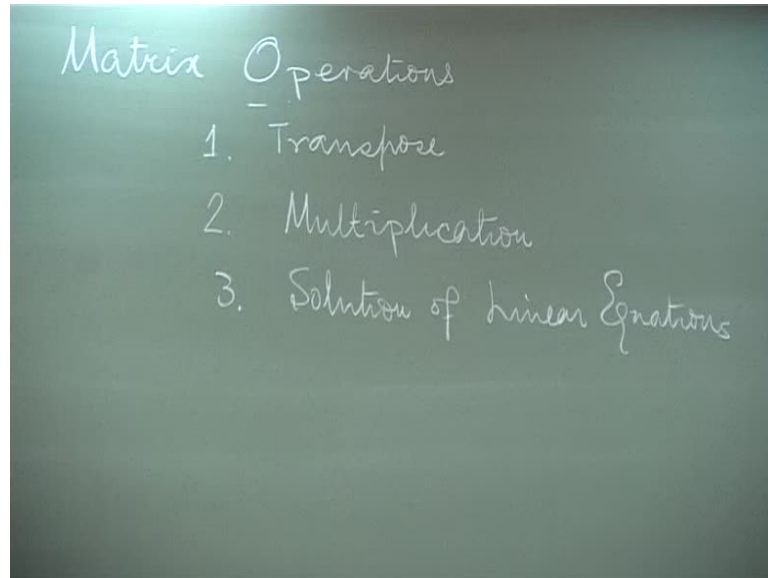


Parallel Algorithms
Prof. Phalguni Gupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 17

(Refer Slide Time: 00:16)



So, I would like to discuss the different problems we faced on matrix operations different parallel machines. You know some machines you will find that are suitable for matrix operations onwards that were not suitable at all. So, if you breakdown the environment you have to decide the type of parallel machines suitable for your matrix operation or something like that, you have to decide accordingly. Now but if I ask that matrix of what are the different type of matrix operations, generally we are encountering; one is

Student: Multiplication.

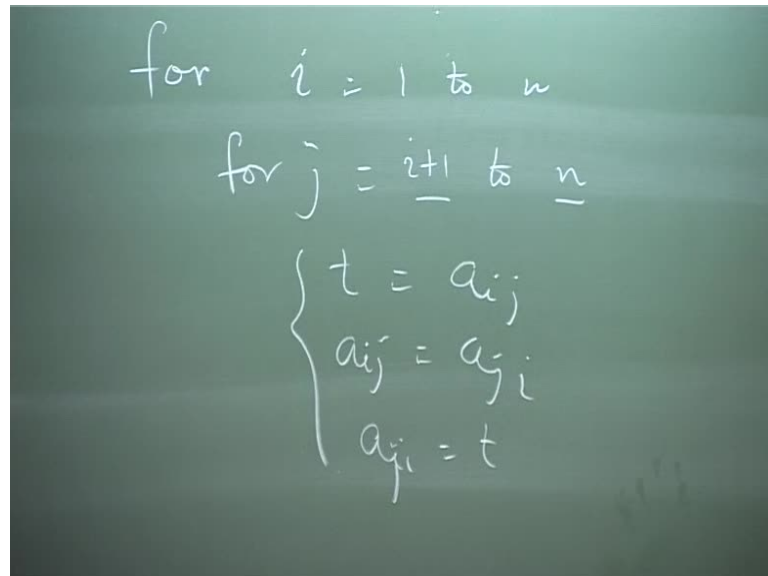
Multiplication and?

Student: Inverse.

Inverse, it is a very difficult problem. So, first one is that simplest one is the transpose of a matrix. So, one is the transpose of the matrix, second one is the multiplication, and third one is not exactly inverse, actually inverse, yes, solution of linear equations, right. This is actually if you think the solution of linear equations actually you are looking for

inverse generally, inverse is used only for the simultaneous linear equation and in reality we do not use the inverse in different ways; we will discuss it later on.

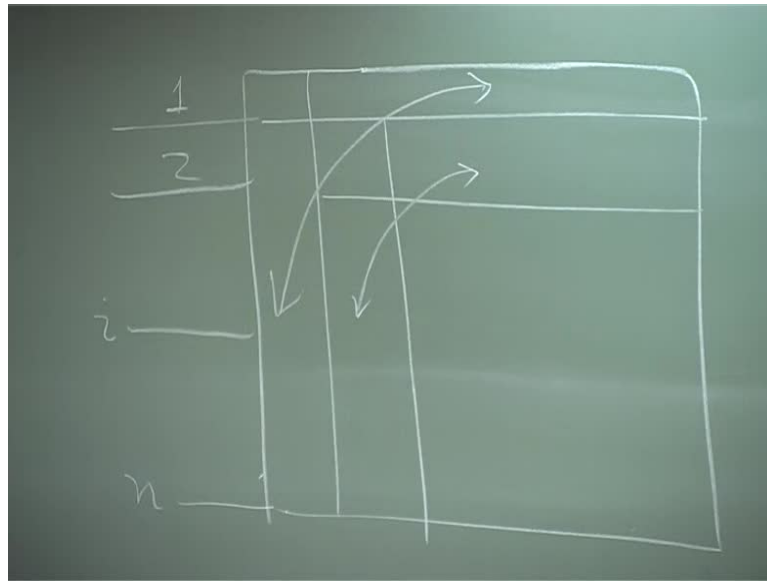
(Refer Slide Time: 02:54)



```
for i = 1 to n
  for j = i+1 to n
    { t = aij
      aij = aji
      aji = t
```

First the problem transpose, you know transpose is not even though it is a very simple of operations in the reality and prove it interties with a j i but it is a very difficult problem to solve, say, for i equals to 1 to n, for j equals to and t equals to a i j, a i j is equal to a j i and then a i j is equal to t, right. And now this two parameters are important because in general you will make several mistakes in the beginning but you should take I plus 1 to n solve 1 to i minus 1, right, yes or no? So, this is your transpose of this. Now if I tell that you write a parallel algorithm for unshared memory model, how many processors on shared memory model?

(Refer Slide Time: 03:56)

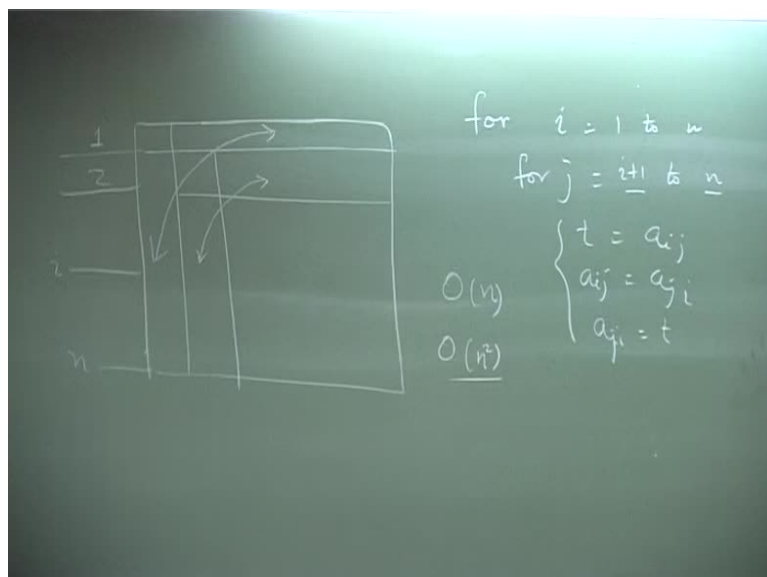


Suppose you have n process and this is the matrix of size n cross n and then first processor is for this one first row; no, tell me, by now you should have some ideas. You have told n processors.

Student: Are they j element?

First processor is first row, second processor is in second row and i th processor in the i th row and n th processor in n , right. So, first processor tasking is to interchange this with this, right.

(Refer Slide Time: 05:11)

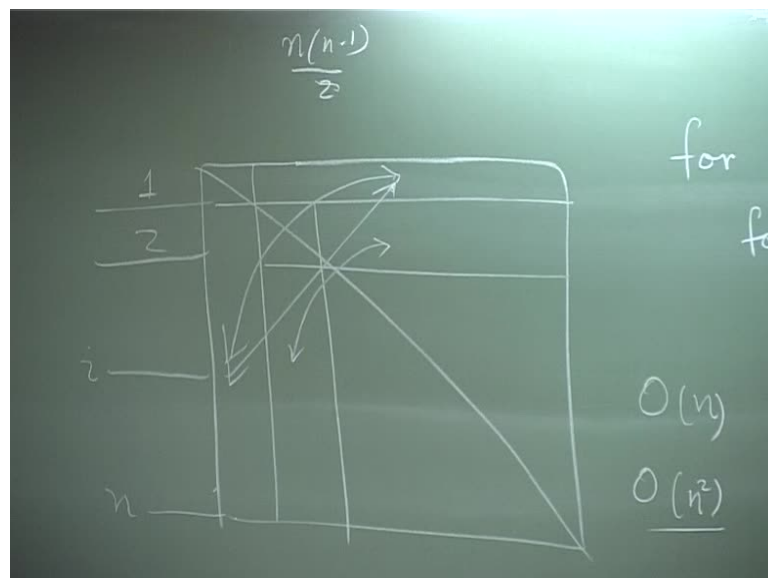


Second processor task is to interchange is it the thing are you looking for? I do not know, is it okay? Yes, no, tell?

Student: Third processor will remain idle for.

Only one job it will be doing, but this is one way in that process your complexity becomes order n because first processor will be doing interchange of n minus 1 according to some condition if you put so otherwise it is order of n time, right. So, worst case you have to take a time needed by the first processor and you will be putting order n time. Now suppose so cost becomes order n square, the cost becomes order n square because order n processor and order n time all become order n square and it is matching with this algorithm, yes, there is no problem in that. Now suppose I use n square processor, right, do you need n square processor?

(Refer Slide Time: 06:16)



We need n square processor because if I have n square processors, these processors are idle, because they will not be doing any job. So, basically you need n into n minus 1 divided by 2 this these many processors basically are of that order minus n may be there, of that many processor for upper triangular matrix, and each processors is changing with this one which takes only one unit of time, agreed. So, what is n square processor; order one time, so cost is order n square which is also cost of $2n$. Now suppose I have mesh connected computer two dimensional mesh now how are you going to do this?

(Refer Slide Time: 07:00)

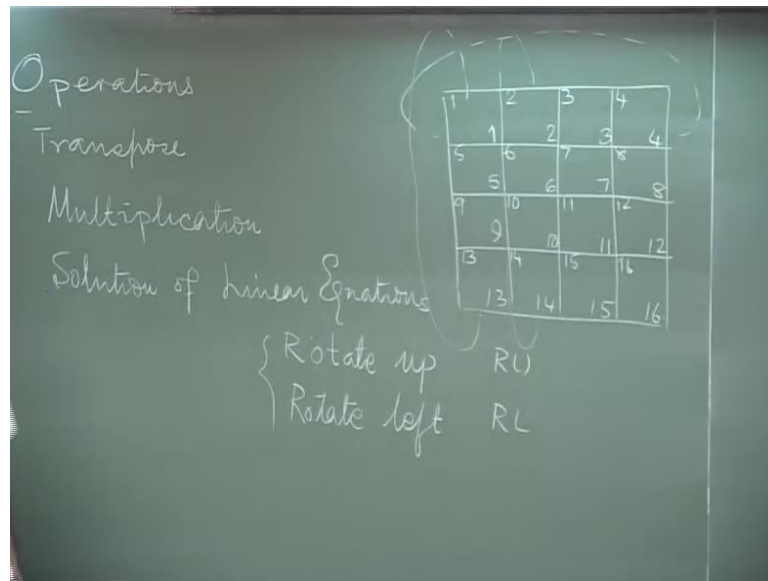
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

matrix

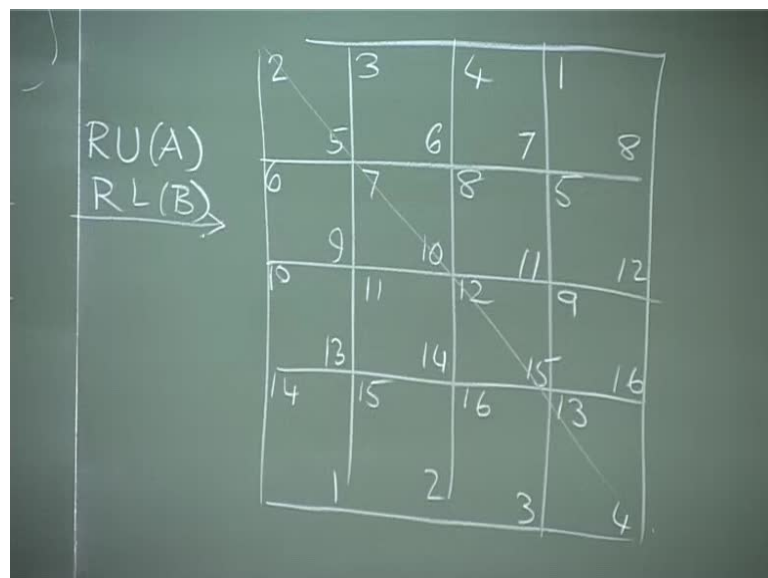
You have a two dimensional mesh. So, suppose you have these data 1, 2, 3, 4, 5, up to 16 and you want to make the transpose, so this 1, 5, 9, 13 should go there and 1, 2, 3, 4, should come here, agreed. Now that means that you have to interchange this with this, this with this, this with this, and same time this, with this and so on, right. So, it is not as simple as you thought in the case of shared memory case, right. So, in order to do that let us first try to do consider this example explains it, and then we will write the algorithm, right. What do we do? We copy this things into another array B. Suppose this is the bottom part is A, and we are copying it to another array B, and we have one thing in the diagonal elements are already transpose; you do not have to transpose it, right.

(Refer Slide Time: 09:06)



Now we define the two operations; one is rotate up, another one is rotate left. Now in order to define these two operations rotate up say we use RU and RL. In order to define these two operations what we did we assume that mesh connected computer wrap-around; wrap-around means this is connected and this is connected, right. You did not bother wrap-around, right. So, once I tell rotate up, so this up means this element will come here.

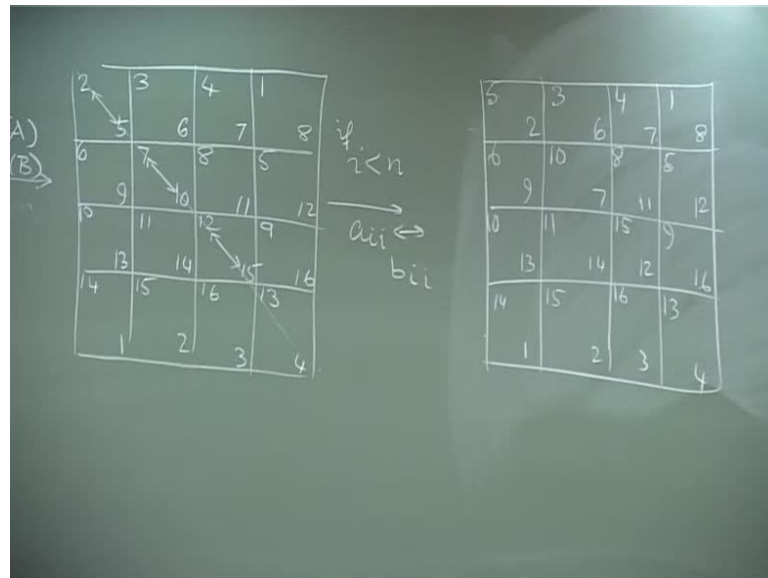
(Refer Slide Time: 09:58)



So, if I perform rotate up A and rotate left B, what i get? 5 9 13 1, 2 3 4 1, 6 7 8 5, 10 11

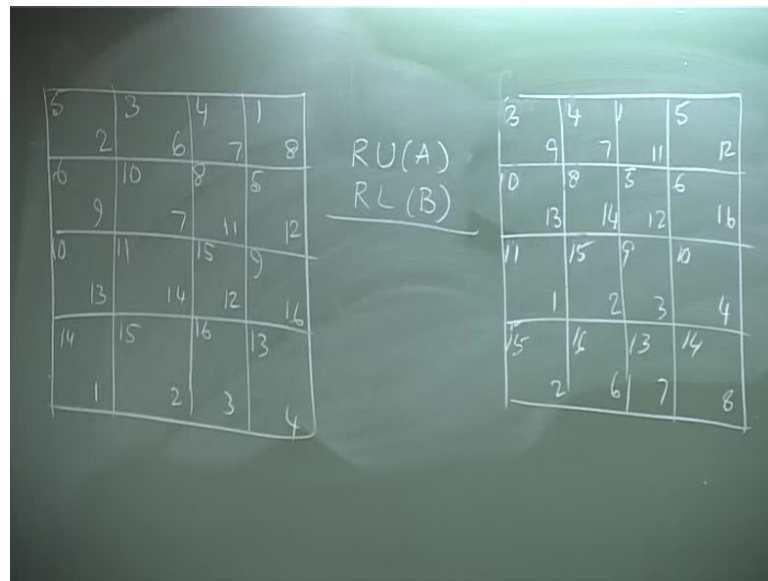
12 9, 14 15 16 13, 6 7 8 10 14 2, 7 has gone up, 11 15 3 4 16 12, right. Now you look at the diagonal elements. If you see the diagonal elements, the diagonal elements are transposable. Actually original matrix it was 2 and 5 that has to be transpose, agreed. Then 7 with 10, this two has to be transposed and 12 and 15 to be transported, right.

(Refer Slide Time: 12:16)



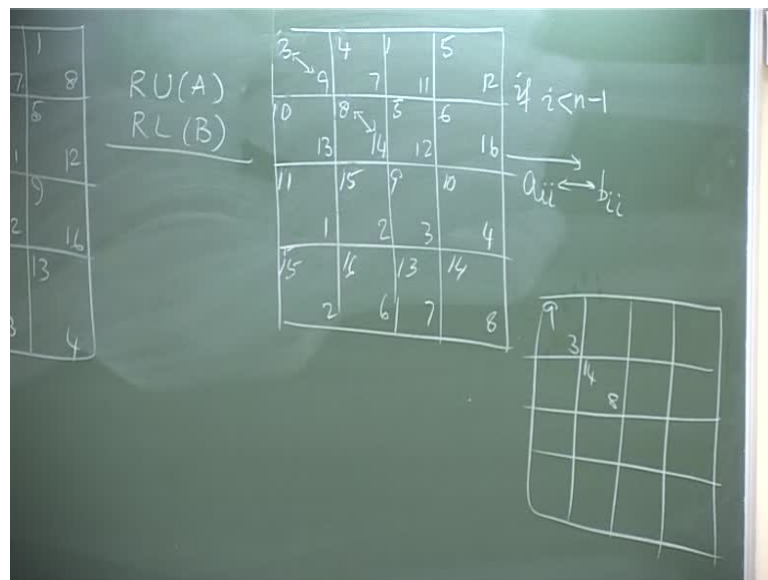
So, if you see the diagonal element these elements are transportable; however, you observe it that this is not transposable. So, after doing that operation if you find that i is less than number of elements then it should be here, i less than n is then $a_i i$ is interchanged with $b_i i$. So, if it is the case then you get here 5, you get here 2, you get 3, you get 6 4 7 1, 8 5 12 8 11 10 7 6 9, 10 13 14 1, 15 2 11 14 15 12 9 16, 13 4 16 3. So, after the first iterations the details will be like this, agreed.

(Refer Slide Time: 13:49)



Now you perform again rotate up A rotate left B, what is there? 6 10 14 5, 10 11 15 3, 8 15; rotate up is A. A is 9 13 1 2, 7 14 2 6, 11 12 3 7, 12 16 4 8 and this side is 3 4 1 5, 10 8 5 6, 11 15 9 10, 15 16 13 14.

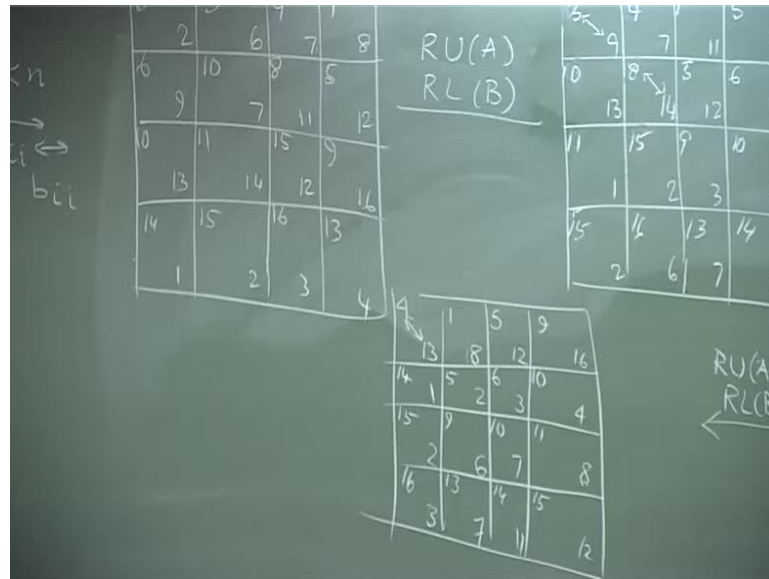
(Refer Slide Time: 15:55)



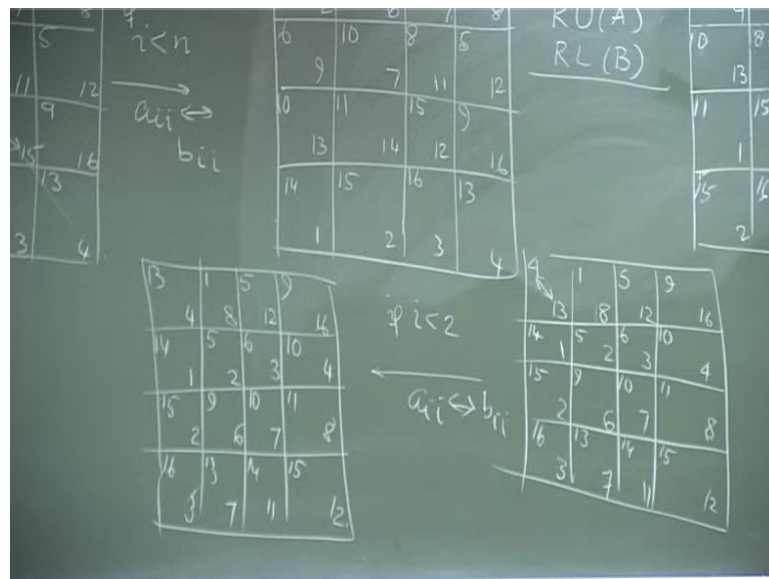
Now after rotate up and rotate left, you observe that this two are transposable elements. You see the original one that 3 and 9 to be transposed and 8 with 14, right. So, this two if you transpose, so here your condition will be if i is less than n minus 1 then a_{ii} is transposed with b_{ii}, agreed. So, what you get here? 9 3 14 8 and rest should be same, is

it okay.

(Refer Slide Time: 17:09)



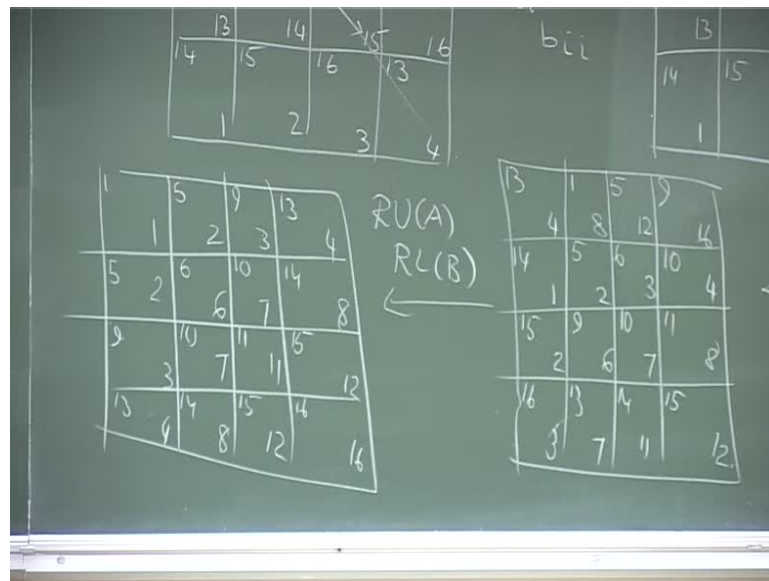
(Refer Slide Time: 19:33)



So, again you do the rotate up A rotate left B, so 3 will be here, 2 will be here, 1 will be here, 13 will come here, 8 will come here, 8 will come here and 2 here, 6 here and 7 is here. This side means 12 3 7 11, 16 4 8, this side it is 4 1 5 9, 14 5 6 10, 15 9 10 11, 16 13 14 15. Now you observe that this to be interchanged, right, 4 with 13 which is lying here. So, next one is there; if i is less than two then $a_i i$ is interchanged with $b_i i$. So, you get here 13 4 1 8 5 12 9 16, 14 1 5 2 6 3 10 4, 15 2 9 6 10 7 11 8, 15 12 14 11 13 7

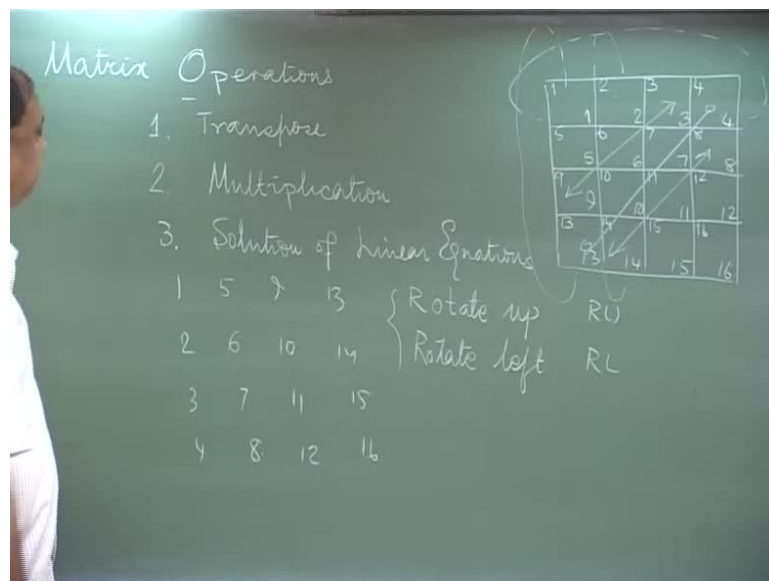
163.

(Refer Slide Time: 20:58)

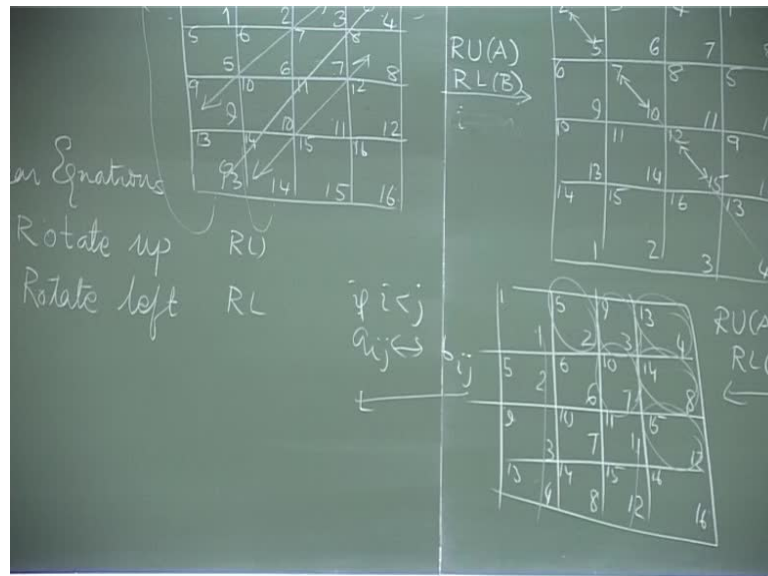


Now you observe that we have almost we have done the transposition work; only thing is that there is a still remaining rotate up and rotate left one and there little manipulation, let us see what happens? You have to rotate up A rotate left B 1 2 3 4, 2 6 7 8, 3 7 11 12, 4 8 12 16; 1 5 9 13, 5 6 10 14, 9 10 11 15, 13 14 15 16, right.

(Refer Slide Time: 22:39)



(Refer Slide Time: 23:13)



Now original matrix it should have been 1 2 3 4, 5 6 7 8, 9 10 11 12, 13 14 15 16. Now this is your A matrix, this is your B; now look at A, so 1 2 3 4 you are getting and then 6 7 8 also you get, you did not get 5, then you got 9 10 11 12, here you see 11 12 you got it, you did not get this one and this one, then 13 14 15 16 you did not get this one, this one, this one, but you got this one, right, yes, and this can be done easily. If i is less than k i is less than j , a_{ij} is interchanging with a_{ji} ; it is just you interchange 2 with 5, so 5 will come here, 9 will come here, 13 will come here, 14 will come here, 15 will come here and so on.

Student: Sir, 4 and 13 are also interchangeable that is in the preparation.

4 and?

Student: In the first preparation we did only three changes in the diagonal. The fourth diagonal is also changes, fourth element also we can change them.

Four and?

Student: 13, the two corners.

Yeah. 4 and 13.

Student: In the transposition also that both whatever is there in the diagonal order changes and then I think we would have got this transpose directly.

So you need not to do this?

Student: Yes sir, because if we do that 9 and 3 change there and 14 and 8 also, because actually what we did is every time we did one less.

Yes.

Student: That is why we got them.

No, but here how many? Here 1, 6, right, 6, so you are avoiding six interchanges?

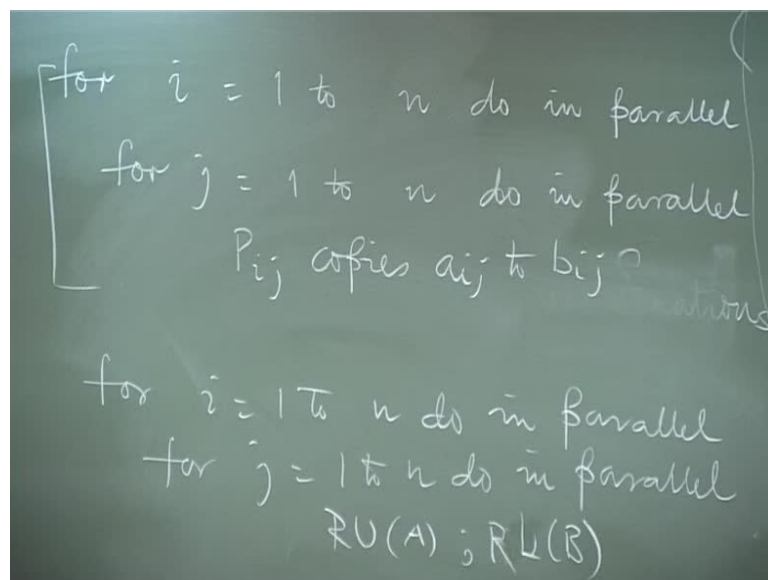
Student: Yes sir. First it is 1, second it is 2, and third it is we are avoiding fully.

Yes, yes, let us see. This is 13 and 14 you want to interchange here?

Student: 13 and 4.

So, in that case 4 will be coming here, that will be coming here. So, here minus B also you will just remove? Yes, so far yes, that is that case that can be done; please check that one. If it is that case then the problem is much simple, then you do not have to do this one.

(Refer Slide Time: 25:55)



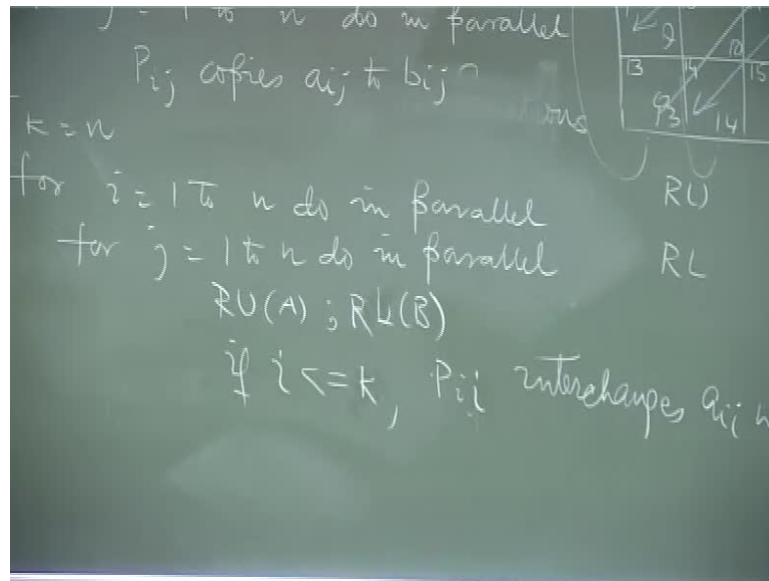
```
for i = 1 to n do in parallel
  for j = 1 to n do in parallel
    Pij copies aij to bij
  end
end

for i = 1 to n do in parallel
  for j = 1 to n do in parallel
    RU(A); RL(B)
  end
end
```

So for i equals to 1 to n, for j is equals to 1 to n because you are making busy all the processor, right, for rotate up and rotate left. So, i can write do in parallel. So, each

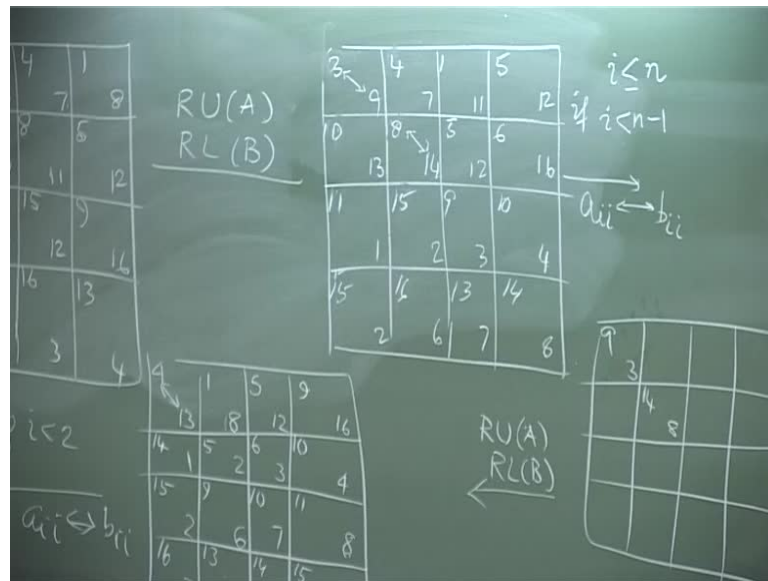
process does what? Rotate up A but before that copy to B is also required, p_{ij} copies a_{ij} to b_{ij} , right. So, this is first job. Second job is for i equals to 1 to n do in parallel, for j is equals to 1 to n do in parallel. You are doing rotate up A rotate up B. See rotate up A means p_{ij} sends that data $p_{i-1, j}$ inter parallel and similar rotate left else p_{ij} sends p_{ij} to p_{ij-1} .

(Refer Slide Time: 27:57)

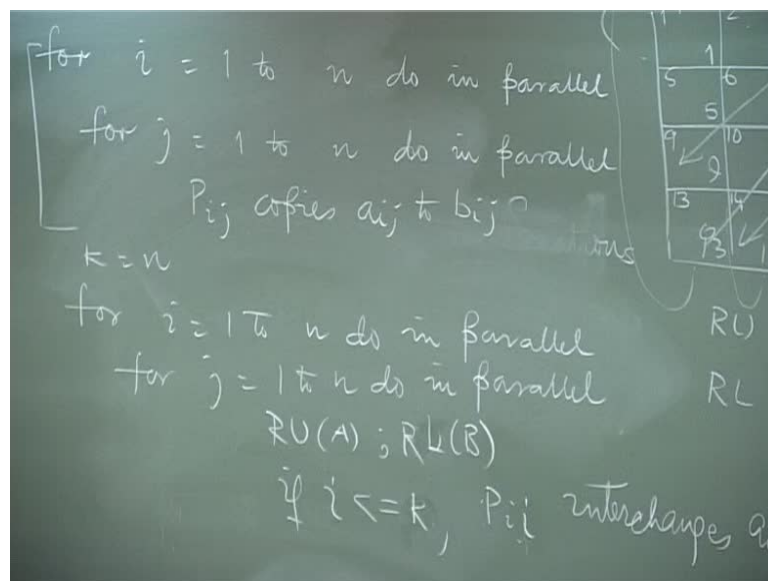


Now next we will be writing if i is less than or equals to k and here somewhere I write k equals to n then interchange p_{ii} interchanges a_{ii} with b_{ii} and then k has to be reduced by 1; oh you do not need to reduce k in that case. If vector is whatever is suggesting then you do not need to reduce the $n-1$ $n-2$. It should be what you are feeling is that it should be always this, right.

(Refer Slide Time: 29:04)

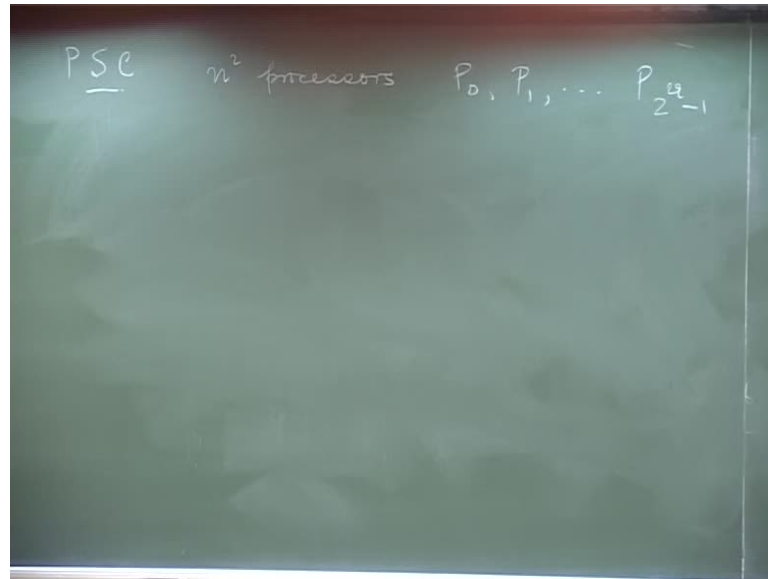


(Refer Slide Time: 29:22)



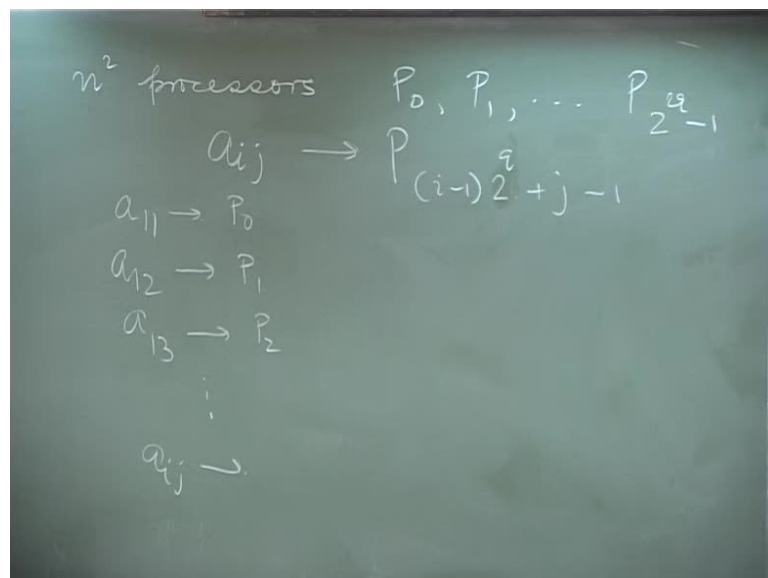
So, if it is that case you do not need have to reduce k. Please check with that because long back i did. So, p i j interchanges a i i with p i i and so I get this the thing. Then the complexity of this algorithm is constant amount of time using a n square process, so cost is order n square which is optimal, because any transpose of a matrix you need n square time. Now what happens to other models?

(Refer Slide Time: 29:49)



Suppose you have a perfect shuffle computer and you have n square processors p_0, p_1, p_2 to the power say 2^q minus 1, right. The processors we assure in the perfect shuffle that is the power of two number of processors, and you remember that it has the three connection, one is the exchange and another one is shuffle times and n unshuffle times and you have n square elements. These n square elements each process contains one element. Now can you give me little idea how can I, you remember one thing that any perfect shuffle algorithm is based on the two properties, right.

(Refer Slide Time: 30:53)

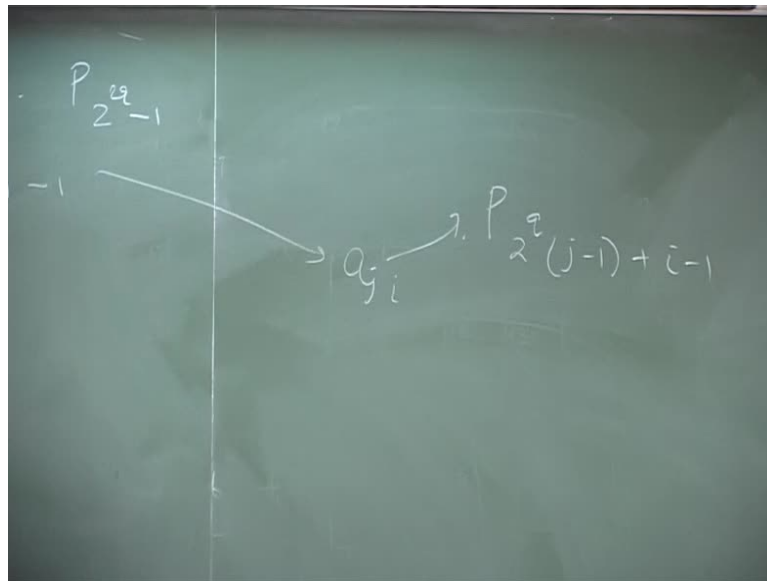


Those properties now can you suggest how a_{ij} is stored in p , then? a_{11} is stored in p_0 , a_{12} is stored in p_1 , a_{13} is stored in p_2 , a_{ij} is stored in?

Student: i minus 1 into j plus 1.

i minus 1, right, because this is n square processor, so this is n square 2 to the power $2q$ is n square. So, 2 to the power q is n .

(Refer Slide Time: 32:37)

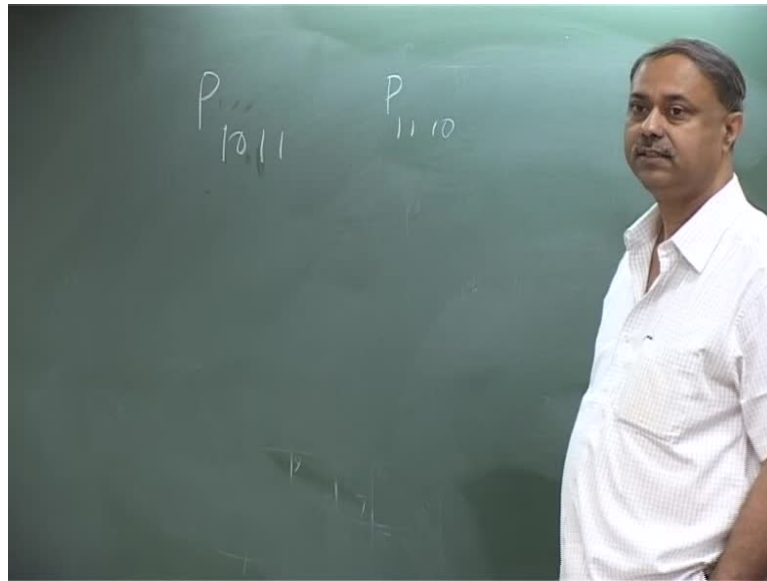


Yes, now a_{ij} is totally in this; a_{ij} has to be moved to the location of this, agreed; that means that processor index will be this. So, the processor $p_{i-1} 2^{2q} + j-1$ contains a_{ij} ; he has to send the detail to the processor $2^{2q} + j-1 + i-1$, yes no, yes no?

Student: Yes.

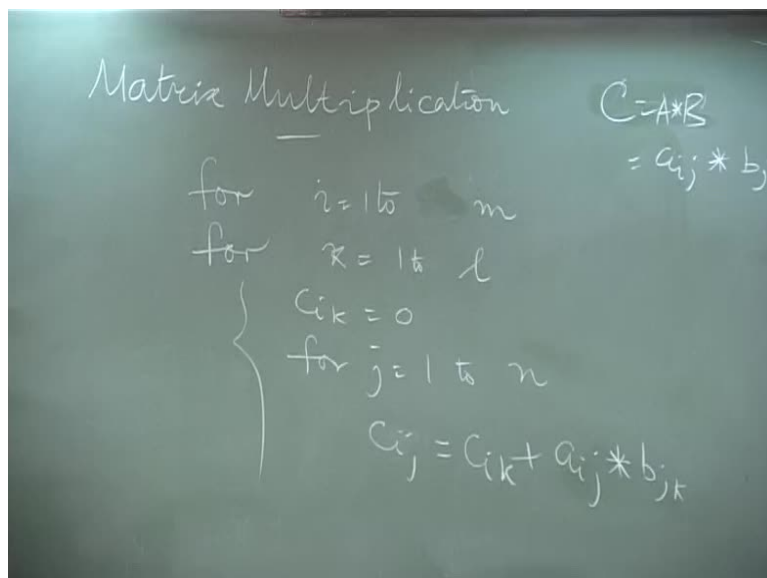
Correct, and this can be done by queue shuffles or queue unshuffles; either left shift or right shift q times if you do it will move to that, yes or no, rotate q times so this will come here $i-1$ will go there, no.

(Refer Slide Time: 33:59)



Suppose you have this index, so rotate q times you will be getting, right. So, if you rotate q times so this tab will come here and this tab will go there. So, you need the queue shuffles to move the element a_{ij} to its respective area. So, you did it basically $\log n$ time to move the data using n square processor, and that cost become n square $\log n$ time. Now this is all about transpose of a matrix. I do not think that these models will be useful or suitable for matrix transposes, right. So, I am not discussing those things

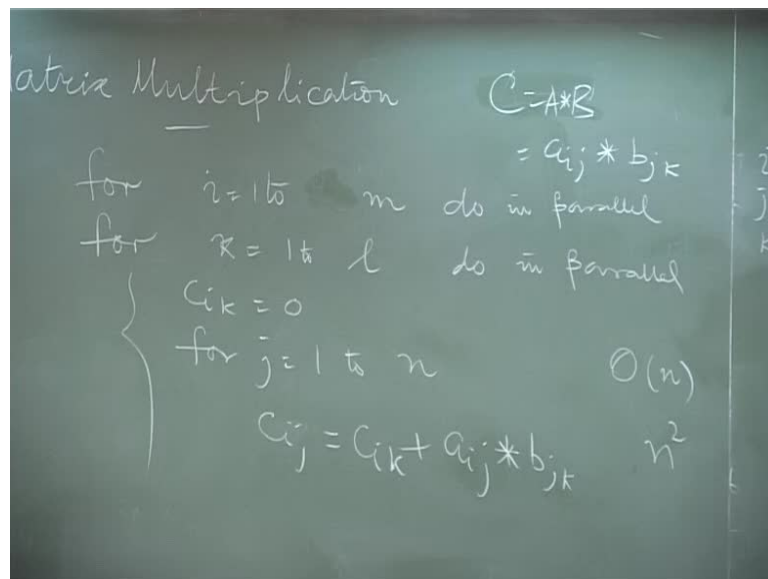
(Refer Slide Time: 35:07)



Matrix multiplications, so let us discuss about first sequential algorithm. It is very simple

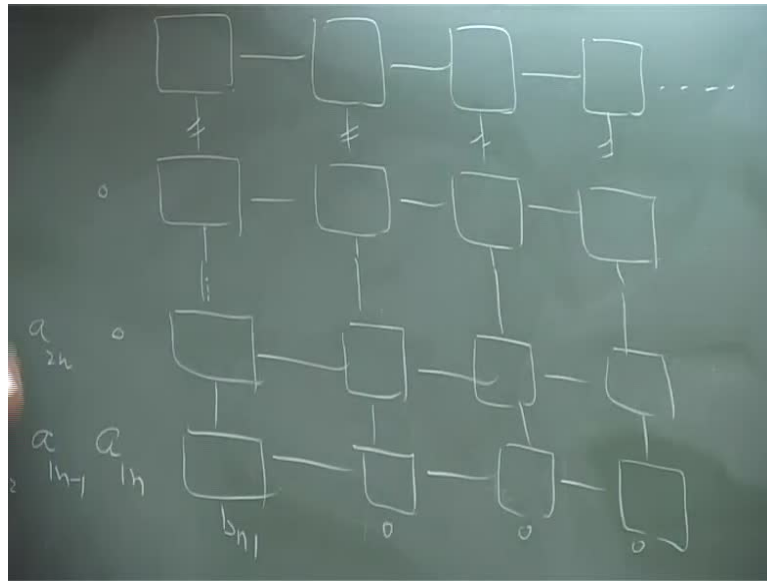
for i equals to 1 to n , for i equals to 1 to say you have C equals to generally we write A star B or that is a_{ij} star b_{jk} and i is 1 to m , j is equals to 1 to n and k is equals to, say, 1 to l , right, for i equals to 1 to m , for k equals to 1 to l , c_{ik} equals to 0, for j equals to 1 to n , $c_{ik} = c_{ik} + a_{ij} * b_{jk}$. This is the simple sequential algorithm which is the classic; I am not talking about the Strassen's algorithm, this is supposed to be sequential algorithms, and if I have the shared memory model with n square processor I can easily write this algorithm on shared memory, right.

(Refer Slide Time: 37:11)



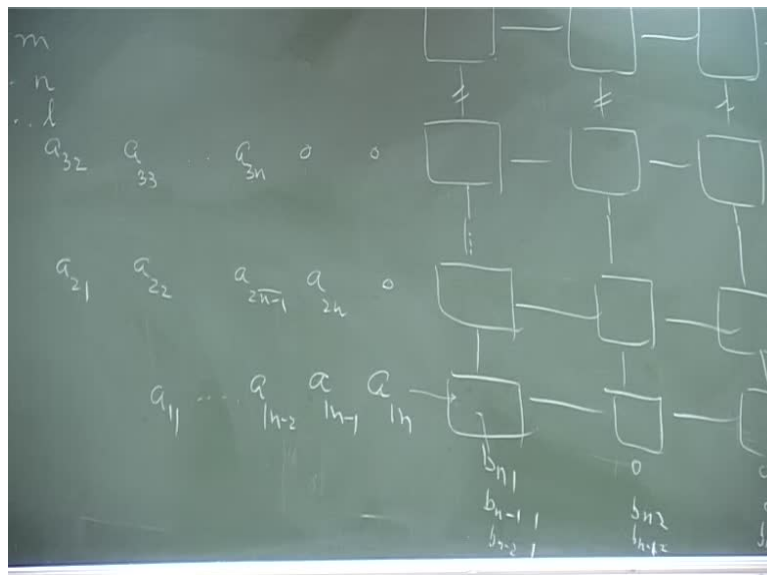
Just simple I will write do in parallel do in parallel and just put in your loop. So, you have order length and complexity using n square processor for this matrix multiplication problem. So, class becomes order n cube and the classical algorithm also at order n cube, that a minus n is also order n . Now today we will discuss only one algorithm and then more of one bridge connected to. See till now we have not considered that input output steps in parallel algorithm. We assume that data is readily available in the processors. As I told you that input output would play a major role in parallel algorithms because I would sometimes define that much more than the executed brand, right. But here in this the algorithm whatever I have discussed this is very clear that input output and other things will be coming into the picture.

(Refer Slide Time: 39:04)



Say, you have the mesh connected, it is not wrapped, it is not wrap around. See one thing you observed that two elements of two matrices are multiplied if first ones column index is matching with the second ones row index. So, that is the thing you have to keep it in mind.

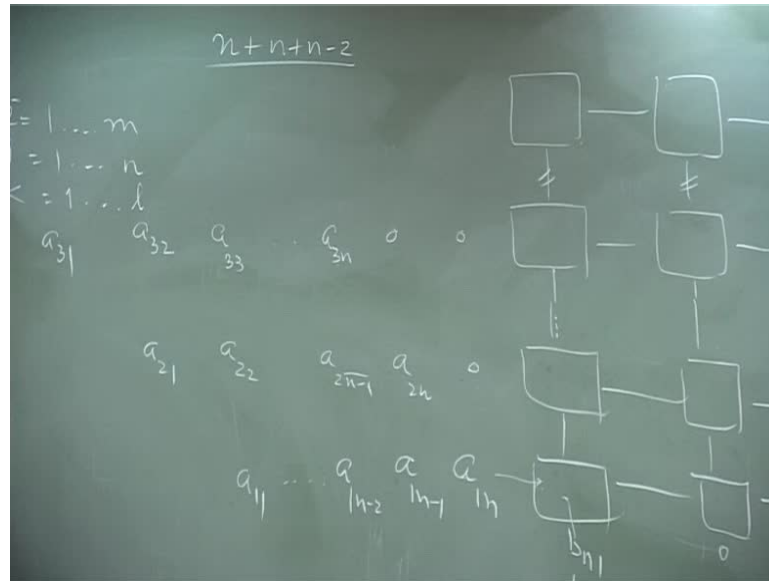
(Refer Slide Time: 40:12)



Okay, let me connect this one; otherwise, you may not get connect this one. See the data is arranged in this form, right, a_{11} a_{12} a_{1n} is nearer to this and then one delay is here, yet to delay and so on; similarly the case with matrix B. Now you move this data up to

one cycle and this data, and you observe that they are multipliable. So, once you are moving this one this data is moved to this place, this data has been moved to this place. So, next cycle this data will be here and this data will be here which is multipliable and which is to be stored in T 11 and at that time these data will be moved here and these data will be multiplied and which is also multipliable and so on.

(Refer Slide Time: 43:22)

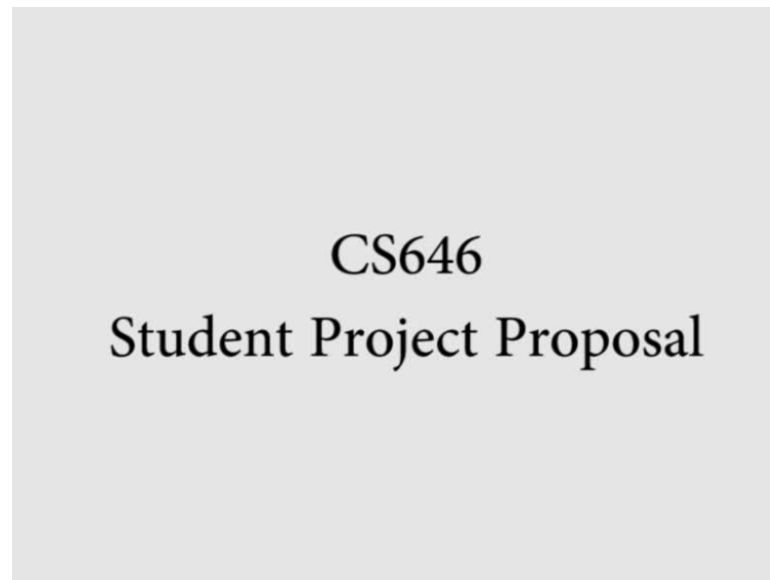


So, actually I have taken n , okay. So, after you will find n plus n plus n minus 2 that many operations all the data will be multipliable and stored in that because this data has to be moved here. This data has to be moved there, the distance will be m plus k plus n minus 2 , because this delay will be factor m plus m minus 1 and that is at k minus 1 and n movements so it is n plus k plus n minus 2 , okay. So, you have n square processor and a $i j$ receipts at any instant of time one data from this side, another data from this side multiplies and stores in its a_{ij} or adds with a_{ij} , then a_{ij} sends the data to $a_{i+1 j}$ and sends this data to $a_{i-1 j}$ and so on, it iterates, agreed. So, after that plus k plus n minus 2 iterations in the matrix multiplication would be ready. So, basically this order and time using n square processor which takes order n cube time cost to multiply the two matrices.

We will discuss in the next class what happens if the elements are stored in the matrix. Now it needs little bit of additional manipulation if you see that if you put a_{ij} and b_{ij} both are in the i th row and j th column of the processor, p_{ij} contains a_{ij} and b_{ij} ; they

are not multipliable, right. p_{ij} and context a_{ij} and b_{ij} they are not multipliable you have to get p_i such situation is like that $p_i a_j$ contents a_{ij} and b_{ji} then only multiplications is permissible or b_{jk} . So, you need to do little operations on that, so we will discuss that one because that will take little time to explain.

(Refer Slide Time: 45:56)



Parallel dynamic programming like with respect to a sub problem of sequential alignment.

Tell me your problem?

Yeah sequential alignment, sequence alignment like the basic problem is you have got two sequences of m and n , say, d_n sequences, and you need to align them; you need to reduce the like I should introduce the problems and show.

Yes, because others may not know because there is another boy who is another team Marking on phonetics.

Student: On same thing sequential algorithm.

Now you explain, first you explain then

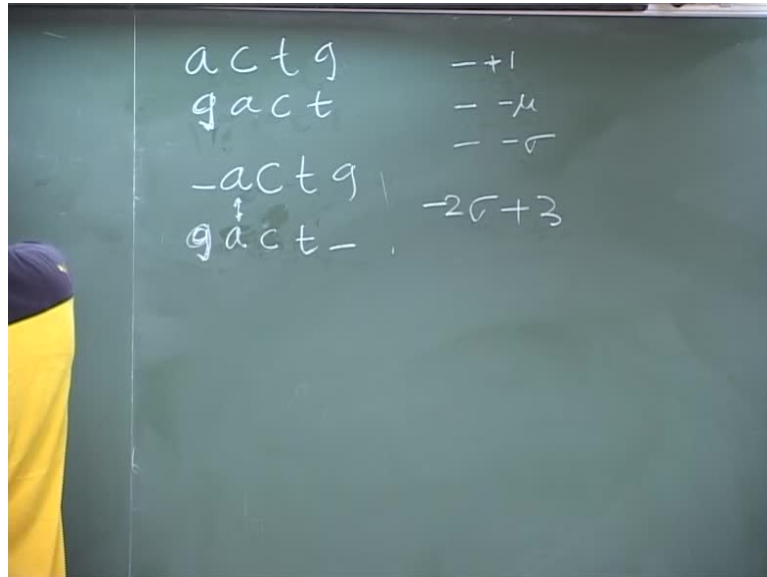
So, like two teams can I mix?

No.

Well, I define a scoring pattern like about the sequential alignment problem.

First you tell the problem then others will find it.

(Refer Slide Time: 47:02)



Ok yeah, so I have got the scoring pattern like, say, these are the two sequences a c t g and, say, g a c t. So, I said that in the case of match the score will be plus 1 and in the case of mismatch the penalty will be mu and for a gap I can introduce gaps and for a gap the penalty will be minus lambda or something.

No, it is not lambda it is sigma.

Yeah.

No what is gap for that. While you explain the problem you tell the problem what is the sequence? A c t is the sequence.

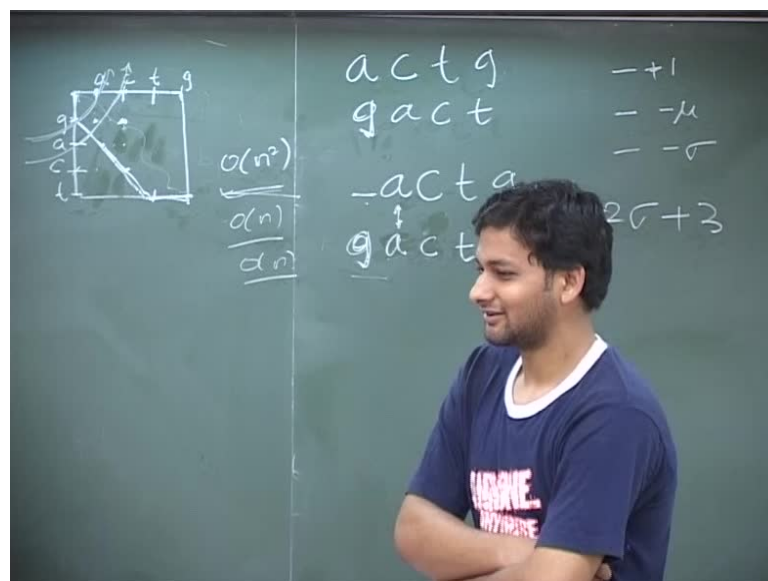
Yeah Yeah I am coming to that what a gap is and stuffs. Well like, so this is the first sequence. Now I can put the g either here or I can put a gap and start the second sequence from here and stuff like that or I can also put start the second sequence here meaning I will put a gap here you know, and I will match the second one second point of the second sequence with the first one of the first sequence and permit. So, if i put it like this, so there will be a gap here obviously. So, here I will be having a penalty of 2 sigma and there are three matches, so it will be plus 3, the score for this particular alignment is

this. So, what I need to do is I need to maximize the scoring.

No no, then it is minus 2 sigma plus 3.

Yeah sorry yeah minus 2 sigma plus 3, so I need to maximize the total score possible. So, this is the basic problem sequence alignment and the dynamic programming approach to this problem like, say, if I have got two sequences of n and n , so if we take order n square.

(Refer Slide Time: 48:59)



So, that will be nothing but like this is the standard way of doing that. So, I construct the matrix of n by n . So, this is the first sequence, and this is the second sequence and every alignment will get a path from here to here from this point to this point and for example this particular alignment like, say, it will be like I start from here and I have got a space here. So, that means I am moving on the second sequence but I am not moving on the first sequence. So, it will be something like this. So, I come down and then there is a match between a and a , so I am moving on both the sequences, So, it is like this and I am moving simultaneously on both and both, and finally I am moving on the first sequence but not on the second sequence. So, this path defines this particular alignment. So, I have got for every such alignment I have some path from here to here.

So, what I need to do is I just need to maximize the scoring on this path. In the dynamic programming approach what I can do is I can for every such position you know, say, I

have got a position here. So I can come into this position either from this place or from this place or from this place. So there are three ways of entering that position and if I know all the smaller problems you know all the best ways till, if I know the best way till here, till here and till here I can easily compute the best way till here. So, in meaning that I got the overlapping of problem, structure and also the optimal substructure, so the dynamic programming approach is applicable here.

And I can do it in order n^2 times you know like by sort of like anti diagonal sort of way like I can compute all these things together, and I can do all these things together, because this will depend only on the things which are lying on this side, similarly on this and stuffs like that. So, the normal sequential algorithm will take order n^2 time and what we need to plan is we need to like we will plan to look at all the different possible parallel algorithms that are there for detail problem sand suppose better problem better algorithm is possible. Well the most basic you know if the parallel algorithm can be like if you have got order n processors you know shared memory model, so you can reduce the time to order n , I mean you can reduce the time to order n in a straightforward fashion because all the diagonals you can do them in a parallel way. You can also use that row by row approach that is like I start from here and I compute the best task till these points I means like first compute this then this and this and then this is this.

No, but in this problem who is doing?

Student: Same problem

So now one of you, they have already reported that this type models.

We can take some other problem in the sequential alignment like there are some of the problems are there.

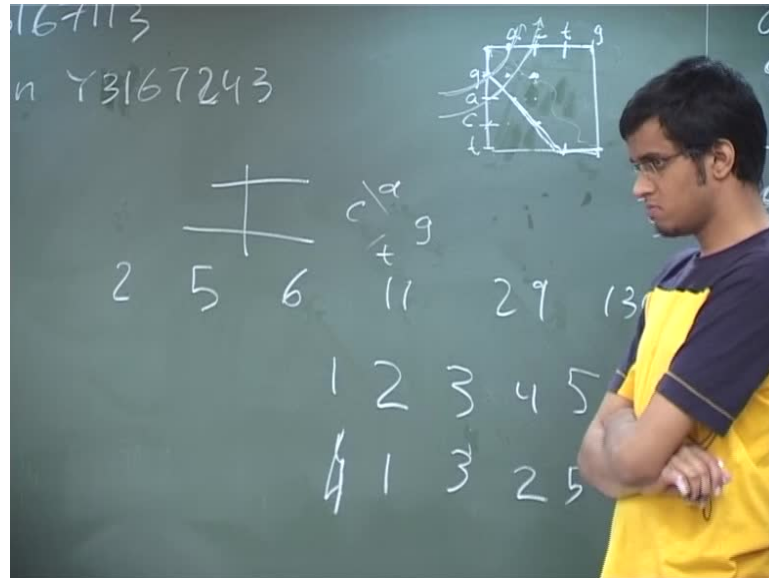
But is student visible also, right, open problem and you are solving.

Yeah it is not exactly open problem like in the traditional way. When you deal a open problem it is like no one can know as many solutions; it is not I am thinking they are so hard.

What is the problem?

One problem is you have given two scales u and v and that is given that i th character of one sequence and j th character of the second sequence, they all basically match and in that case you have to find the best length.

(Refer Slide Time: 53:21)



You are fixing like you have got two sequences, say, I fix this, these two have to be aligned. So, now get the best alignment for that two sequences and also you can also try the circular alignment. You have got two strings and like exactly it is like so I can cut them anywhere like sort of, so you have got two such circular strings, so what I want to do is I want to align them.

See I can tell you one problem, you can try that one. A sequence is known as k sorted; k sorted means an element i th element should be in the high position; that is the i th solution should bring in the high position. Instead of i th position just it may be at a distance of i like,

High position or less than high position.

Or greater than also, you try.

What do you mean by distance?

Distance when at i should be here instead of here it may be put in there suddenly, say, you write down a number a sequence of integer sorted sequence, positive like integer

sorted sequence.

Ok actually I am not talking about this type of three, I am talking about you know what is integers sort.

Ok, I do not know.

You do not know? Integer sort is that a permutation of n integers first n integers.

Ok ok.

Right, so one two three for a total of five, five elements are there. So, one two three four five, write down one two three four five and another permutations you write up for five elements.

For the same five elements?

Yeah, right. See 1 2 3 4 5 is your perfect sequence, the distance is zero. So, three has occupied its own position.

Ok.

Right, no no no, not neechae or ooper also

Ok fine fine.

I am talking about the bottom one, say, it is that you are telling that it is pre-sorted, why it is pre-sorted? That every element are at a distance of plus minus 3, right. So, I am telling that it is a pre-sorted. So, if k equals to zero it is a sorted sequence, k equals to one then k equals to one means it is almost sorted; it is almost sorted only one distance may be here or there, you understood? Two sorted means it may be two elements away or two elements this way, right.

In the actual position.

From the actual positions, so can you think about that position? Then I will sort it why shall I do your thing.

Yeah.

So, can we have order n algorithm it must be a function of k , k sorted function of k in such way when k equals to i will write zero, then it should be of that sorted $n \log n$ time, right. So, you think that one because this is somebody has done already or worst case you take whatever the other one you told that given two sequence, one is fixed.

(Refer Slide Time: 57:43)

	D1	doc	Di
t1	4	0	10
terms			

So, my work is basically around the bigger position of a matrix to singular value decomposition. So, I mean this is basically used like in any search engine. Basically there are two things terms and documents. So, this is like in a search engine there is a matrix which has terms on this side and documents on this side, okay. So, for in every document there is a term and it may be the repetition of terms in the documents. So, this matrix basically contains this documents has how many times this term like document D_1 and term t_1 . So, if t_1 comes four times in these documents, so this entry will be 4 and this term in D_i ; if this comes ten times this entry will be 10, may be zero, may be zero. So, for a search engine this matrix is very large; I mean it may be around thousand cross thousands may be more than this. So, basically to store these types of matrixes what is basically done? They have to store it in some reduced form structure, okay.

(Refer Slide Time: 58:58)

$$A = U \underline{\underline{\Sigma}} V^T$$

R

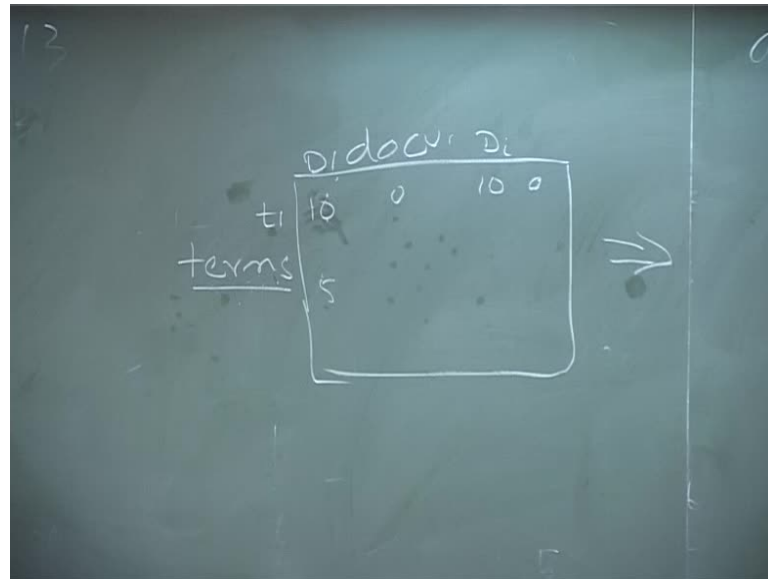
$$A_R = U_R \Sigma_R V_R^T$$

So, like this is a matrix, so this is basically converted into its singular value decomposition like U this and V transpose. So, U and V is the original matrix; orthogonal means U transpose into T. These are the original matrix and this sigma basically contains the singular values of this matrix. So, this is stored and after this we will take the rank of A like k this is the k rank, and it is stored as A_k is equals to U_k sigma k and $V^t k$. This is another form, I mean to store plus the ranking.

K th position. What is A_k ?

A_k is rank of k matrix with rank k. So, basically our main intention is to find this thing.

(Refer Slide Time: 01:00:16)



Now this is very dynamic matrix, I mean documents are adding and terms are adding very frequently. So, let us say once you find this thing then again after sometime a document added and this number gets changed to 10, this number gets changed to 5. So, again we have to find out this thing again. So, what I am doing? I am finding basically the parallel algorithm to find this singular value decomposition. So, this is basically the problem.

Student: How can I solve it theoretically? Even if you add a new document for document D_1 the terms will be like you know, so even document implement will change, right. See you are not adding new document. You are just changing the original document.

Documents can be added, but right now I mean I am basically intending to change in terms, so I mean this will change, this will change, this will change.

What he is telling that can there be possibility to have new terms?

New terms, sir this is I have not searched there.

No, this is you have $t_1 t_2 t_3 \dots t_n$ terms are there, can there be another terms at $t_n + 1$.

$t_n + 1$.

Student: Or in a given document t_1 the number of occurrence of t_1 can be changed.

Number of occurrences?

Number of occurrences can be changed.

Student: No, we are modifying the file.

Modifying the file, modifying the document, this is all or I think terms can also be changed but I did not searched about this whether terms and documents will be changed, i just find out.

No no, I am not sure that how you are changing, documents are getting changed.

I mean documents are changing very.

Contents of the documents.

Yeah, I mean these web document are changed in very frequently. So, these ranks the web page ranks are changing very frequently.

No, ranks are different but how content of them.

Student: Sir another thing let us say I am only maintaining the n documents and a particular document is not being accessed that frequently now. So, it moves down the list out of the n and some other documents come in. So, let me say my i th document has moved out of the list. So, instead of the i th documents I am having the new document. So, that is equivalent to saying that the terms in the document have changed.

What he is telling are you planning to keep the information of the terms with reference to the highest degree of frequencies

(Refer Slide Time: 01:03:14)

$$A = U \Sigma V^T$$
$$A_k = U_k \Sigma_k V_k^T$$

No, basically I am just giving the application; I mean this thing can be used here. I mean I am basically.

No no the matrix size, matrix let me answer that one; the matrix size is it constant or variable?

Matrix size, so whenever this will be calculated let us say t equals to 0 this matrix size this is some n cross n , and it may be of after t times n plus 1 cross n plus 1. So, for n plus 1 cross n plus 1, this will again be calculated. So, calculation of this thing is.

See here there are two components my feeling; given a matrix A n cross n you are updating the terms and all the contents of that t i j , whatever, and you are modifying; that is one thing. Another one is that no matrix also dynamically.

I am not changing the matrix size; I am just changing the entries.

Contents of but you are assuming the matrix as n cross n . So, that problem is then simple because if you change the matrix size the problem becomes more complex.

So this is basically the idea.

In that case if you are considering the matrix size is same, are you arranging these data t 1 t 2 t n rows based on the frequency number, highest frequency will be the first row second highest frequently.

No, I mean just changing the entries and just converting in to some compact form.

So, most referred thing should not be in the first.

Most referred things should not be in the?

In the first row.

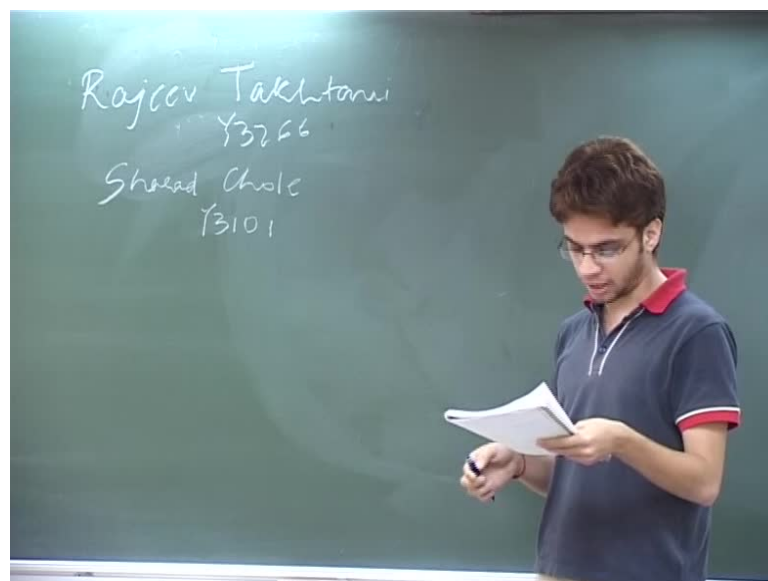
No no no sir.

But I do not know but may be function is like this, sequential solution agrees, right.

I mean there are there are some ways already exist to find these singular values of a matrix.

Of course it exists; let us see because in case inverse will be coming there, so I don't know.

(Refer Slide Time: 1:05:42)



Sharad.

It is Chole.

Sharad Chole

Yes sir.

Sir I have two problems in mine. Sir one is like parallel minimum spanning tree problem, like minimum spanning tree in algorithms.

So where does the algorithm exist?

Yes sir, what I have found is one particular paper has given two parallel algorithms defined like pulse synchronization in parallel like not shared memory model but they have given two algorithms with they are defined time complexity. So, what I suggested like I will go through the algorithms I will read through them and like afterwards suggest an algorithm in a different model make a terminal model.

They are not models side base, okay

Yes sir in this thing.

What is the second problem?

Sir there is a clustering problem that is like hierarchical clustering.

This is you are doing, right.

There is one problem is single-source shortest path problem.

Nithya is doing it similar type of thing.

Student: Sir here it is different.

No, that you can show. Now you tell.

Sir one more problem is like single-source shortest path problem

Ok, explain.

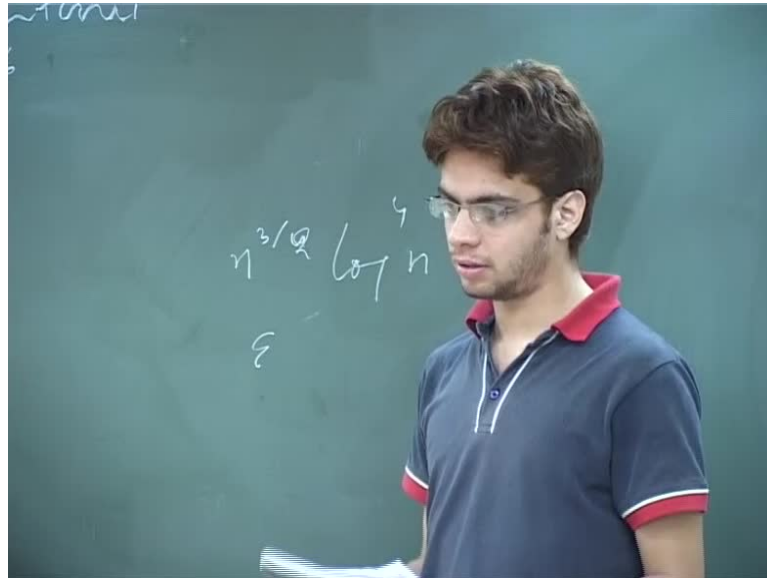
Sir actually like given a problem with one particular node in the graph we tend to find like shortest path to all the nodes in the graph. So, they are like Dijkstra's algorithm can be used to finding that but they tend to find like using the shortest like.

Why you cannot find it? These are the algorithms easily be implemented.

Yeah yeah, but we tend to find some parallel algorithm to do that. There are many

parallel algorithms for that also but like we want to tend to do bind on the given algorithm, because there is one give algorithm by Iris author called Canas.

(Refer Slide Time: 01:08:10)



So, he has given like a reference which tells like he uses which are the cost of n to the power 3 by 2 $\log 4 n$ and there is one more algorithm that has been given recently by author Dutt has given a reminder there. So, he tells like he can like there is a label like epsilon in that complexity. So, like we can rewind on the value of epsilon can be reduced to this. So, like there is a possibility in lambda.

So, what are the problems you have finished?

Sir, we will do minimum span tree and single source shortest problem.

You need to figure at one thing. So, any comments you want to give to him because both the problems you know are classical problems and by now you get several form solution to this problems; only thing is that if you change the model you can think about it how many way; otherwise, I do not know you have to work very hard.

Sir, in this particular problem the minimum span tree problem is not unlikely. It is the problem solution that has the people you have found is like given in the Reynolds model; it is perhaps like this is a similar algorithm on those lines in the shared memory model.

But I am giving that what preference actually that is the thing, You have to work really

hard, right.

Or any updates, I will keep talking to you.