 2 p r n minus 2. So, what I am dividing? I am dividing this n minus 1 p r minus 1 into n minus 1 groups each is having n minus 2 p r minus 2 groups.

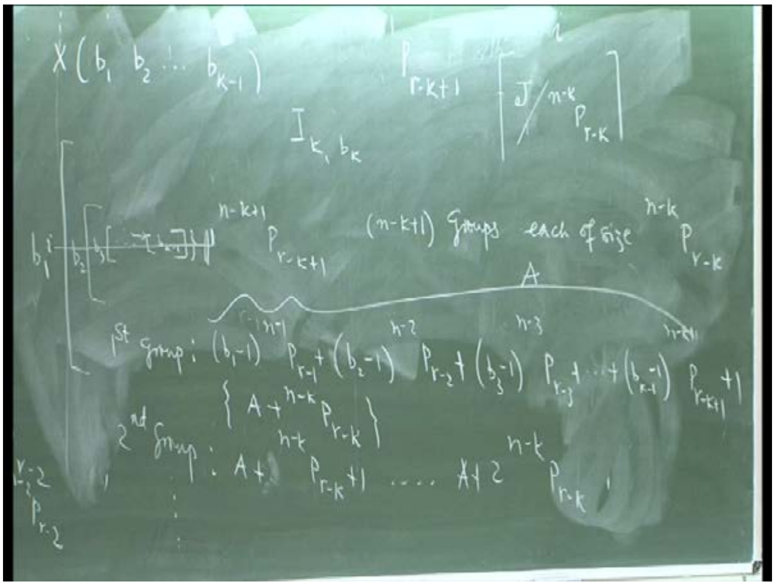
And remember here this element is 1 b one th area is 1 and this is b 1 and now thinking about the second element. Now second element first n minus 2 p r minus 2 elements should be 1. So, I would like to have some procedure to bring interchange these two, right. Similarly, the case with this and so on, right. What I will be doing that I have divided into n minus 1 group each is of size n minus 2. So, I can get a group number. Let us assume that i th row is belonging to b two th row. Let us assume that your i th row now belongs to b two th row. So, this b two th group I will perform the operations I 2 b 2; that means second element will be interchanged with b 2 plus 2 minus 1, second element will be interchanging b 2 plus k plus b k minus 1, right, is it okay.

So, you can find out similar to this case the first group contains the row index. Can you tell me what are the row index for first group of b 1 x b 1 what is the starting index of this, starting to this prior to that there are several searches. So, what is the index of this? Index will be?

Student: x n minus 1 p r minus 1.

Up to this, it is belonging to other rows. So, first I need to know the starting index; starting index is b 1 minus 1, right, and it is n minus 1 p r minus 1 plus 1, right, that is the starting index of this. What is the n index of this, this one? So, it will go up to b 1 minus 1 n minus 1 p r minus 1 plus n minus 2 p r minus 2, right. What is the starting index of this and n index of this? It will be b 1 minus 1 n minus 1 p r minus 1 plus n minus 2 p r minus 2 plus 1 to b 1 minus 1 n minus 1 p r minus 1 plus 2 times n minus 2 p r minus 2 and so on. So, this is the first group of x b 1, this is the second group of x b 1, third group of x b 1 you can define and there are how many groups? n minus 1 groups, right. So, once you know the group number let it be b 2. So, b 2 group what I will be doing in the operations that modify that b i 2 with b 2. So, that interchange you will be making. Now at that oval at the k th iteration let us assume now what happens at the k th iteration.

(Refer Slide Time: 30:08)



Let us assume at the k th iteration, right, that I have a subset of as I define x b 1 subset of rows at k th iteration I have a subset of rows x b 1 b 2 b k minus 1, right, and can you tell me what is the number of rows in x b 1 b 2 b k minus 1? Can I say when it was b 1 only b 1 k is 2 when only b 1 was there it was false n minus 1 p r minus 1, right, when only if b 1 it was n minus 1 p r minus 1, when b 1 b 2 it is n minus 2 p r minus 2. Now when I have a set of rows where you have performed the operation b 1 b 2 b 3 that means what, what is the logical indication? Logical meaning of this is that b k minus one th group or the b k minus 2 th group of dot dot of b k th group b two th group or b one th group, right.

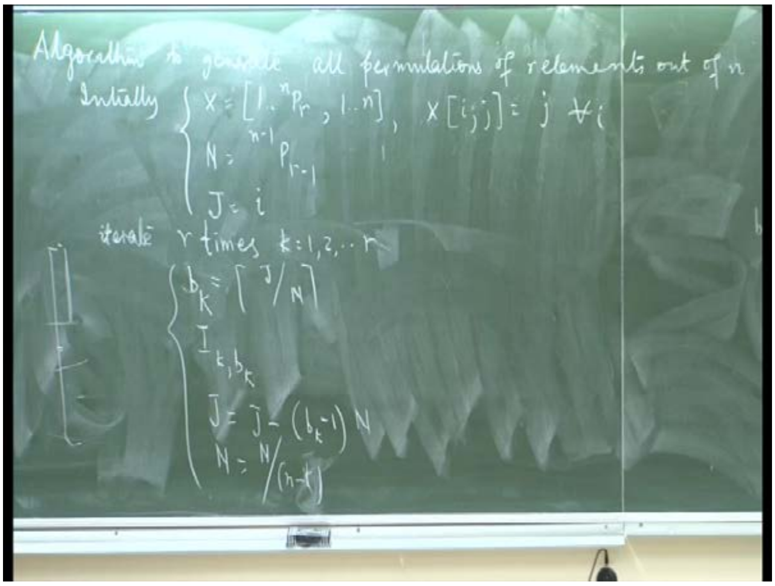
What it means that initially i had b one th group then I got b two th group and within that I got b three group; I am talking about b k minus one here that group, I am talking about these many elements these many groups which is of size n minus 1 combine with k plus 1 p r minus 1. Now these groups are divided into how many groups n minus k plus 1 groups, right. So, these groups this size is n minus k plus 1 p r minus k plus 1. Now these groups are divided into n minus k plus 1 groups each of size what n minus k p r minus k. So, on dividing this n minus k plus 1 p r minus k plus 1 group into n minus k plus 1 groups each of size n minus k p r minus k, right. Now can you tell me what is the starting index of the first group?

So, first group starting index b 1 minus 1 n minus 1 p r minus 1. So, up to this is over then b 2 minus 1 n minus 2 p r minus 2 plus b 3 minus 1 n minus 3 p r minus 3 b k minus 1 minus 1 n minus k minus 1 p r minus k plus 1 plus 1, then there is the starting index, right. So, this is of b 1 minus 1 n minus of p r minus 1 it is coming here plus again you have to go down b 2 minus 1 n minus 2 r minus 2 and so on plus 1 is the starting index and first groups n index will be whatever it is here. Suppose this it is A plus n minus k p r minus k that is the end index if I assume this part is A, agreed. Now the second group is A plus 2 times n minus k p r minus k plus 1 to A plus, is it 2, no, A plus 2 times n minus k p r minus k and similarly is the case with the n minus k plus 1 rows.

So, I now know the starting index and end index; I have divided into n minus k plus one groups. So, i get the b k th group and for this I performed again I k b k for the b k th group I performed this iterations that is sub k th element will be interchanged with k plus b k minus 1, okay. So, when k equals to r you will be getting all the necessary arrangements and pick up the first r elements to get your arrangements, okay. So, let us thing little generic way, suppose you have i th row and picking up the i th row. Now you have to find out what is the value of b k, right, for the i th row. What is the group number at the k th row iteration of r, what is the group number at the k th iteration of r?

So, let us assume that j is the serial number of i th row where you have performed b 1 b 2 b k minus 1; that means after k minus 1 iterations the j is the serial number of i, is it okay, then you have that b 1 which precedes all the array. Now here you have the i th row, i th row is this. So, j is this one, this index is j. j is the serial number of i with it x b 1 b 2 b k minus 1, right. Now you have to find out the group number of i; group number of i is nothing but j divided by n minus k p r minus k. So, that if I know I can find at every iterations what is this serial number or in which group it lies? So, once I know this one rest of everything is very simple just interchange.

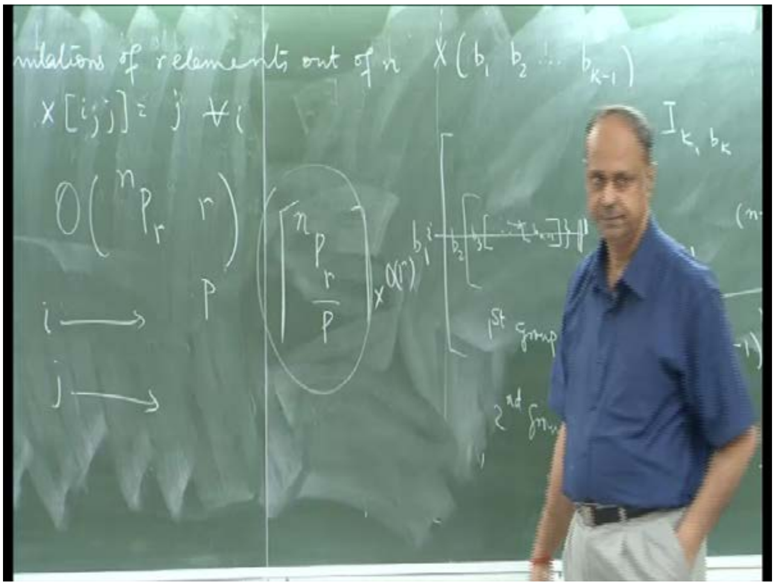
(Refer Slide Time: 39:04)



Now let us write down the algorithm to generate all permutation of r elements out of n elements. Initially let us define an array x when x i j is j for all I, and also we assume that n is equals to n minus 1 p r minus 1 and also let us assume that j is equals to i; that is I am thinking about the i th row j is equals to I, so this is your initialization. Now what I have to do? It has to iterate in r iterations. So, iterate r times. First you find out the mu j or group number r times, say, k equals to 1 2 r. So, b k I have to find out; b k is nothing to J by N that will give you the group number. Then you finally do the mod and interchange k with b k, right

Once you have performed this operation you have to now modify your j and modify your n. What is your j? j is whatever initial j was there minus b k minus 1 into N, right. So, that will give you now modified j initial is now I have divided into this. So, I want to know what is the new j, what is the new positions that many has to be subtracted. So j is j minus b k minus 1 into N, and what is now your new N. New n is N by n minus what? Say first time it is n minus 1 p r minus 1, second time it will be n minus 2 and so on. So, can I write it is n minus k plus 1 n minus k, k is 1 this k n minus k. So, let us reduce by n by n minus k. So, this is your steps to be iterated. So, to get the one permutation you need to iterate r times and you have n p r rows.

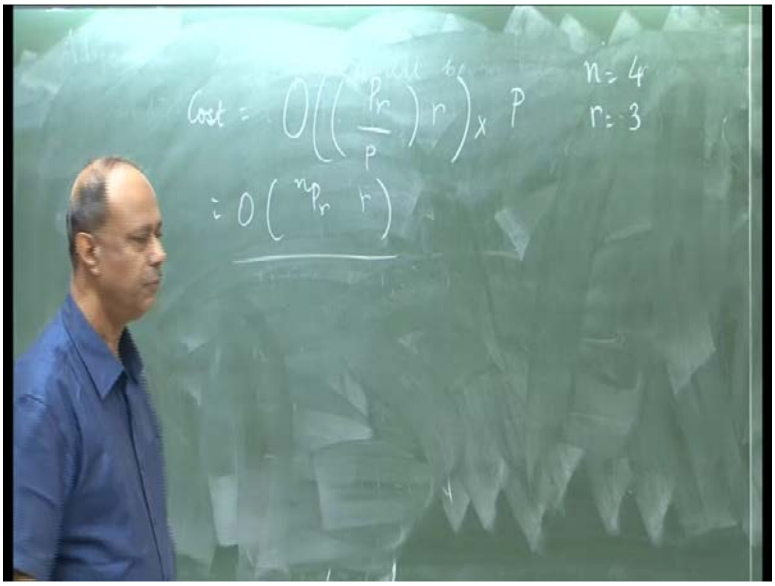
(Refer Slide Time: 43:47)



So, basically if I use a sequential algorithm you need n p r into r, right, r iterations and for each row you will be doing the complexity terms order in terms p r into r, right. Now you observe that if I have to generate the i th permutations I do not need the help of j th row, i th independently I can do, right. While I compute the i th permutations I can easily independently compute because they are independent. So, if I have n p r processors then I can generate all n p r permutations after r iterations plus they are independent. So, this algorithm can be implemented on any parallel machines, right, the complexity of this algorithm will become order r.

So, if I have p processors, right, to generate n p r permutations of r, I need n p r divided by p times and each time you take order r times, right, because you have the p processors, each processor will be generating that many permutations. So, what happens? You have the p processors and I employ the p processor to generate n p r permutations. So, each permutation where processor will be told to generate n p r divided by p that many permutations, right, and to generate one permutation you need order r times. So, the total complexity of these algorithms using the preprocessor is n p r by p into order r. Now what is the cost of this parallel algorithm? If you remember the definition of the cost of parallel algorithm is nothing but the worst case time complexity multiplied by the number of processors.

(Refer Slide Time: 46:21)

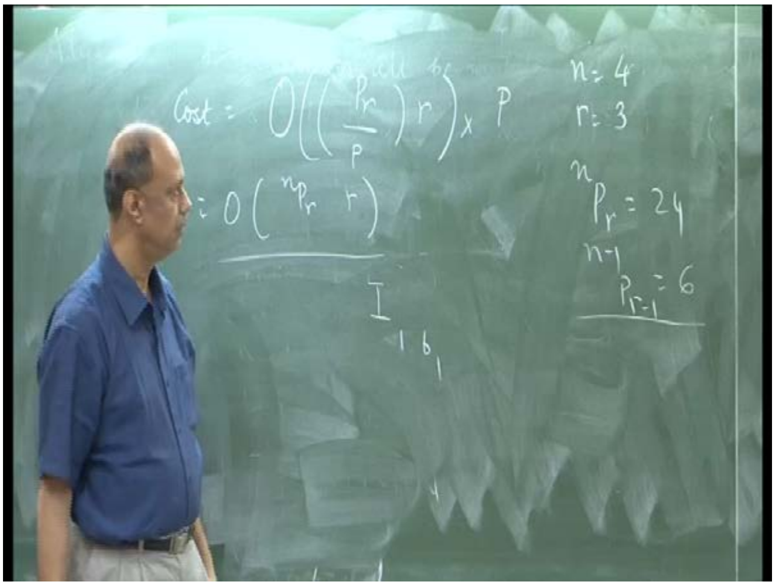


Cost is worst case time complexity is n p r by p into r plus number of processors like that processors which is order n p r into r, right, there is the complexity. Now let us consider one example to illustrate this form 5 factorial by 3 factorial you got 2 factorial, it is a big number, 5 factorial 2 factorial, 5 into 4 into 4 put the big number 4 factorial by 4 factorial is 24. Let us consider n equals to 4 and r equals to 3. Let us see how it works. I know that there are 24 numbers let us assume we want to do this; otherwise you may not feel good, right.

(Refer Slide Time: 47:40)

So, initially there are 24 rows 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23. Initially numbers I told that they have identity 1 2 3 4, 1 2 3 4.

(Refer Slide Time: 49:26)

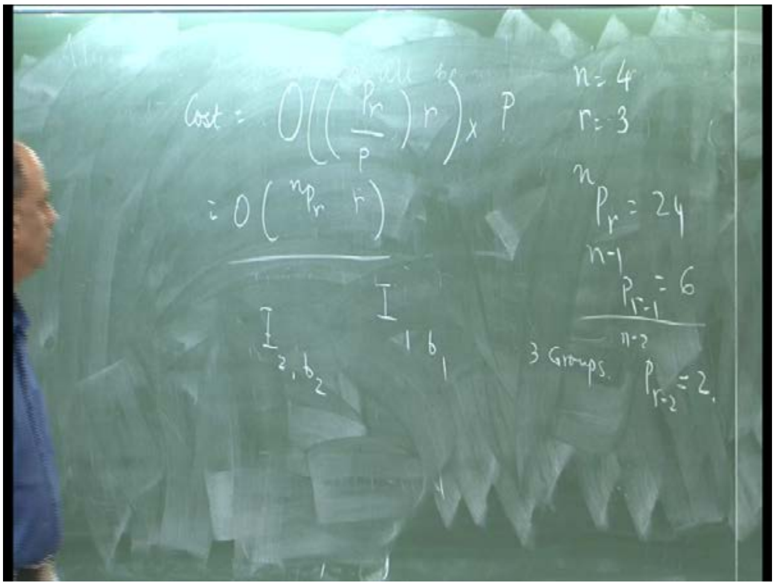


So, these are the 24 rows as I told you that you have n p r rows which is 24. Now this 24 rows are divided into four groups, right. Each group is having size n p r minus 1 which is 6. So, this if you divide into the four groups which is of size 6 groups. Now this is first group, this is the second group, this is the third group, and this is the fourth group. Now first group I will now be performing the operations b 1 is this; b 1 is 1 2 3 4. So, if I have to perform this one first group b 1 is 1, so no interchange, b 1 is 1. So, 1 plus 1 minus 1 is 1, so no interchange.

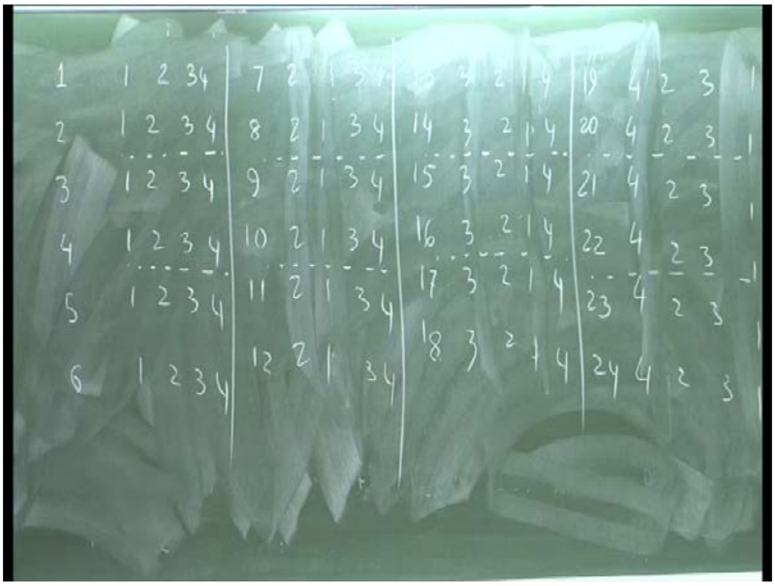
(Refer Slide Time: 50:22)

But here this will be interchanged 2 1, 2 1, 2 1, 2 1, 2 1, 2 1. Here this will be interchanged with this one, right. So, it is 3 3 3 3 3 3, 1 1 1 1 1. This will be interchanged with this one, right. Now I have four groups, first element is 1 2 3 or 4. Now consider this group. It has common elements 6 elements. These six elements are divided into n minus 1 groups; n minus 1 means basically three groups.

(Refer Slide Time: 51:25)

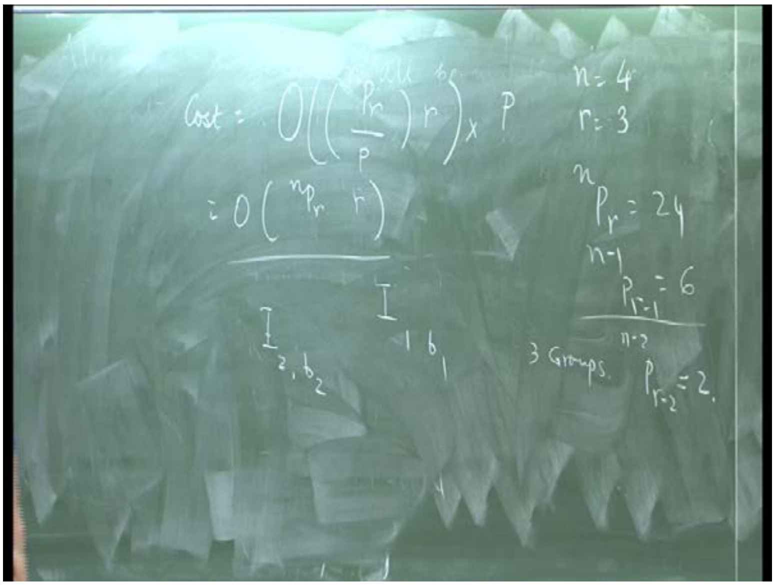


(Refer Slide Time: 51:43)



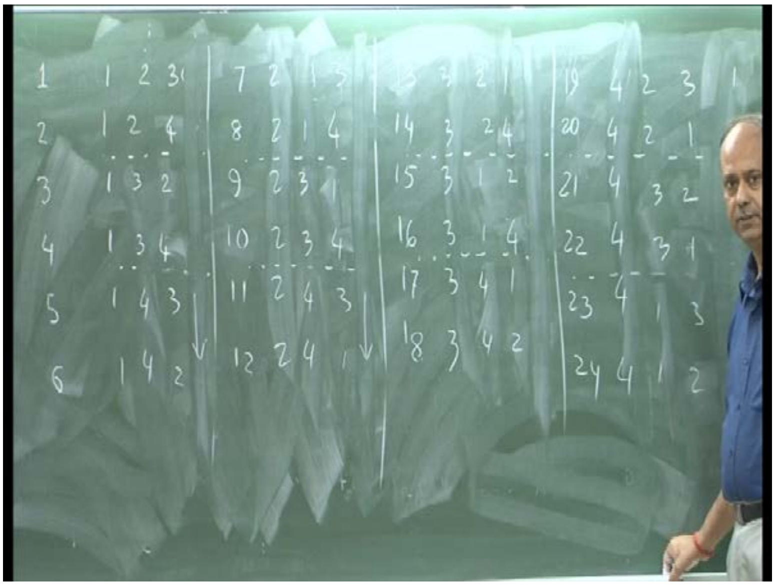
Each group is of size what? n minus 2 p r minus 2, n minus 2 p r minus 2 is 2. So, you have divided into the three groups. This is also three groups, this is also three groups, three groups, three groups.

(Refer Slide Time: 52:04)



So first group I 2 b 2 then first group will be interchanged with the group number plus 2 minus 1. So, this one will be interchanged, what is b 2? b 2 is 1, 2 plus 1 minus 1, what is that? 2 plus 1 minus 1 is 2. So, for this group no interchange as it is. For this group b 2 is 2, 2 plus 2 minus 1. So, this will be interchanged with this.

(Refer Slide Time: 52:44)



3 2 3 2, right, and for this group this will be interchanged with this 4 4 2 2. For this group there is no interchange, for this group this will be interchanged with this one 3 3 1 1, this will be interchanged with this, 4 4 1 1, for this group no interchange and here it is 1 2 1 2, 4 4 2 2, 3 2 3 2, 1 1 2 2, right. Now this is one group, this is another group; this is another group, one group on another each of size 2. Now again we will be dividing into two groups and each of size one and interchange with that. So, what happens in the third group this will be interchanged 4 3, this will be interchanged 4 2, this will be interchanged 2 3, this will be interchanged 1 3, this will be interchanged 4 1, this will be interchanged 4 3, this will be interchanged 4 1, this will be interchange 4 2, 2 1, 2 3, 1 2, 1 3. So, this is the operations we have done.

Now I am telling that first r elements will give you the arrangements; that means I am removing this one, I am removing this one, I am removing this one, I am removing this one. Now you check whether you are getting all the cases or not 1 2 3, 1 2 4, 1 3 4, 13 2, 1 3 4, 1 4 3, 1 4 2, 2 1 3, 2 1 4, 2 3 1, 2 3 4, 2 4 3, 2 4 1, 3 1 2, 3 2 4 and so on. You observe that all of them are distinct and they constantly contain r elements, all the number of elements is n p r. So, continuously you will be getting n p r distinct permutations, right. And you observe also that the ordering is not in lexicographic order because here 1 2 3, 1 2 4, 1 3 2, 1 3 4, 1 4 3 then 1 4 2, this is not lexicographic precedes, right.

Similarly is the case 2 1 3, 2 1 4, 2 3 1, 2 3 4, 2 4 3, 2 4 1 and so on. So, it is not lexicographic permutation, but it generates all the distinct permutation of n p r size of r arrangements out of n, right. The disadvantage with this technique is that it needs a huge array n p r rows and n columns, right, and if I use preprocessors the algorithm can be made easily adaptive. Since, the generation of one permutation does not depend on another one, so at last it holds good. So, it can be implemented on any parallel machines using preprocessors the complexity will become order n p r by p into order r. So, this is one algorithm. Next algorithm we will be discussing on differential structures. So, I think I will not be covering today that one; we will be discussing this on the next class. So, before that I assume that you have knowledge on balanced binary tree.

See we design a new data structure based on the height balanced binary tree, right which has n blocks to generate the permutations and which is also adaptive and can be implemented on any machines, right. Now if you remember that height balanced binary tree is a binary tree where height is of order log n and height of the left sub tree minus height of the right sub tree is plus minus 1 lying between plus minus 1; that is height will be left or right, it may be minus 1 more than the right sub tree height or plus 1 or may be of the same height, right, and insertion of any node in the balanced binary tree is also order of log n, deletion also takes order log n and you wanted to make use of this data structure we will modify this data structure so that this can be used for our permutation arrangement reverse generations. See we define a problem is of that structure that given m records which form a balanced binary tree and an argument k, your aim is to find a nonnegative integer u such that if I insert k plus u or the right of k plus u in that m record is u plus 1.

And if you get that u plus 1 you insert that node at the u plus one th element and then rebalance the tree for future generations. So, that is the problem. So, first we will be discussing about this problem, and we will discuss how you cans solve it. Then we will see how we can make use of it to generate the lexicographic permutations, because our algorithm aim is to generate the lexicographic permutations. We will also be discussing another formation algorithm which is generally we have n elements and r distinct objects you want to select, but suppose what happens that n elements which are not distinct. The problem is little different that if the elements are not distinct and you want to generate r out of n then some of them will be repeating them again. So, how to generate that permutation also we will be discussing and along the algorithm; so let us stop here. We will discuss in the next class.