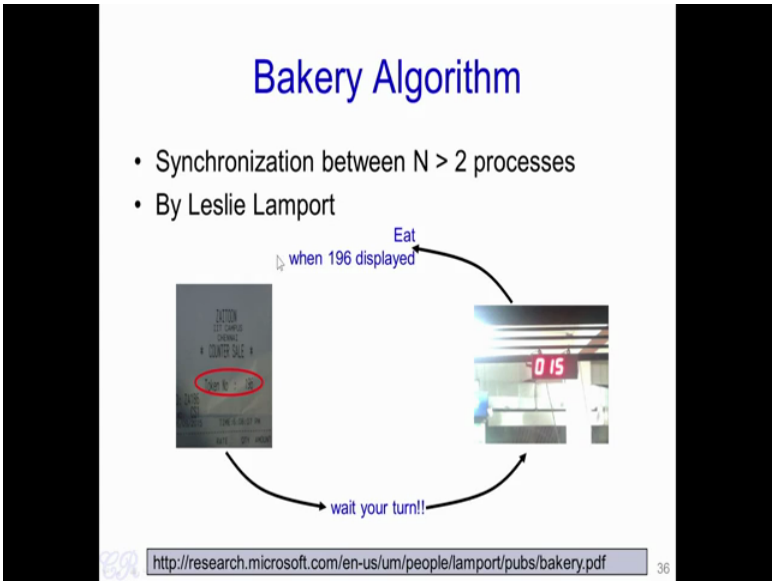
Introduction to Operating Systems
Prof. Chester Rebeiro
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Week - 06
Lecture – 26
The Bakery Algorithm for Synchronization

In this video, we look at the bakery algorithm, which is a software solution for the critical section problem. Essentially compared to the Peterson Solution, which we seen previously, this algorithm is most suited for larger number of processes. Essentially when we have the number of processes which is greater than 2 then the bakery algorithm would work efficiently.

(Refer Slide Time: 00:43)

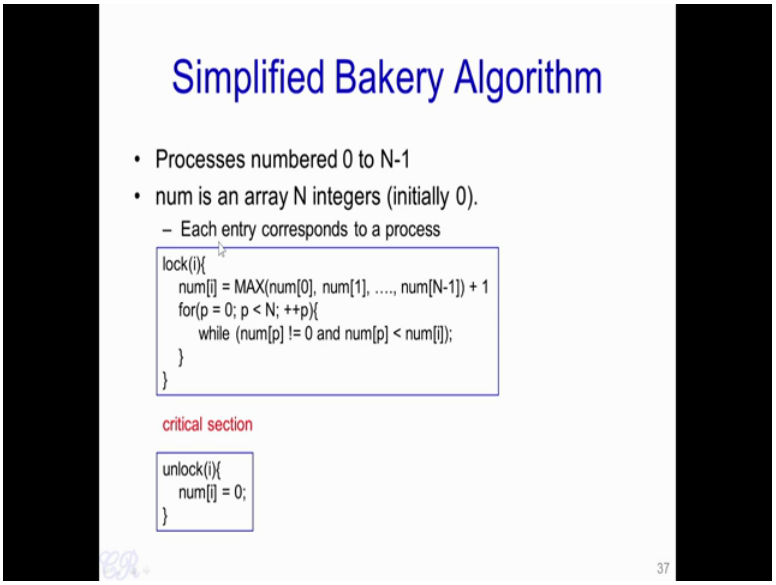


So, the bakery algorithm was invented by Leslie Lamport, and you can get more information from this particular website present here. So, the essential aspect of the bakery algorithm is the inspiration from bakeries and banks. In Some bakeries, what we see is that when we enter the bakery, we are given a particular token have for instance in the bakery, we would be given token with a number present over here.

In this particular case, we have a token number 196 which is present. Now we need to wait for some time until the token number 196 is called out. So, we have a display over here which periodically would set a token number, and when the token number of 196 is

displayed then you are able to get your food from the bakery and you could eat. So, essentially when we look at this from synchronization aspect, we see that we are trying to synchronize the usage of a particular counter. So all people who have such a token should wait until their number is called then sequentially each person depending on when the number is called goes into the counter, and is able to collect whatever he or she wants and for instance eat.

(Refer Slide Time: 02:16)



So, we will see how the bakery algorithm is used to solve the critical section problem. So, we will start with the simplified analysis of the bakery algorithm. So, this particular algorithm is used to solve the critical section problem when there are N processes involved and all these N processes access the same critical section. Now there is also a global data which is shared among these N processes and this data is known as num. So, the length of num is size of size n; essentially each process as a particular index in this num array.

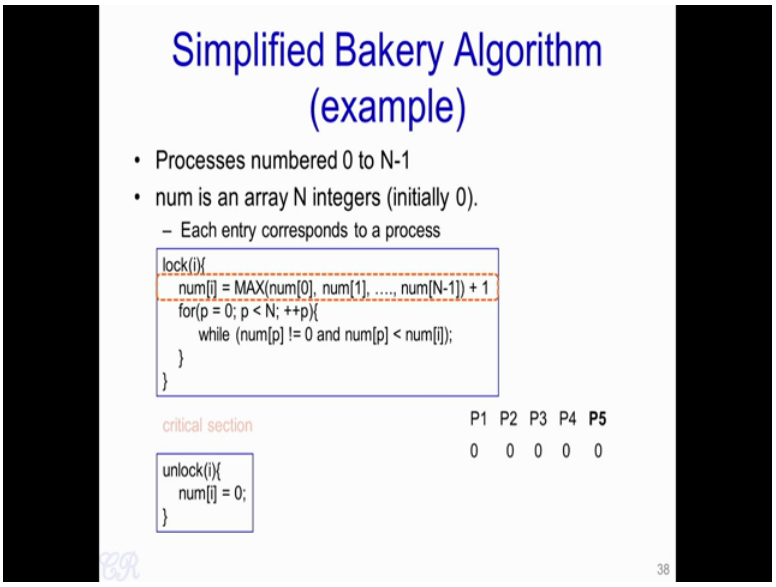
So, for instance process 0 would have a flag corresponding to num 0, process 1 has num 1, process 2 has num 2 and so on. Secondly, at the start of execution, this value of num is all set to 0s. Now in order to enter a critical section, a process would first need to invoke the lock call with the value of i. So, i here is the process number for example, a process id. So, we have N processes so, the value of i could be from 0 to N minus 1. And at the

exit of the critical section, the function unlock i is invoked where the value of num i is set to 0.

Let us see what lock and unlock is actually doing internally. So, when a process invokes lock of i where i is its number; it is corresponding num value. So, num of i is set to the maximum value essentially this particular function MAX is going to look at all the numbers corresponding to all of the processes and get the maximum from that and add one to that. So, num of i is going to get the highest number which is present among the shared num array. Second there is for loop, which scans through all processes. So, p equal to 0 to p less than N plus plus p.

And within this for loop, there is a while loop, which checks two things it checks that num of p is not equal to 0 and num of p is less than num of i. So, essentially this particular while loop will break, when either num of p is 0 or at this particular condition is false. So, essentially num of i is less than or equal to num of p. So, essentially, the process I will enter into the critical section only if it as the lowest nonzero value of num of i.

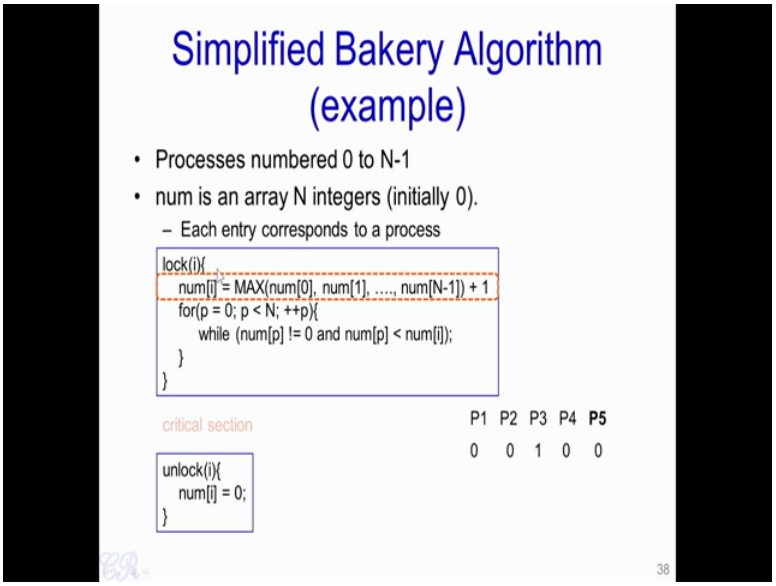
(Refer Slide Time: 05:25)



Let us look at this within example. Let us assume that we have five processes P 1 to P 5, and these are the num values. So, the num array is also having five elements, and each of these elements corresponds to our process, so num of 0 corresponds to process P 1, and num 1 corresponds to P 2, num 2 corresponds to P 3 and so on. Now let us also assume

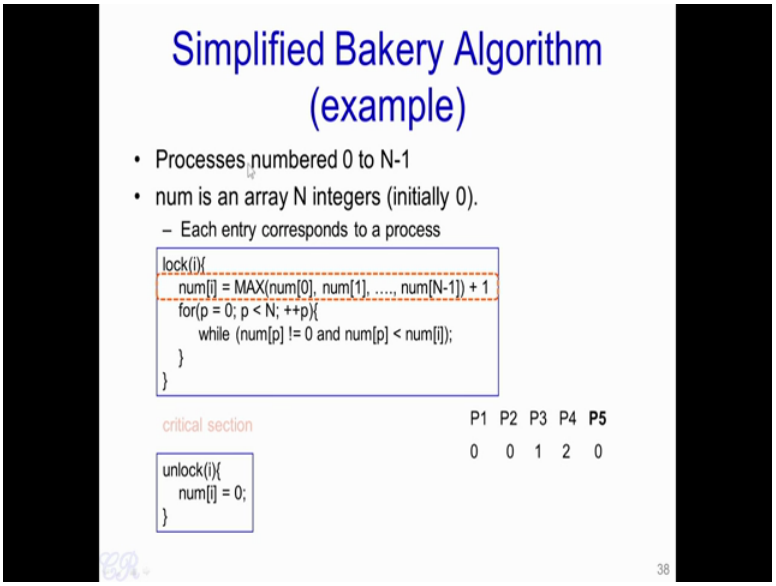
that all these processes almost simultaneously invoke the lock i function, essentially all these processors want to enter into the critical section almost simultaneously. Then what would happen.

(Refer Slide Time: 06:11)



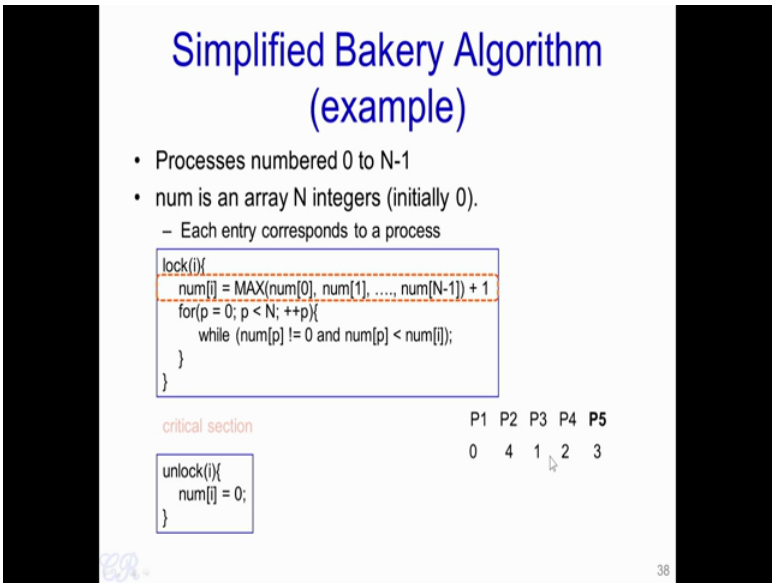
Let us say process P 3 begins to execute. So, it comes here and it finds the CPU finds that the MAX of all these numbers which are present is 0, so num of a corresponding to P 3 is set to 1.

(Refer Slide Time: 06:29)



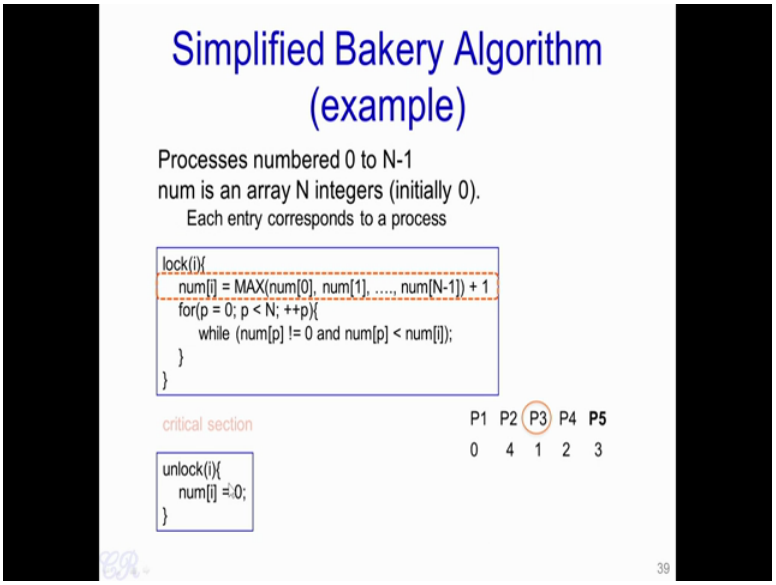
Then let us say P 4 executes and it is get a value of num of 4 equal to 2.

(Refer Slide Time: 06:39)



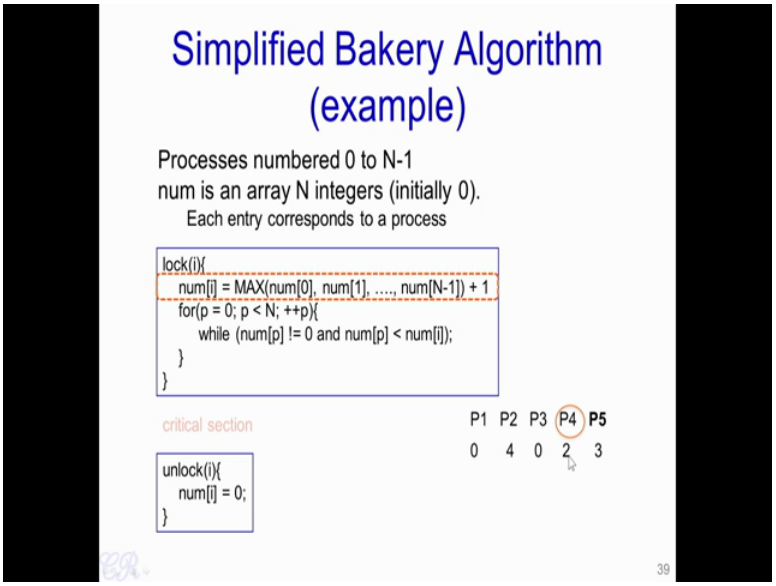
Then P 5 executes, and P 2 executes and they get corresponding values of num as 3 and 4. Now, let us say that we come into this part of the loop, and we see that the process with the lowest nonzero value of num would enter into the critical section. In this particular case, we scan through all these particular values of num and we see that P 3 has the lowest value. Therefore, P 3 needs to execute in the critical section.

(Refer Slide Time: 07:10)



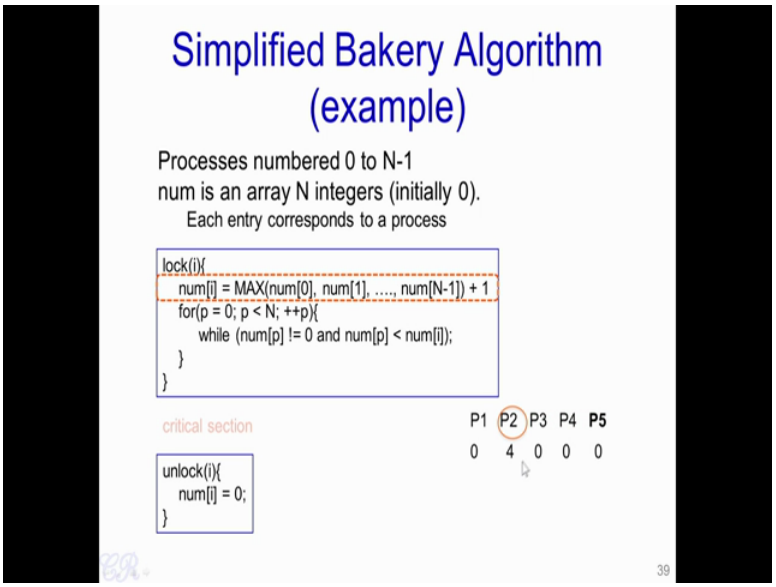
So, P 3 executes in the critical section and at the end of the critical section it sets the corresponding value to 0 then because other processors are also waiting in this particular loop.

(Refer Slide Time: 07:25)



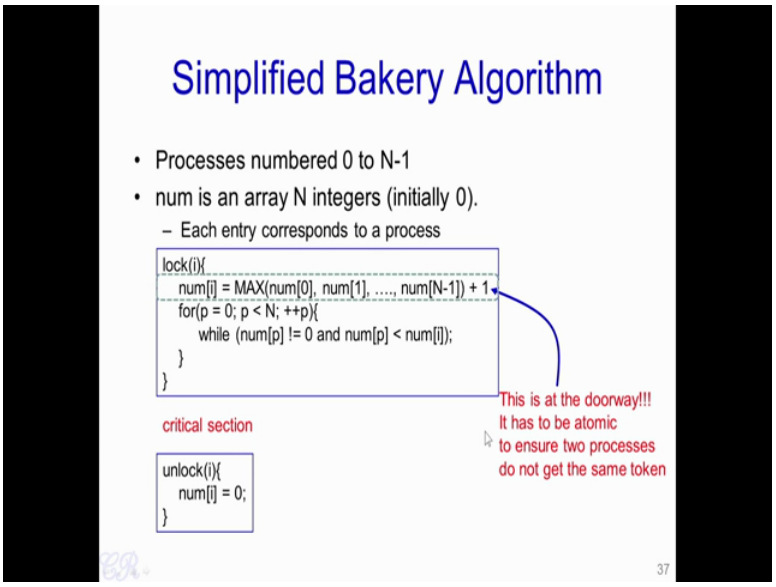
So, the next lowest number which corresponds to the process P 4 and has a value of 2 would get to execute in the critical section. Therefore, process P 4 enters into the critical section; and at the end of it, the number corresponding to process P 4 is set to 0. And then process P 5 executes and then process P 2 executes.

(Refer Slide Time: 07:44)



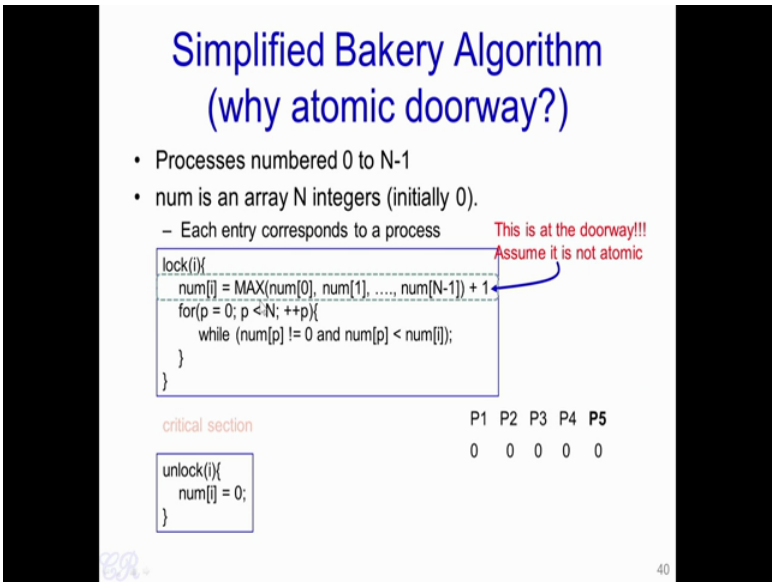
So, process P 2 would execute, because it is the only nonzero number which is present so, at the end of the P 2 execution, we get all values of num which are back to 0.

(Refer Slide Time: 08:06)



So, one requirement or one assumption that we need over here is that this particular assignment of MAX needs to be atomic; essentially this is required to ensure that no two processes get exactly the same token.

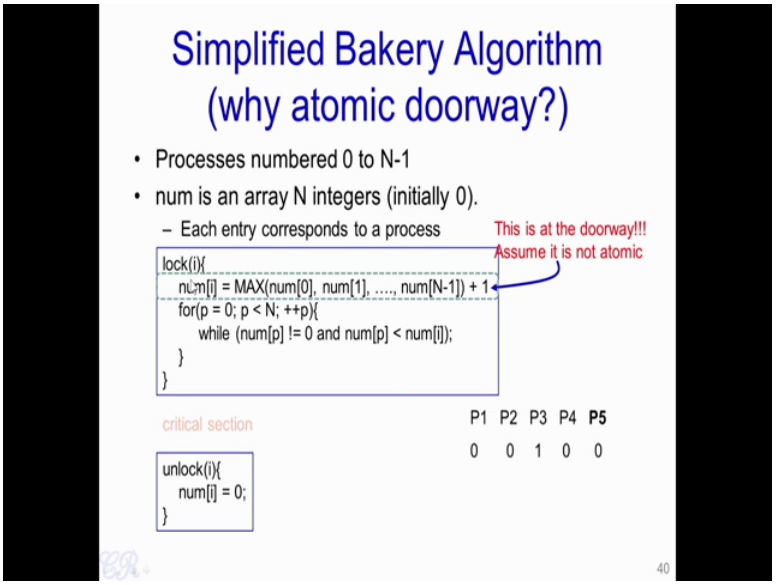
(Refer Slide Time: 08:23)



Essentially, it means that when a particular is executing this particular statement that is find the MAX of all these numbers and adding 1 to it then no context switch can occur.

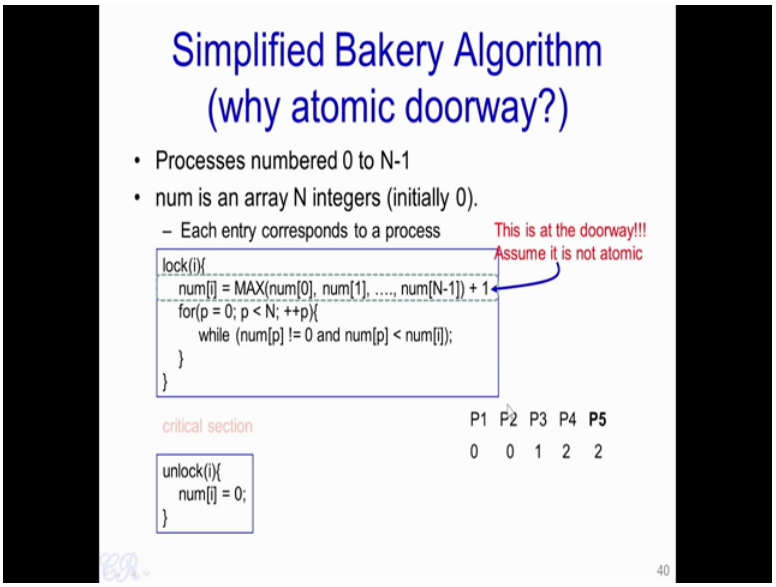
This entire statement executes as a single entity. So, the reason why we make this particular assumption is that we need to ensure that no two processes get the same number. Let us see what would happen if actually have two processes having the same number, essentially what would happen if this doorway or this statement which is known as the doorway is not atomic. So, we will take our example of the five processes and we will look with respect to this particular example.

(Refer Slide Time: 09:19)



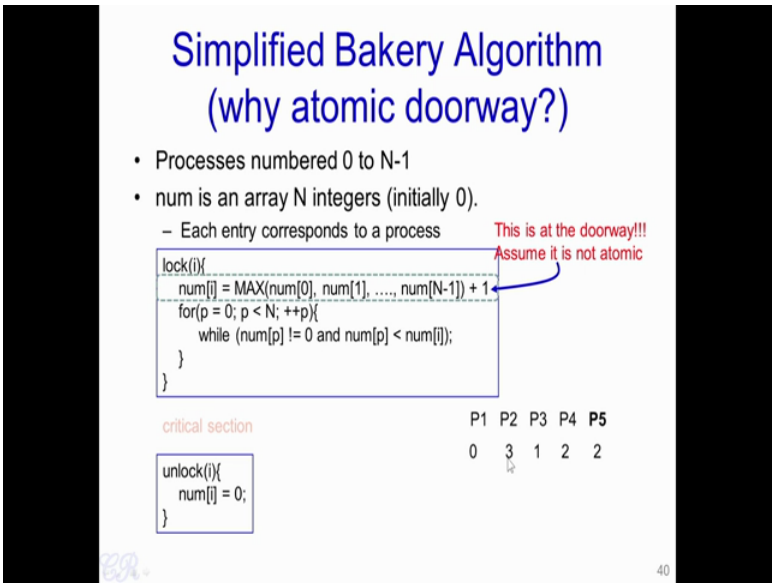
So, as usual let us say process P 3 invokes lock first, and it obtains a number 1, because it is the smallest number all other numbers are 0.

(Refer Slide Time: 09:34)



Then let us assume that process P 4 and P 5 simultaneously execute MAX, resulting in both of them getting the value of 2.

(Refer Slide Time: 09:45)

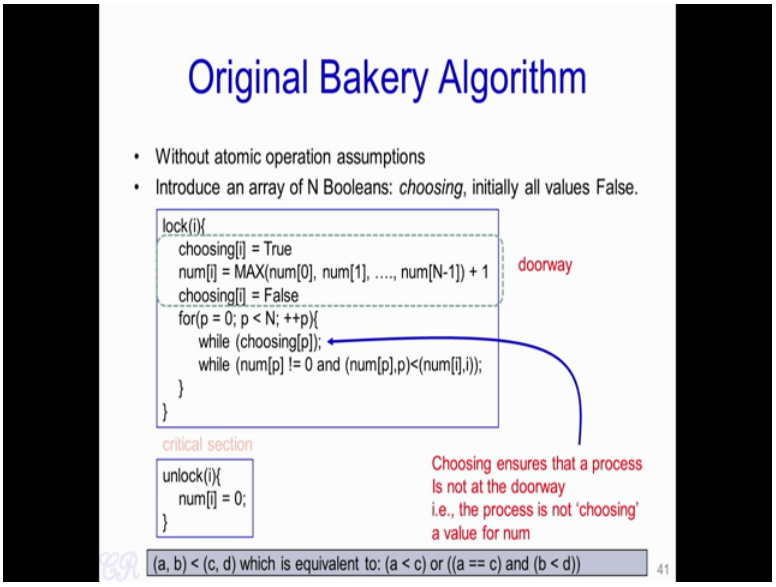


And then of course, we have the process p 2 which gets the value of 3. Now, what would happen in the second part of this lock? So, we would see as usual process P 3 is going to execute first, because it has the lowest number. And once it exit from the critical section, P 3 is going to set it is corresponding number to 0, therefore this number corresponding to P 3 is set to 0. Now next there are two small numbers corresponding to P 4 and P 5

which are equal. So, as a result of this, we have process P 4 as well as process P 5 which enter into the critical section simultaneously.

And thus we do not achieve the mutual exclusion, therefore it is required that this MAX operation is atomic. So, this will ensure that no two processes get the same value for num; and thereby it will ensure that the critical section is executed exclusively by a process at any given instant of time. Next what you are going to look at is the relaxation of this particular assumption so, we are going to look at what is known as the original bakery algorithm where we do not require making this statement atomic.

(Refer Slide Time: 11:05)

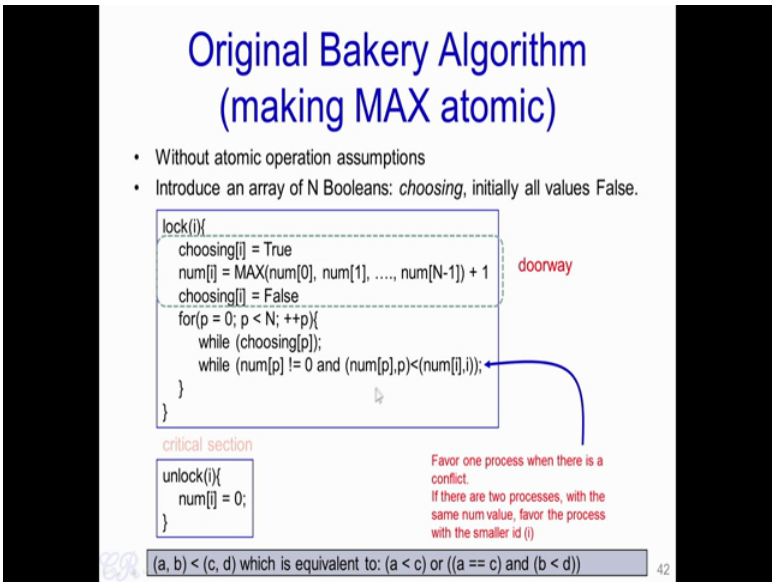


The original bakery algorithm is as follows; in addition to the shared array, num, which is present as before. We also have a shared array called choosing. So, this is the Boolean array and could have the value of true and false; and this length of this array is N that is each process will have a particular element in choosing. So, essentially this particular choosing is set to true, before the process could invoke MAX. And after this particular MAX function is invoked and 1 is added then the process would set it is choosing value to false.

So, there are also some minor changes in the second part of the algorithm. First, we have statement called while choosing of P, which is present here So, this particular statement would ensure that a process is not at the doorway that is it will ensure that the process is

not currently being assigned a new number through this MAX that is the process is not choosing a new value of num.

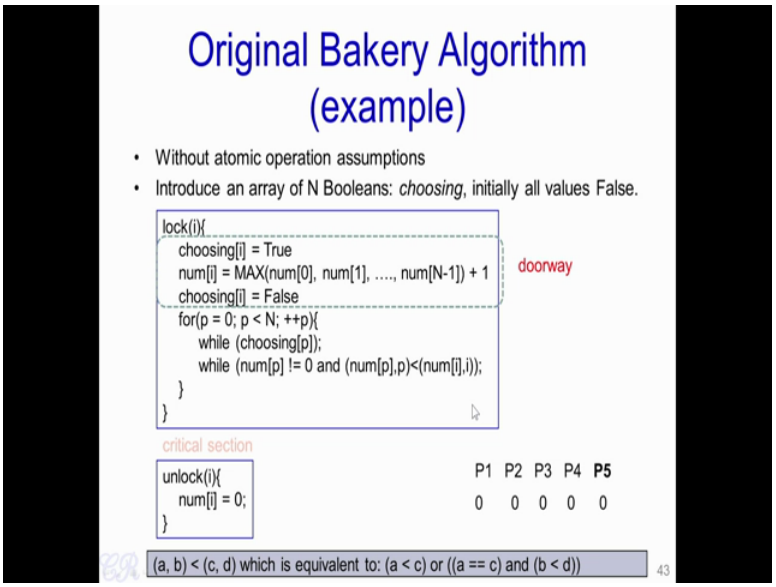
(Refer Slide Time: 12:26)



Secondly, we have the second part of the while loop which essentially changes invoke here in the condition check, which we have a tuple num of p comma p checked with less than num of i comma i. So, what this condition checking means is written over here and it means the following. If a comma b less than c comma d is the same as a less than c, or a equal to c and b less than d.

This particular complex looking check is used to break the condition when two processes have the same num value. So, as we have seen before if two processes are given the same value of the num then this condition is used to going to resolve the issue, and ensure that only one of these two processes would enter into the critical section. When num of P is equal to num of i that is both of the numbers have the same value we need to favour one of the processes In such a case, we favour the process with the smaller value of i.

(Refer Slide Time: 13:44)



Let us see with the same example as we seen before. Let us look at this example again and let us say as usual process P 3 executes this MAX first and is given the smallest number then P 4 and P 5 happen to get the same number of 2 and then process P 2 gets the next highest value of 3. Now, let us look at the second part of this locking. So, the first process to execute in the critical section is quite obvious that is P 3, because it has the lowest number, so P 3 executes. And at the end, it will set the value of num of 3, 2, 0. Now the next process to execute could be either P 4 or P 5.

So, how do we choose between these two processes we have seen that both num of P and num of i are two. In such a case, therefore, in order to favour one of the processes, we look at the second part that is p and i, so, based on this we favour the process which has a lower number, and therefore, process P 4 executes in the critical section. So, after P 4 executes it is value is set to 0, and then quite naturally P 5 executes.

And after P 5 executes as usual P 2 will execute that is we see the addition of choosing the Boolean array over here as well as or more complex conditional check would help resolve the need for an atomic operation of MAX. So, this is the original bakery algorithm which was proposed by Leslie Lamport; and it efficiently helps to solve the critical section problem when the number of processes is greater than 2.

Thank you.