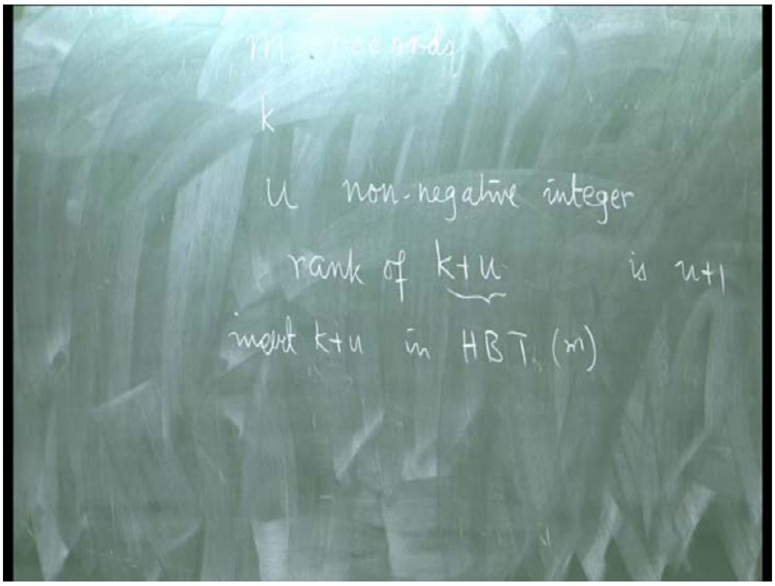
Parallel Algorithms
Prof. Phalguni Gupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 19

So, in the last class we discussed about the generation of permutations of r elements of n, but the problem we told that it will not generate the permutation in lexicographic order. So, today we would like to discuss about the problem of generating the permutations in lexicographic order. So, to design that yesterday last class we told that we have to design a new data structure which is basically not viewed in the modified function of all that is binary case.

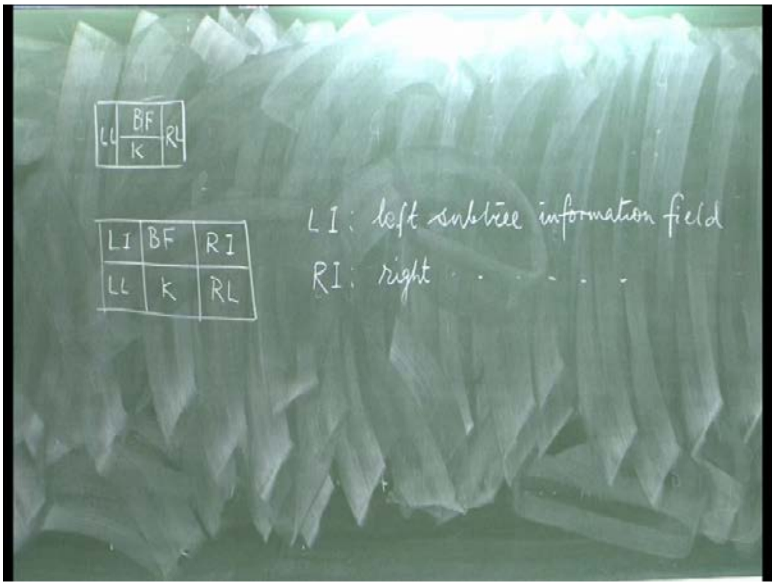
Now, if we remember correctly that balance binary tree having the certain properties that the height of the left sub tree and height of the right sub tree of a binary tree or binary sub tree should be if I take the difference between these two, it should be either plus 1 minus 1 or 0. So, from the definition or from the view what is a binary sub tree, the height of the bell binary tree would be restricted to order log n, right. So, if it is that case then substitute on the binary search tree, it will take order log n time; if I have to insert then no leaf binary search height value binary tree. Then you can go to the last row or to the left row in log n time, and then after insertion it may lose the property of values factor that is the plus minus 1. So, you may need to do the revalues six of that binary search tree becomes again high balanced binary trees.

(Refer Slide Time: 02:38)



Now here let us define a problem for a given m distinct networks who is forming high-valued binary trees, right, and there is a key k. Now the problem is to find a nonnegative integer u. The problem is to find a nonnegative integer u such that the rank of k plus u in your m records is u plus 1. So, what it means? That you have to find out the u which is a nonnegative integer such that the rank of k plus u if basically I want to insert k plus u in the m records such that its rank is u plus 1. So, rank of k plus u if we insert in m records is u plus one and once you get that k plus u you insert in the height balance tree of m records, so insert k plus u in the height balance tree with m records, right, and once you insert it, it may lose the property of balancing. So, you need to do the rebalancing, so this is the problem.

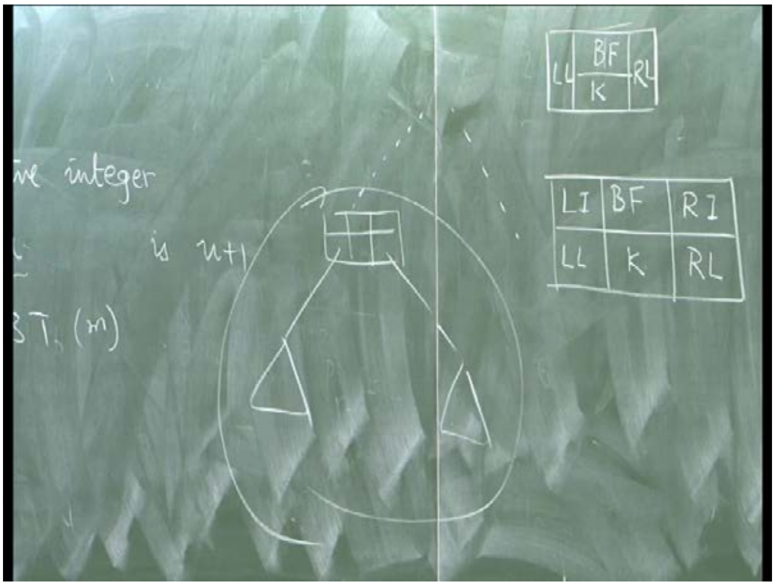
(Refer Slide Time: 04:43)



Now if you remembered that in any height balance binary tree that a node consists of four fields; one is the left leg field, another one is right leg field and the balanced factor and key, right. So, a node consists of four fields; one is balance factor which contains value zero, one or minus one. Then left leg it points to the left sub tree and right leg it points to the right sub tree which is also a balanced one and k is the key field where the data is stored, right. In all case this known structure has been changed; instead of having the four fields here we are keeping the six fields.

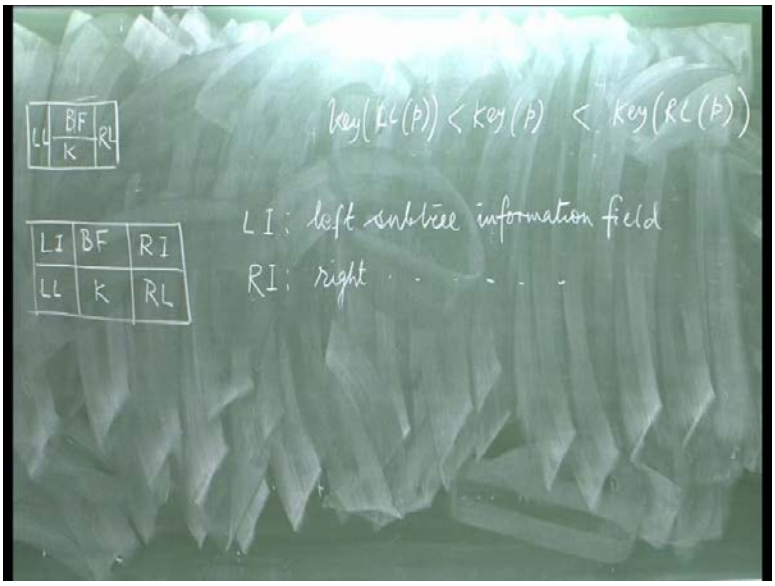
The six fields are one is balanced structure, one is key, one is left leg, one is right leg, and here I am putting another in LI and RI. Li is the left sub tree information field and RI is right sub tree information field. Now what it means? What I want to tell to LI? LI will give the information about the number of nodes in the left sub tree. RI will give the information about the number of nodes in the right sub tree.

(Refer Slide Time: 06:53)



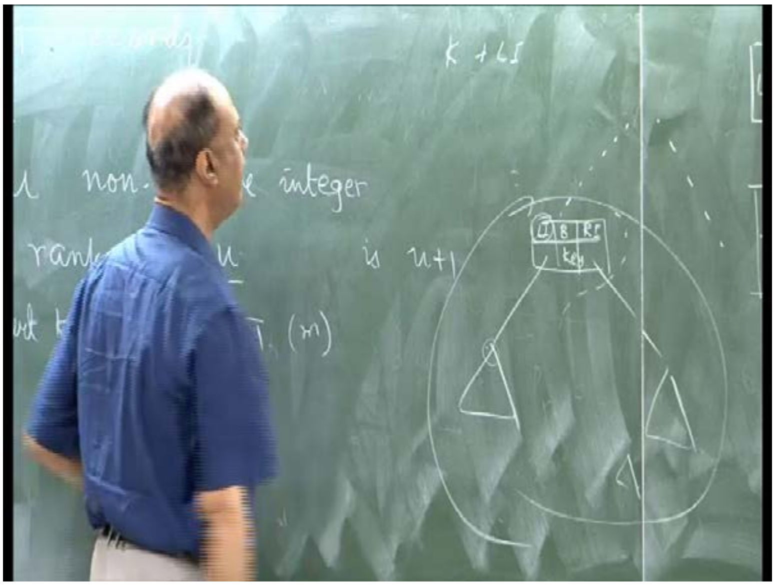
So, if I have a node here. So, you may have other structure here and so on. So, to know the number of nodes in this sub tree I can find out LI plus RI plus 1; that will give you the total number of nodes in this sub tree. So, the problem you can define or you can redefine into three sub problems. One I have to find out the rank of k plus u, right; that means I will find out one such view such that the rank of k plus u is u plus 1. Once you get that k plus u you have to insert that k plus u in your tree in your balance binary tree height balanced binary tree and after insertion it may lose the property of balance tree, so you have to rebalance it.

(Refer Slide Time: 08:35)



Since I have decided since it is a binary search tree and this m nodes or m records are distinct in nature for any node p the key of p satisfies this property, right; that means the key of this node will be larger than this one the key of this node and smaller than this one. Now we know that height of this tree is log m because there are m records, so height is log m and whatever searching procedure we followed to find that u, u has to travel from here to here up to the leaf node to take the final decision. So, from here to leaf node height is order log n. So, you can easily know that or easily show that to get such a u it will take order log n time.

(Refer Slide Time: 09:54)

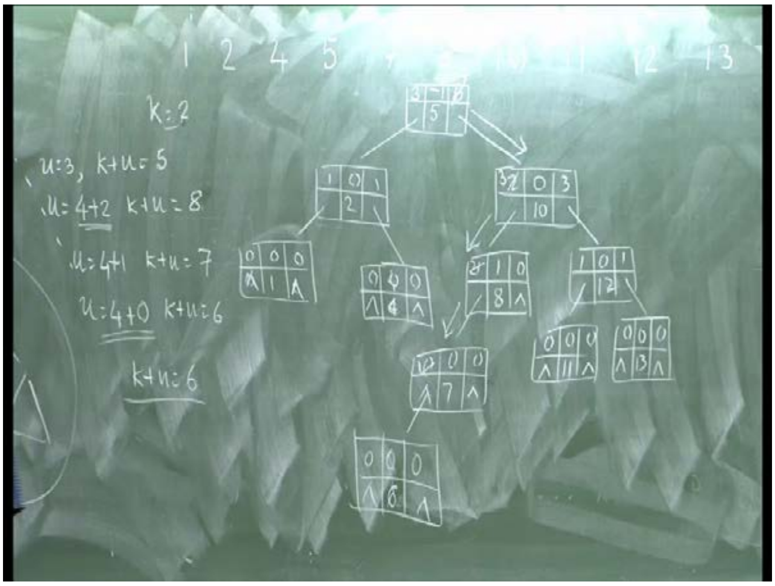


Once you know the u you can insert that node a new node and the leaf node and then balancing may be required, you have to do the balancing which takes order long n time. So, general procedure see I do this left information field, I know there are information field, I know that there are risk factor and I know the key value of this node. Now the k to be inserted or k plus u to be inserted in to this node, you try to understand this one that either it will follow this path or it will follow this path. Now if it follows this path then at least these many nodes will be smaller then all these values will be smaller than whatever you are going to insert, because k plus u will be in this side, is it okay.

Now if it is that that means u value will be at least these many nodes, right, and if it follows this path then u value will be added to this side not this side. So, what I will do that u will be these LI. Initially I will think that u is your LI. Now you check k plus the

LI; LI is the left information field. If you find that this is smaller than your key value k plus LI is smaller than this value, then the path to be followed is this one. If you find the k plus LI is greater than or equals to this value then we have to follow this path, right. And suppose I am here then again your LI is this one k plus LI to be this step and you have to follow this path or that path and so on. Let us consider one example that will help you to understand how it works.

(Refer Slide Time: 12:25)



So, I have 1 2 4 5 7 8 10 11 12 13, four plus ten data I have taken, right, four plus four, and let us assume that my value is like this. This is our work procedure, this is a merged binary tree of 10 nodes and I want to my k is set to 2, my k is 2. Now how can I proceed? So, what I told that if I have to pump this path then there are four nodes at least you will have on left of u k plus u. So, here I start with u equals to 3 k plus u, u equals to 3 I get it is 5. Now this 5 is equals to this, so that is not right; you will not get u plus 1. So, what you have to do? You have to follow this path, okay.

So, once you are following the right sub tree; that means the node will be inserted in this area not on the left side. Once you are inserting the node in this side then this node will be increased by one. So, that node will be inserted to this side. Now what happens in that case? That your u becomes gradually 4 because at least four nodes will be there; now you are here. So, that contains two, okay. So, k plus u is your shift to eleven are known, k plus u is 6 plus 2 is 8, right.

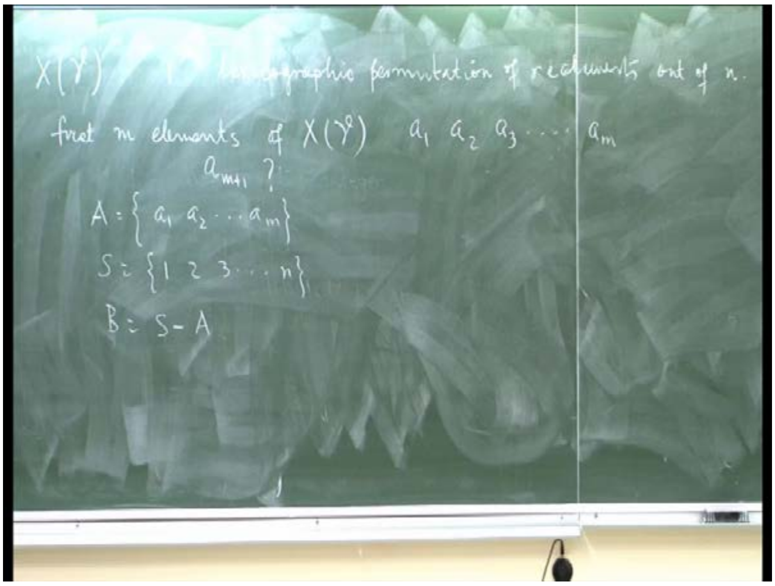
Now you see that 8 is less than 10. So, the path is this side, right. So, once that you are only in this path this number will be increased by one, because now node will be inserted somewhere here so it will be increased by. Now what is your u? u will be still 4, because this has not travelled towards the right, it is travelling towards the left. So, it is u will be still 4 plus expected one is 1 plus 1, k plus u becomes now 5 plus 2 is 7. Now 7 is less than 8, so the path will be followed by this one; so your u is now 4 plus 0 k plus u is 6.

So, this will be iterated by one which is 2, 6 is less than 7. So, you got the position where to be inserted. This becomes one, right, and your u is 4, okay. So, what is happening? That means k plus u is 6, and the rank of k plus u is 4 plus 1 is 5 because both that are smaller than this, so k is plus 1 is fine, u is 4 u plus 1, so rank of k plus u is 4 plus 1is 5. Now up to this is now the problem is commencing loses the property of balanced factor, right.

Now you could have done the rotation technique to solve this one, but only rotation will not help; therefore, I am keeping track about the number of left information number of nodes on the left sub tree n number of nodes on the right sub tree. So, that information I will lose it. So, you have to be very careful; see if the rotation technique if I follow then basically it loses the property of here, right. It loses the property of balancing here, yes. So this is simple where 7 will come up, 8 will come down and 6 will go up, and corresponding while we are changing the balance factor you have to change the information of the LI and RI.

So, the total time complexity for this is order log n. So, this problem I have solved now that given m records distinct records which formed to the high value binary tree and an argument k you can always find a nonnegative integer u such that k plus u is u plus 1, and if you get that one you will insert k plus u into height values tree and rebalance the tree and also adjust the left information field line of this 10 values, right, and this can be done in order log n time. So, these data structure will be used to generate the lexicographic permutations.

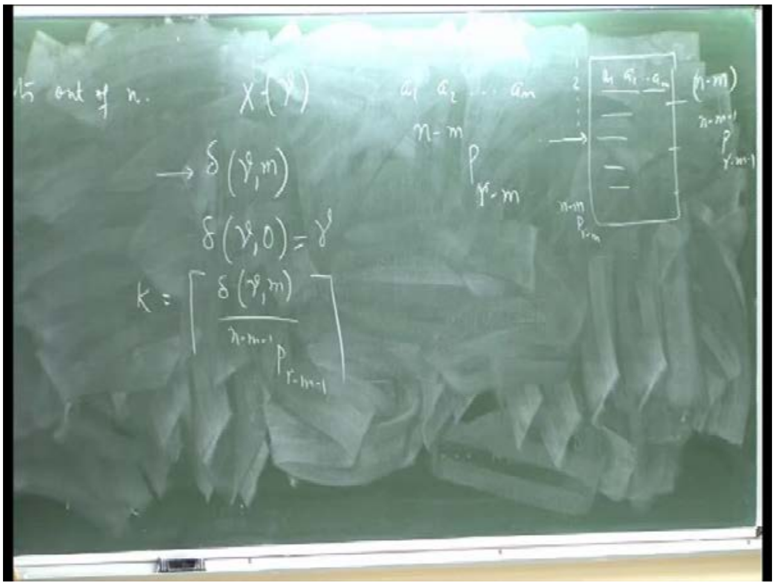
(Refer Slide Time: 22:03)



Let us assume that x gamma is the gamma x lexicography permutation of r elements out of n, so comma x lexicographic permutation of r elements. Lets us assume that I know first m elements of x gamma; let us assume that first m elements are known to you. I want to find out that what is the m plus 1 th element. Let us assume these are a 1 a 2 a 3 a m. Now I have to find out what is a m plus 1. So, if I can find out that a m plus 1 then I could make it for any m i can find out and that will give me the lexicography permutation x gamma.

Now let A be the set of a 1 a 2 a m. Now this permutations are obtained from first n integers 1 2 3 4 up to n. Let us assume that s is your 1 2 set of n integers, and you have to find B 5 S minus A; that means the set of all the elements which are in S but not in A, right. Now my m plus 1 th element a m plus 1 must be 1 whole square elements from B, right. So, that is the thing I have to find out, what is that element from B?

(Refer Slide Time: 24:43)



Now we told that x gamma with the gamma lexicographic permutations and its first m elements are a 1 a 2 a n. From the definition of the permutations there are several permutations you will find whose first elements are a 1 a 2 a m. There exists several such permutations whose first n elements are a 1 a 2 a m, right and number of such permutations is n minus m p r minus m. So, there exist n minus m p r minus m permutations whose first elements you will finding a 1 a 2 a m, right, and x gamma is also is one such permutation. So, each I arrange them in lexicographic order, these n minus m p r minus m permutations in lexicographic order, then x gamma may be lying somewhere here, okay.

Let us range them 1 2 3 up to n minus m p r minus m and x gamma might lie here. Let us assume that h gamma m is the position from the top, basically, instead of x confusion with this; let us put it delta gamma m. Delta gamma m is the position of x gamma from here. So, appears to be delta gamma 0 is nothing but your gamma.

Student: The m is the variant word so that will mean that.

No, the m indicates that number of elements you have already generated number of elements. So, there are m elements that are already generated. These elements are a 1 a 2 a m, right.

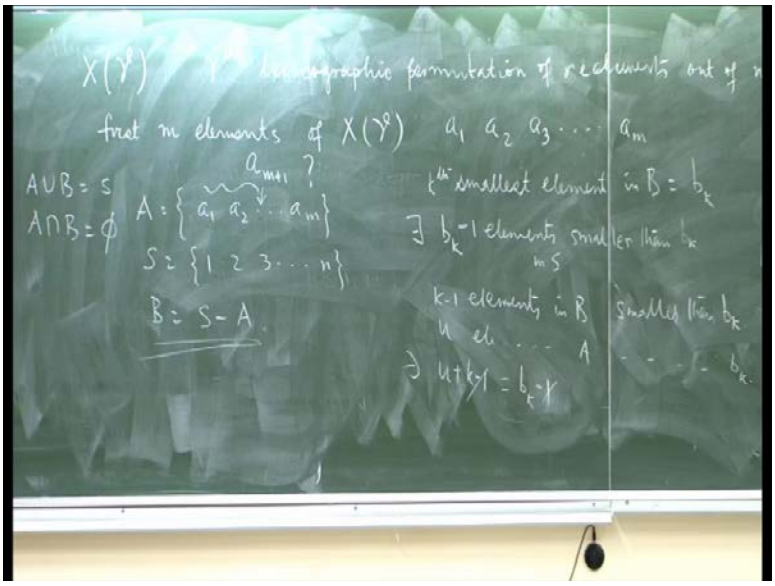
Student: And the rank of x gamma is?

Rank of x gamma in this table is delta gamma n, right, and obviously delta gamma 0 will be zero, nothing else can be done, that initial position is gamma, right. Now what is given to you? Given to you is delta gamma m and also you have n minus m p r minus m; this is given to you for defined value of m. Now one property in permutation if you see that the occurrence of m th element is a m and m th element is a m plus 1 that number of suspicious change, right; that means these s minus m p i minus m you will be dividing into n minus m groups. Each group is having n minus m minus 1 p r minus m minus 1, in the last class also discussed this.

So, we divide it these n minus m p r minus n into n minus m groups. Each group is having m minus n minus 1 p r minus n minus m permutations, right. So, let us assume that k is the group number for delta gamma m, because to find out the group number delta gamma m divided by n minus m minus 1 p r minus 1, and if you tell take the serene function that will give you the group number of delta gamma m which is skipped. Now since we are assuming that in lexicographic order, this k will help me to find that what is the m plus 1 th element, right.

This key is nothing but the k th smaller b because B is the set which contains all the elements which are not in A but in S, right, because I have to arrange in the lexicographic order in the k th smallest element in B, right. But remember one thing that i do not have B I have A, because I have generated a 1 a 2 a 3 here. So, k cannot create the value of k th smallest element of B from A. So, that is there; if i can do it then I am through.

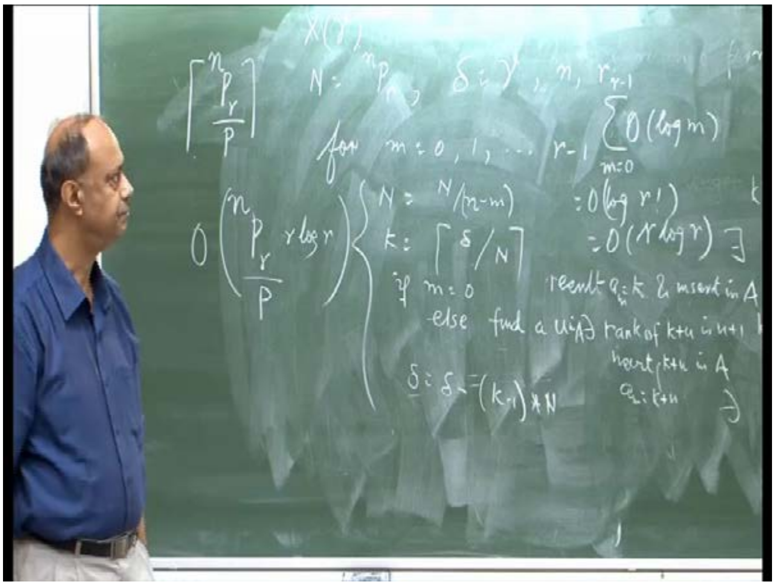
(Refer Slide Time: 30:43)



Now, let us assume that k th smallest element in B is, say, b k, let us assume. Now remember one thing that A union B is your S and A intersection B is phi, right and S contains the first n result numbers. Now each k th smallest elements in B is b k then I can tell there exists b k minus 1 elements smaller than elements in S smaller than b k, because this is the first n natural numbers, b k is one of the element here. So, there exists b k minus 1 element is smaller than b k, elements in A smaller than b k, right. So, if b k minus 1 element is smaller than b k here and k elements are here because in the k minus 1 elements smaller than your b k.

Here k minus 1 element in B smaller than b k, because b k is the k th smallest element, b k is the k th small zone there exists k minus 1 smallest element in B k minus 1 elements for in B smaller than b k, right. So, there exists one u here, right, so that here one u and here k minus 1, so that this two addition will give you b k minus 1, right. So, there exists u elements in A smaller than b k such that u plus k minus 1 is equals to b k minus 1, right, k minus one elements in B smaller than b k and u elements there must be some u elements in A which is smaller than b k, so u plus k minus 1 is equals to b k minus 1 that means u plus k is equals to b k. What it gives you that given k I have to find out one u in A such that rank of u plus k is u plus 1, right, so that algorithm that structure we have to discuss. So, i can find it out such a key such a u. So, if I know k from A i can find out u so that u plus k is equals to your b k.

(Refer Slide Time: 34:28)

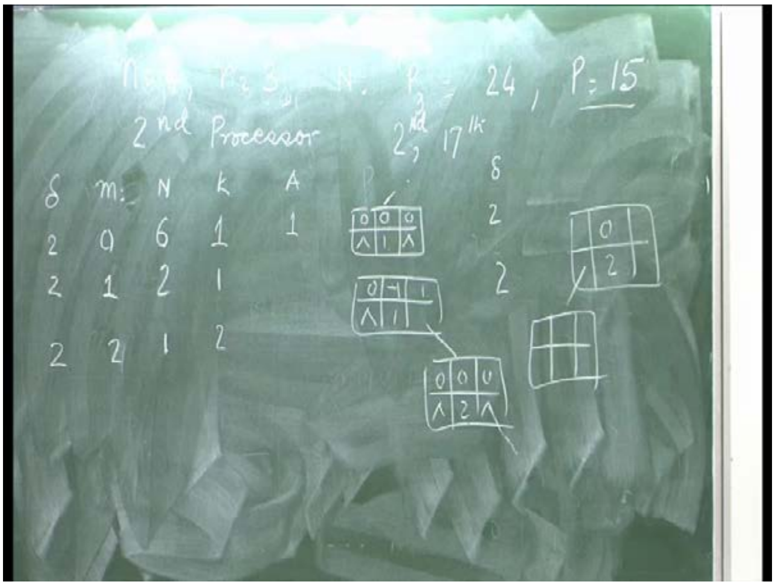


So, based on this idea we can have I have to find out say x of. So, initially N is equals to n p r and delta is equals to gamma from the definition and you have N, N is known to u and also r is known to u. Now for N equals to 0 to r minus 1 you first define your N, N is N by n minus m. So, that will give you n minus n p r minus 1 r minus p r minus n. Now k is nothing but delta by n, right, yes. So, if n equals to zero that is first iteration then your result is k, result is just stay to a k else, so whether this k and insert in A else find a nonnegative integer u such that rank of u in A such that rank of k plus u is u plus 1 and inside K plus u in A.

By insert I mean not only insertion you have to do the balancing also, and now i got what is my, say, here basically u is 0, right. So, I can write here a n equals to this, and here also I can write a n equals to k plus n, right, so that you got your a 1 a 2 a 3 a plus n. Then you have to modify your delta, delta is equals to delta minus k minus 1 into n, right, so that you can find out that index of delta in the next formation that is there. So, this I think all about your algorithm. So, you observe that to obtain the gamma f lexicography permutation you have to do the r iterations, and here you need searching technique because you have to insert an element in a balanced binary tree, so at the m th iteration the height of the bell binary tree is log m because m elements are there log m.

So, order log m and there are order log m and the m th iteration is actually log m minus 1. Let us assume log m and it will be iterated on that. So, summation over order log m which is nothing but log r factorial because this will come m equals to 0 to r minus 1 and this is nothing but order log r factorial which is nothing but order r log r, right. So, to generate one permutation you need order r log n time, right. Now there are n peer such permutations you have to generate so which states n p r log r stats. You observe that this algorithm is basically adaptive because it does not need; every person can do it on his own, right. So, if I have the p processors then the time indicates to find out that n p r by each processor will be generating n p r by p permutations n p r by p permutations; each processor will be generating. So, the time needs n p r by p r log r, but if you see that it is not cost optimal algorithm, but they are very simple to implement.

(Refer Slide Time: 40:50)



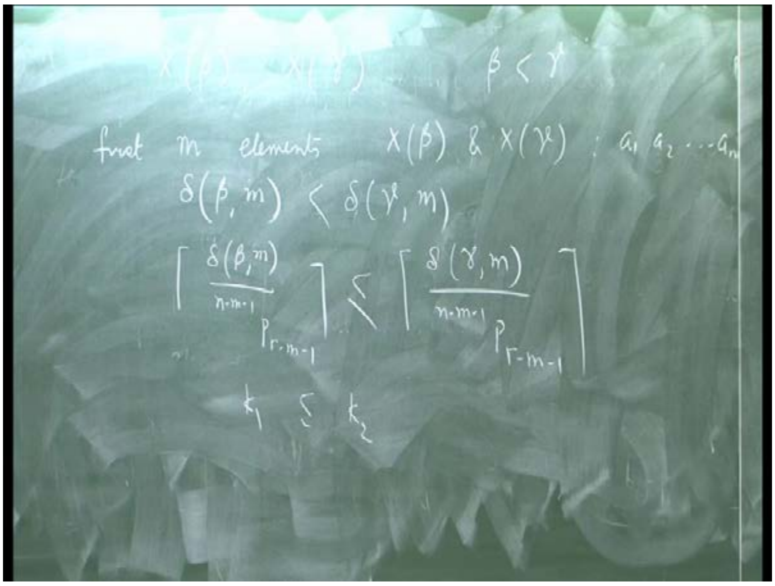
Let us consider one example that will help you to understand. Suppose you have n equals to 4 r equals to 3 and what is your capital N 4 p 3, it is 24, right, and the iteration of the number of processors is you have 15, p is equals to 15. So, let us see what the second processor will generate, right? So, second processor will generate the second lexicography permutation and seventeen, so it will second permutation and seventeenth permutation, right, because you have the 15 processors, so second processor will generate the second lexicography permutation and seventeenth lexicography permutations.

So, you have delta value is initially second 2, then is coming what are the parameters you need? What is your initial length? Initial length is 24. So, 24 your m is zero. So, m value is 0, what is your n value? n value is n minus 4 which is 6. What is your k value? k value is 1 because 2 divided by 6 is 1 and recedes m equals to 0, so it will be 1. So, your tree will become 1 0 0 0; this is your initially it will generate.

Now what is your delta value updated delta value you have to do it, delta is delta minus k minus 1 into n; k is 1 so 0, delta is still 2, right. So, delta is 2, n is 1 what is your N value? N is 6 divided by 3 is 2. What is k value? Delta by 2 is 1. What is your A? A is now you have to insert one here k plus u. So, k plus u to obtain this you see that I will be changing k plus u, u is zero. So, it is is one, one is this. So, it will be left of it, so it will become zero; this side is minus 1 1 1 2. So, this is your at the same iteration what will be the delta value? 2 minus k is 1, so zero it is still 2, right. So, delta value 2, n is 2, what is n? Delta divided by 4 minus 2 is 2, 2 divided by 2 is 1 and k is now to 2, because 2 divided by 1 is 2.

Now this 2 to be inserted here, so 2 plus 0 is 2, 2 is greater than 1. So, it will follow this path, right. So, 2 plus 1 is 3, 3 are greater than 2. So, it will be inserted there. So, you will be inserting here but doing that it will be losing the property of balance, so it will become now. It will generally lead more like to permutations if because that is the ultimately you have to show because this is the thing what we started. So, you have to show that first part I want to tell that whatever number it generates, the number will be lying between 1 and n, because there my set A and set B they are belonging to the set of s and s contains 1 to n. So, one part is that all the numbers whatever numbers it will be generating that will be lying between 1 and n. Second part is that in the case of permutation n p r permutations that there are n blocks of one block will contain first element one, another block will contain second one first element two and so on and the size of the each block is fixed that is n minus 1 p r minus 1.

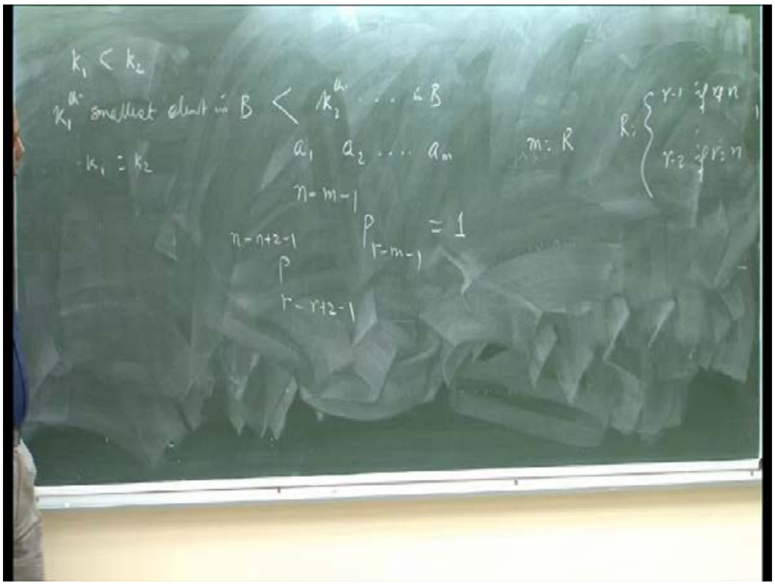
(Refer Slide Time: 47:37)



Now you have to show that it obeys the lexicographic property, let us take the two permutations x beta n x gamma and with the and also in that b is less than gamma, right. So, you have to show that x beta precedes lexicographically l x gamma; that is the next point beta less than gamma you have to show that x beta precedes lexicographically x gamma, right, for beta less than gamma. Let us assume that it both of them have generated 10 elements first 10 milligrams. Both of them have generated first 10 elements and they are set that first 10 elements of x beta and x gamma they are a 1 a 2 a n, right.

Now we have also defined delta beta m is the index of beta x beta within that group and also we have defined delta gamma this is slope and since beta is less than gamma it obeys the property of these ones, right, because these are just an index. So, it obeys the property. Now you will be dividing or you have to find out the group number which is given by delta beta m divided by n minus m minus 1 p r minus m minus 1 less than, right. So, once I divide it may not obey this property less than property, it may be less than or equal to. So, that it is a group number. So, there may be several permutations that belong to the same group. Let us assume that group number of this division gives you k 1. So, k 1 is less than k 2. Now each it is less than, k 1 is less than k 2; that means what? You will be finding a nonnegative integer u, right, in s k 1 plus u; k 1 is defined from k 2 and it will get the smaller value, because k 1 is the smallest element in B, k 1 th smallest element b and k 2 as smallest element of b. They are two distinct numbers if k 1 is less than k 2.

(Refer Slide Time: 51:12)



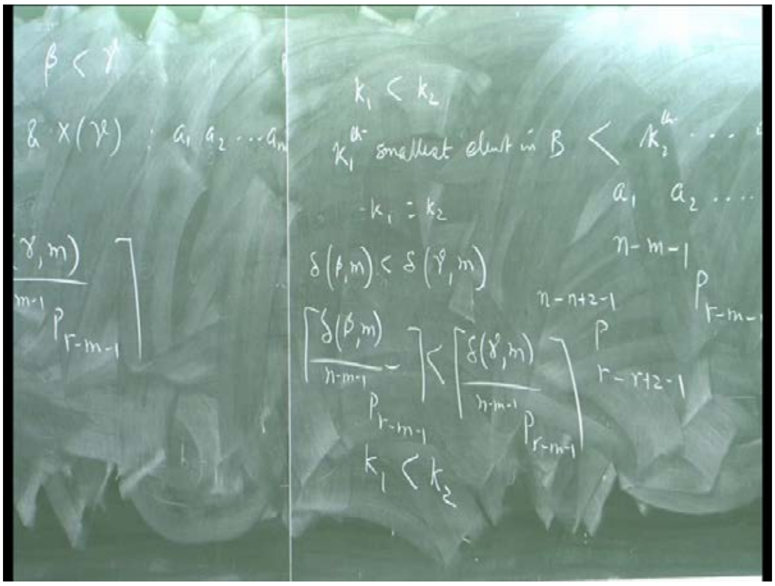
If k 1 is less than k 2 then k 1 th smallest element in B is always less than k 2 th smallest element in B, right, because k 1 is less because B is the distinct numbers. So, in that case with in terms of A, if I tell in terms of A, then also we will get because if we had two distinct numbers. So, if k 1 th element is smallest element less than this then it obeys the property of lexicography. So, by any means if it is less than then there is no problem, but if it is not less than if they require then the issue comes. Now, let us assume that k 1 is equal to k 2, right. Let me assume that it generates all same numbers. Now, through a contradiction I would like to show that it is not possible.

Let us assume that you have generated the same a 1 a 2 a m and m is equals to, say, R. What is R? R is nothing but r minus 1 if r is not equals to n and r minus 2 if r equals to n; that means if I want to generate permutation of r elements out of n, if r is equals to n then this you want to generate the permutation of n elements out of n, then whatever is the case may be then I am going to show that last two digits are fixed. I will come through an example to this one; otherwise, last one digit is fixed. So, let us assume that I have a 1 a 2 a m where they are the same and m equals to R.

Now I can show n minus m minus 1 p r minus m minus 1, for this m value you can find out this is all equal to 1. This you put when r equals to n, m equals to R minus 2; R minus 2 means n minus p r minus m, m is r minus 2. So, it is becoming 1 p 1 which is 1.

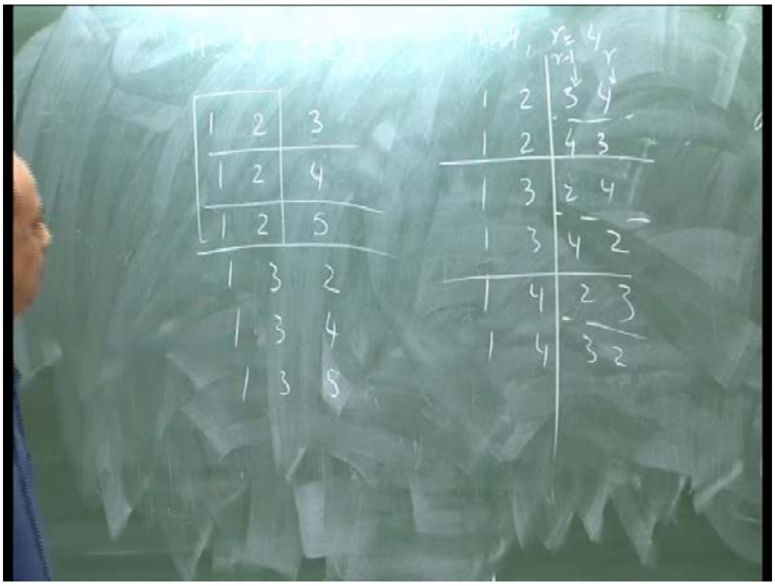
Similarly if you put when r is not equals to n then you will find n minus r plus 1 minus 1 and then this is also becoming to 1. So, if a 1 a 2 a m m equals to R, your this value is 1.

(Refer Slide Time: 54:53)



Now what happens in that case? In that case you have delta beta m less than delta gamma m you are dividing it beta m divided by n minus m minus 1 p r minus m minus 1, right, but this variable is one; I had shown that this one. So, this k 1 is less than k 2 and k 1 it is not possible at this at least when capital r k is it is k 1 is less than k 2, and that means that k th k 1 smallest element in B is less than k 2 th smallest element in B and at least in the last capital r k this will be defined, and so that will prove that it obeys the lexicographic permutation; that means x beta precedes lexicographically x gamma. Now this part you have to show now that what I am telling that if r is not equals to n the last element is fixed. And if r is equals to n then last but two elements are fixed.

(Refer Slide Time: 56:35)



So, let us consider one example, say, n is 5 and say, r is 3, right. So, if I have 1 2 3, 1 2 4, 1 2 5, then 1 3 2, 1 3 4, 1 3 5 and so on, right. When r is not equals to this is r is nt equals to n. What I am telling when r is not equals to n then each of them is having one group because these are fixed, right, they are grouping to the same group, but only one group size is 1, that is fixed, that is the n minus m minus 1 p r minus m minus 1 is 1, this is 1 1 1, and when n equals to 4 and r equals to 4 you have 1 2 3 4, 1 2 4 3, 1 3 2 4, 1 3 4 2, 1 4 2 3, 1 4 3 2, and here you observe that once I have that n equals to 4 n equals to r then after then r minus 2, this is r minus 2 case, r minus this case and r minus 1 and r.

So, this two case it is fixed, say, it is three into one whole group because it is one one group. So, group size is 1, n minus n minus 1 p r minus 1 group size is 1, right. So, that is the thing; that is the basis of thing that even if it generates the same numbers of two capital R m equals to capital R, but in this case it will generate the two distinct numbers which obeys the lexicographic preceding. So, this is about the lexicographic generation of permutations where elements are not this state.