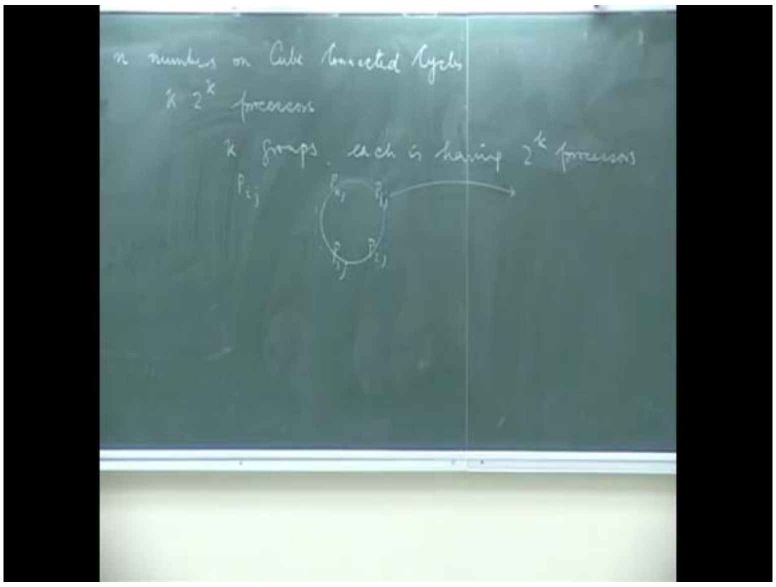
Parallel Algorithms

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

Lecture - 13

As in the last class we were discussing about the some example that is on sum of n numbers on different models. And we discussed on mess connected computer. We also discussed of perfect sample and also then hypercube. Did we discuss on butterfly? Yes, Butterfly also. Now, cube connected cycle. Did we discuss on that? Yes. Sum of n numbers on. Butterfly is done right? Okay, let us do one first cube and then, we will go for butterfly.

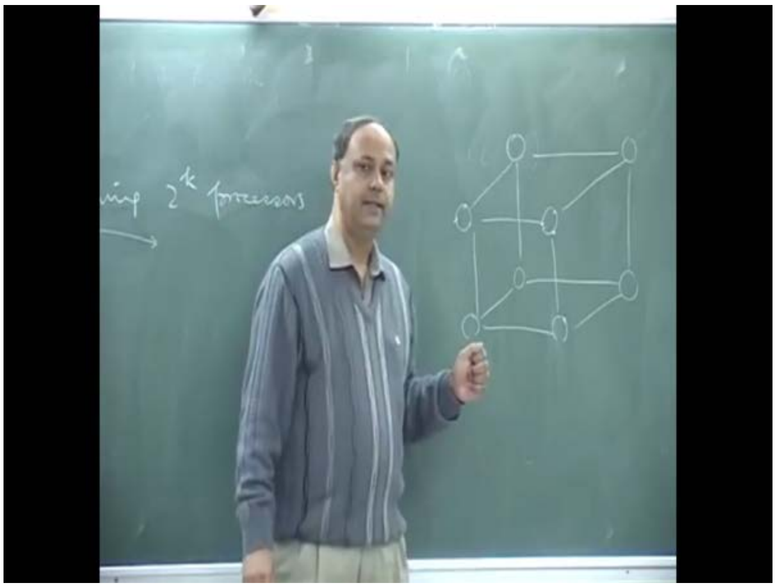
(Refer Slide Time: 00:48)


n numbers on Cube Connected Cycles
k 2^k processors
k groups, each is having 2^k processors
P_{i,j}

Cube connected cycle and if you remember in the cube connected cycles you have. And this k into 2 to the power k processors, you divided into k groups. Each is having 2 to the power 2 to the power k processors. So, let us also assume that P i j is the processor in the i-th group and j processor in the i-th group and all these processors of the different groups of the same j, they form a cycle. So, this is P i 1, P 1 j, P 2 j, P 3 j and P k j. This is the sachet and P i of j is connected through it P i j. P 1 j is connected with one processor of another cycle. Agreed? Which is P 1 k, something like that or P 1 n.

Now, in the case of finding the sum; can you suggest some method how are you going to do it? Say, it is the combination of butterfly and hyper cube. So, what I want there? Since, it is a combination of butterfly and hyper cube, if I consider this cycle is a node only 1 node.

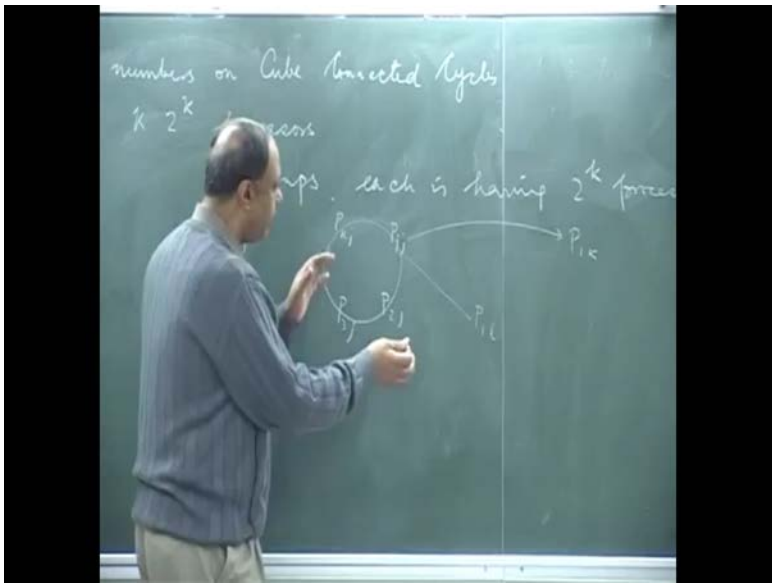
(Refer Slide Time: 03:46)



Then similarly, you will have. So, each of them is this cycle. Each of them is the cycle then, I get a hyper cube. Now, can I introduce my hyper cube algorithm to collapse it and get the result here, yes or no? This is connected, This node is connected with this node exactly at one part. So, I can collapse it and I can get these data here similarly, I can get the data here. But, if I collapse this reaction so, this data will move to here, this data also will move to here.

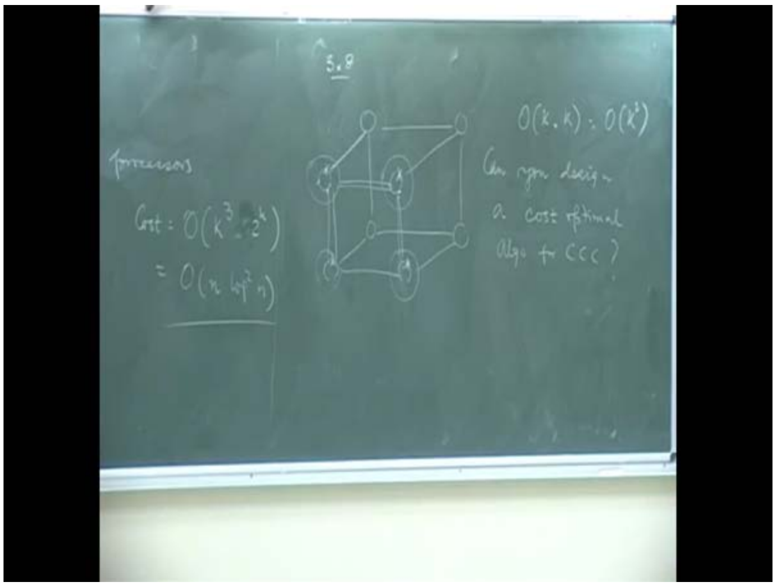
Now, if I collapse this, this data will move to here. Agreed? Now if I do that, what problem I may face can you tell me? This one thing you remember, this is connected with this node through only one connecter but, you observe the 2 connecter are laying here and these 2 data is node lying here, it is separate here and here. Is that ok? So, every time whatever data you said you have to added in that cycle and then, bring it to the position so that you can transcript the data. Is it clear? I need complete details.

(Refer Slide Time: 06:18)



Say the processor P 1 j is connected with P 1 k; now, if you bring this data here it will come to this P 1 j. Now, next time you will be collapsing, this is connected with P 1; P 1 something else; P l l say but, while you get the data from here to here you need to bring them into another positions.

(Refer Slide Time: 06:53)



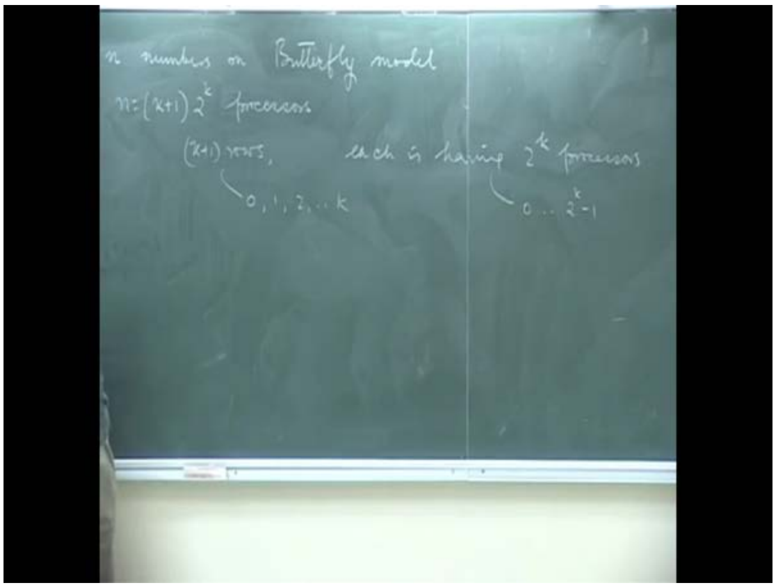
Say for this case, if I have 3 into 8 processors. So, this 3 forms cycle; this 3 forms cycle and so on. Now, if you collapse the data from this side to this side. Similarly, this data is lying here; similarly, this data is lying here. So, what happens? You can send only 1 data which is added here but, that is not required. You need to get the sum of all these 3 elements that has to be sent. You understood? This contains the 3 element. So, you first make the sum send it here so, the data is ended here. Similarly, you make a sum send it here radiant circuit to this processor; similarly, send it data is added to this then, they will added to this. Now, in these radiant, this contains a sum of 4 elements; this contains sum of 1 element; this contains sum of 1 element; this contains sum of 4 elements, 1 element, 1 element; 4 element 1 element 1 element; 4 element 1 element 1 element.

Now, in this you added, in this you added, you get a sum of 6 elements. Bring it to that appropriate position so that this goes up, this goes up. So, this contains the sum of 7 element, this contains sum of 7 element; this contains sum of 4 element, this contains some 4 element, this contains sum of 1 element, this contains sum of 1 element. Now, you add all these element; add all these elements and you send it to here, that will give you the sum of all 24 elements. So, you need little book keeping to do all this one. Yes? So, what is the time complexity you are expecting in that?

You observed at every stage every stage you need in the worst case k additions. Why was k? That say front side you are making low addition but, this side you make the k addition, which is linearly connected. So, you need k order k time. Then, you are collapsing, it which take order one time. Next time again k-th additions and again order 1 time. So, what is the time complexity is coming? Order k log n times log k times. So, order k into log of 2 to the power k which is k. So, it is order k square time. Is it ok? This first k is for adding the using the cycle and then collapsing will be done k types where, 2 to the power of k . So, it is order k square times. So, cost is order k cube, 2 to the power k and this is nothing but, order n log square n which is a way of factoring log square n.

Now the problem is, can you design a cost optimal also for CCC? So, this is the things you have to now think. That is if possible to design a cost optimal parallel algorithm on cube connected cycle. So, first you look for additive algorithms and then, we find out for one value of p, you get the cost optimized algorithms. Say, here the load is somewhere else, load is that cycle. Unnecessarily we are treating, the putting additional effort on the cycle. So, you have to look into that, whether it is possible to design cost of parallel algorithms on cube connected cycle.

(Refer Slide Time: 12:22)


n numbers on Butterfly model

$n = (k+1) 2^k$ processors

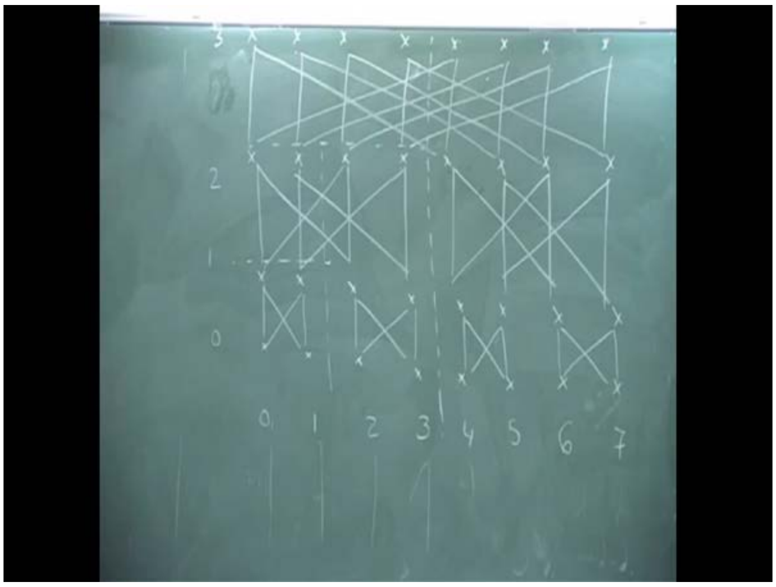
$(k+1)$ rows, each is having 2^k processors

0, 1, 2, .. k

0 ... $2^k - 1$

Now, the next one is the sum of n numbers on butterfly. Here you have assumed, n is equals to k plus 1, 2 the power k processors; k plus 1 groups or rows which is having 2 to the power k processors. Now, rows the number 0, 1, 2 upto k and processors are number; something like that. Did I start with 0, 1, 2 k or 1 2 k plus 1? There should not be any problem on, 0 to k.

(Refer Slide Time: 13:28)



And it is in the form, if I remember 0, 1, 2, 3, 7 and you we are telling that 1, 2, 3. This is your 4 into 8 that is, 32 processor some butterfly network. Now, in order to find the sum of n numbers; now, let us show that, this processor contains one element and you observed that you observe that this is linearly connected and also you can set the data from here to here also by login time. This is also login time, this is also login time. So, from any processor to another, to set that to another processor you take a maximum or a login time. The strategy is very simple that, you target is to bring all this data from this to this place and where were you are bringing, you are just adding. Then, the problem reduces to this row .

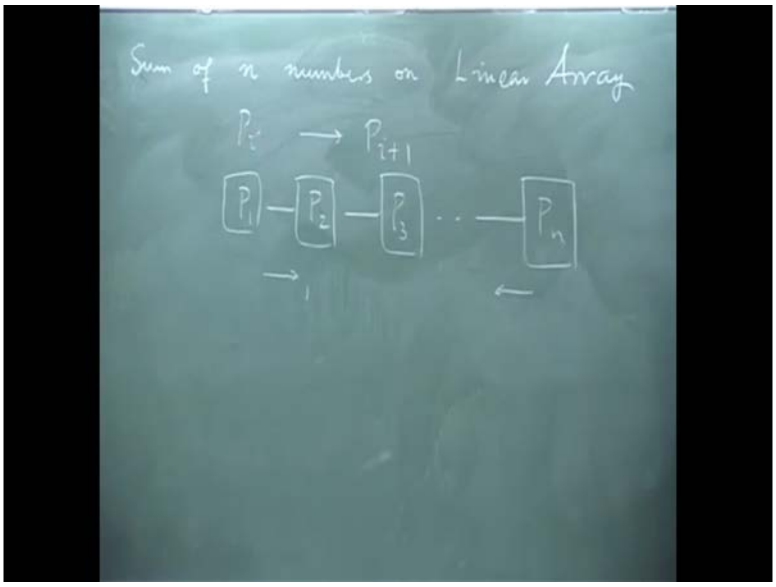
Now, this half will be valid or good if you can collapse this part to this side then, your factor will come in this part. Again you divide, you bring this data from here to here then, you bring the data here then, you will cut and paste. first state these are not connected, which are not connected. See, my point is, how long if I will bring this data I will discuss. But, my point is that, the best way is that, you divide it in such a way that you can divide into the 2 groups. You bring this data here; then, you bring this part here; then, your deduction is another by butterfly. Again partition, this bring it here, you bring down then, you get another butterfly model. this is the idea.

Now, one thing you remember that at this stage to bring this data from here to here it will not take less than more than login time, whatever way you do it. So, can you prove this, that will not take more than or a more than time. So, the idea is you bring this data here; somewhere similarly this data you can bring it here. Agreed or not? Now, if you bring this data here and add it, you bring this data here and add it here; next level you bring this data and add it. So, all this 4 elements you have done not of half order login time.

Now, you get the data to this part. This connecting this, order one time then, you bring this data here, another 1 time. Then, this is the some butterfly and you only take it. So, every stage you do not need more than or a login time and size is login so, it takes logs square n time. Now, but they are very costly, n into log square . This, what you observe that if we use the calculating system that the data is coming up and then, possibly you will observe that butterfly gives better results. Otherwise, most of the time, most of the processors are remaining ideal.

Say for example, after collapsing as you have seen in that case of hypercube, here also it is the case that k that, after collapsing 50 percent of the processor are ideal. Which is and as you keep the processor ideal means your cost is increasing, you are not make, you are not being, you are not able to make that able to use the . So, that is the bad point of butterfly model like hypercube.

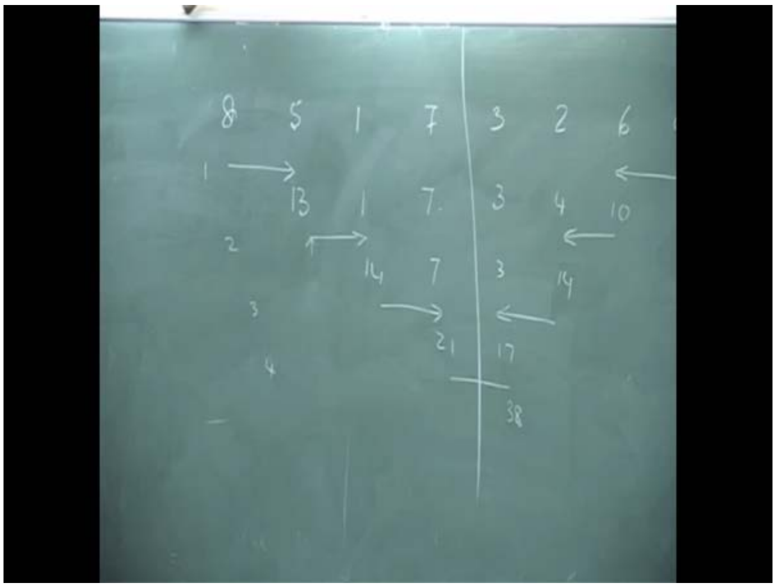
(Refer Slide Time: 20:59)


Sum of n numbers on Linear Array
$P_i \longrightarrow P_{i+1}$
$\boxed{P_1} - \boxed{P_2} - \boxed{P_3} \cdots - \boxed{P_n}$

Now, let us have the sum of n number problem on linear array. That is, very simple structure you have. P i is connected with P i plus 1, provided it exists and the structure is P 1, P 2, P 3, P n.

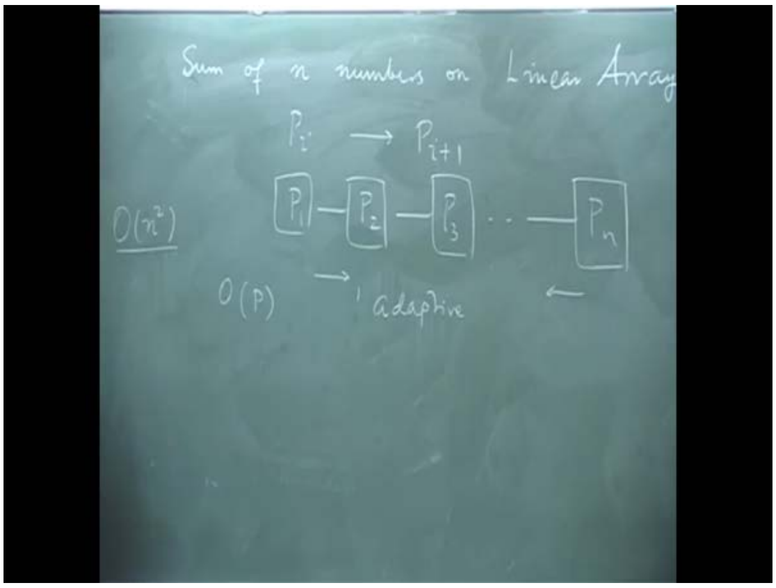
Now, can you tell me or suggest me how to find this sum of n numbers? Suggest some method which is very good. . So, what he is suggesting is that while you are adding this side it starts collapsing from that side also. Then, what is the complexity you are expecting? . See that the data is here.

(Refer Slide Time: 22:32)



Let us take one example also; 5 1 7 3 2 6 4 and 8. What this method is that you add this and . Sure you get 13, 1 7 3 4 10, 14 7 3 14, 21 17, 38. This is the method you have looking for. So, how many steps you made? 1 2 3 4 5, right. Then, suppose you have 16 elements, 16 you need . Sure? So, what is a you are talking n by 2? So, complexity wise you are not gaining anything, what you have to do. Just you have to take this element here and add this element, that element, that element and so on; which takes order n times. Why to take such a complex thing that you will be dividing this, you will be dividing this into the 2 parts then, you have to keep always a pointer to move which one you have to move right and which one you have to send back all those things. Nothing is free of cost, it will add some complexity. However, if I see this, number of parallel addition will be less but, complexity wise you will not be gaining much.

(Refer Slide Time: 25:07)



Sum of n numbers on Linear Array

$P_i \longrightarrow P_{i+1}$

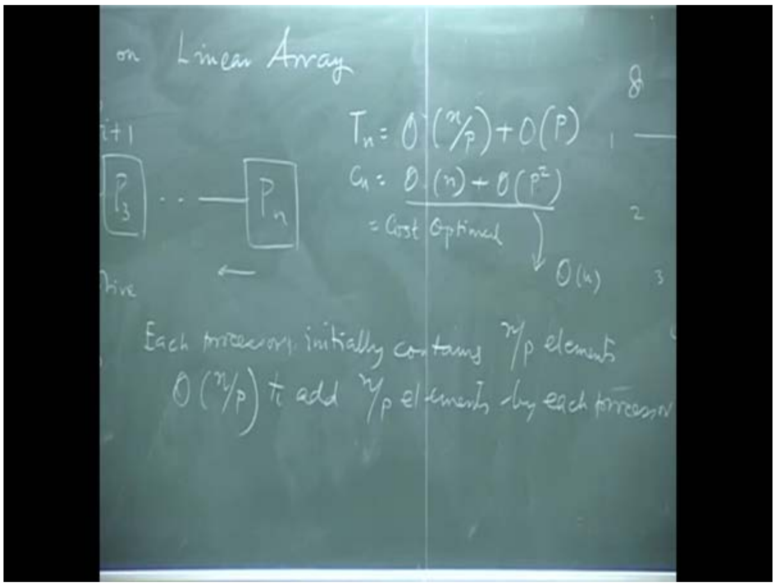
$\boxed{P_1} - \boxed{P_2} - \boxed{P_3} \cdots - \boxed{P_n}$

$\underline{O(n^2)}$

$O(P)$ $\longrightarrow$ adaptive $\longleftarrow$

So, the cost is order n square. You have used n processor, order n time order n square time and order n square is the cost which is far away from that cost of humanity. However, you can think about this whether if it is an adaptive you need order P times. In case of adaptive you need P times to find sum of P numbers.

(Refer Slide Time: 25:48)



on Linear Array

$T_n = O(n/p) + O(P)$

$C_n = O(n) + O(P^2)$

= Cost Optimal $\rightarrow O(n)$

Each processor initially contains n/p elements

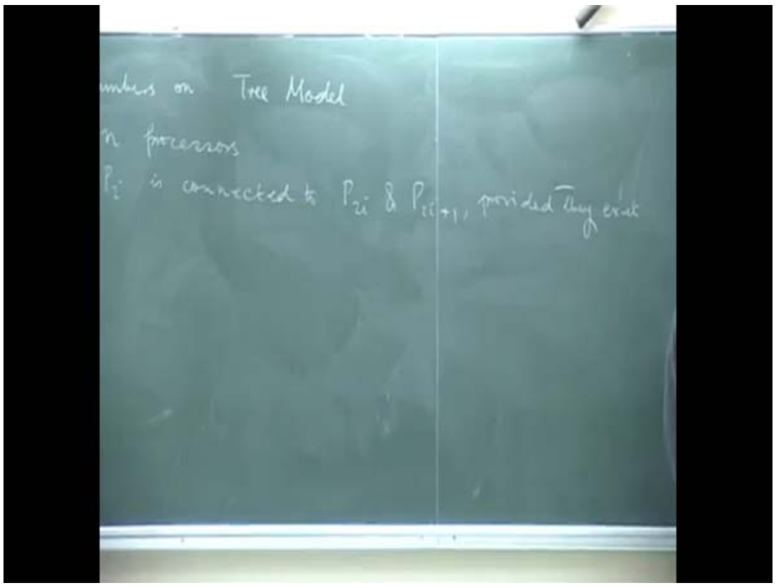
$O(n/p)$ to add n/p elements by each processor

Now, if I assume that there are n elements . Each processor initially contains n by P elements. Let us assume that the P processors and n elements, each processor initially contains n by P element and each processor is asked to find the sum of these n by P elements sequentially. So, you need, if I add, I need order n by P, to add n by P elements by each processor's. Now, you have P processor, each processor contains the sum of n by P elements. So, if I collapse them and add; then, I need order P times to get the sum of n numbers. So, the total time T n becomes order n by P plus order P. This n by P is for sequential addition of n by P elements, this P is for parallel addition of P elements by P processor.

So, cost of sum of n elements becomes order n plus order P square or it becomes order n plus order P square. Now, these will become cost optimal if, this is order n. Agreed or not? This algorithm becomes cost optimal if, this is order n. Now, this becomes order n if p square is order n. That is, if p is square root of n.

So, basically you have square root of n processor, each processor contains square root of n elements and then, it performs sequentially addition of square root of n elements by each processor then, use parallel algorithm for finding the sum of square root on square root n elements, which takes order square root of n, order square root of n. So, total time of this will become order square root of n, cost becomes order n.

(Refer Slide Time: 28:59)

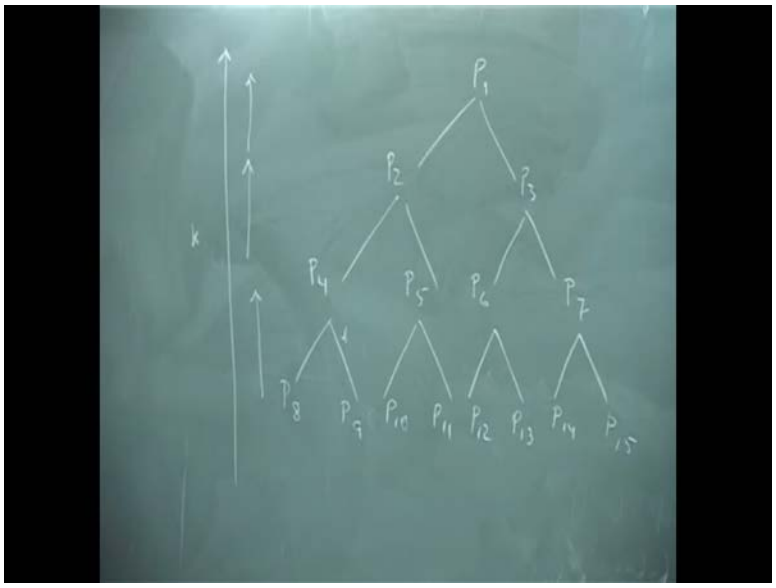

umbers on Tree Model

n processors

P_i is connected to P_{2i} & P_{2i+1}, provided they exist

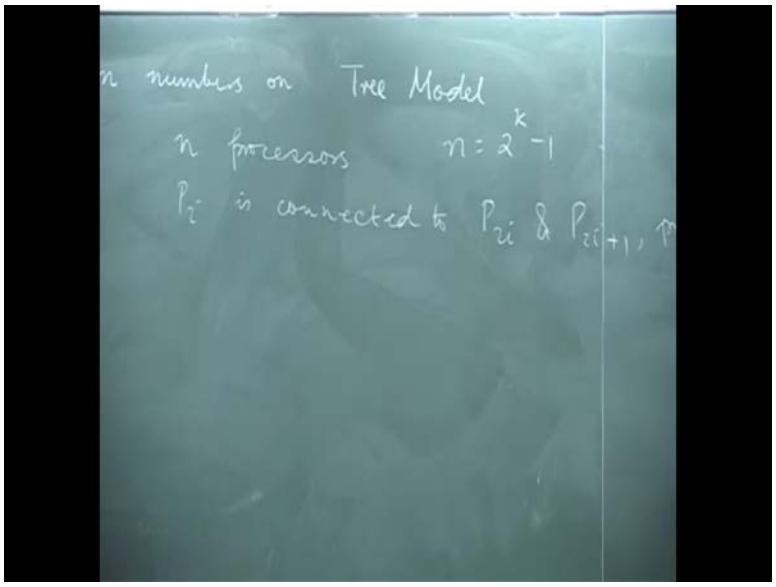
Next one is the sum of n number on 3 model. And here you have assumed that you have n processors, P i is connected to P 2 i and P 2 i plus 1; provided they exist.

(Refer Slide Time: 29:52)



So, here every processor you have most 3 connections and you thing it is P1, P2, P3.

(Refer Slide Time: 30:29)


n numbers on Tree Model

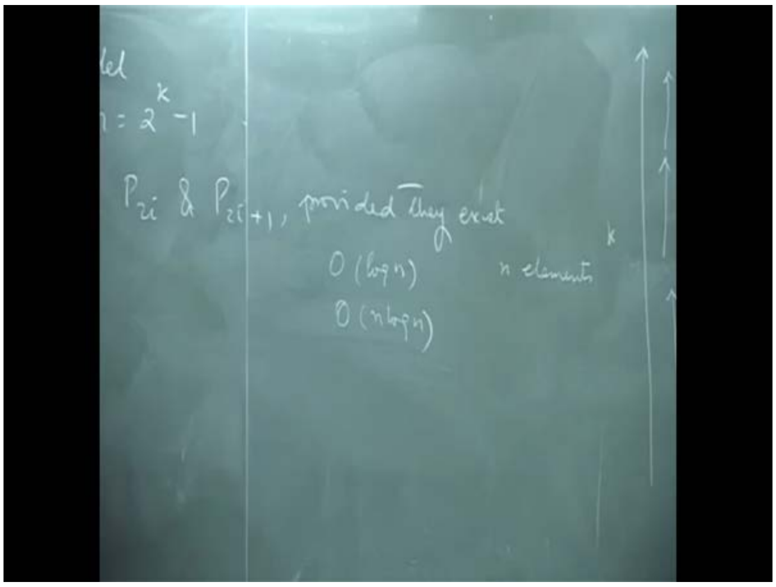
n processors $n = 2^k - 1$

P_i is connected to P_{2i} & P_{2i+1}, P

So, n can be expressed in the form of 2 to the power k minus 1. And height of this height of this is k. What is the height? K. Now, here also it is actual that every processor initially for case one element, every processor contains one element and we are interested to find the sum of n numbers. So, how to do it? The collapsing can be again; so, bottom over approach. You take this element and add with this, both the element come up and add with this. Similarly, these elements come up and add. Next step: so, this contains sum of P elements and this contains sum of P elements and so on. So, this contains sum of 7 elements; this contains sum of 7 elements.

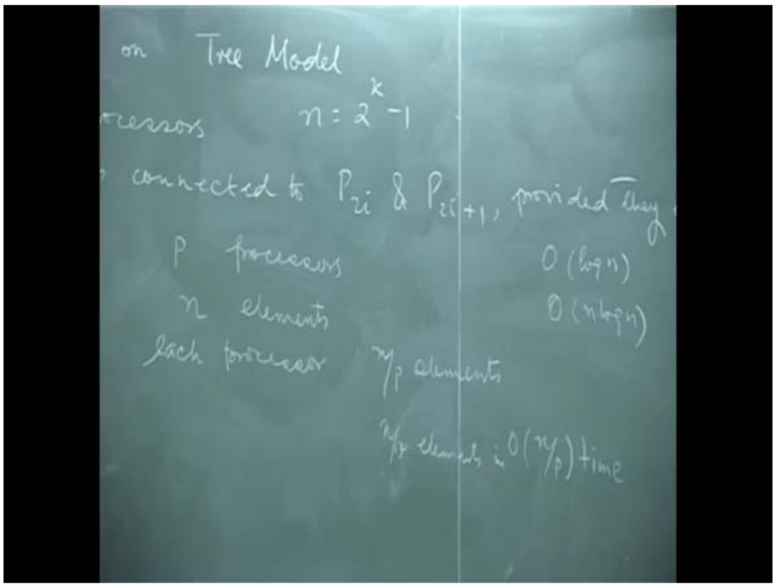
So, what is the time complexity? Pardon, what all? K, which is of order log n. And one thing you remember, even though we are telling the both the processor say for that simultaneously to this but, processor is only one processor so, delay will be there. Agreed or not? Because, this is data simultaneously P 9 cannot send the data so, p 4. So, this should be some delay factor from data to this data but, this is very nominal.

(Refer Slide Time: 32:32)



So, you have order at login time, order login time to find the sum of n elements. So, cost is order log n. See, you remember this cost is increased because, the processor most of the processor are remaining at every interactions. So, you have to make this people busy so that the cost comes down.

(Refer Slide Time: 33:07)



on Tree Model

cessors $n = 2^k - 1$

connected to P_{2i} & P_{2i+1}, provided they

P processors $O(\log n)$

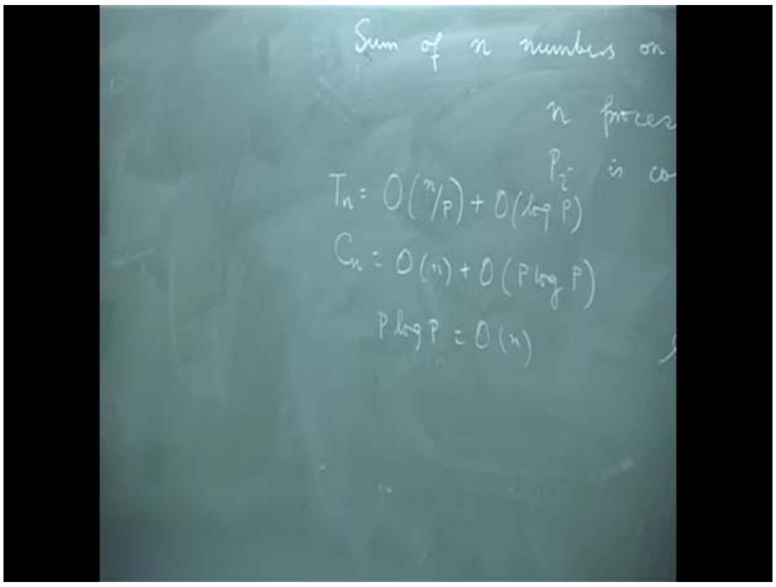
n elements $O(n \log n)$

each processor n/p elements

n/p elements in $O(n/p)$ time

Now, let us see what happens to additive algorithms if we have P processors, P processors which form the k model n elements. Each processor contains n by p elements. So, sequentially these processors perform the sum of these n by p elements which takes order n by p times. Sequentially, each processor is thus to find the sum of n by p elements which takes order n by p time. Then, these P processors are used to find the sum of this P element of k model which takes order log P times.

(Refer Slide Time: 34:18)



Sum of n numbers on

n proces

P_i is co

$T_n = O(n/p) + O(\log P)$

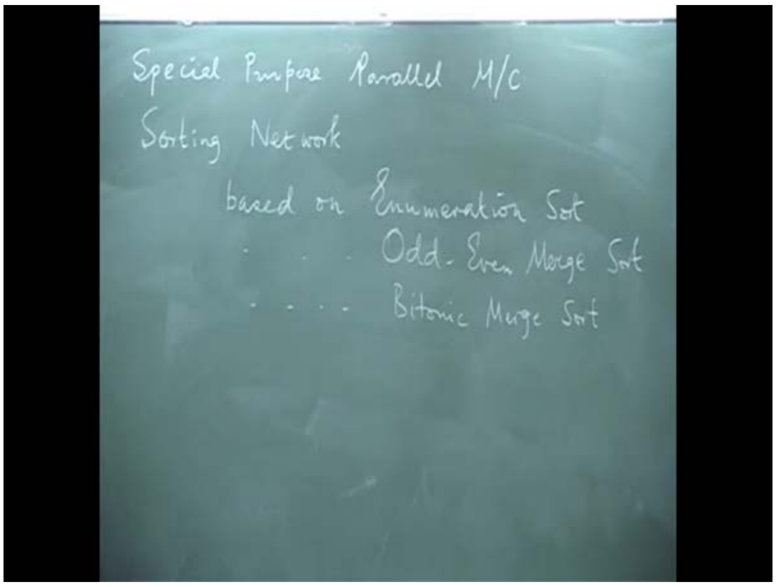
$C_n = O(n) + O(P \log P)$

$P \log P = O(n)$

So, the total time to find the sum of n element using P processor becomes order n by p plus order log p. And that means cost becomes, order n plus order P log P. So, this becomes cost optimum only if P log P becomes is order n. That is, P of the order n by log n. P is of the order n by log. So, you can verify it.

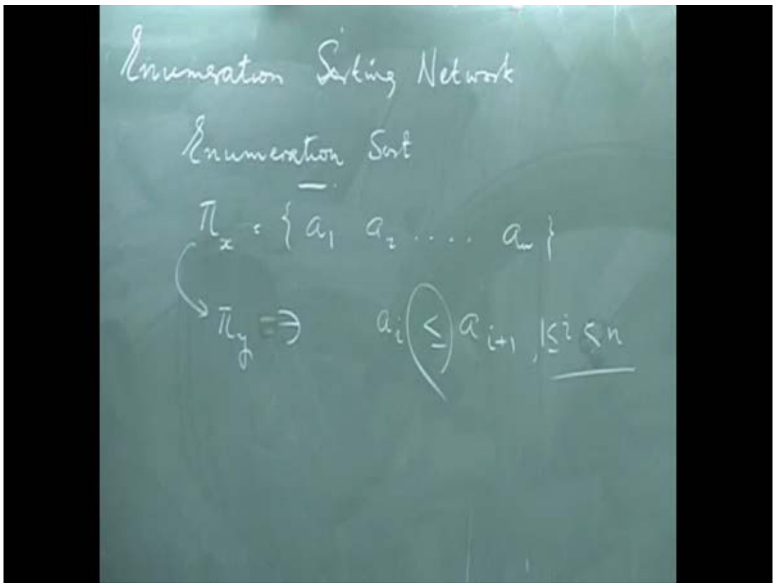
But, see that is finding the sum of n elements on one dimensional P have been model which is similar P model. So, yes or not? Because, you will not gain much or to linearly addict on that. So, you can derive cost optimal parallel algorithm on that. Similarly is the 2 dimensional pyramid models. So, you can think about how to write the parallel algorithms on 2 dimensional pyramid models. How to write or how to find the sum of n elements on 2 dimensional pyramid models. So, once you know the finding the sum of n element, you know the how to find the minimum, maximum all those things. This same class of problems you can really solve on parallel machines.

(Refer Slide Time: 36:14)


Special Purpose Parallel M/C
Sorting Network
based on Enumeration Sort
. . . Odd-Even Merge Sort
. . . . Bitonic Merge Sort

Now, we will be discussing the special purpose parallel machines. Say, here we will discuss mainly sorting network. There are three types of sorting networks we will be discussing. First one is based on enumeration sort based on enumeration sort; second one is based on odd-even merge sort; third one is based on bitonic merge. So, these are the 3 special purpose sorting network that will be discussing. Today, I will try to finish the enumeration sorting technique, enumeration sorting network.

(Refer Slide Time: 37:42)



Enumeration Sorting Network

Enumeration Sort

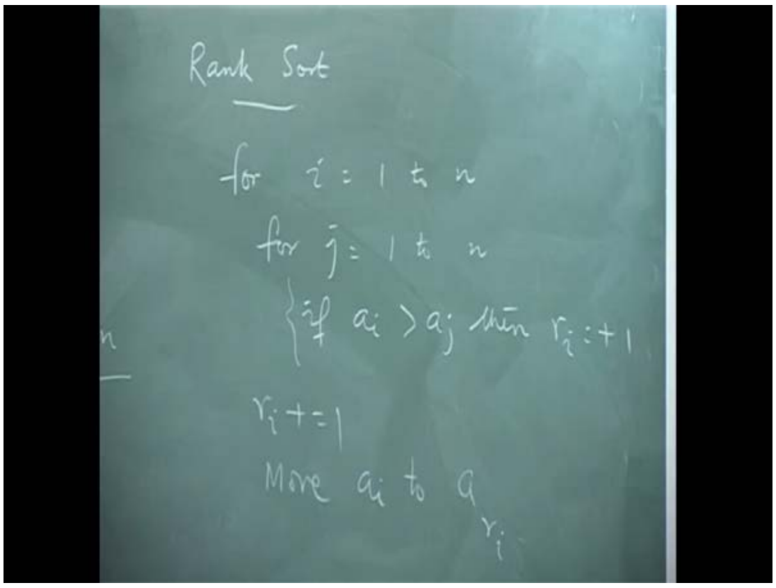
$\pi_x = \{a_1\ a_2 \dots a_n\}$

$\pi_y \ni \quad a_i \boxed{\le} a_{i+1},\ 1 \le i \le n$

So, what is enumeration sort, first let us understand that one. Now, before that, what is sort? What is sorting? Can you tell me, what is sorting? You have n elements there a 1, a 2, a n. This is arbitrary additive. What you want to do? Order in some form either in increasing order or decreasing order.

So, basically I can write this is a permutation of n elements. Say, phi x. You want to get phi y such that, a i is less than or equals to a i plus 1. i is greater than equals to n. You are looking for a permutation phi 1 of n elements say one equivalent n such that, the i is less than a i plus 1. That is your. Now, this need not be, this symbol is not be less than or equal. It is some operator. Now, in the enumeration sort which is sometimes known as a rank sort; which is a old sorting technique. What it does? That you observe to tell whether a i is greater than a j, a i has to be compared with a j. So, what happens? That you take one element, you compared this element with every other element. And you see what is his rank with reference to every other element. Then, you move the data to that respected right element position.

(Refer Slide Time: 40:11)



Rank Sort

```
for $i := 1$ to $n$
    for $j := 1$ to $n$
        {if $a_i > a_j$ then $r_i := +1$
    $r_i += 1$
    Move $a_i$ to $a_{r_i}$
```

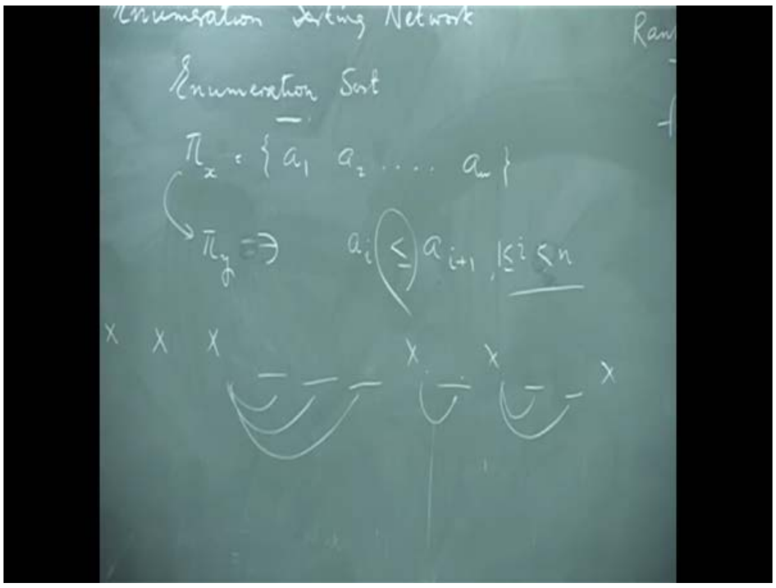
That is, if I see that way then, for simple i equals to 1 2 n for j equals to 1 to n. If a i is greater than a j then, rank r i is increased by 1 else, we do not know whatever . Now, if you know r i is 5 that means the rank of a i is . No. That means, what it does that if I find r i is 5 it indicates there exists 5 elements which is smaller than a I; there is a 5 element which is smaller than a i. So, length of a r is same. Agreed? So, finally you do r i is increased by 1 and move a i to a r i position. move here a i to a r i position. So, that is the simple algorithm you could rank sort algorithms and it is very simple to understand also.

Agreed? Yes.

Now, here is the base assumption is the element of this state. This is the base assumption the base elements of this states. Now, what happens if the elements are not to be restricted? If there exists some elements which are same because, this will not work in this case. Agreed or not? So, how are we going to handle that? Is it difficult for us to handle? Can you suggest some method? Is it possible to modify these sorting technique if some of the elements are not district. We can keep an array in which we can keep an count of each and everything. Yes. And then we can modify according to that every time that particular elements occurs then we subtract the one form that particular question in a time. Like suppose, a occurrence is a particular sequence then, I keep an array in which I could in ever time I subtract 1 from this side good which I am keeping.

But, how you will know that 5 rockets are ? Once I will scan the whole sequence. What you want to do there? You want to scan the whole sequence first and then you compute the frequency of each element and then, if they arrive empty otherwise, I am moving to next empty . How do you know that? Means, we need after we attend a storing instead of putting into a r i, put it in a different v r i. We can introduce some random ordering amounts equal elements that initially, the positions are different. So, I felt that the element occurring at the right is less than element occurring at . No, let us understand one thing. In this case what happen if require, first let us understand this one. If requires, what happens? .

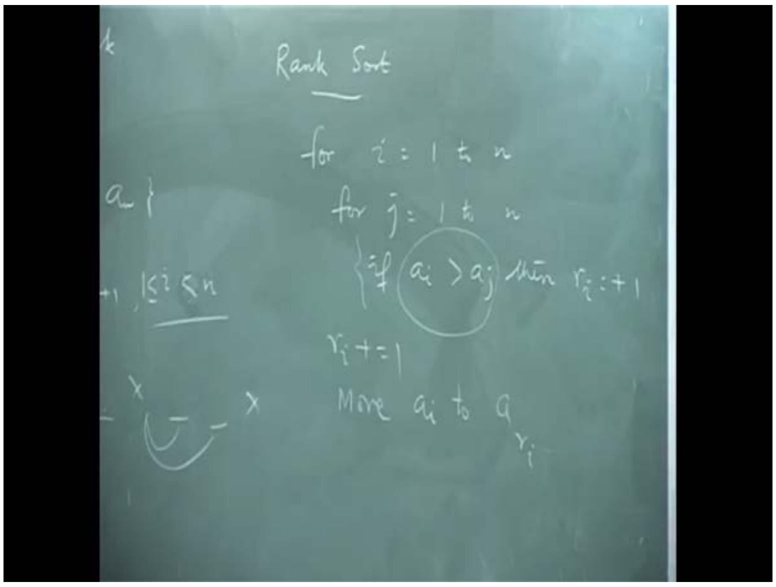
(Refer Slide Time: 45:33)


Enumeration Sorting Network
Enumeration Sort
$\pi_x = \{a_1\ a_2 \dots a_n\}$
$\pi_y \ni\ a_i \leq a_{i+1},\ 1 \leq i \leq n$

So, you may find in the senior that element is here, element is here, element is here, blank, blank, blank, element is here, blank, element is here, blank, blank, element is here like that senior you will be getting. That is the only thing you will be getting, nothing more than that. And this blanks you can eliminate by collapsing that is not problem. But, by that process you will be losing this information, how many times this element occurred that instead of it occurred 4 times, you will be writing only one time. So, you can fill this later on also. You just can anything, scan and fill this value by this and so on; this can be done. But, this cannot solve our problem.

Do you know what is the stable sort? have we got the sum? This quick sort is the stable sort. In which the 2 elements which are occurring longer than because if I have the 2 elements which are if it is a distinct then occur in that are sorted sequence then, it same order in the. Yes. Quick sort is not a stable sort because if I have the 2 elements, if it is a district in that form there is no problem but, if it is not a distinct then, quick sort will not give you any guarantee that the first occurred element will occur first, second element occurred the same element will occur second time. That is not the . So, same thing by doing that, that information you will be missing. You get the sorted sequence but, it will not give you the step rule one that first occurred element because, suppose I have an element 3, along with the 3 some more information may be there. 3 is a one indexive but, then we color this, we write this, there are all those things. But, if you put them here, next 3 also have been the different color different height, that also is a replicating, that may not solve by purposes. That is why that stable sorts we some times need.

(Refer Slide Time: 48:11)

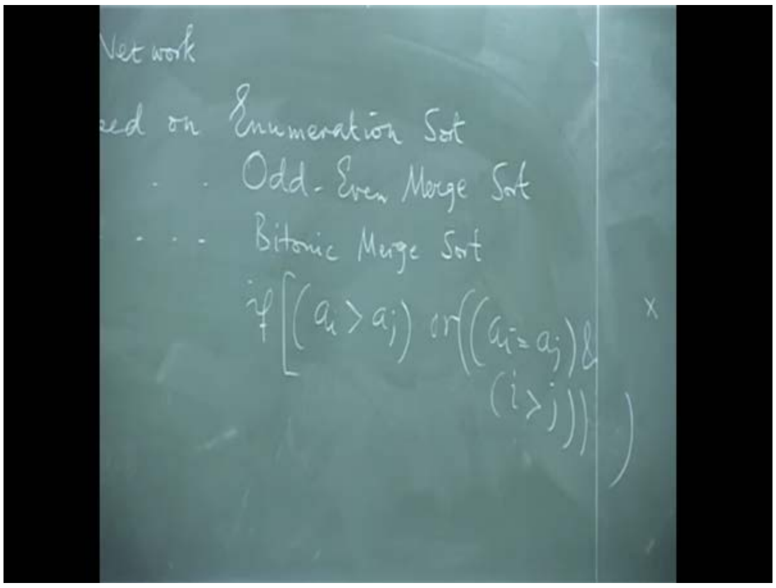


Rank Sort

```
for i := 1 to n
    for j := 1 to n
        { if ($a_i > a_j$) then $r_i$ := +1
    $r_i$ += 1
    Move $a_i$ to $a_{r_i}$
```

Now, you can handle this one very easily if I change this condition.

(Refer Slide Time: 48:20)



Net work

ed on Enumeration Sot

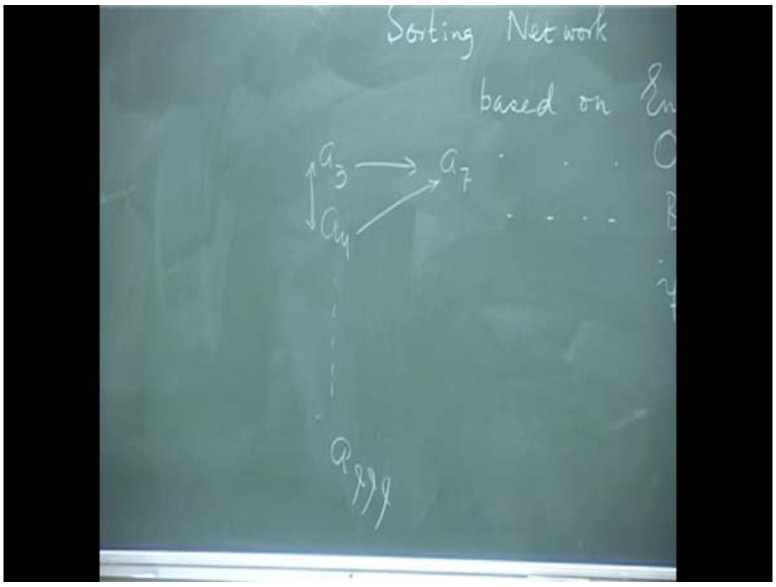
. . . Odd-Even Merge Sot

. . . . Bitonic Merge Sot

$\text{if}\left[(a_i > a_j) \text{ or } ((a_i = a_j) \& (i > j))\right)$

Now, the condition will be, if a i is greater than a j or a i is equals to a j and i is greater than j then, you increase your time. i is what? i is , i is element index define this greater than this j-th element in the j index then you try to increase. So, this gives you the guarantee, this give you the guarantee in the first obtained element has the less rank with the next accrued element. Agreed? No, you will not get.

(Refer Slide Time: 49:25)

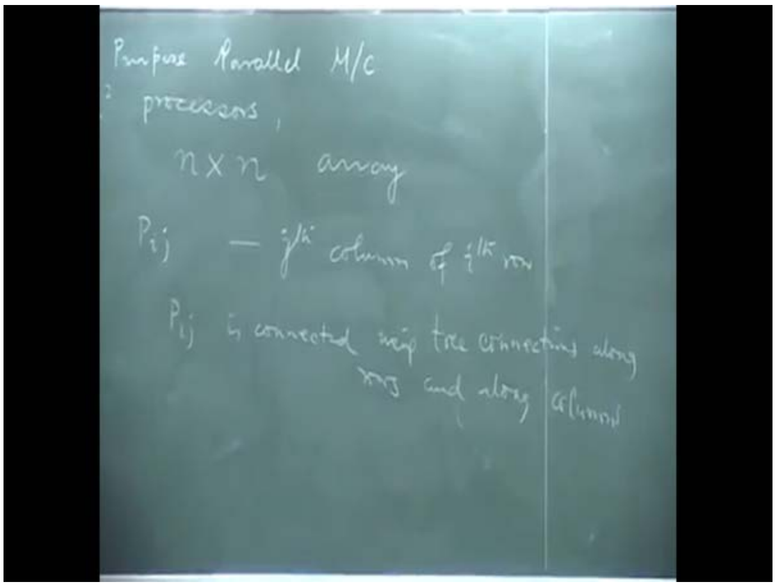


See a i is; say question as an example, a 3 should move to a 7 position and a 4 and this 2 are same should also showing that a 7, what are able to do? a 7 equals? Because we have moved a 3 to a 7; yes, we have a 7 will be equal to a 7. So, what we will do with this rank increase by 1. Whose rank? A 4's and then, moving to a 8. This, this and this are same, now you tell me. are over there in. Then? So, that means complexity increase. No, no.

So, there is giving over read hand in that case. The additional payment all those still will be coming. So, this simple way to handle this is, if you find the same then, you just check the i index. If i occurred for first, he should be given the first chance and i in demand must be distinct, there is no impurity in that, i and j cannot be equal. So, that stable thing all that can be taken care by enumeration sort.

Now, this sorting network which is been designed based on this strategy is as follows.

(Refer Slide Time: 51:06)



Purpose Parallel M/C

n^2 processors,

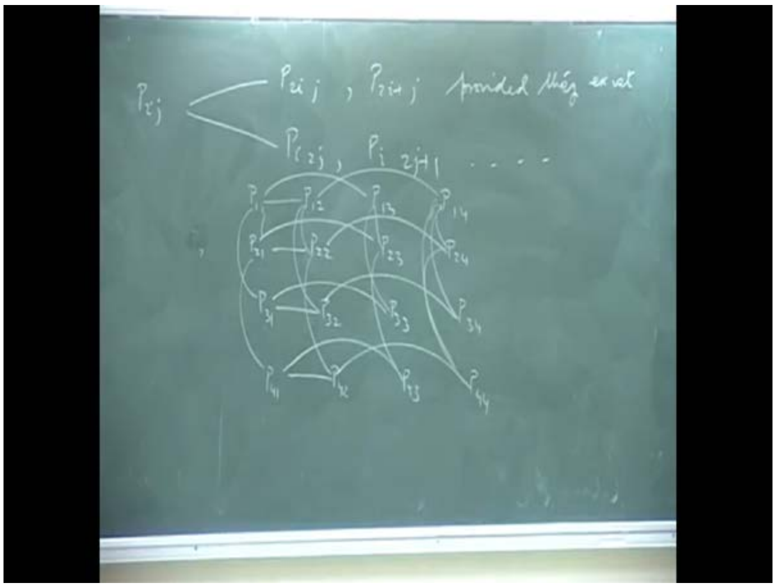
$n \times n$ array

P_{ij} — j^{th} column of i^{th} row

P_{ij} is connected using tree connections along rows and along columns

And this processors are arranged in the form of 2 dimensional array. So, this n square processor are arranged in the form of 2 dimensional array and processor. So, P i j is the processor in the j-th column of i the row. P i j is the processor in the j-th column of the i-th row. Now, P i j is connected using 3 connections along rows and along columns, what it means?

(Refer Slide Time: 52:33)

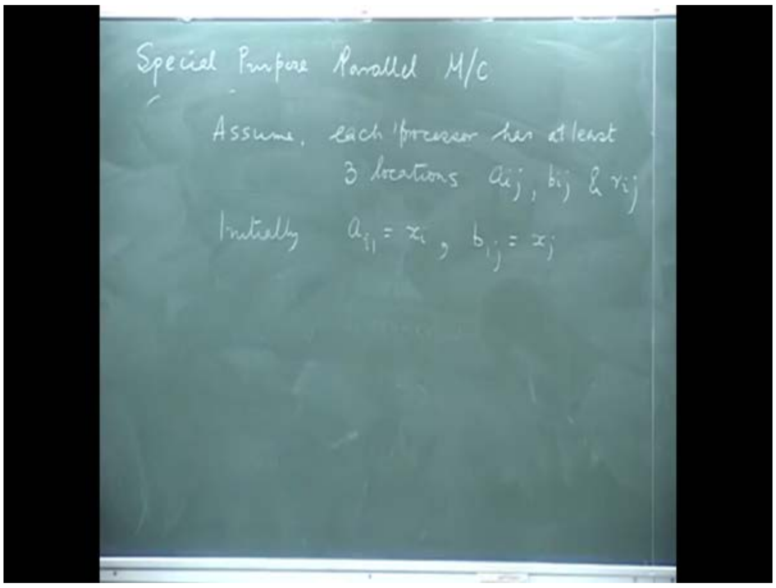


The P i j is connected with P 2 i j, P 2 i plus j provided they exist and also P i 2 j, P i 2 j plus 1 exists. That means, suppose we have P 1 1, P 1 2, P 1 3, P 1 4, P 2 1, P 2 2, P 2 3, P 2 4, P 31, P 3 2, P 3 3, P 3 4, P 4 1, P 4 2, P 4 3, P 4 4 these are the second processors you have.

So, connections are like that; P 1 is connected through it then, this is connected to this. So, this is the connected model so that it follows the 3 connections, if I see from this direction and also you see from this direction. Only thing is that it is very complex, if you have . But, to send the data from one corner to another corner you need how much time? Can you tell me? To send the data from one corner to another corner, how much time do you need? . Yes. Worst case it is order log n because, your n elements are there they form a treat on itself. So, height is . So, what from one corner to another corner, what does he is telling is may be 2 login timings are needed to transcript the data from one corner to another corner.

So, this is a network or model we are using for enumeration sort. Initially, it is assumed the element; see one thing remember that why should we discuss parallel algorithms for some problem, we are silent about the input part or output part. How input will be coming, that part we are not because, input takes huge amount of time and this input, if you consider then, complexity may fail. Suppose, n elements you have and this n elements you want to keep in all the processor, how much time you will need? You will need log n time to broadcast the data from corner to another corner for 1 element. So, log n element to be broadcasted. You understood? It takes lot of time. So, what we assume that initially the elements are stored here and the elements are stored here in this case.

(Refer Slide Time: 57:29)



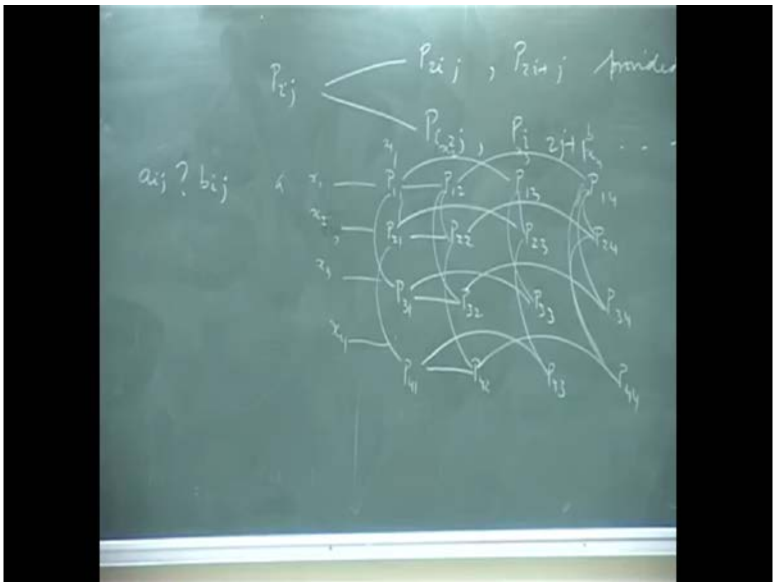
Special Purpose Parallel M/C

Assume, each processor has at least 3 locations a_{ij}, b_{ij} & r_{ij}

Initially $a_{i1} = x_i$, $b_{1j} = x_j$

So, each processor which is assumed that each processor has at least 3 memory locations, 3 locations a i j, b i j and r i j; a i j, b i j and r i j. r i j is for keeping the rank of element and a i j and b i j to keep the elements. Initially, a i 1 is x i and b 1 j is x j.

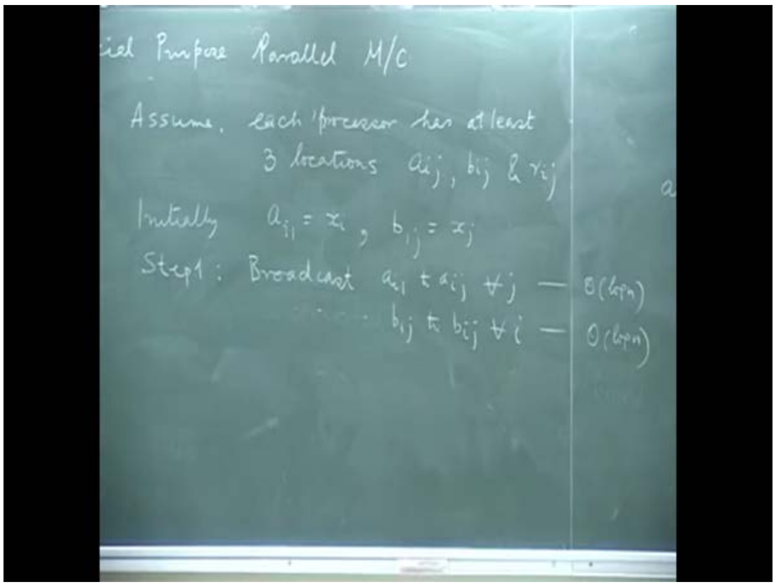
(Refer Slide Time: 58:45)


P_{ij}
$P_{2i\,j}$, $P_{2i+1\,j}$ provided
$P_{i\,2j}$, $P_{i\,2j+1}$ ...
a_{ij} ? b_{ij}
x_1
x_2
x_3
x_4
P_{11} P_{12} P_{13} P_{14}
P_{21} P_{22} P_{23} P_{24}
P_{31} P_{32} P_{33} P_{34}
P_{41} P_{42} P_{43} P_{44}

That means, here you have x 1, x 2, x 3, x 4 this element contains and here this is x 1, x 2, x 3, x 4, this is b elements and this is a elements.

Now, what happened that you observed that in the rank sort every elements wants to be compared with every other element. So, what happens? That this a I will be broadcasting to all the processors, b will be broadcasting to all the processing. So, a i j and b i j are the 2 elements x I will be compared to x j basically. a i j will be compared with b i j, what does it mean? x i will be compared with x j and then, we will put rank r i j is 0 or 1 based on that. Then, we collapse it and add it sum of r i j and keep the data here. And once you get the rank, you broadcast that element to the appropriate positions. So, first part is that you broadcast a i 1 to all a i j and similarly, b 1 j to all a r i j.

(Refer Slide Time: 1:00:34)



ial Purpose Parallel M/C

Assume, each processor has at least 3 locations a_{ij}, b_{ij} & r_{ij}

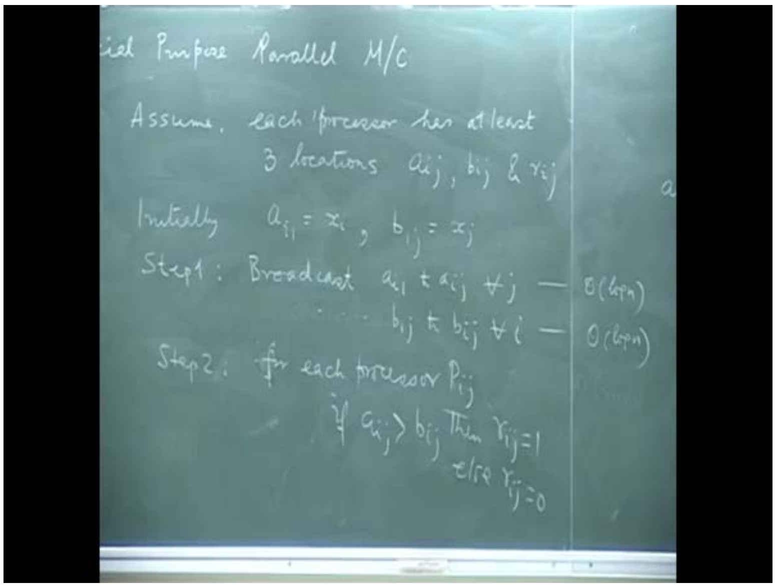
Initially $a_{i1} = x_i$, $b_{1j} = x_j$

Step1 : Broadcast a_{i1} to a_{ij} $\forall j$ — $O(\log n)$

b_{1j} to b_{ij} $\forall i$ — $O(\log n)$

So, step one is broadcast a i 1 to a i j for all j then, broadcast b 1 j to b i j for all i. So, this element would be broadcasted to, this element x 1 has to be broadcasted to this elements; x 2 would be broadcasted to this, x 3 would be broadcasted to this and so on because x 3 is nothing but a i and p 1. Similarly, this will be broadcasted. So, any processor will contain one a i j and one 1 b i j. Again, a i j is your x i and b i j is your x j. This can be done in order log n time, this also can be done in order log n time. Agreed?

(Refer Slide Time: 1:01:55)



ial Purpose Parallel M/C

Assume, each processor has at least
3 locations a_{ij}, b_{ij} & r_{ij}

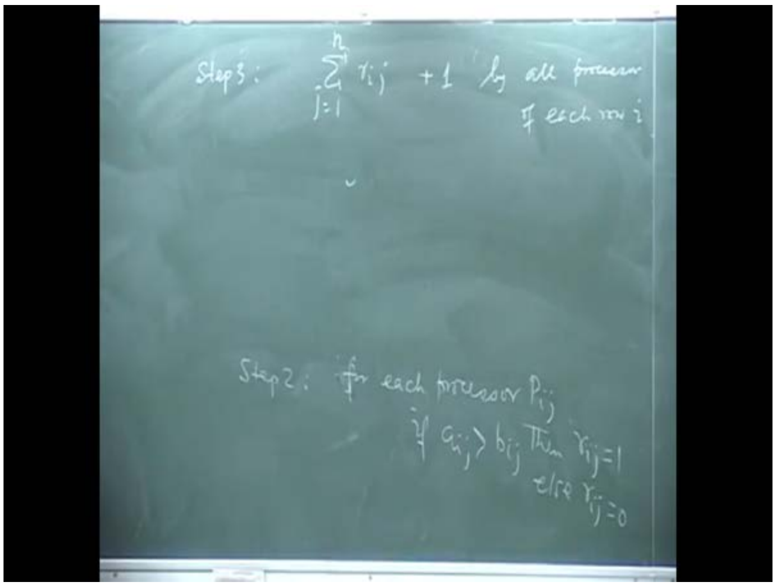
Initially $a_{i1} = x_i$, $b_{1j} = x_j$

Step1 : Broadcast a_{i1} to a_{ij} $\forall j$ — $O(\log n)$
b_{1j} to b_{ij} $\forall i$ — $O(\log n)$

Step2 : for each processor P_{ij}
if $a_{ij} > b_{ij}$ Then $r_{ij} = 1$
else $r_{ij} = 0$

Now, step 2 if every processor, for each processor P i j, if a i j is greater than b i j then, r i is 1 else r i j is 0. What it means that a processor P i j is compared a i j with b i j and if you find that a i j is greater than b i j that rank on a i j is 1 otherwise it is 0. So, every processor contains now, r i j which is either 0 or 1. Now, if I sum that this side, I get the rank of x sort with reference to every other element. Agreed or not?

(Refer Slide Time: 1:03:05)

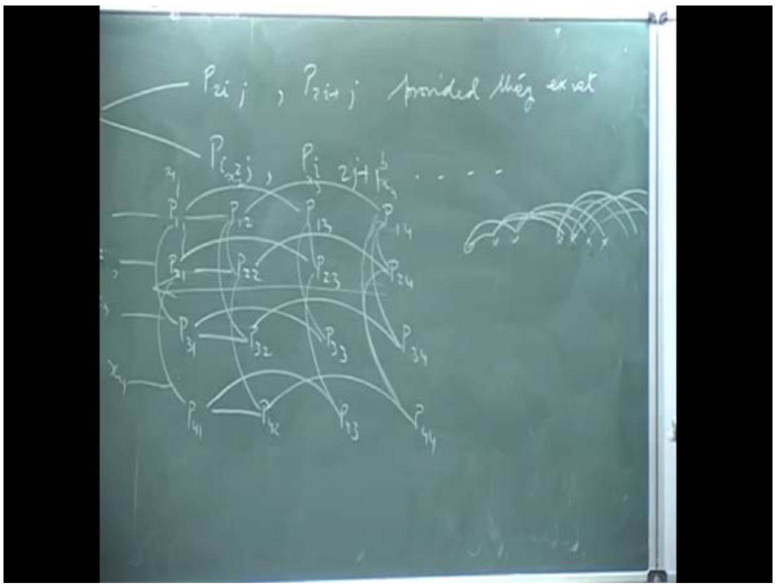


Step3 : $\sum_{j=1}^{n} r_{ij} + 1$ by all processor
$\forall$ each row i

Step2 : for each processor P_{ij}
if $a_{ij} > b_{ij}$ Then $r_{ij} = 1$
else $r_{ij} = 0$

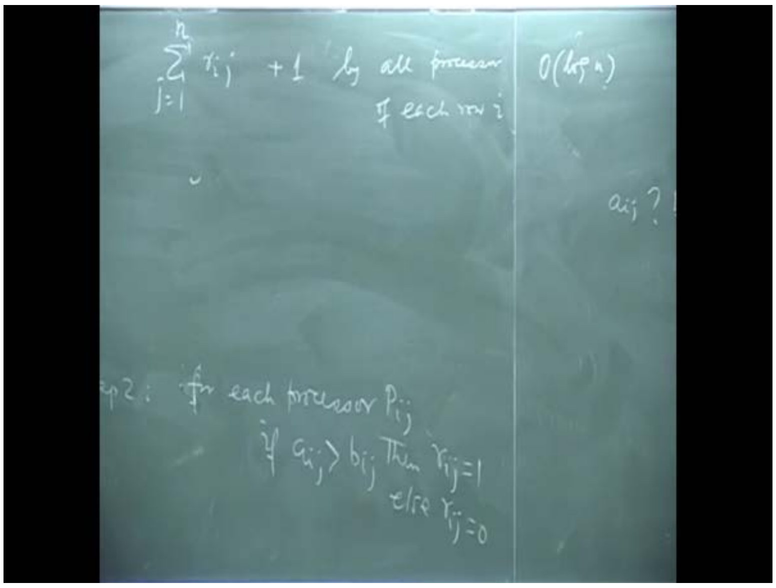
So, if it is a case then, step 3: Obtain summation over r i j over j equals to 1 to n plus 1. This 1 is as I did in the case of say find his own rank. So, obtain this by all processors of each row.

(Refer Slide Time: 1:03:51)


$P_{2i\,j}$, $P_{2i+1\,j}$ provided they exist
P_{11} P_{12} P_{13} P_{14}
P_{21} P_{22} P_{23} P_{24}
P_{31} P_{32} P_{33} P_{34}
P_{41} P_{42} P_{43} P_{44}

Now, this can be done by collapsing that is the reverse. See you have, it is the 2 connection then, for this you have 2 connections and for this you have 2 connections. Then, you have this as 2 connections, this as another 2 connections. So, what I am telling that this contains 0 and 1 by 3 connection model, you send this 2 elements model here and add it. So, you get the sum of 3 elements here, sum of 3 elements here, sum of 3 elements here. Now, you send this 2 elements to here and you get the sum of 7 element, this 2 elements you send it here, sum of 7 element; this 2 elements you get it and this sum of 15 elements. And now, it can be sufficient to cover these things to get these elements. So, you need order log n time to compute this one by collapsing method.

(Refer Slide Time: 1:05:01)



$\sum_{j=1}^{n} r_{ij} + 1$ by all processor $O(\log n)$

$\forall$ each row i

a_{ij} ?

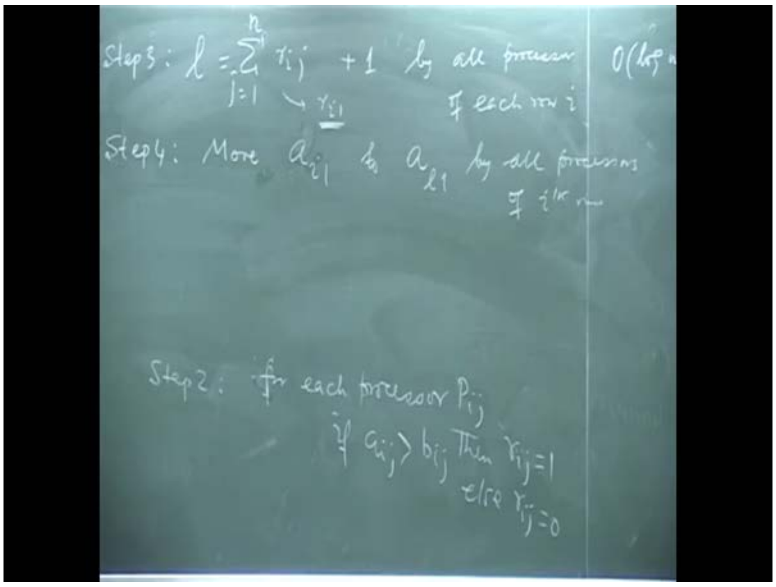
ep2: for each processor P_{ij}

if $a_{ij} > b_{ij}$ Then $r_{ij} = 1$

else $r_{ij} = 0$

So, once let it be say let it be j, not j. Some say let us assume that this is equals l. What it mean? That means rank of x I is there. Now, if I have to store it then, step 4.

(Refer Slide Time: 1:05:38)



Step3: $l = \sum_{j=1}^{n} r_{ij} + 1$ by all processor $O(\log$

$\rightarrow r_{i1}$ $\forall$ each row i

Step4: Move a_{i1} to a_{l1} by all processors

$\forall$ i^{th} row

Step2: for each processor P_{ij}

if $a_{ij} > b_{ij}$ Then $r_{ij} = 1$

else $r_{ij} = 0$

Step 4 that moved a i 1 to a l 1 by all processor of i of i-th row. . So, you put it in r i 1, you stored it in r i 1, is in r i 1. That is this rank element here, r i 1. Now, move a i 1 to a r i 1 area. So, from here you. So, what you have to do?

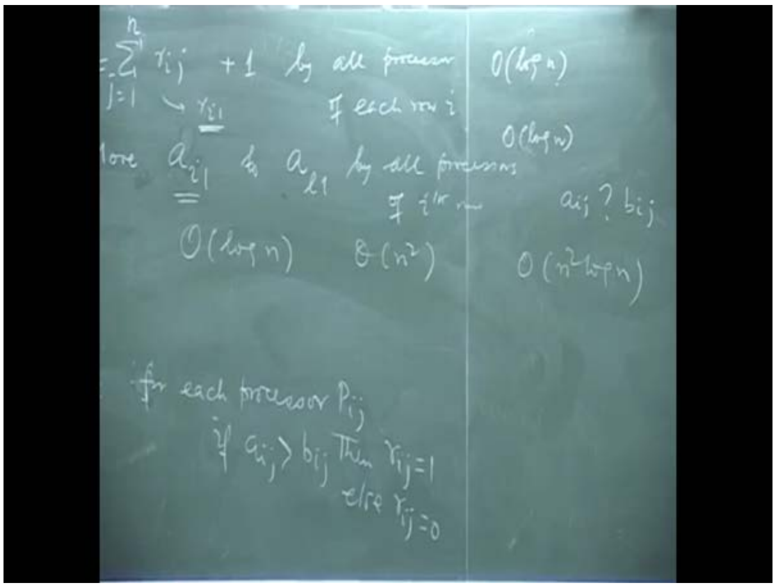
(Refer Slide Time: 1:06:49)

You have a i 1 and you have a l. So, this data may go to some a m 1, this may move to a m 1, this data has to be come as to be brought here. This one has to be brought here, this 1 move out obviously because one position only 1 element will be there.

Now, how to bring this element to this position that is done. So, in order to do that, you first suit 3 connections, this direction I can move it to a i l using 3 connections I can move it to a I l. Agreed? If I can move it, I can move it to a l i using column connection. a i r 1 move to a i r moves to a l i using column 3 connection. From there using collapsing method along the row, I can move the data to l i l, a l i. Yes or no? First, I moved the data from here to here, from this to this, this to this.

Yes, only 1 value you have a i r. See this contains l. So, this moves to a i r. .Yes, let them go to this. So, this will go to a l l then, a m l then a m 1. So, this takes order n time. So, this takes order log n time.

(Refer Slide Time: 1:09:00)



$= \sum_{j=1}^{n} r_{ij} + 1$ by all processor $O(\log n)$

$\rightarrow \underline{r_{i1}}$ $\forall$ each row i

$O(\log n)$

Move $\underline{\underline{a_{i1}}}$ to a_{l1} by all processors

$\forall$ i^{th} row $a_{ij} ? b_{ij}$

$O(\log n)$ $\theta(n^2)$ $O(n^2 \log n)$

for each processor P_{ij}

if $a_{ij} > b_{ij}$ then $r_{ij} = 1$

else $r_{ij} = 0$

So, you observed that the total time complexity is order log n using order n square processor, cost is order n square log n to sort n element. Order log n times very first, order n square processor and but, the cost is order n square log n.Now, back part is that the model is very complex. Every processor we are having in row direction 3 and column direction another 3 connections at most.

Now, the problem remain or you have to try at home is this that there is no way why a i 1 likes to send the data to a l 1 and a l 1 wants to send the data to a m 1 that is no conflict. If that is the conflict then there will be a . So, what it how to prove it? Why we are sending a i 1 to a i l then, there will only 1 element who will sending the data to a i 1 to a i l. There is no way a l 1 will conflict with the a i i because that will go to a l 1, a l l, a l I using other 3 directions. Agreed? Now, when you send the data from here to here and so, you are using different because, this a l l and this is a i l. You are sending the different column connections to send the data from here to here and from here to here, similarly, in that case. So, there is no possibility of having any conflict between the 2 elements while we send the data from one corner to another corner. So, any doubt in this? Or you go through first and then. This is available in on sorting and searching, on all sorting and searching.