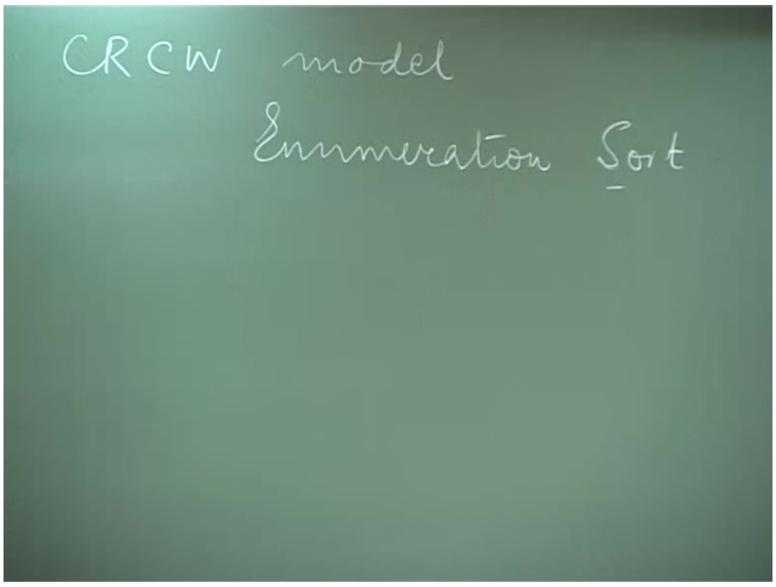
# Parallel Algorithms

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

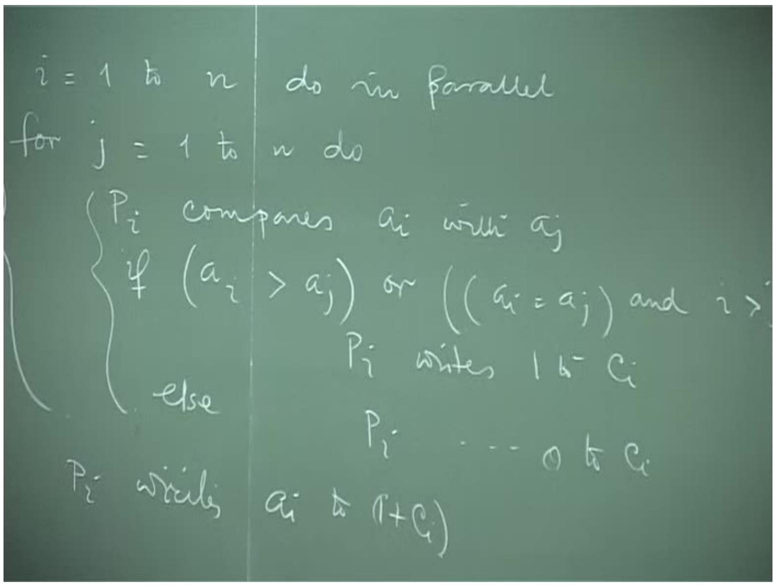
## Lecture - 14

So, in the last class we are discussing about that shared memory SIMB, shared memory model, sorting algorithms; and just to complicate the merging technique and on CREW model, only in a CREW model.

(Refer Slide Time: 00:59)


CRCW model
Enumeration Sort

(Refer Slide Time: 04:32)

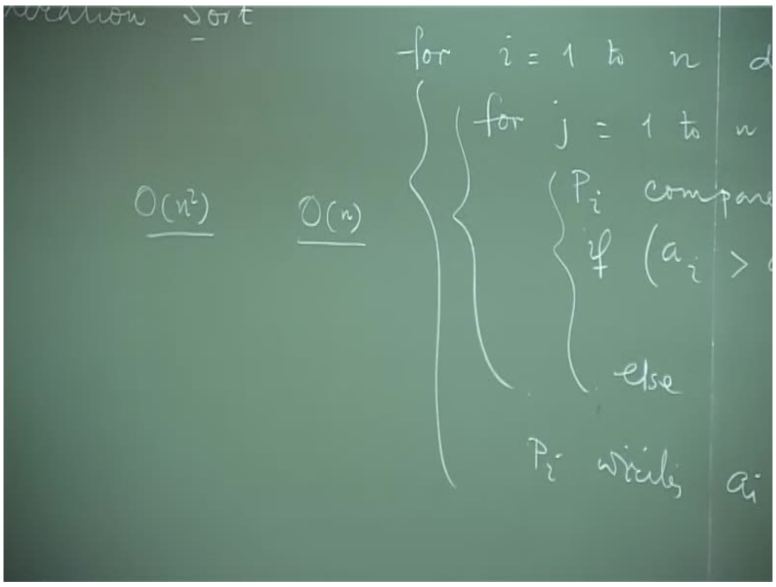


```
i = 1 to n do in parallel
for j = 1 to n do
    P_i compares a_i with a_j
    if (a_i > a_j) or ((a_i = a_j) and i >
                P_i writes 1 to C_i
    else
                P_i ... 0 to C_i
P_i writes a_i to (1+C_i)
```

Now first let us discuss the sorting on CRCW model an d the sorting technique is enumeration sorting technique and you remember that enumeration sort is nothing but, that every element is comparably every other elements and he keeps the rank and then remove the data into that positions. So, here also same thing, for suppose I have n processors and n elements to be sorted for i equals to 1 to n do in parallel; p i compares a i with a j. If a i is greater than a j and or a i equals to a j and i is greater than j p i writes 1 to c i, c i is the counter, else p i writes 0 to c i, p i writes a i to 1 plus c i.

So this is your algorithm that for i equals to 1 to n, n processors accumulated and this were iteration for n because every element has to compared with every other elements. If p i compares with a i with a j, and then if a i is greater than a j or both of them are equal, but j i is occurred try to i, then p i writes 1 to c i, because right and else p i writes 0 to c i. So this c i gives you the counter to move, where to move right, and p i writes a i 2 1 plus c i right and this writes 1 to means and 1 to c i basically add one to c i.

(Refer Slide Time: 05:06)

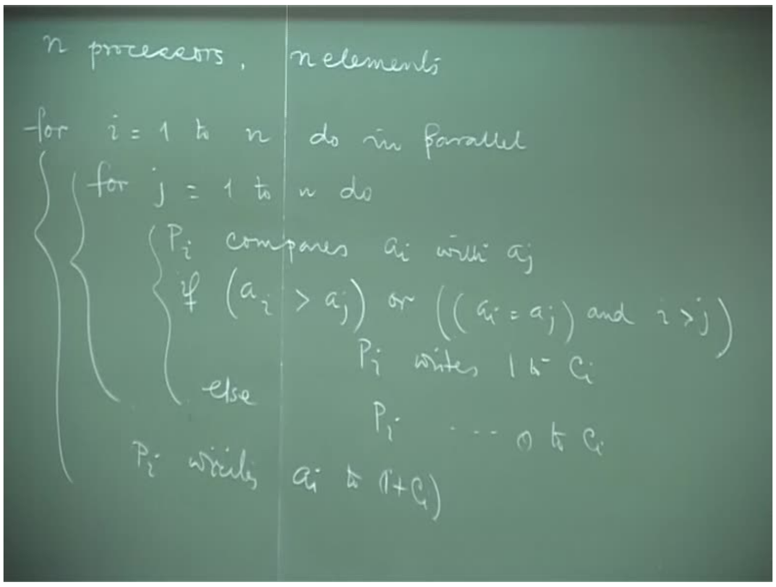


eration Sort

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

```
for i = 1 to n d
    for j = 1 to n
        P_i compare
        if (a_i > 
        else
    P_i writes a_i
```

(Refer Slide Time: 05:11)



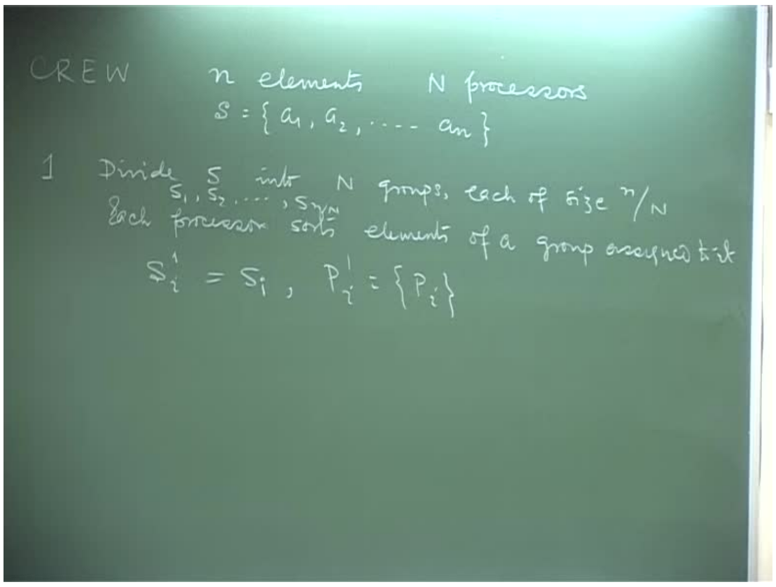
n processors, n elements

```
for i = 1 to n do in parallel
    for j = 1 to n do
        P_i compares a_i with a_j
        if (a_i > a_j) or ((a_i = a_j) and i > j)
                P_i writes 1 to c_i
        else
                P_i ... 0 to c_i
    P_i writes a_i to (1 + c_i)
```

So, this is ordered n time complexity and cost is cost is alright n square which is matching with your enumeration sorted algorithm but, it the obviously it is not cost optimal because enumeration sort itself is not the cost optimal algorithm. It does not give you n log n time complexity agreed now this algorithm also can be thought that way that suppose i have n square processor suppose i have n square processor then also you can use it only p i writes one to c i. Here you will be writing for j equals to one to n do in parallel do in parallel and p i writes one to c i right this you just actually instead of that concurrent write is coming that several processor wants to write into c i. So, basically that procedure that I have mentioned that while I will not to handle and not to handle the concurrent write there is several module you can several way you can thing one is that you assume that minimum of the cost or index should be allow to write or some of that values can be allowed. So, here sum of 1 should be written into c i right.

So, we assume that that will take constant amount of time so this whole algorithm will take constant amount of time to sort n elements using n square processor the cost is becoming order n square time becomes order one in that case is it.

(Refer Slide Time: 09:12)



CREW n elements N processors

$S = \{a_1, a_2, \cdots a_n\}$

1 Divide S into N groups, each of size $n/N$

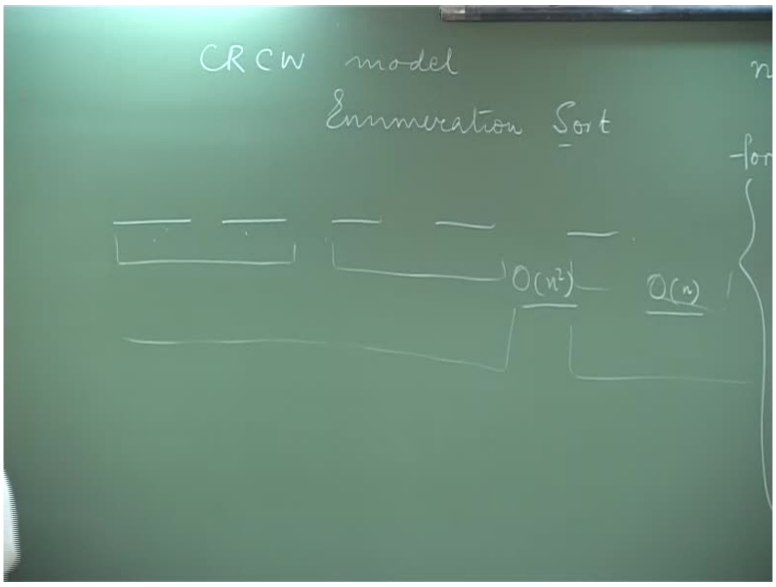
$S_1, S_2, \ldots, S_{n/N}$

Each processor sorts elements of a group assigned to it

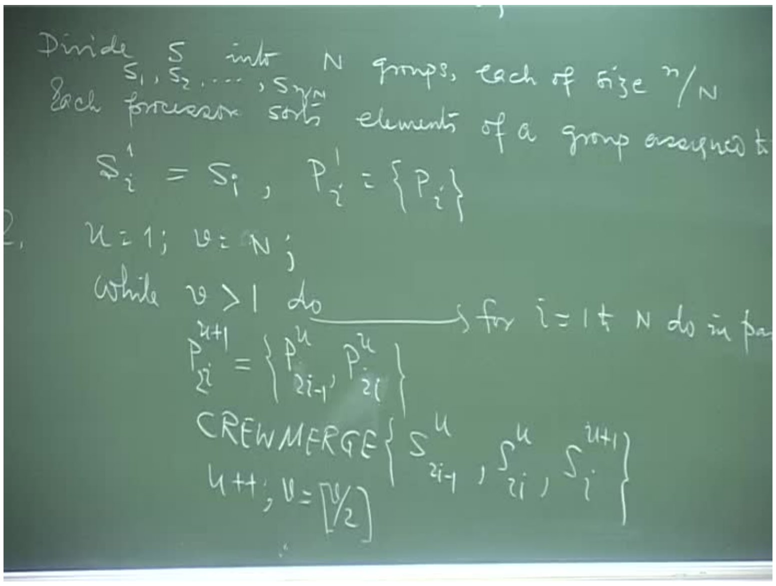
$S_i^1 = S_i$, $P_i^1 = \{P_i\}$

Now, the next model is CREW models, here you have n processors n elements and n processors and this n elements are in the sequence s now initially initially you divide divide s into n groups each of size n by n each processor each processor so you have basically n rows each group is having n by n elements. And now so let us assume that s 1 is to s 3 s n s 1 s 2 s n by n these are the groups right and p i is address to s i. So, each processor sorts the elements of s i right let us do it using any sorting algorithm may be quick sort.

(Refer Slide Time: 09:55)



(Refer Slide Time: 13:53)



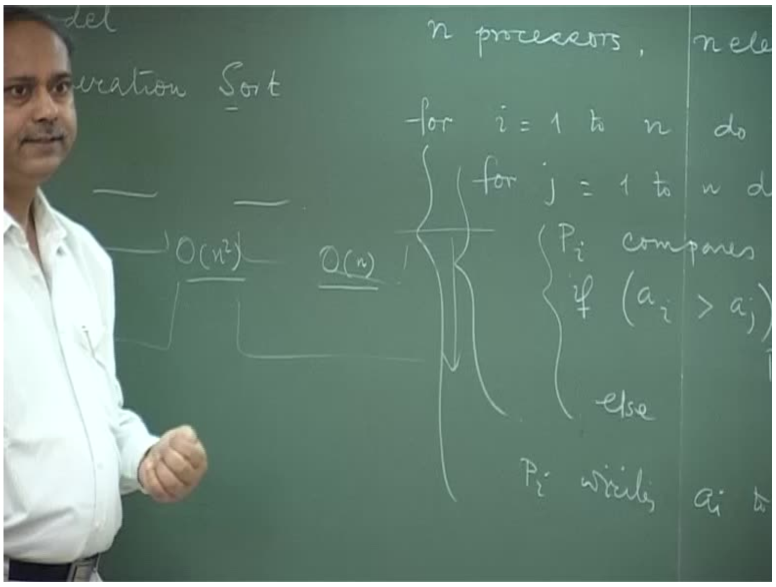
```
Divide S into N groups, each of size n/N
       S_1, S_2, ..., S_N
Each processor sorts elements of a group assigned t
    $S_i^1 = S_i$ , $P_i^1 = \{P_i\}$
2.  $u = 1$; $v = N$;
    while $v > 1$ do ——— for $i = 1$ to N do in pa
        $P_i^{u+1} = \{P^u_{2i-1}, P^u_{2i}\}$
        CREWMERGE$\{S^u_{2i-1}, S^u_{2i}, S^{u+1}_i\}$
        $u{+}{+}$; $v = \lceil v/2 \rceil$
```

And you assume s i 1 is your s 1 s i and p i 1 and the processor p i that is the set of processors is defined by p i 1. Now, what happen that you have you have several sorted sub sequences and you know now CREW merging algorithms? Right you know CREW merging algorithm that we discuss in the previous class so use two processor to merge this two two processor to merge this two two processor to merge this two next time use 4 processor. To merge these 4processors to merge this next is eight processors. To merge this and so on right that algorithm we have discuss so same idea will be using here. This u e and v are the u is that number of iteration of highest step, highest step how many process should be handle that parameter that u is initially 1 this is of 1 to right that is the first stage u is giving the stage number and v is that number of number of processor and apply your number of iterations. You have to do number of iterations you have to do that is that you need to do the log n iterations right first second third and log n iterations so v will take care that one.

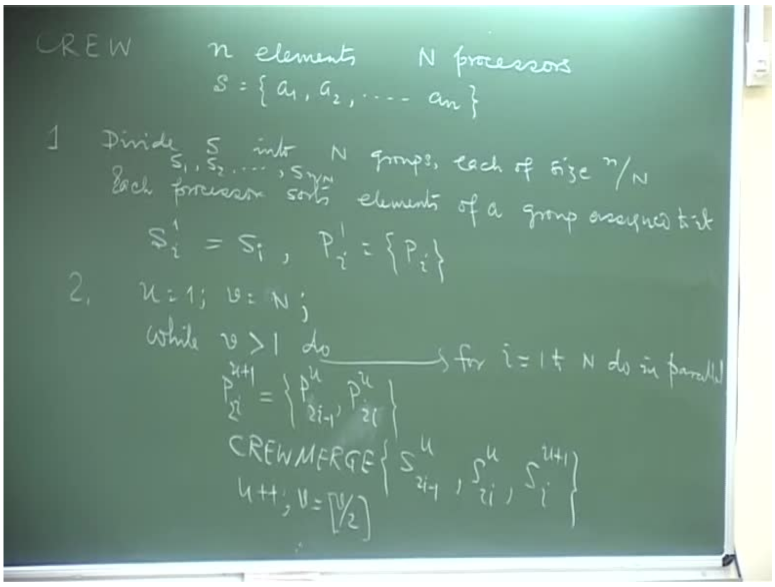
So, while these greater than 1 so you have to merge to it. So, you define the number of processors you are going to ampler is p i 2 u or p i 2 u is equals to p i u and p i plus I, i am writing correctly or not right now it should be 2 y and 2 y minus one processors here you have to write for i equals to 1 to n do in parallel.

Why those 1 to n do in parallel? This n processor will employ so t2 i this is 2 u or u plus 1. This will be u plus 1 is p 2 i minus 1 then i equals to 1 for p 1 and p 2. Whatever say that is having here right so that 3 processors would be employing to fall CREW merge which routines. This s u i minus 1 u s 2 i u it give you s i u plus 1 right, you have been merging this 2 sequence shows u plus 1 and then this simple thing will be u is increased by one and v is decreased by sorry it reduced v by 2. So, only thing is that then it should be that n is always divided by 2 n is always divided by 2 right.

(Refer Slide Time: 14:18)



(Refer Slide Time: 15:10)



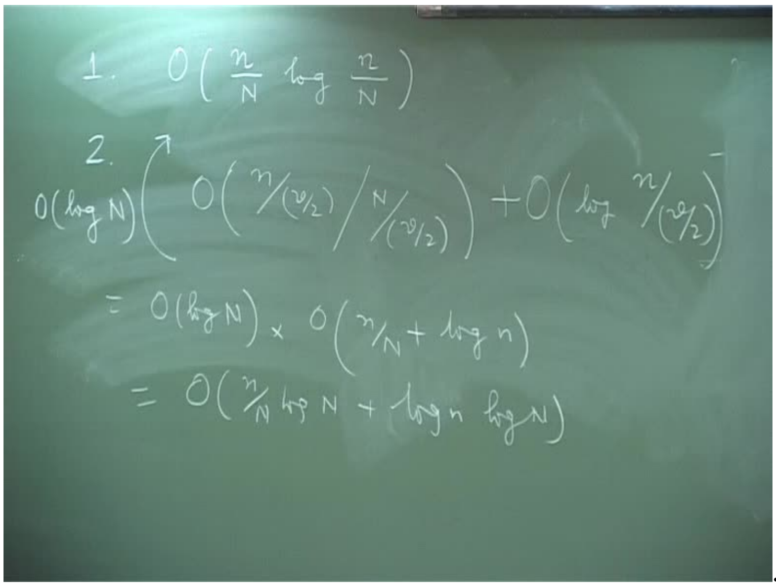
```
CREW        n elements        N processors
            $S = \{a_1, a_2, \ldots a_n\}$

1  Divide S into N groups, each of size $n/N$
          $S_1, S_2, \ldots, S_{n/N}$
   Each processor sorts elements of a group assigned to it
     $S_i^1 = S_i$ ,  $P_i^1 = \{P_i\}$

2.   $u = 1$; $v = N$;
     while $v > 1$ do ——→ for $i = 1$ to N do in parallel
         $P_{2i}^{u+1} = \{P_{2i-1}^u, P_{2i}^u\}$
         CREWMERGE$\{S_{2i-1}^u, S_{2i}^u, S_i^{u+1}\}$
         $u{+}{+}$; $v = \lceil v/2 \rceil$
```

So, here i issuing that you can always carry 2,2,2,2 but, if it is not divisible by 2, what happens you just bring down here to merging anything agreed so if it is not divisible by 2. Just you have to write one line that this is as it is and it show merging just this will be transplicate to this nothing else and u is increased by 1 v is decreased by 2. So, this is the your algorithm agreed any confusion at this stage please let me know which one yes these form of is you are activating all the processors. This all the processor you are activating nothing more than that right, this is not iterating once you tell that all this processor you activate to. Now, once you make the all the processor activated some processor is used for merging this two sequence. Some processor will be used to merge this some processor will be used to merge these two sequence and so on. Which are those processor to define that we are used this one which are the subsequence to be merge to define that we have find this one and this algorithm would be here.

(Refer Slide Time: 23:23)



1. $O\left(\frac{n}{N} \log \frac{n}{N}\right)$

2. $O(\log N)\left( O\left( \frac{n}{(v/2)} \Big/ \frac{N}{(v/2)} \right) + O\left(\log \frac{n}{(v/2)}\right) \right)$

$= O(\log N) \times O\left(\frac{n}{N} + \log n\right)$

$= O\left(\frac{n}{N} \log N + \log n \log N\right)$

.

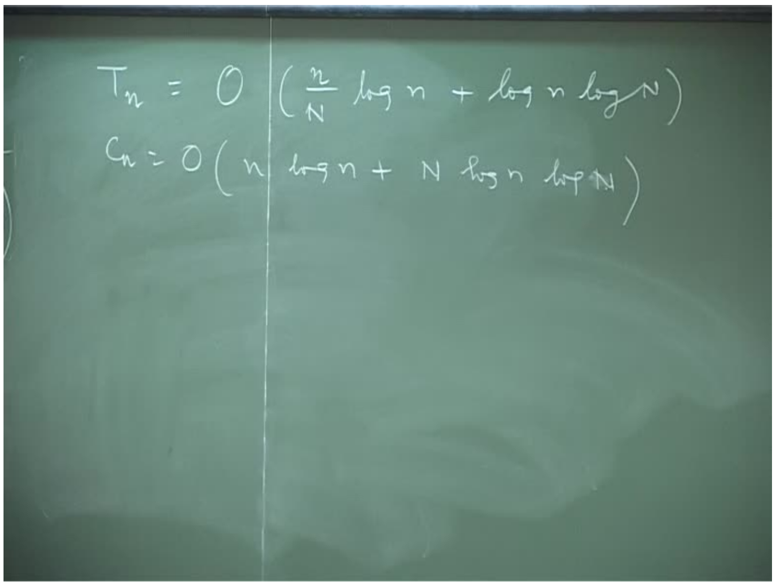
And u will be increased by stage next stage. So, that is why u is in plus plus and size will be that this half right. So, number of textures will be half so that is why it is v by 2. Is it now if I have to compute the kind complexity for this for step one how much time you need how much time capital line? How can you do that the capital line? Yes? So, step one takes this first time, because you are using and sorting all these so the first one deficit does not take much time. Because it is logically only or it conferral read is allowed so there is a problem.

Now, in the step two you observe there is a y loop there is a y loop and every time you are having it right. So, this will be iterated log n times agreed log at the higher time what is the time complexity. You have to find out at the given time now you go back here to election note, you see how much time you need for merging the two sequence. What is the time you see the merging two sequence we discussed this one merging is where is the follow we are looking every time looking for what is that, what do you mean by this follow first you tell me what do you mean by this follow we activate but, this is not that iterating element right.

This is to failure this many processor you are activated so these i is depended on this. So, how can you tell that this is outside the follow? What is the time complexity you remember if you are m n m something, there right what is that time complexity? Say here up to model log there n by n plus do not write m.

First log n n by n plus log n did you get that that algorithm. You got it or not CREW to CREW merge algorithm. Have you got in your lecture not but, way yes or no yes must and stood be getting is not there binary sort finding the j smallest (( )) suppose things is that it is a bit of binary sort no end of that before before median parallel. So, what is the time complexity we obtain order n by n log n not there max up r n. Yes, then log or log then if I request to s equals to n so it will become is it n log n is coming later for finding the cost algorithm. So, this is the thing I am writing, this is small n because this is the size of the elements in the sequence this is the number of processor you have employ, this is the number of processors you have employed, this is the size of the sequence. So, n by n plus order log n n is here running by v by 2 yes. So, this can be written as order log n into order this is n by n plus log n right.

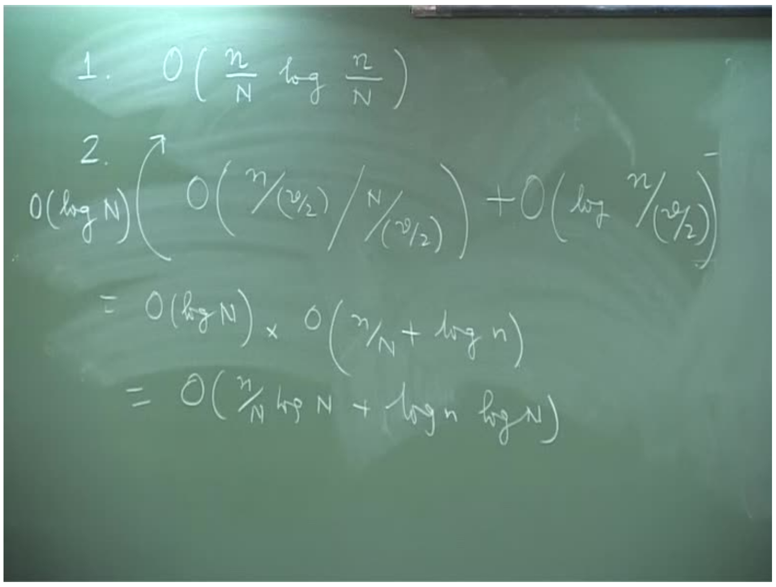
(Refer Slide Time: 23:49)



$$T_n = O\left(\frac{n}{N}\log n + \log n \log N\right)$$

$$C_n = O\left(n \log n + N \log n \log N\right)$$

(Refer Slide Time: 28:22)



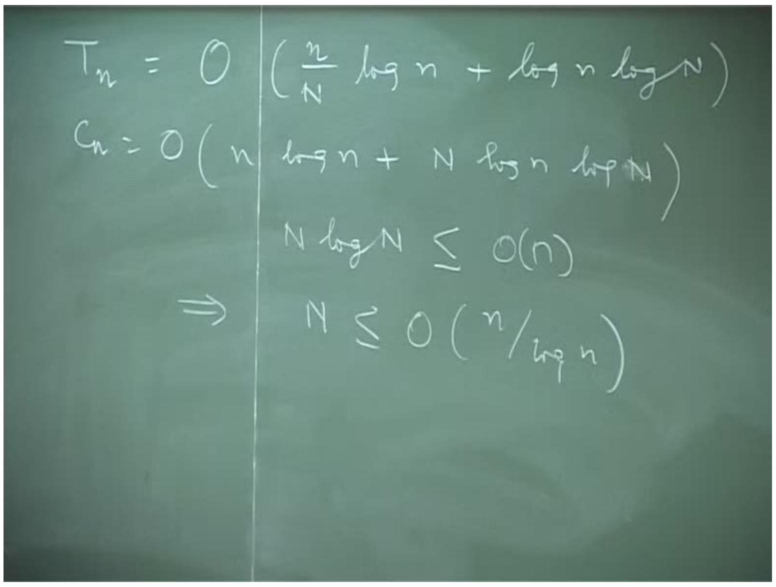
1. $O\left(\frac{n}{N}\log\frac{n}{N}\right)$

2. $O(\log N)\left(O\left(\frac{n/(\nu/2)}{N/(\nu/2)}\right)+O\left(\log\frac{n}{(\nu/2)}\right)\right)$

$= O(\log N) \times O\left(\frac{n}{N}+\log n\right)$

$= O\left(\frac{n}{N}\log N + \log n \log N\right)$

(Refer Slide Time: 24:05)



$T_n = O\left(\frac{n}{N}\log n + \log n \log N\right)$

$C_n = O\left(n \log n + N \log n \log N\right)$

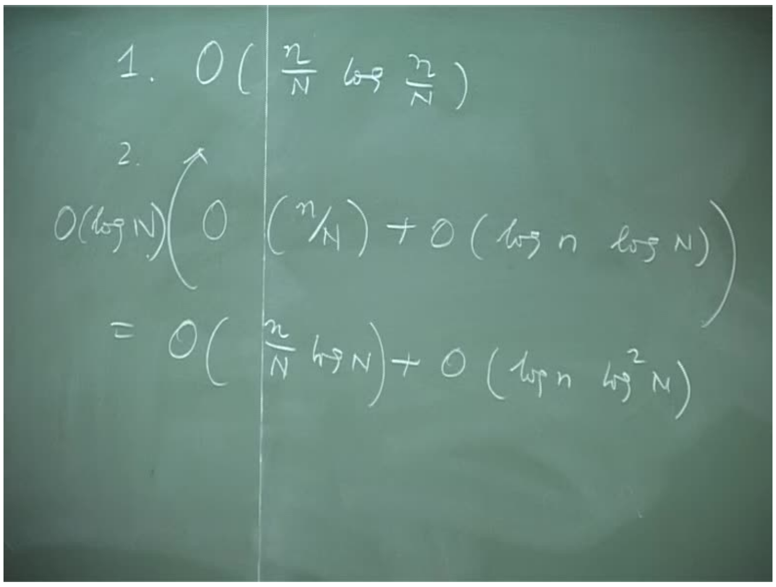
$N \log N \leq O(n)$

$\Rightarrow \quad N \leq O\left(n/\log n\right)$

So, which I can write order n by n log n plus log n log n so to the T n is order n by N log n plus log n log n right. Because this is n by N and this is larger if I write this is log n. Then this get canceled right this get cancel write log n minus log n will be coming last log n will be coming. So, log n plus this agreed or not so cost becomes order n log n plus n times log n log n tell me here is lot of problem I get T n is equals to n by N log n and you have the step one n by n log n by n so this capital n will become minus n by n log n and this is plus. So, this gets cancel right then these becomes cost optimal if if capital n log N is less than equals to order n that is I can write N is less than equals to 0 n by log n, right because capital N is smaller than smaller so this become cost optimal if.

Now, if I use this one for a CREW model you have to pay extra money for this broadcasting excise because, this is concur and read and call another broadcasting will take another log capital N capital N, because capital N elements will be there. So, you have to broadcast it same thing happened here also because, this is you are build binary set if you remember right so that binary set will not be allowed. So, here also you need log n factor so log capital N factor will be added into that and if you add this one then it will no longer it will give you the cost optimal I will get right now.

(Refer Slide Time: 27:59)



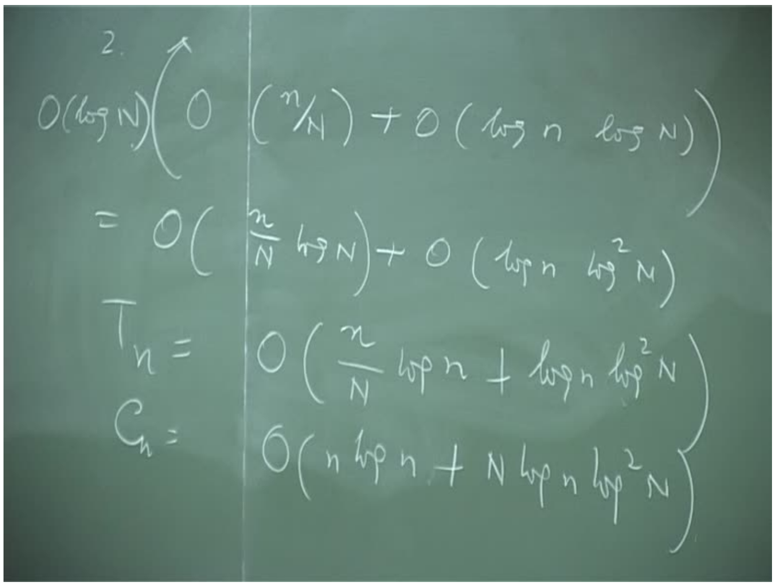
1. $O\left(\frac{n}{N} \log \frac{n}{N}\right)$

2.

$$O(\log N)\left(O\left(n/N\right) + O(\log n \log N)\right)$$

$$= O\left(\frac{n}{N} \log N\right) + O(\log n \log^2 N)$$

(Refer Slide Time: 28:43)



2.

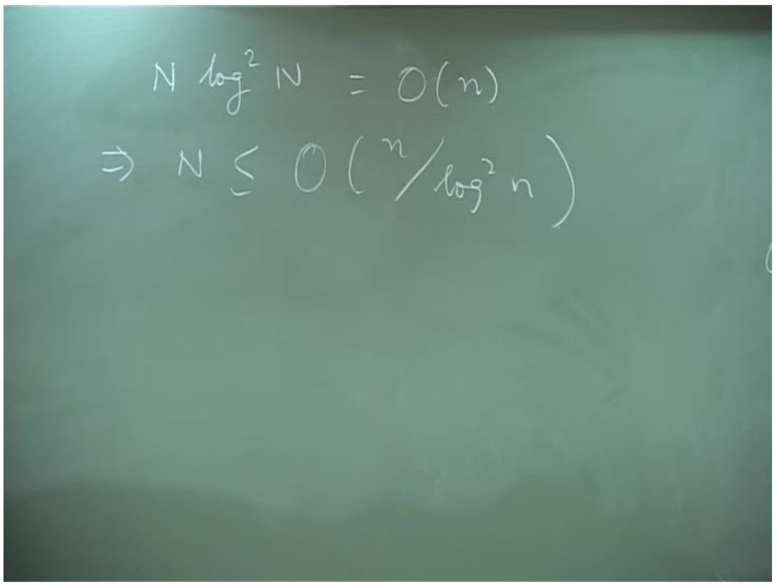
$$O(\log N)\left(O\left(n/N\right) + O(\log n \log N)\right)$$

$$= O\left(\frac{n}{N}\log N\right) + O(\log n \log^2 N)$$

$$T_n = O\left(\frac{n}{N}\log n + \log n \log^2 N\right)$$

$$C_n = O(n \log n + N \log n \log^2 N)$$

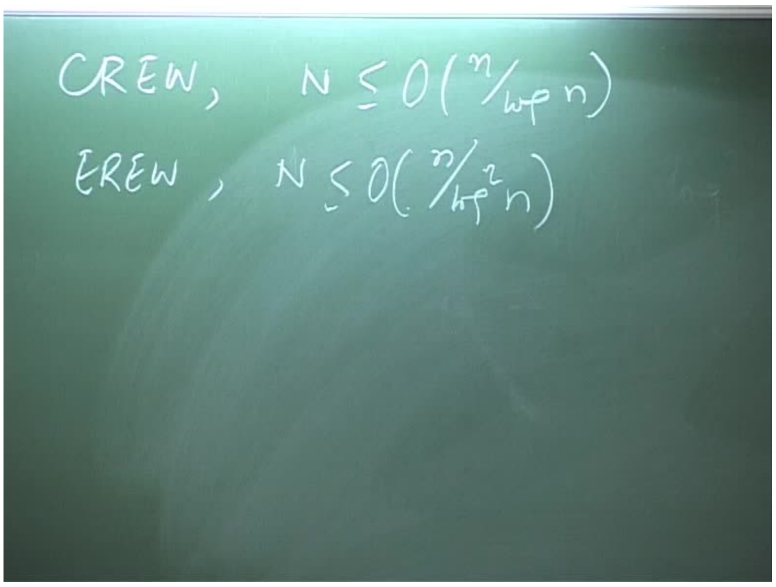
(Refer Slide Time: 29:13)



$$N \log^2 N = O(n)$$

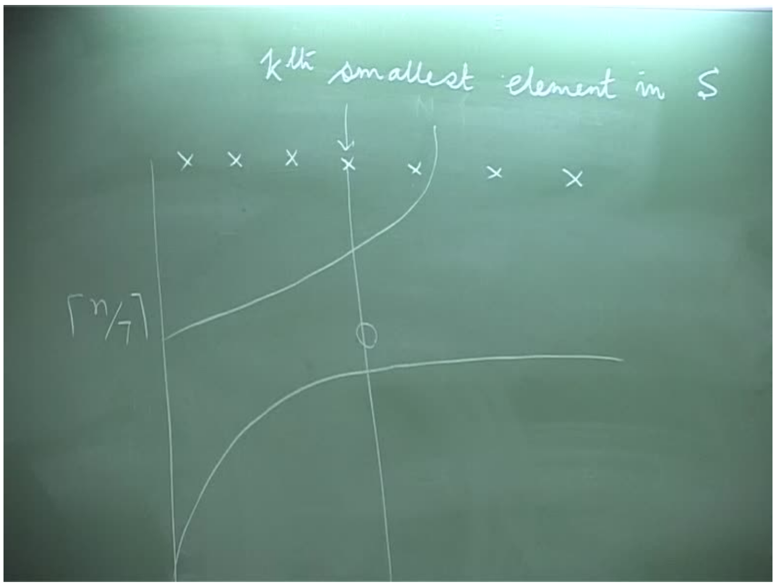
$$\Rightarrow N \leq O\left(n/\log^2 n\right)$$

(Refer Slide Time: 30:00)



What happens if I replace here CREW model by EREW model right, we discuss here CREW model by b d l parallel elements. You remember right in that case what is the time complexity is coming for that? Can you find out in that case in step one will take order n by N log n by n. Step two takes obviously this connection you cannot get it of it. Now you can be CREW model here instead of CREW model EREW merge, now what is the time complexity my recoat tells it is order n by n plus order log n log n is nothing. So, this gets you order n by N log n plus order log n logs square n. So, if I add this to I get the t n is order n by n log n plus log n log square n and the cost become so, this as to be this as not make it cost optimal. This is as to be n log N this is as to be n log N that is n logs square n is equals to order n right. So, which we can write n must be less then equals to order n by in case of CREW model EREW model it is cost optimal. When n is lesser equals to n by log n in case of EREW model it is cost after. When n is less then n by logs square n right now the forming both of them are cost optimal that is but, point is that you know EREW model that we have you have size is much, much less of that CREW model.

(Refer Slide Time: 33:20)

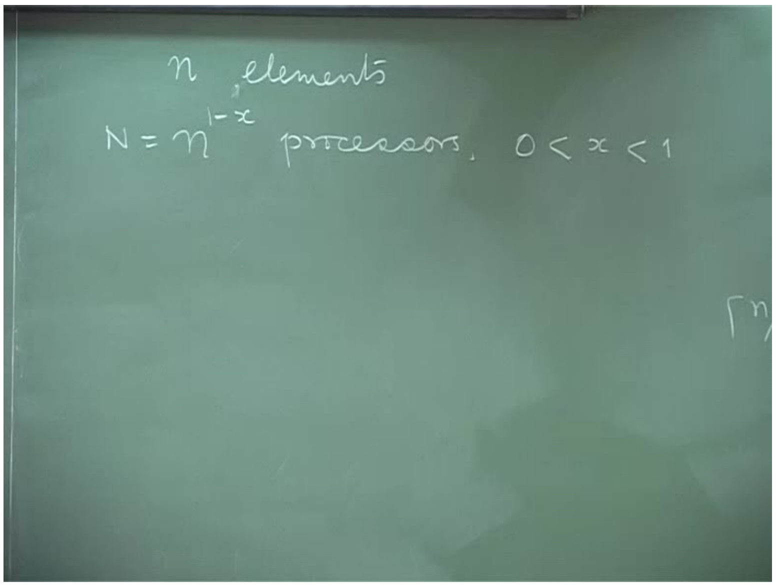


So, can I think about some alternated so you know to see the better algorithm for CREW model, we need to we need we need to go back little that is we are interested to find out finding the k-th smallest element k-th smallest element and from a sequence of n element. Right, that I think you have discussed this thing earlier also k-th smallest element in s is having n elements. Let us think about sequential algorithm first and then we will go for the parallel algorithms and then we will try to leave that one to find out the sorting algorithms for EREW models.

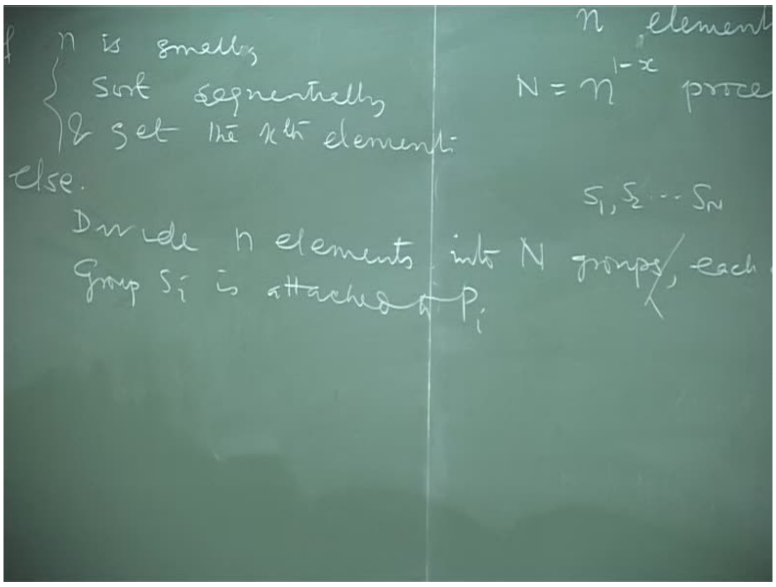
That in the cases sequential algorithms if you remember the (( )) is that you divide this sequences this as into s by 7 groups each group is having 7 elements right. Now later on it has been shown that 7 is also the constant, any constant number of elements right, any constant number of elements in each group only to decide that you must have the best known minimum number of comparison base sorting algorithms for those number of elements. Suppose if this sequential case you divide it of n y 7 groups, each is having 7 elements. In case it is not because subsequence may have component 0 and then you sort it. Sort each. Both which takes constant evolved operations, then we find out the median of this middle elements by recursive process. Suppose, this is the middle element. This middle element is used to divide this 4 sequence into the c groups. The elements which are smaller element, are smaller than this elements which are larger than this. Now if you find this elements open number elements is smaller than this is both the k, then k-th element is lying in this row. If you find that row the total number of elements is smaller than k but, larger than this properties line here that the bigger element is set in the k th element either lines is reduced to this right.

So, time complexity is shown that sorting case finding the bigger of medians, that is recursively you will be finding. So, recursive call will be there and after that dividing to the 3 groups which takes several times and then again you are finding that it has been shown that in that case, then only 1 4-th element will be discarded from your searching zone. So, remaining 3 5 4 elements will be there. So, you have to succeed size in this to t 3 n by 4 and it is the 2 dimensional recursive routines elements. But, ultimately show to find the k-th element right, it is said that lars algorithm so which obtained in 1973 are something like that.

(Refer Slide Time: 34:52)



(Refer Slide Time: 37:00)

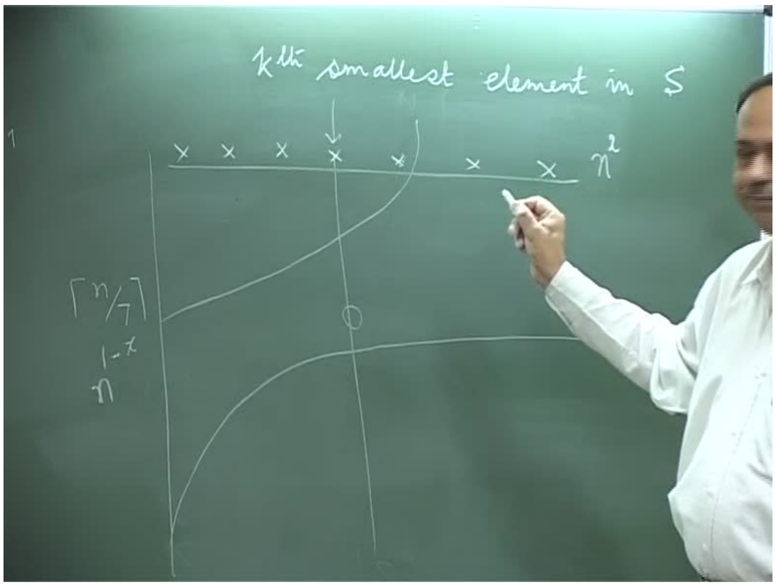


If n is small,
{ Sort sequentially
& get the $k^{th}$ element.
else.
Divide n elements into N groups, each
Group $S_i$ is attached to $P_i$

n elements
$N = n^{1-x}$ proce
$S_1, S_2 \cdots S_N$

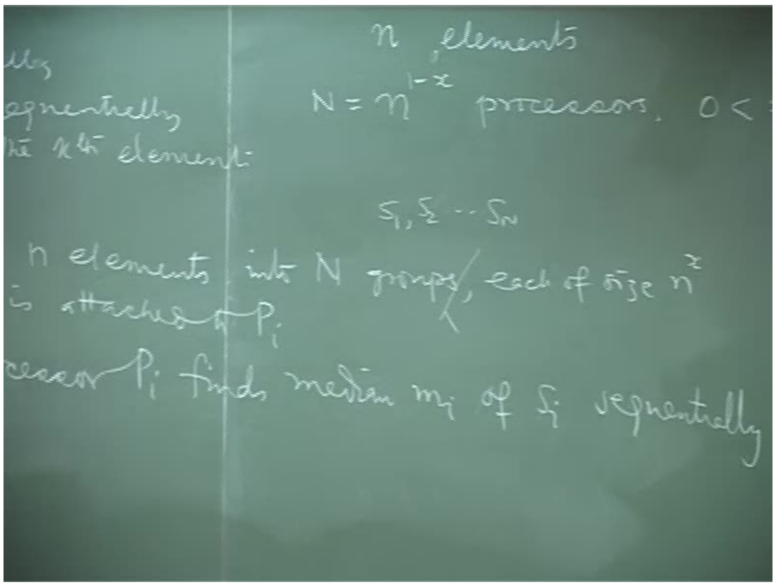
So, same idea can be used to find the k th element in s. Here we are issuing n elements number of processor. Is can be expressed in the form of n to the power 1 minus x. x is lying between 0 and 1. Now, if the number of elements is very less or constant, so, what you do with the case of Lars algorithm is you directly go to the k-th element, you sort it and get the k-th element right. So, if the number of elements, if n, is small than just sort sequentially and get the k th element.

(Refer Slide Time: 37:32)



Else if it is not small then only the question of using the parallel process is coming to the picture. You divide these n elements into capital n groups. Each group is having n to the power x elements. Divide n elements into n groups each of size n to the power x group i or n group. Let us name it here s1 s2 sn group. si is attached to pi so basically it corresponds to the sequential algorithm what we have done here, these instead of 7 elements we are putting n to the power x elements. We are putting n to the power of x elements instead of 7 elements and here your total number of rows is in the n to the power 1 minus x 1. Each processor you are going to the x and sequentially I want to find out the median of this n to the power of x elements agree.

(Refer Slide Time: 38:11)



lly

equentially

the $k^{th}$ element

$n$ elements

$N = n^{1-x}$ processors. $0 <$

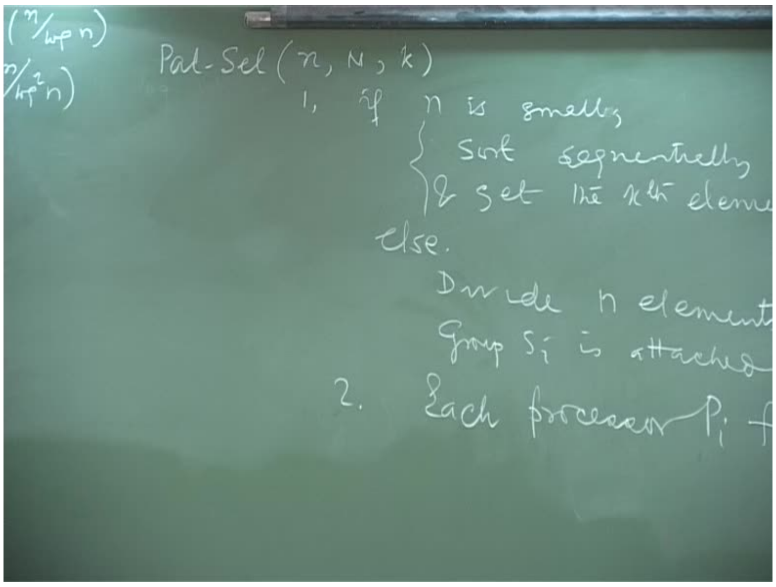
$S_1, S_2 \cdots S_N$

$n$ elements into $N$ groups, each of size $n^x$

is attached to $P_i$

cessor $P_i$ finds median $m_i$ of $S_i$ sequentially

(Refer Slide Time: 38:32)



$(n/\log n)$

$n/\log^2 n)$

Pal-Sel$(n, N, k)$

1, if $n$ is smaller

{ Sort sequentially

& get the $k$th eleme

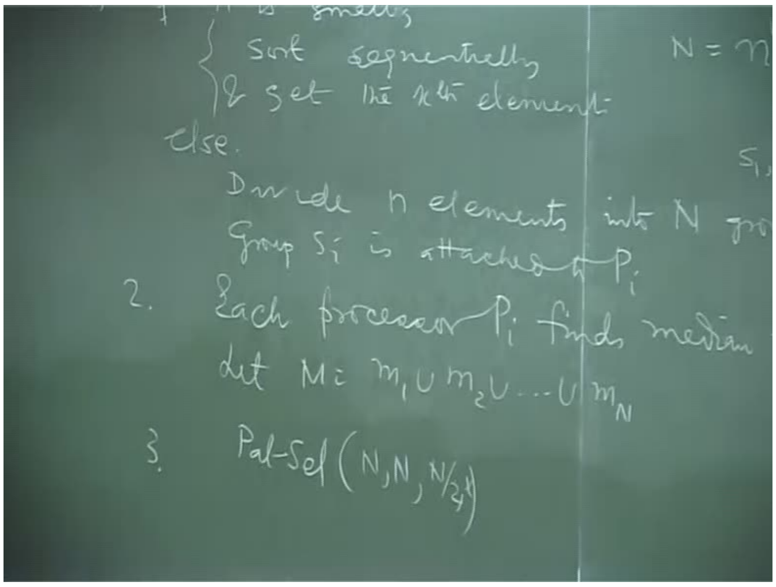
else.

Divide $n$ element

Group $S_i$ is attached

2. Each processor $P_i$ f

(Refer Slide Time: 38:29).



is smaller

{ Sort sequentially

& get the $k$th element

else.

Divide $n$ elements into $N$ gro

Group $S_i$ is attached to $P_i$

2. Each processor $P_i$ finds median

Let $M = m_1 \cup m_2 \cup \ldots \cup m_N$

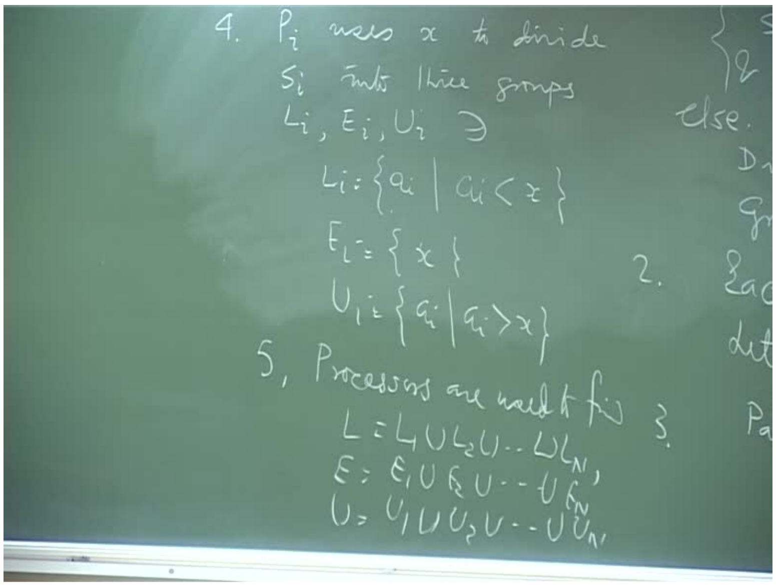
3. Pal-Sel$(N, N, N/2)$

$N = n$

$S_1,$

So, each processor pi finds median mi of si. Sequentially let m is m1 union m2. Now we have got the median of this subsequent you have to find out the median of medians, right. So, you can call this pal select. So, you call pal select u m capital n you have capital n and you have to find out n by 2, right. You have much more than the elements than the number of processor is requiring. So, through that you got the actually I need the returned thing, right, I think I put here x, here also return is x. So, either the n by 2-th element which is x. So, this x is the partitioning element to divide into the 3 groups, agreed.

(Refer Slide Time: 42:16)



4. $P_i$ uses $x$ to divide $S_i$ into three groups $L_i, E_i, U_i \ni$

$L_i = \{a_i \mid a_i < x\}$

$E_i = \{x\}$

$U_i = \{a_i \mid a_i > x\}$

5. Processors are used to find

$L = L_1 \cup L_2 \cup \dots \cup L_N,$

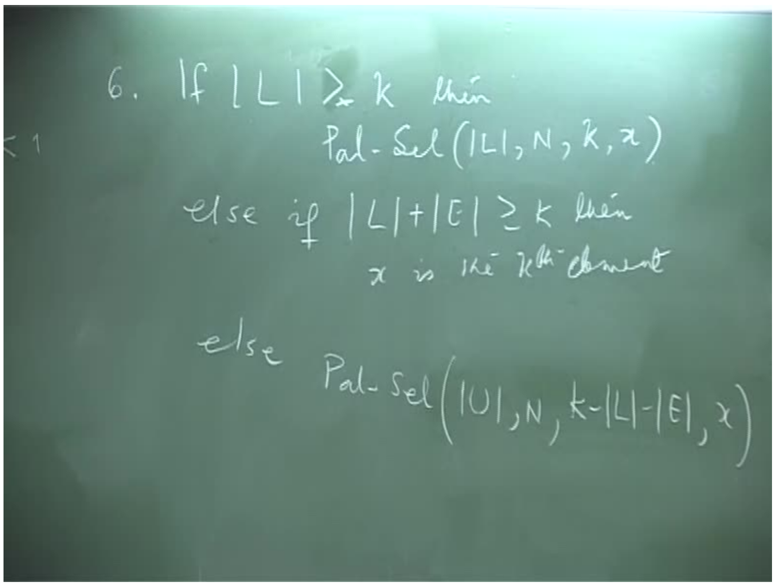
$E = E_1 \cup E_2 \cup \dots \cup E_N,$

$U = U_1 \cup U_2 \cup \dots \cup U_N,$

So, pi step 4, I will write here. Step 4 is pi, uses x to divide si into 3 groups, li, ei and u such that, li contains all the element ai, where ai is less than x. ai is containing all the elements x and ui contains all the elements. ai where ai is greater than x so that is the thing you did in the case of sequential same thing we did for each processor.

Step 5 is nothing but, now you have to find out globally total number of elements smaller than x, total number of elements equal to x and total number elements of u. I did the other ways you will be not able to know other 1. So, processors are used to find l, where is l is nothing but, l1 union l2 union ln, u e u1 union e2 u, and n you agreed. So, then you will get l e and u. So, for l you find out the size of l. If you find the size of i is less than k then you are calling this (( )), on this if you find that size of k is sum of these 2. Then you got already that this is, your k th element. Otherwise you will be reducing getting the k th elements from there that is step 6.

(Refer Slide Time: 43:59)

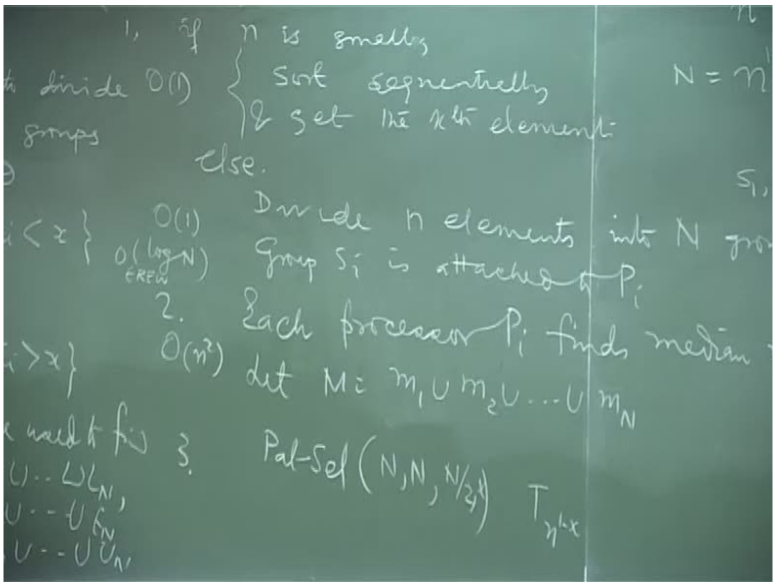


```
6. If $|L| \geq k$ then
        Pal-Sel$(|L|, N, k, x)$
   else if $|L|+|E| \geq k$ then
        $x$ is the $k^{th}$ element
   else
        Pal-Sel$(|U|, N, k-|L|-|E|, x)$
```

Step 6 is if size of l is less than k greater than equals to k, then you call pal select. From l you have total number of processors you have to find out the k-th elements return is your x, else if l plus e is greater than or equal to k, then x is the k-th element, else if pal call select from u n, then this k minus l minus e then return is x. Now, tell me where is the problem? First what we did if the size is small sequentially you sort it and find the k-th element. Otherwise you divide this n elements into n groups each group is placed in n to the power x and group si is let us to pi. So, sequentially pi finds the bigger element of si and this middle elements form m and then sequentially you call pal select algorithm and to get the median of medians that middle elements, you are using by each processor to divide into the 3 substrate li ai and ui li could solve the elements smaller than x. ai contains bigger elements and ui contains all the elements greater than x.

Now, you have to combine this li ui and ai. So, you have combined them using all the processors and finally, after obtaining that the procedure whether the size of the l is greater than r k or not. If it is greater than or equals to k that means k-th element is lying in l. So, you are calling this 1. If it is not and if it is forming this 2 equal greater than this then k-th element is your x, otherwise, the k-th element is lying in u. So, u n k minus l minus e x agreed.

(Refer Slide Time: 46:51)


1, if n is smaller

divide O(1) { Sort sequentially

& get the k-th element

groups else.

$< x \}$ O(1) Divide n elements into N gro

O(log N) EREW Group $S_i$ is attached to $P_i$

2. Each processor $P_i$ finds median

$> x \}$ $O(n^x)$ Let $M = m_1 \cup m_2 \cup \dots \cup m_N$

need to find 3. Pal-Sel$(N, N, N/2)$ $T_{n^{1-x}}$

$\cup \dots \cup L_N,$

$\cup \dots \cup E_N,$

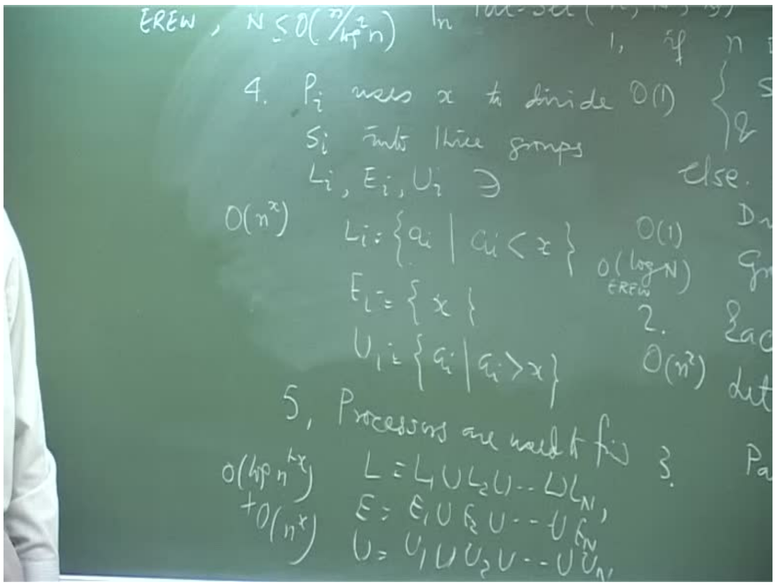
$\cup \dots \cup U_N$

$N = n$

$S_i,$

So, in order to find out the time complexity, this takes constant times. Now, this is constant, this is also a constant time in the case of EREW model. In the case of EREW model you have to broadcast. So, log in the case of EREW model it will come order log n time, this is for EREW model. Now, each processor finds the median independently. Size of s x is order n to the power x. So, this will take order n to the power x. This in the same time if p n is the time then what should be the time here? t n to the power 1 minus x right because, size is n to the power one minus x.

(Refer Slide Time: 47:00)


EREW, $N \leq O(n/\log^2 n)$

4. $P_i$ uses x to divide O(1)

$S_i$ into three groups

$L_i, E_i, U_i \ni$

$O(n^x)$ $L_i = \{a_i \mid a_i < x\}$ O(1)

$E_i = \{x\}$ O(log N) EREW

$U_i = \{a_i \mid a_i > x\}$ 2. $O(n^x)$

5. Processors are used to find 3.

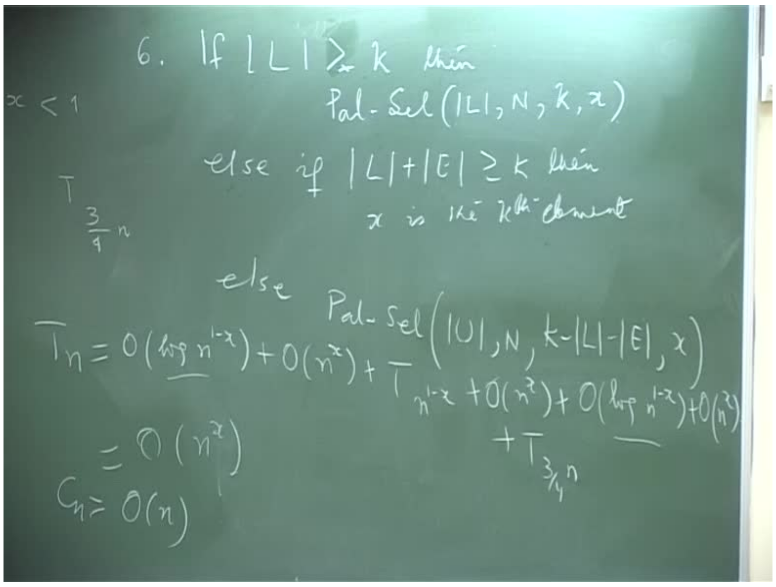
$O(\log n^{1-x})$ $L = L_1 \cup L_2 \cup \dots \cup L_N,$

$+ O(n^x)$ $E = E_1 \cup E_2 \cup \dots \cup E_N,$

$U = U_1 \cup U_2 \cup \dots \cup U_N$

Now pi is dividing his subsequence into L i E i and U i. So, this will take order n to the power x, where you have only n to the power x, and this you will be moving here and here agreed. Now, tell me about this 1, how can you do it? Yes, so what you have to do? You have to find out the total number of elements L i because, once you are computing this one, you can find out the number of L i's right. So, if I know the (( )) number of L i number of l1 number of l2 number of l3. So, this can be done through cumulative sum. So, what should be the starting address of the next processor? Starting address of the processor which takes log n times. So, log n agree or not, it takes log n times and then you will be moving the data right, each person moving the data of n to the power x only.

(Refer Slide Time: 50:41)



```
6. If |L| ≥ k then
        Pal-Sel(|L|, N, k, x)
   else if |L| + |E| ≥ k then
        x is the kth element
   else
        Pal-Sel(|U|, N, k-|L|-|E|, x)
```

$x < 1$

$T_{\frac{3}{4}n}$

$$T_n = O(\underline{\log n^{1-x}}) + O(n^x) + T_{n^{1-x}} + O(n^x) + O(\underline{\log n^{1-x}}) + O(n^x) + T_{3/4\, n}$$

$$= O(n^x)$$

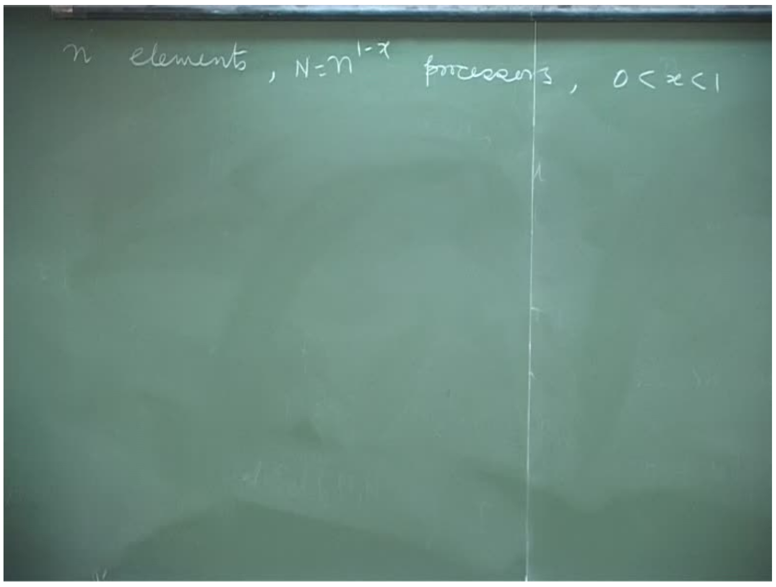
$$C_n = O(n)$$

So, here this 1 will take order log n to the power 1 minus x, that is log n plus n to the power x. That means the data movement in the worst case will be moving. Yes, so, you got l and then this is log 2. This side will be t size of 3 by 4n because that is the worst case. You can extract only 1 part of the element from your succeeding rows. So, remaining in your size.

So, that time complexity become t n is equals to order 1 plus order n to the power x. Here, order 1 is shown up, that you should try log n time log n to the power order log n to the power 1 minus x plus order n to the power x plus p n to the power 1 minus x plus order n to the power x plus order log n to the power 1 minus x plus order n to the power x plus t 3 by 4n.

So, these are this 1 this 1 they are smaller than this n to the power x right. So, this is the primary 1. So, this will become order n to the power x as we did in the case of sequential algorithms. Same sequential algorithm technique, yes. So, it be so the t n is order n to the power x, so, what happens c n c n is order n because that was the n p n is p is n to the power 1 minus x. So, if I multiply it I get order n which is cost optimal right.

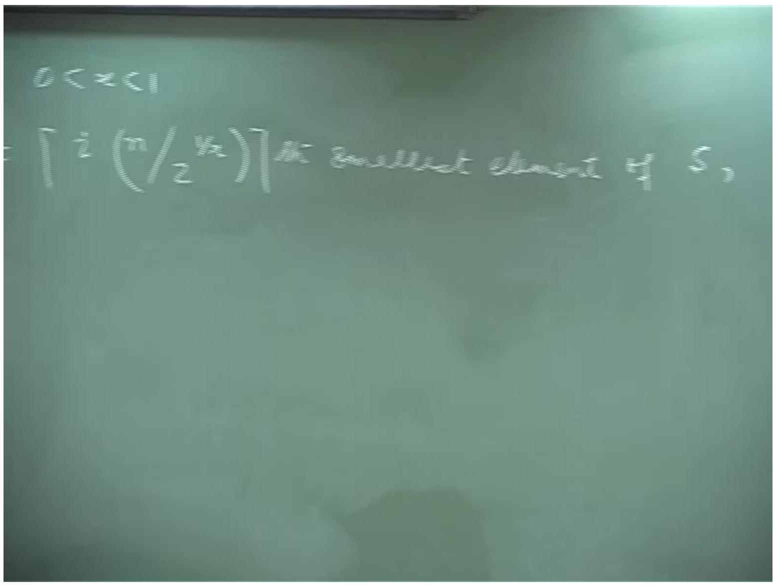
(Refer Slide Time: 51:23)



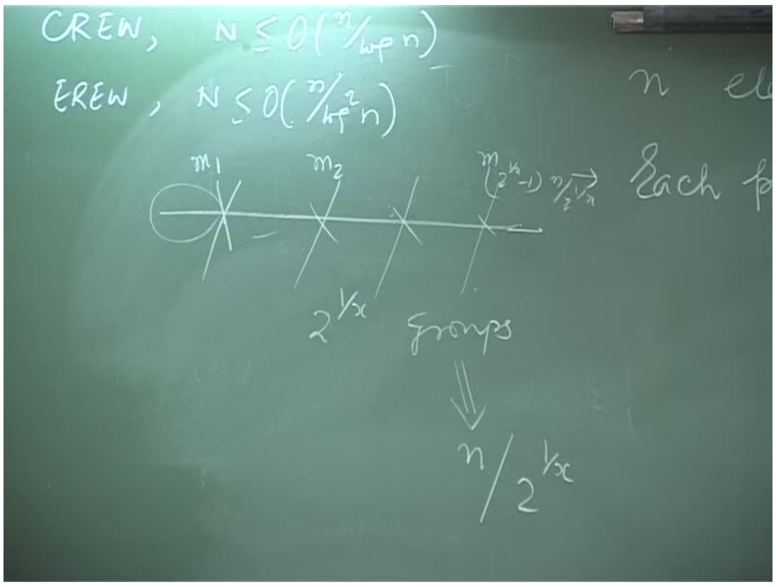
n elements, $N = n^{1-x}$ processors, $0 < x < 1$

So, this true for EREW model so on. So, I have considered all the things. From the EREW model you have n processors an elements and n to the power 1 minus x processors take this satisfying this 1 x.

(Refer Slide Time: 53:34)



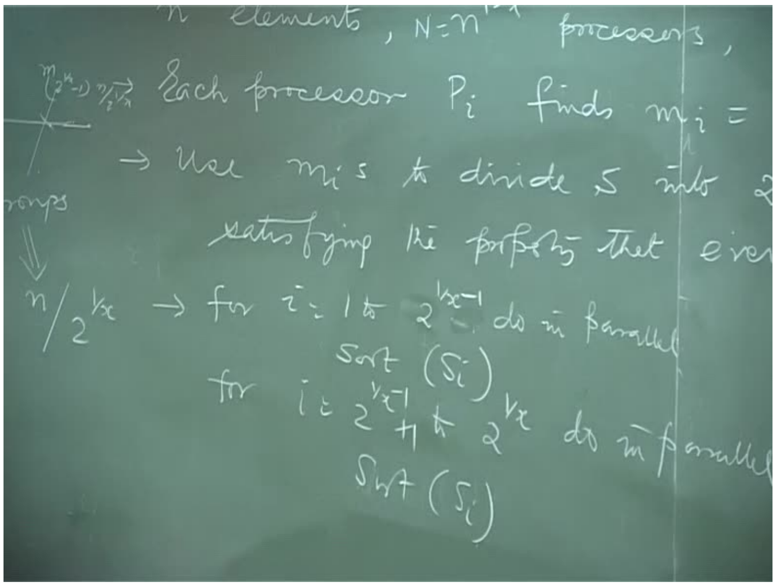
(Refer Slide Time: 54:39)


CREW, $N \leq O(n/\log n)$
EREW, $N \leq O(n/\log^2 n)$
$m_1$
$m_2$
$m_{(2^{1/x}-1)}$ $n/2^{1/x}$
n el
Each p
$2^{1/x}$ groups
$n/2^{1/x}$

Now each processor pi finds mi. This processor pi find mi is nothing but, i into n by 2 to the power 1 by x, smallest element of x, where i is 1 2 3 up to this step. This basically what we are making that you have s element, you are dividing into 2 to the power 1 by x groups, right. This is your m1, this is your m2 and this may be your m2 to the power 1 by x minus 1 and n by 2 to the power 1 by x element. So, that many groups you are making in such a way that all this elements will be smaller than this elements, all this elements will be smaller than this element and so on, right. And if i is having how many elements, in that case each group of size, what is a group size total number of elements is n, and you are making that many, so, both of groups are easy in this, except the last 1.

See I am finding the first n by 2 to the power 1 by x for this element, 2 times n by 2 to the power. So, this are equidistance many number 1 element will be in this side, may be another, may not be existing, visible. That is why may come out that 1 but, 1 or 2 elements plus minus this side will be there but, what happens you divide this m 1's, you can find m's. You can find out now, once you find out the m's you can use this same to find found the s to bring the elements which are smaller than m 1's this side, larger will be this side and so on. So, this groups are satisfying the property that all this elements to be smaller than this elements all this elements will be smaller than this elements so on, as you did in the case of finding the k element on your EREW model again is it ok.

(Refer Slide Time: 59:33)



n elements, $N = n^{1-x}$ processors,

→ Each processor $P_i$ finds $m_i =$

→ Use $m_i$s to divide S into 2

satisfying the property that ever

→ for $i = 1$ to $2^{1/x-1}$ do in parallel
    Sort $(S_i)$
  for $i = 2^{1/x-1}+1$ to $2^{1/x}$ do in parallel
    Sort $(S_i)$

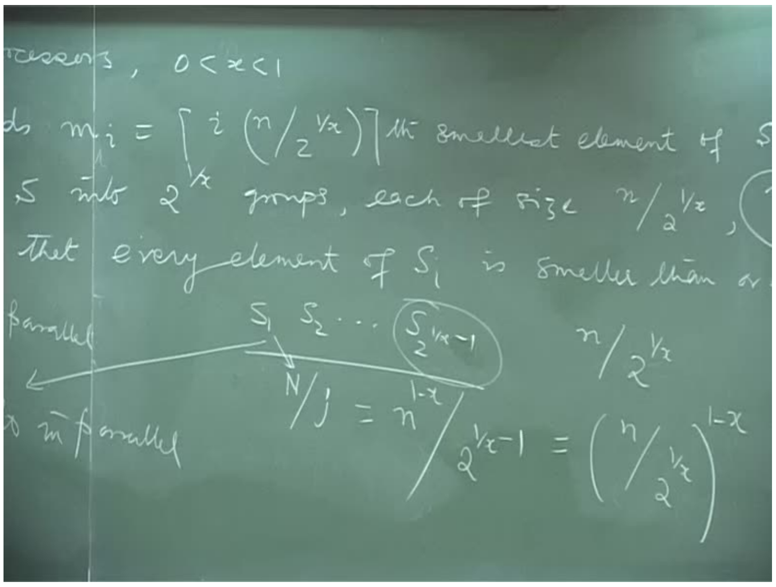
$m_{(2^{1/x}-1)\, n/2^{1/x}}$
roups
$n/2^{1/x}$

Yes, so this means m i's use m i's to divide s into 2 to the power 1 by x groups each of size n by 1 to the power 1 by x and this groups can be s1 s2 sj, sj plus 1, s2 j were j is equals to 2 to the power 1 by x minus 1. So, this finding the properties that equal to all elements of a cycle 1 for all i. Now basically you have s1 is your sub sequence which call the elements smaller than this and so on. So, recursively I will tell why recursively is possible later, on a recursively you employs some person to do this 1 merge s 1 sort s 2 and so on. If I can do that then you have told me sort right. So, this elements if I sort this element just sort and so on, then the problem is solved right. So, I can do it for i equals to

1 to n is it n 2 to the power x 2 to the power 1 by x do in parallel sort sij 2 to the power x minus 1 also for i equals to x 2 to the power 1 by x to 2 to the power 2 j is what a is 1 by x agreed or not. Here it will be minus 1, yeah. So, up to 4 this i will come to this again 2 power x then this is minus 1 plus 1 by x do in parallel sort si.

See basically what we are doing this fault and then this fault that is nothing over to this fault. You do sorting for each of them after completion of this that will do the sorting of this fault. Why it is so? Because in the less number of processor, we employ already there is at. Can I use it recursively or not that is the thing you have to be sown and you have to show that time complexity. See this part is simple for each processor pi find mi, mi right. This is finding that k-th element then use pi, use mi to divide s into this groups that you did li li ui ei and then you will be reading of this, so that will be found here and then this 2 part

(Refer Slide Time: 1:02:29)

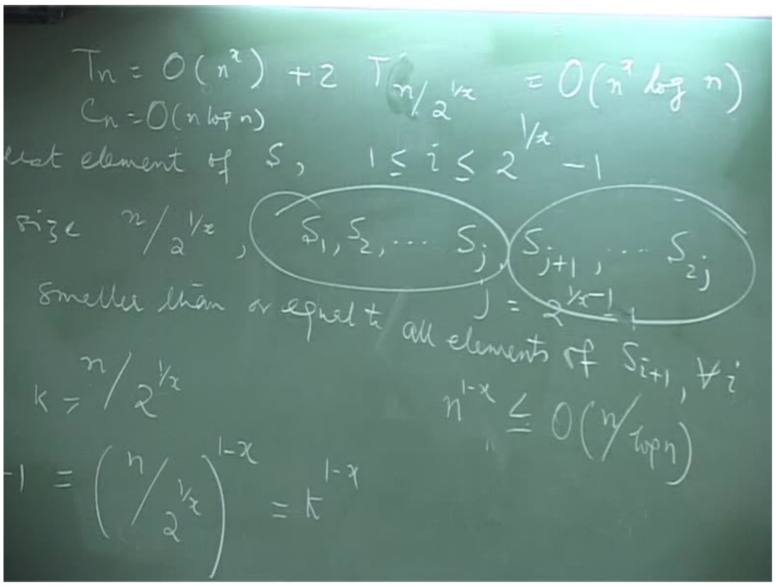


ocessors , 0 < x < 1
ds $m_i = \lceil i (n/2^{1/x}) \rceil$ th smallest element of S
S into $2^{1/x}$ groups, each of size $n/2^{1/x}$,
that every element of $S_i$ is smaller than or
parallel $S_1\ S_2 \dots S_{2^{1/x}-1}$ $n/2^{1/x}$
$N/j = n^{1-x} / 2^{1/x-1} = (n/2^{1/x})^{1-x}$
in parallel

So, here you observe you are employing how many sequences are there s1 s2 s2 to the power 1 by x minus 1 that many sequences you have here, right. I am merging is s1 s1 sj so that many sequences you have, and what is the size of the each sequence, when upon 2 to the power 1 upon x right, and I am employing for each sequence I am employing that many processor. Well I have total number of processor are having j sub sequences. So, each sub sequences can have n by j processor, agreed or not. No maximum i can give now more that is right, because total number of processor is capital and there are j's sub sequences so n by j n is what n is n to the power 1 minus x j is what 2 to the power 1 by x this minus 1 is not here minus 1 is here that was that, means minus 1. So, this is minus 1. So, it becomes 1 minus x so n by 2 to the power 1 by x 1 minus x right. In the gate becomes n by 2 to the power 1 minus x 2 to the power 2 minus x. Which 1 see how many sub sequence you have. j sub sequences right, and how many processors you have capital n. But, is that processor to that so, what I want to say that recursively I must be able to use it recursively I must be able to, means bigger than elements, I should be able to use n to the power 1 minus x processors then only you can recursively call this routine agreed or not.

Now you have j sub sequences and each sub sequences size will be n by 2 to the power 1 minus x 1 by x and you have total number of processor n. So, j sub sequence you can give at most each processor n by j processors, because total number of processor is capital n, j sub sequences. If you keep that equal number of processor to each sub sequence then it is n by j processors which is nothing but, n to the power 1 minus x there over 2 to the power 1 minus x minus 1 which I can add any by 2 to the power 1 minus x to the power 1 minus x. So, your total number of element suppose it is k, then I can write this k to the power 1 minus x form so if you ordered number of processor is n to the power 1 minus x if you have k you have the number of processor k to the power one minus x. So, you can use recursively to sort s1 s2 sj using that same total number n to the power 1 minus x processor. You have not taken any additional processor agreed. If it is the case after that again same way you can do this one right.

(Refer Slide Time: 1:05:42)



$T_n = O(n^x) + 2\,T(n/2^{1/x}) = O(n^x \log n)$

$C_n = O(n \log n)$

st element of $S$, $1 \le i \le 2^{1/x} - 1$

size $n/2^{1/x}$, $S_1, S_2, \ldots S_j$, $S_{j+1}, \ldots S_{2j}$

smaller than or equal to all elements of $S_{i+1}, \forall i$ $\quad j = 2^{1/x-1}$

$k = n/2^{1/x}$ $\quad n^{1-x} \le O(n/\log n)$

$-1 = \left(n/2^{1/x}\right)^{1-x} = k^{1-x}$

So finally, you will be getting that sort sequence. So, you observe these takes order n to the power x time. This take order n to the power x times agreed or not, because that we have already done it. So the time complexity t n becomes order n to the power x plus 2 times T n by 2 to the power 1 by x, because size is reduce to n by 2 to the power x. And the solution to this is right, it is yes, if it is the case then c n is because number of processor is a n to the power 1 minus x. So, you get c order n log n. So, this is always cost optimal right. On the way what is the constant on the processor? n to the power 1 minus x. You must be able to write in this form right, which is always you can find out order. It is always less than it goes to end by log n form. So, this becomes cost optimal on EREW model. You can if n is less than equals to n by log n, which is matching with CREW model.

So, what are the things we have covered? We have covered all sorting algorithm, we are covered merging algorithms, and we have covered finding the k-th’s smallest element, right. Yet to be covered searching k-th’s smallest element another one I will be covering and then matrix study and a graph algorithms right and then common tricks.