Computer Organization and Architecture
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 40
(Part – 2)
Concurrent Programming in Hardware

Student: [FL].

Yeah.

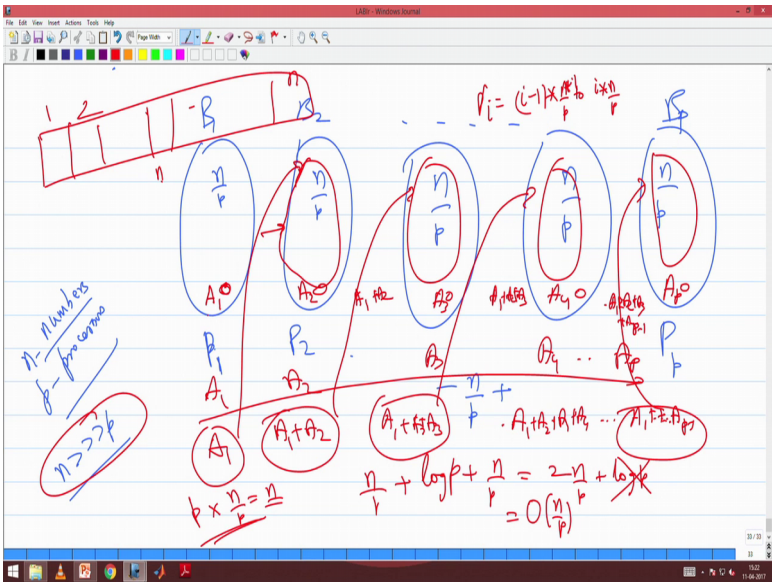
Student: [FL].

First you finish of the entire prefix sum completely.

Student: [FL].

Then you will get another you will get all these fellows. So, just let us say I will over write it, you want to say or you can even over write it or I can put it in a new location also. So, I over write it then at the end if you want to which 1 you want.

(Refer Slide Time: 00:30)



Student: 150.

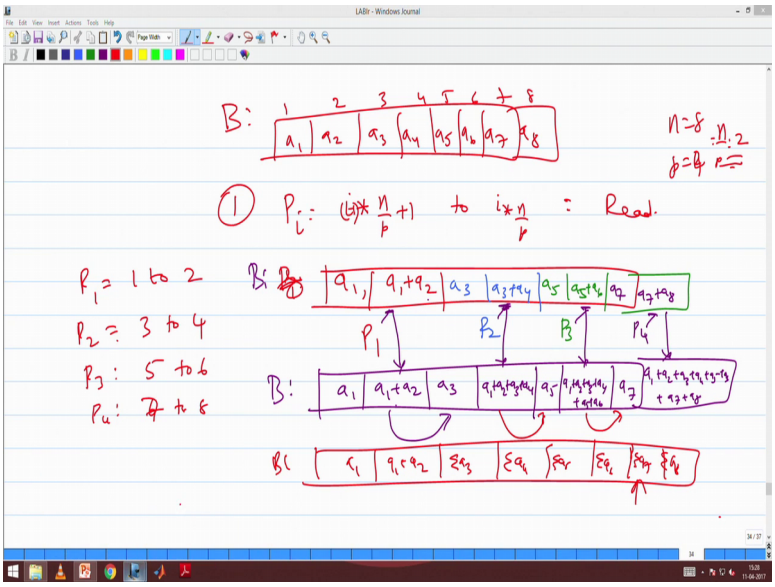
One fifty then go to the second fellow and take 150 here meaning it will be already available

see this is one full array of n this will be 1 2 till n this is 1 array, array of n elements. P 1 will read from 1 to n by p P 2 will read from n by p plus 1 2 2 n by p right. So, pi will read from i minus 1 into n by p to i into n by p, i minus 1 into n by p plus 1 to i into n by p right. So, ultimately in the array itself everything will be stored and go and read from that array. So, there is no.

Student: (Refer Time: 01:33).

Let me say no we will have 2 processors and 8 numbers ok.

(Refer Slide Time: 01:34)



A 1 a 2 a 3 a 4 a 5 a 6 a 7 a 8 and this is array b with the index 1 2 3 4 5 6 7 8 ok. P 1 let us say 4 processors right pi read from i into n by p plus 1 to i minus 1 into n by p plus 1 to i into n by p this is the first step read this is step number one. So, what will be P 1 read P 1 will read 1 to what is n by p n is 8 p is 2 p is 4 n by p is 2.

P 1 will read from 1 to 2 P 2 will read from 3 to 4 P 3 will read from 5 to 6 P 4 will read from 7 to 8. So, so then what will, so then what will P 1 do P 1 will compute a 1 and a 1 plus a 2 P 2 will compute a 3 and a 3 plus a 4 and store in which location this store in. So, what will happen do let us see, so i ok.

So, P 1 will do this. So, P 2 will do a 3 and a 3 plus a 4 will be done by P 2 a 5 and a 5 plus a 6 will be done by P 3 a 7 and a 7 plus a 8 will be done by P 4 this it will all do in the same array this is the bi array again 2 right and this will take 2 2 steps n by p steps after that what

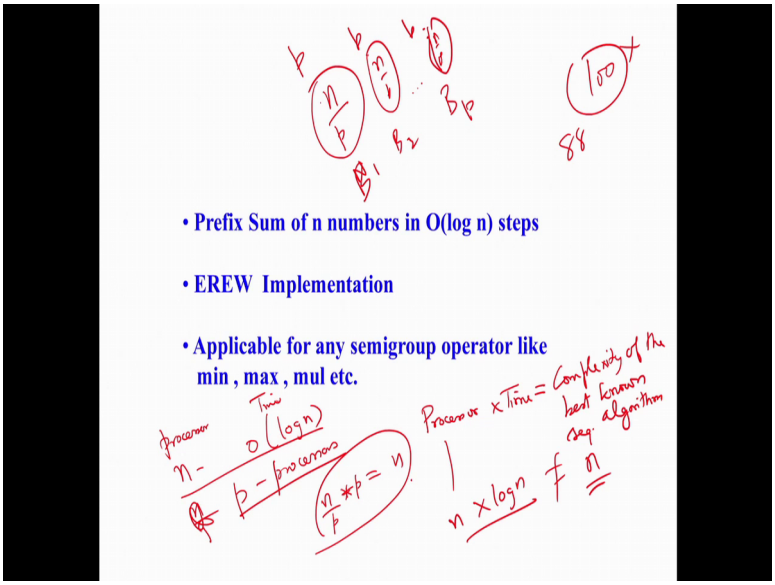
will P 1 do.

So, then I will now P 1 P 2 P 3 and P 4 will take a 1 plus a 2 a 3 plus a 4 a 5 plus a 6 a 7 plus a 8 and now it will do what you do a prefix sum. So, then what will happen, now at the end of this a 1 a 1 plus a 2 a 3 a 1 plus a 2 plus a 3 plus a 4; a 5 a 1 plus a 2 plus a 3 plus a 4 plus a 5 plus a 6; a 7 a 1 plus a 2 plus a 3 plus a 4 plus a 5 plus a 6 plus a 7 plus a 8 this is what will be there in B. Because it does the prefix of 1 2 3 and 4 whatever I put it as a dot here as you see here those things should be saved here in this location is the prefix sum.

Now what happened now P 1 will take a 1 plus a 2 and add it to a 3 while that is going P 2 will take this number and add it to a 4 and then P 3 will take. So, now, let me just say P 1 will take this and add it to this, this will take this and add it to this take this and add it to this.

Now, I will have all the things. So, in B I will have a 1 a 1 plus a 2 sigma till a 3 sigma till a 4 sigma till a 5 sigma till a 6 sigma till a 7 and of course, 8 are you getting this or just want you can query any element and take. So, now, I want what is b 4 B 7, I want I can take (Refer Time: 06:27).

(Refer Slide Time: 06:43)



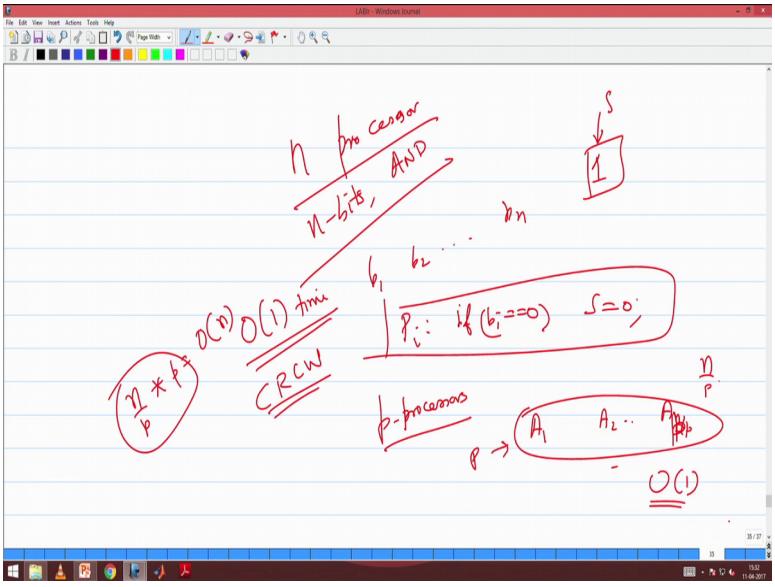
Are you able to follow right this is the program the doubt is answered correct boss.

So, what did we learn now from this lamar now what is the time complexity I had n by p time sorry. So, I had I took n by p plus log p right its n by p times and p processes this n n by p plus p processes n by n by p into p n right. So, the pt product the processor time product is

equal to the complexity of the best known sequential algorithm; that means, this particular algorithm is optimal.

What is optimality in the context of a parallel programming? The processor into time product should be equal to the complexity of the best known a sequential (Refer Time: 07:37) is it ok, all of you its fine.

(Refer Slide Time: 07:51)



Now when n processors ok. Let me just am given n processors and n bits which I want to and un decent bits what is the time complexity for the best time complexity .

Student: Login (Refer Time: 08:16).

Login, but can I do faster and of these bits. So, let me say b 1 b 2 to b n and 1 bit call s . So, initially I will make s as 1 right initially s as 1. So, each pi will see if b i is equal to 0 s is equal to 0 only 1 step and if b i is 0 then s is equal to 0. That means if all the bits are 1 the answer will remain at 1 if at least 1 of the bit is 0 the answer will becomes 0. So, what sort pram model this algorithm will require, what sort of pram model this algorithm will require Palaghat. Exclusive right, exclusive rights.

Student: Concurrent right.

Concurrent right.

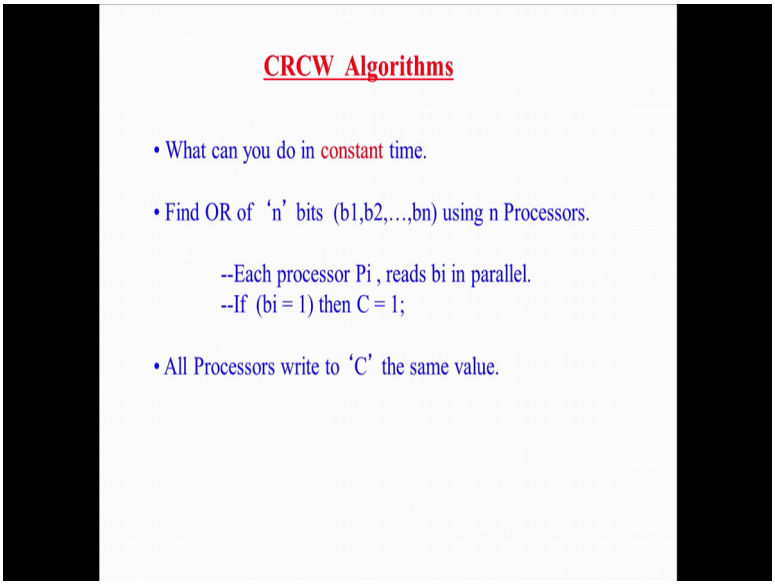
Student: Concurrent right.

Everybody is trying to write into s, but not all the processors if they want write they will write the same value. So, it does not matter who writes first for example, let us b 3 and b 6 b 0 right irrespective of whoever write first b 3 write first or b 6 write first the answer will be the same. So, that is what we told right in concurrent. So, this is 1 step at the end of this step please note that there could be several fellows who would be willing to write to s, but all of them will write the same value. So, it does not matter whichever way we execute it.

So, how much time it will take for me to find the n bits order 1 time , but in which model CRCW suppose I am given p processors how will I do this I will split it into n by p. So, I will get the a 1 a 2 to an by p in sorry ap in n by p time. So, I will find the each processor will find the and of those n by p bits let me call it a 1 a 2 to ap.

Now, I have p processors and p bits now this is take order 1 times. So, in n by p time into p, this is order n, this is optimal algorithm. So, I can design an optimal algorithm which will take n by p time to find the and of p bits, able to do this.

(Refer Slide Time: 11:24)

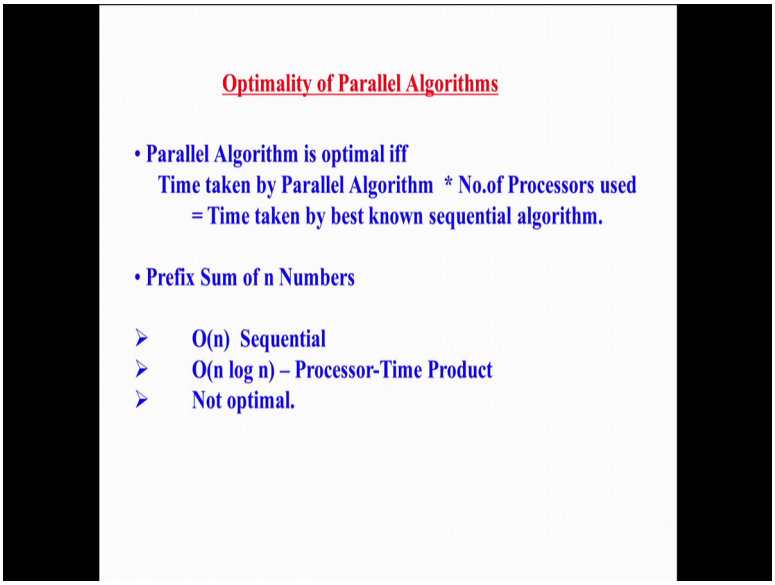


So, how I will find r same way what I should do s 0 if bi is equal to 1 s equal to 1. So, that way I can find the bitwise r and again do it parallel time ok.

Now, let us do the last part next 5 to ten minutes which is indeed bit more complex. So, I can do the r of n bits in constant time I can do the and of n bits in constant time and all the processes that write into that common location in this case it was you know c they write the

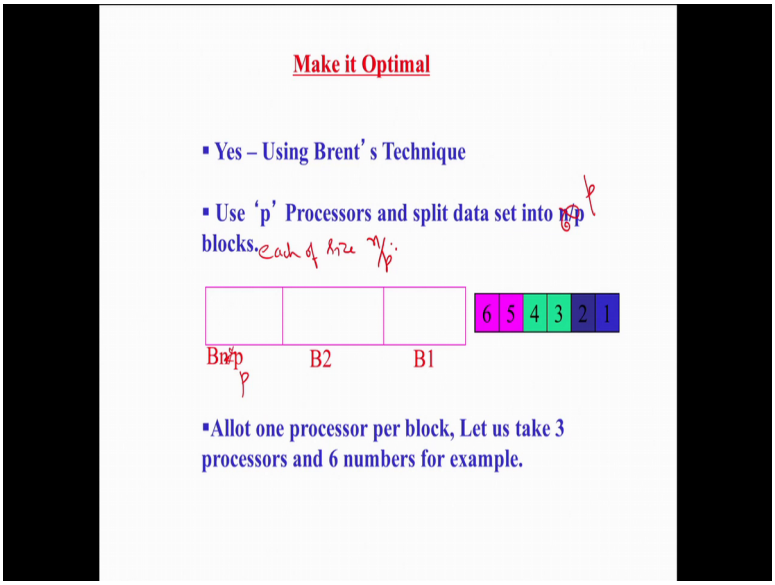
same value. So, this is much suited for its very very clearly suited for the CRCW pram model.

(Refer Slide Time: 12:15)



So, optimality of parallel algorithm parallel algorithm is optimal if only if time taken by the parallel algorithm into number of process used is equal to time taken by best known sequential algorithm.

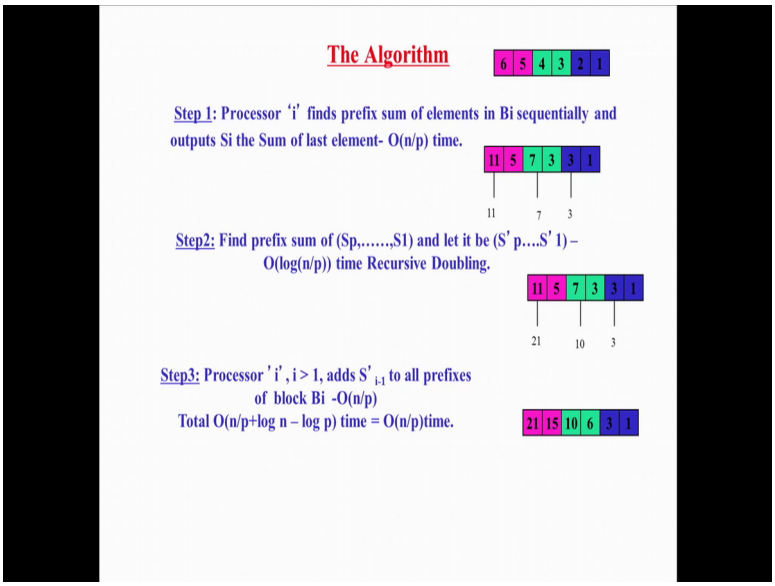
(Refer Slide Time: 12:30)



So, this is how we make it optimal that lemma this is that the procedure that I followed in both the cases of splitting it n by p and then you know.

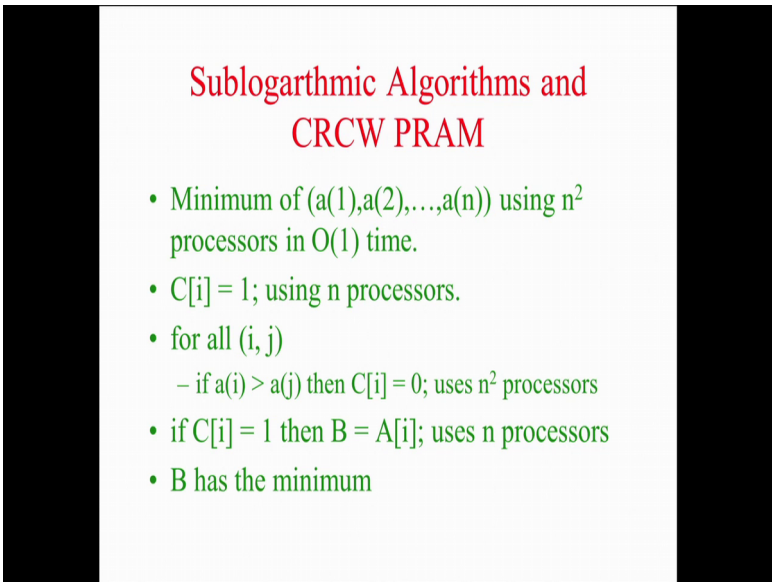
So, use p processors and split data set into n by p blocks sorry p blocks right this is wrong p blocks allow to 1 processor per block let us take p blocks each of size n by p and allow to 1 processor per block and let us take 3 process and 6 numbers we did this right. So, we already saw.

(Refer Slide Time: 13:17)



So, this is this was actually invented or proposed by Brent. So, this is sometimes called the Brent's technique sorry is called the Brent's technique or it also called Brent's lemma both means same.

(Refer Slide Time: 13:41)



Now let us do a much more fascinating algorithm of trying to find the minimum of n numbers what is the time complexity to find the minimum of n numbers, n sequential time complexity so order n.

Now, let us see suppose I want to find the minimum of a 1 to an using n square processors suppose I am giving you a n square processors I can do it in constant time right this is algorithm. Let me have n location called C i; C i equal to 1 and i have n locations that will be using n processors constant time i can finish, now for all i cum r j if a is greater than a j then C i is equal to 0 right that is solve that will use n square processors.

Now, note that except for the minimum element every element would be set to 0 except for all the minimum element I could have more than 1 minimum element also right because if a is greater than a j I consider all the pairs i cum r j except i not equal to j of course, if a is greater than aj then I make C i is equal to 0; that means, there exist a number lesser than that.

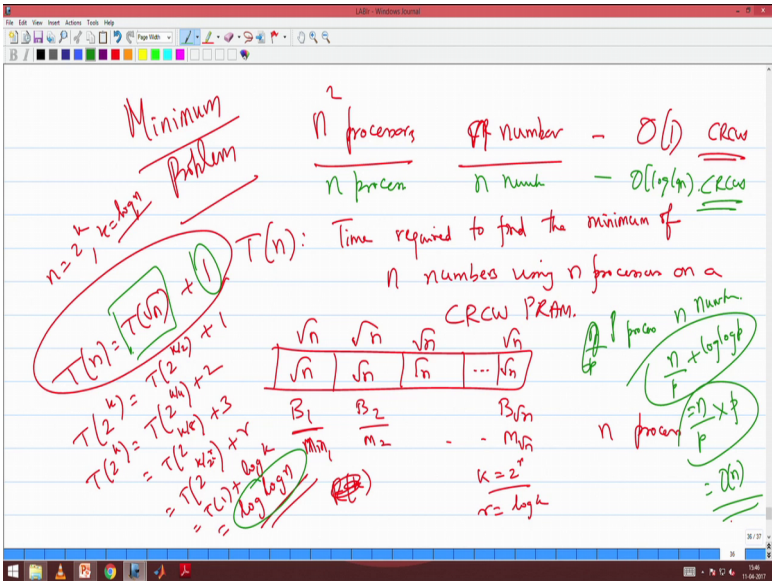
So, at the end of this if C i actually become 0 at the end of the second step that is for a if C i actually become 0; that means, there exist a number which is lesser than correct. So, at the end of second step if C i still one; that means, there is no number lesser than that and hence that is the least number now what you do is C i equal to 1 then b equal to ai again it uses n processor right and if more than 1 fellow or the same then their corresponding case will be one.

So, each of these processors will write into b b is a common location as you see here . So, b is a common location many process may try to read, but all of them will carry the same value if they do not carry the same value then they would have been set to right. So, at the end of this step please note that at the end of this step C i will still remained at the end of this for all ij step C i will still remain 1 only if it is the minimum if it is 1 among the minimum that could be many elements with the same minimum value and then i go and say if C i equal to 1 then I said b equal to a i then several fellows start writing into b, but all of them write the same value.

If two of them a and a j sorry if C i and c j actually writes 1 C i equal to 1 a i and a j are different suppose let us assume a i and a j are different then what would have happened either a i would a greater than a j or a j would be greater than ai then correspondingly either C i or C j would have become 1 0 now I say both are one.

So, that is a contradiction here and hence. So, using n squared processors using n squared processors in I can find the minimum of n numbers in constant time.

(Refer Slide Time: 17:22)



So, n squared processors n numbers this is about minimum problem n squared processors n numbers constant time on CRCW right ok. Now, how I will do it. So, let T n be time required to find the minimum of n processors minimum of sorry n numbers using n processors on a CRCW pram ok.

Now, what I do if I have given n numbers I will split up into chunks of root n each can I split it into chunks of. So, let me call it as b 1 b 2 to b root n and for each 1 of them I will assign root n processors am given n processors right. So, for each 1 of them I will assign root n processors and solve them recursively.

So, T n is equal to T root n T root n is time required to find the minimum of root n numbers using root n processors on a CRCW. So, am. So, am assigning root n processors to each of them and solve it recursively. So, T n will be T root n at the end of it what the n I will have 1 number for each let me call min 1 m 2 to m root n. So, I have n numbers I have root n numbers now I will have n processors because the process are now relived there.

So, I have root n numbers and n processors how much time will it take for me to find the minimum 1 because n number n and n square processor is 1 root n numbers and n process. So, this is T n is equal to T root n plus 1 have you getting this. So, this means let without loss

of generality let n be equal to 2 power k. So, T 2 power k is equal to T 2 power k by 2 plus 1.

Let me say that R k let us say. So, this is nothing, but T 2 power k is equal to T 2 power k by 4 plus 2 this is equal to T k by 8 plus 3 this is T 2 k by 2 power r plus r and it will stop and k is equal to 2 power r right T 2 or r equal to log k when r equal to log k this will be T 1 plus log k and what is k k is log n. So, this will give you a log log n algorithm.

K is log n right n equal to 2 power k means k is log n. So, this will give you a log log n . So, the idea is that if I have root n processors, if I have n processors and n numbers I will fit it into chunk of root n root n each and give root n processor to each 1 of them. So, that will take after that I will finish off in log log n time after that it I will get root n numbers and I will have no n processors root n numbers and n processor I will take constant time.

So, that is how you get this this 1 here, 1. So, and since every every process is executing concurrently I do not need to multiply this with any factors because all route n sub problems are solved concurrently each by the set of processor. What is T root n? Time required to find the minimum of root n numbers using root n processors on CRCW.

Now, I am giving that process. So, it starts working and then this is how I derive so that T of n is actually log log n. So, n square processor n number order 1 time n processor n number order log log n on the CRCW n processors n numbers, now what will happen if I say n by p processors sorry p processors and n number. I will just took I will take the minimum of n by p each then I will have n by p elements. So, I will have p elements and p processors if I have p elements and p processor it take log log p right. So, n by p plus log log p since p is very much less than n log log p is still very much less than n. So, this is equal to n by p into p processor I will get order n. So, I can use the same Brent's technique to show that I can get an optimal.

So, 2 interesting thing number one is that if n square processor constant time I can find the minimum of n numbers that is the first simple, but very interesting asset. I had done a how will you use it to actually find this rigorous relation and how can I get a order log how can I use it to get an order log log n algorithm and then finally, how to optimize it .

So, with this we sort of understand what is a CRCW pram and EREW pram CRCW is very close to EREW. So, am not taking any specific example there now we can start writing. So, if you are writing some major parallel program we should first understand classify this and start thinking in the direction else you know you may not get that real speed up that you are

looking for fine any doubts are you able to understand this any doubt. So, if there are no doubts, ok.

Thanks.