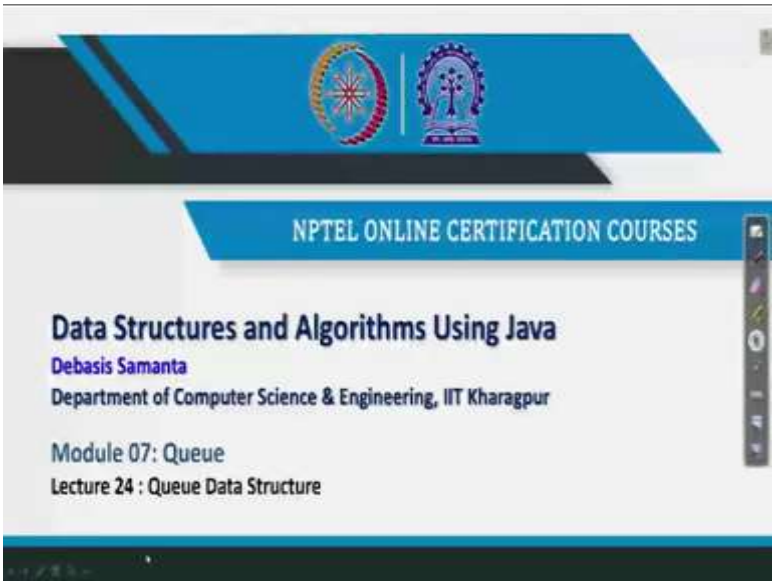
Data Structures and Algorithms using JAVA
Professor Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 24
Queue Data Structure

(Refer Slide Time: 00:33)



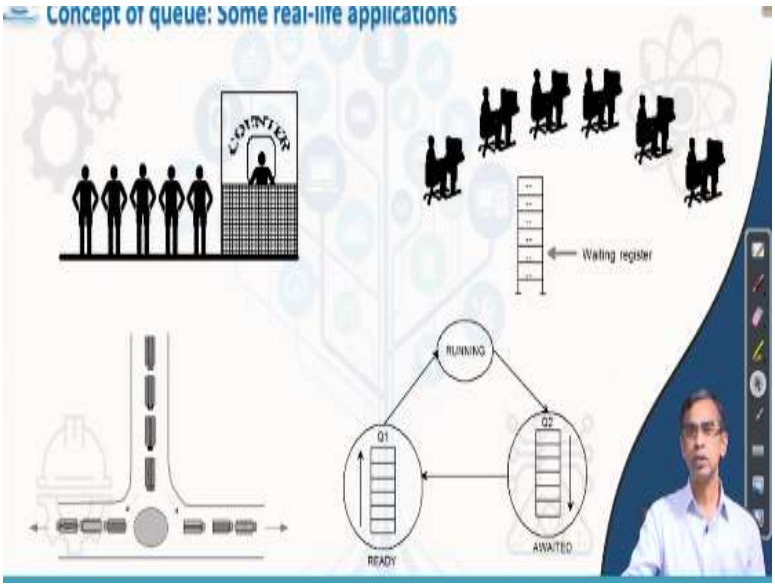
We shall start another data structure it is called Queue. So, today we shall cover about this data structure first and then different operations of queue mainly the operations are called enqueue, dequeue and status it is just like push-pop like and then we will try to cover different types of queues.

So different queue structure those are extensively used in many program design and finally we will conclude this lecture giving hinting some application of queue structure actually. So this is the topics for today.

(Refer Slide Time: 01:18)

Now let us have the concept about queue. The name queue is very common in our daily activity. So, usually if visit any counter so you have to be in queue like this one. So queue is there, so queue means it basically some order.

(Refer Slide Time: 01:39)



So, here as I have mentioned one example so it is basically customers standing in front of a counter actually there is a queue of customers, so this is usual. Now there are some

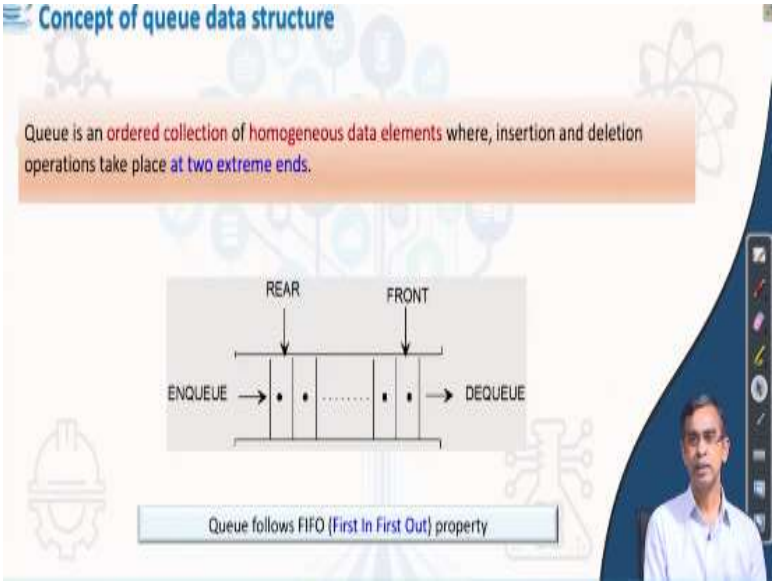
applications also available when you are writing some program particularly in a distributed computing environment where many client programs are running and there is a server which has to serve different clients actually.

So, the client server computing where we have to maintain that which should be, I mean CPU should be given to which client alike. So they are basically all the clients should be queued that mean they are in queue and then server will process each client according to the queue. So you know queue is because first who basically appear first should be served first and that is why for example is called LIFO last in first out, but here queue is called FIFO first in first out.

So who join the queue first will be served first, so this is the concept it is there. Now, so there are many other real life application where the queue is there. For example there is a traffic and controlling the traffic so for a better control of the traffic and there is a automatic traffic controller if it is there they basically allow you to follow the queue concept so that the traffic can be better managed or controlled.

So many applications here and there actually like stack is also extensively used to solve many algorithms. So queue is also very much important to solve many programs actually.

(Refer Slide Time: 03:46)



Now let us decide in what way queue is different from the other data structure that we have learned so far. Now like stack or array or linked list it is also an order collection. So that means all element are in order, order means in which order they added into the list so that is

the order actually. It is not that sorted order or random order. So it is an order collection and like stack, array, linked list it is also homogenous set that means all elements should be of same type.

But there is a basic difference between the stack and queue. Stack allows operations at one end only whereas queue operations, queue allows operations at two ends. So, there are two ends in queue. One end is called the front and another end is called the rear. So, if you want to perform removing an item from the queue then it should take place from the front of the queue.

On the other hand, if you want to add an item into the queue then you should do it rear end. So there are two. So, front is just similar to we can say bottom and rear is basically similar to top in case of stack it is there. Now regarding this removal operation actually it is called the dequeue operations and similarly removing inserting an item into the queue this is called enqueue operations. So this is a two name is given instead of pop and push like.

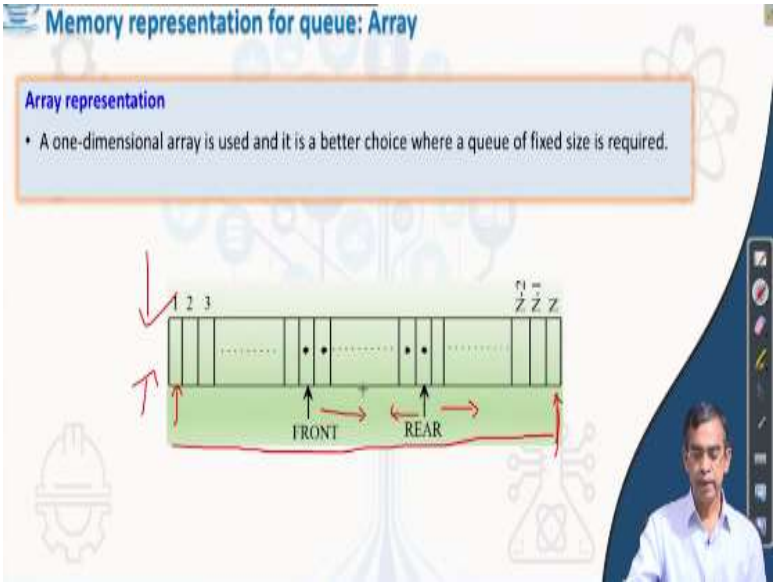
Pop and push are basically is a marked operation for stack, it is not usually followed to mark operation for deletion and insertion in case of queue or other structure like. So it is customarily use the different method to make it distinguish that this operations are basically is a standard operation or marked operation for a particular data structure.

Anyway so insertion and deletion namely the enqueue and dequeue these two operations can be done at two ends like rear and front. And as it allow to remove an item based on the property or policy called first in first out this structure is also called the FIFO structure just like stack is called LIFO structure it is called the FIFO structure.

(Refer Slide Time: 06:37)

Now, let us come to the discussion of how a queue can be stored in a memory for your program. Now you can use either array or linked list anyone. Array is very convenient, very simple so you can have it, but array if you use array suffer from the limited size. So that is why it is little bit it is encouraged to have it, but linked list represents the best representation by which a queue can be maintained.

(Refer Slide Time: 07:12)



So, today we will discuss about first the array implementation of the queue or array representation when the array is required. Now here okay in this figure as you can see this is a total size of the array that we are considering. Now because of the operation was going on

so it is not that front is always here and rear is always here because these two pointer can move both way I mean this front always move in this direction rear also moves in this direction.

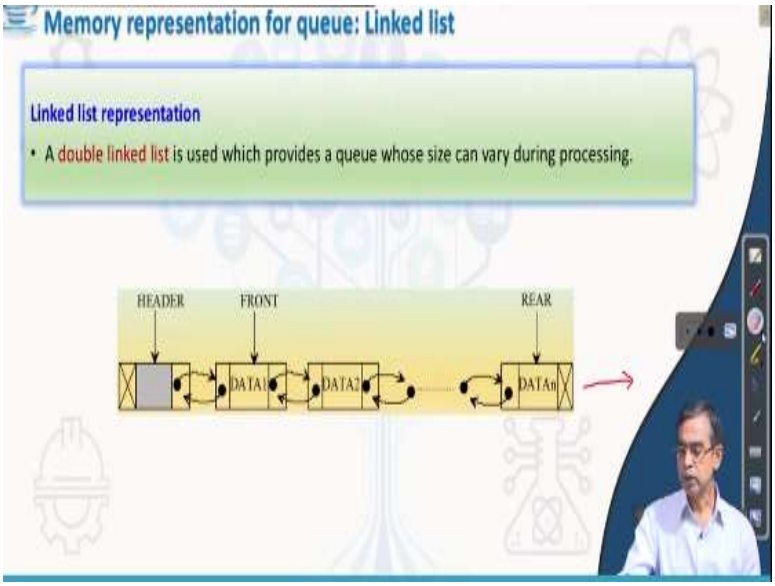
This is because I am sorry the front will move in this direction rear is move in this direction it is basically the idea it is there. Now so while you okay at the very beginning you can start while you are starting the operation both are basically front and rear are here and when queue is full then you can see front is here and rear is here.

So, that basically gives you an idea about whether queue is empty or it is full that can be decided by knowing the location or pointer or counter value of the front pointer and rear. So front and rear are the two pointer to indicate the front most element and rear most element on the queue, you understand? So, this is a concept it is there and this is an array of size queue.

So we can say total capacity of this queue that means using one dimensional array representation is N actually. Now sometimes it may have that the front and rear are same so it indicates that there is only a single element there, but this element can be at any position of the array. So those are the very things to be taken into care when you consider about array representation.

But all those things are not coming into linked list representation because it is most convenient way anyway but array representation is simple so that is array structure is there, but it has to be taken many things while you implement the different operations on it.

(Refer Slide Time: 09:50)



Now so I told you linked list representation is the most suitable for a queue representation now again question is there whether single linked list or double linked list? So, here the idea is the double linked list should be followed to represent I mean to maintain a queue because here you can see both the pointers front and rear should move both direction, for example, whenever you want to add it.

So go on adding list in this direction so basically when you go for enqueue so it is basically enqueue is say equivalent to insert at end if end is the end of the list actually. So this is the idea and when you go for removing the removing is basically delete at front. So this is the two operation for the linked list is sufficient to maintain a queue and if you want to move here in this direction both direction so double links is there.

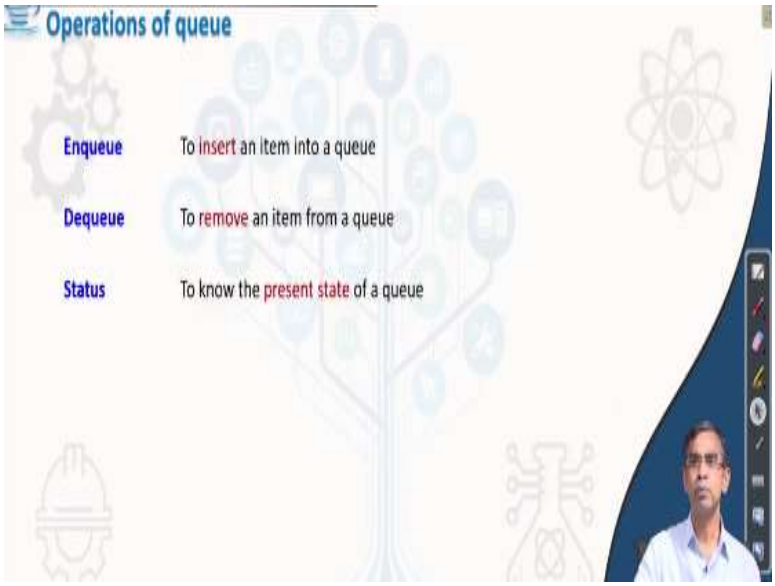
Anyway so single linked list or double linked list, double linked list is better that is why our developer follows for stack queue implementation double linked list, but any single linked list also can be used, but you have to be taken care about how the movement if you want to do the backward movement and everything you cannot do that.

So double linked list or single linked list that is not an issue actually but linked list is an important one data structure that also can be considered to build another data structure called the queue. And here operations are two end rear and front that is end and front. End actually the insertion can be taken place and at the front you can make the deletion of the item it is there. So this is the idea about the linked list representation of queue.

(Refer Slide Time: 11:48)

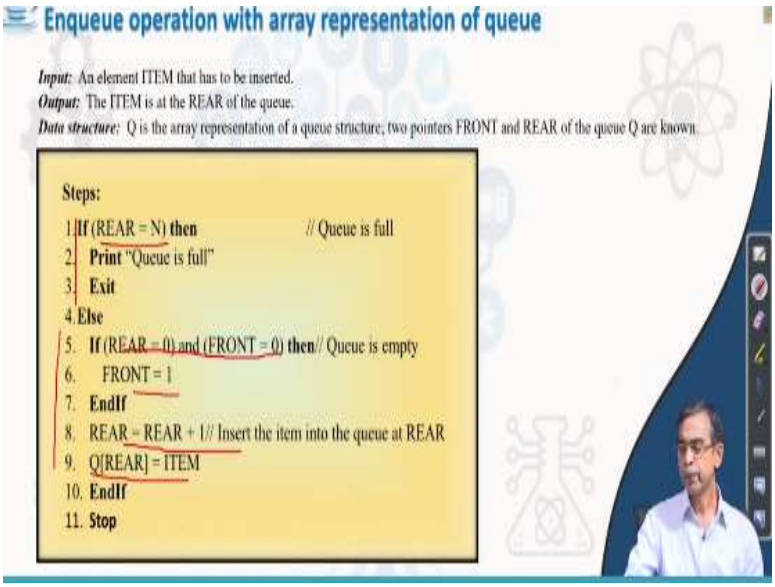
Now we will discuss about operations of queue. First, we will discuss queue operations using array and then finally we will be there in the linked list.

(Refer Slide Time: 11:59)



As you know so operations are pretty simple, operations, pair of 3 types of operations like enqueue, dequeue and then status, status representing size, availability, empty all these things are the same thing just like stack is only there. So, enqueue is basically insertion decrease for deletion from the item from the queue.

(Refer Slide Time: 12:20)



Now I will just give an idea about array which is very pretty simple. So you just simply pointer, two pointers are to be maintained there. So, the pointer is called the front and rear. Rear pointer is basically says the location of the array where the elements are to be added and then the front is basically from where the element are to be removed.

So this algorithms is given to you, you can just follow this algorithm how enqueue operation can be there. So, before doing the enqueue operation you have to check whether we have already achieve to the full of the array actually full of the array is basically I mean full of the queue is basically marked with whether rear position come to an end of N that means that is the total bottom of the array you can say.

So if it is there then we can say that queue is full. Now here you can see queue is full does not mean that all the elements in the array are occupied by the queue elements. Because say suppose in some situation rear is at 5 and sorry the front is at 5 and rear is at 10, so we can say that 6 elements are there.

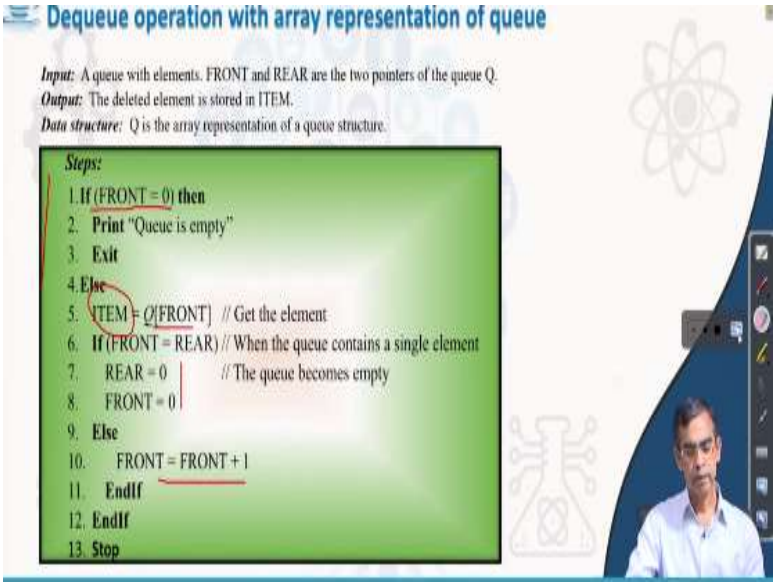
But suppose total capacity of the array is 10 so rear is at 10 and front is at 5 this does not mean that array is full because 6 elements are there out of total 10 elements, but 4 elements are somehow we will not be able to add because if some elements comes there then we can go on adding it is like this.

Now if it is full then definitely the enqueue operation will fail otherwise this is the operation that you can do it if rear equal to 0 and front equal to 0 this basically express one extreme condition that it is basically queue is empty and then in this case front equal to 1 and rear will be rear because it is 0 because there is no elements are to be there.

Anyway front equals to 1 rear equals to 0 in that case. Now once you do insertion there so front will be increased by 1 and then rear also will be incremented by 1 because insertion takes place and then if you want to go for removing an element that need to perform the dequeue so dequeue will take place from the location 1 only.

And here we are assuming that array index is starting from 1 to N like if you 0 then accordingly the index and assignment will be there and here is actually queue rear is item, item to be inserted there. So using this array and if you follow this algorithm and you can go on inserting elements into this (()) (15:20) you can see it actually how it works. So, this is the enqueue operation.

(Refer Slide Time: 15:33)

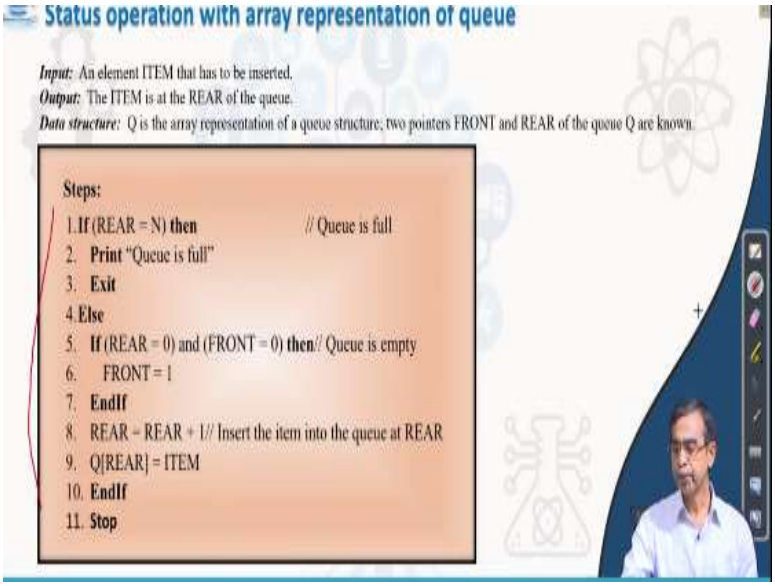


Like enqueue the dequeue operation is also there and this is the algorithm for dequeue operation just similar to here. Now before performing dequeue operation you have to check whether the queue is empty is not. So queue is empty when front comes to 0 that means there is no more elements to be removed here and if front equal to 0 then queue is empty otherwise you can come here queue front this basically empty the elements to be removed so take the read the elements at the front location.

Let the element be item and then if front equal to rear that means you know so rear position and front comes to the rear we can say that element the stack becomes (exhaust) queue become exhausted. So if it is exhausted then we can set the rear equals to 0 front equal to 0. So again queue will start from very beginning like this one. You can consider the situation of queue in front of a counter like it is like that.

And then otherwise you can see front can be increased by front equal to 1 so front go on incrementing this one and this is the end of the things it is there. So here actually idea about this one that you can do it how the dequeue operation can be take place. So this is the dequeue operation.

(Refer Slide Time: 17:04)



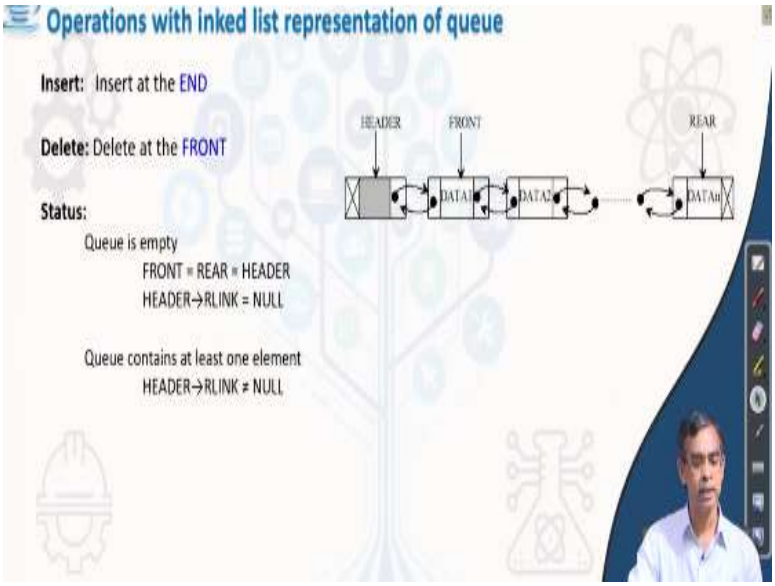
And this is the operations with array the another operation is size, empty and everything just only front and rear pointer everything simple arithmetic that you can follow to check that whether a queue is empty or queue is full or queue contains N number of elements etcetera. So those are the simple arithmetic for the queue having the location of the front pointers and rear pointers.

So this algorithm you can follow to understand that how it work. Now let us proceed with the discussion about for some other so linked list based implementation. We have discussed about array implementation.

(Refer Slide Time: 17:52)

Now linked list using linked list how the queue can be maintained let us discuss about it. As I told you, so if you use linked list and if you want to perform insertion then insertion should be at end and if you want to perform dequeue operation, so dequeue should be at front and this way we can ensure first in first out what is called the policy.

(Refer Slide Time: 18:28)



Now here is an example how the insertion at front and insertion at end and deletion at front can be done. We have already learned about while we are discussing about the linked list structure. So these are the few pointer manipulation that you have to do in different situation

like queue is empty and you can see that knowing the front and rear pointer are actually you can check easily whether queue is empty or queue is not empty.

And in this case full condition never achieved because it will dynamically go to infinite length until memory is not available like. So this is the case that you can think about whether the queue is empty or queue is full in this structure and I do not want to discuss about the inserted front or deleted end all these things are there that is obvious insertion we have already experienced with while we are discussing linked list structure given the double linked list also no issue.

So this is about operations with linked list structure representation of the queue and this is the status of the queue and they are basically indicated by the front and end pointer or front and rear pointer you can say.

(Refer Slide Time: 19:42)

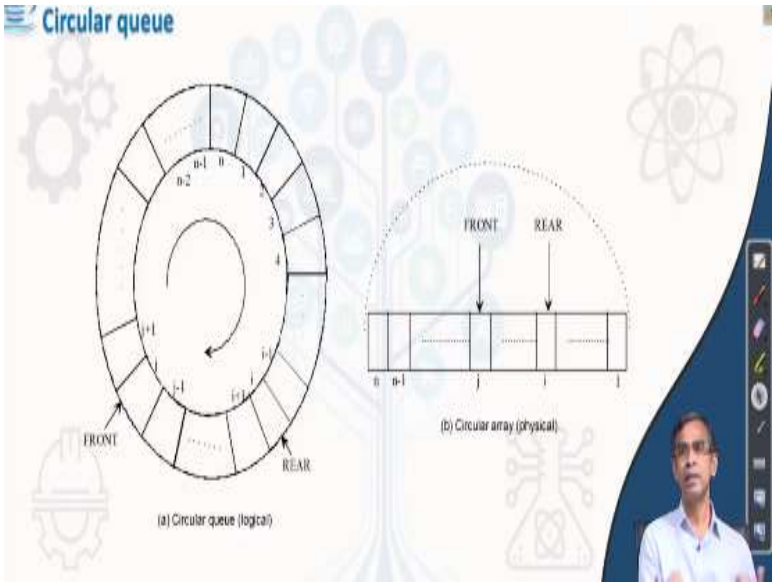
Now there are many variations of queue structure those are for different applications it is used we will discuss about the different varieties of queue there.

(Refer Slide Time: 19:55)

Let us first discuss about circular queue. Now circular queue is basically is an okay you probably know circular linked list. So, it basically from an end if we want to go to front then actually the last node that is the N most node should not include the null linked field rather it will point the header node or first node you can say, say header node. So this way a circular link can be maintained.

Now if we follow a circular link for example for an array in order to implement the queue operations then such a queue is called circular queue.

(Refer Slide Time: 20:49)



So typically a circular queue will look like this. So circular queue you can note as I was telling about say front is at 5 rear is at 7 it indicates that only few elements are there. Now suppose front remain in the same and then go on adding that means rear is increasing. So one time rear will come to an end where N is the total of elements available I mean size of the array.

So after that you see front in at the front for first 5 elements I mean 5 locations rather or 4 location you can say is remain empty but still if we want to add few elements we will not be able to add because stack is showing, I mean queue is showing as full because rear reaches to an end, but it could be occupied those are non-empty location or non-occupied location which are at the front position.

So the idea is that whenever rear reaches at this end, but still at the front some elements are available we can move the rear to that and go on adding this one. It is same also front also so this way front and rear instead of going to an end marking it can loop to a circle. So if we follow this kind of logic in order to represent a queue then this structure is called queue structure and logically a queue will look like this is a circular ring like.

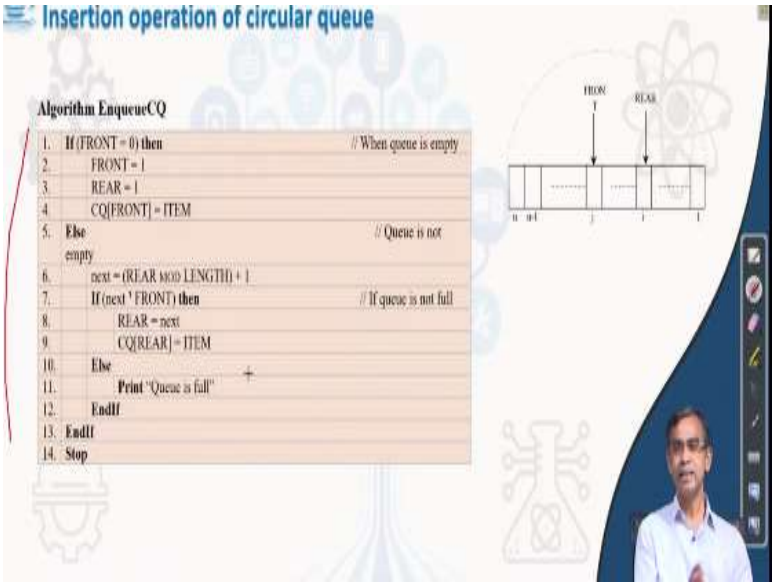
So the queue will go on and for the circular queue now you can decide whether a queue is empty or not. So, queue is empty if rear and front both are say minus 1 otherwise queue will be containing some elements and the number of elements will be decided by the simple arithmetic between rear and front pointer calculation. So this is basically the idea about it and that is the circular queue concept it is there.

So circular queue concept is basically used for the computer science students as a puzzle like because how we can do whether insertion, where it will go so lot of questions particularly if you are preparing for GATE and everything so many question usually post-related the circular queue concept because it is little bit complex.

Complex in concept, complex in pointer manipulation and everything but anyway once you understand about it logically a circular queue is just like a circular ring sort of thing, but only the thing that movement of pointer needs to be taken care in what way in what direction they can move and how many elements are there and which is a current location of front and rear those are basically systematically you have to follow.

And if you do it your program will become very easier actually. So this is about concept of circular queue.

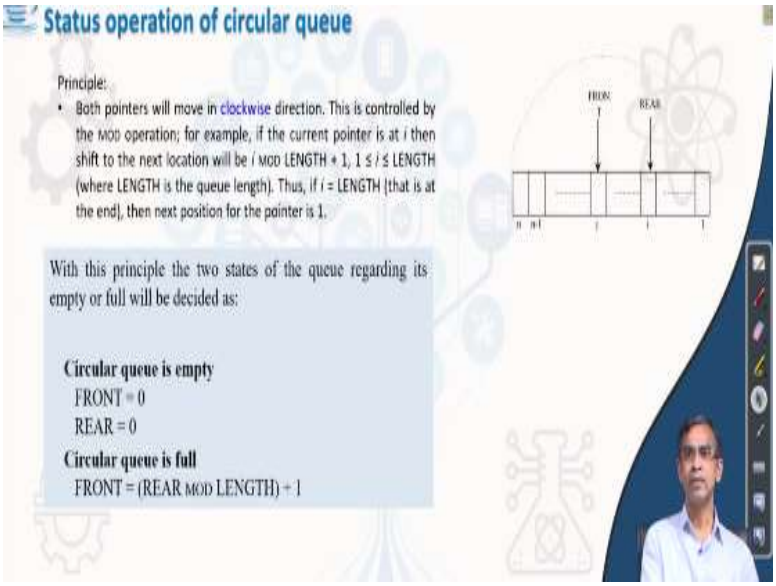
(Refer Slide Time: 24:00)



Now operation of circular queue you have to have the enqueue and dequeue. I will not go for detail discussion about it you just given an idea bout algorithm here I advise you to check the algorithm and try to implement by writing program and then you see whether this is working in a circular fashion and if you implement this algorithm for the circular queue it is called the enqueue circular queue CQ and then you can compare the enqueue operation for non-circular queue representation of array.

Then you can check it how it is working there and there and so this idea about I just left as an exercise for you to have it the real programming and you can check it. So, this is the circular queue and the next varieties of queue circular queue enqueue and dequeue operation just a dequeue operation you can follow.

(Refer Slide Time: 24:56)



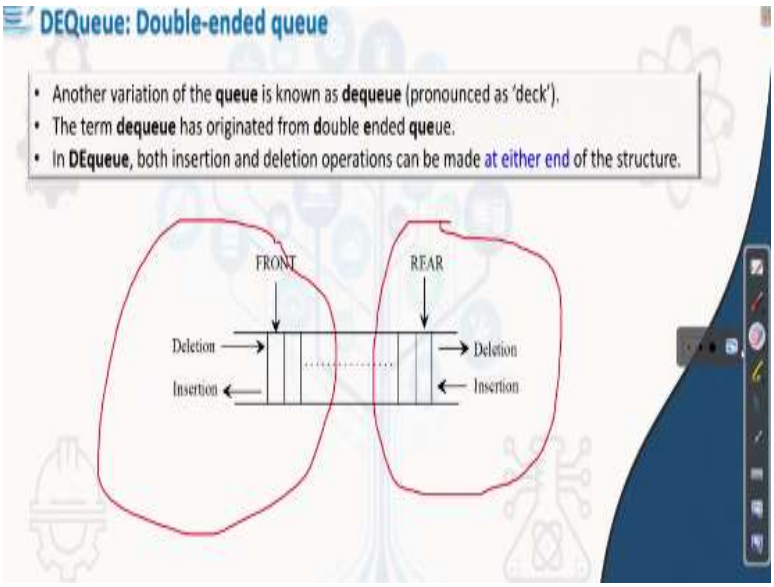
Status operation also how many elements are there if given a rear probable location and front location like in a circular queue simple arithmetic calculation you check that what should be the arithmetic calculation between the two and everything so it is just like rear plus front plus 1 minus 1 whatever it is there you check it and then you can find how many elements present there.

So you can write on pen and paper also practice it just calling some enqueue and dequeue operations at random and check it I will give the programming in the next session of implementing all these things then you can understand that how a queue can be implemented using JAVA programming and that idea of implementation you can extend it two different types of queue actually and then you can have the idea about it.

(Refer Slide Time: 25:41)

Now so this is the one about one type of queue it is called the circular queue other than circular queue some other type of queue is also has some importance they are called double ended queue.

(Refer Slide Time: 26:01)

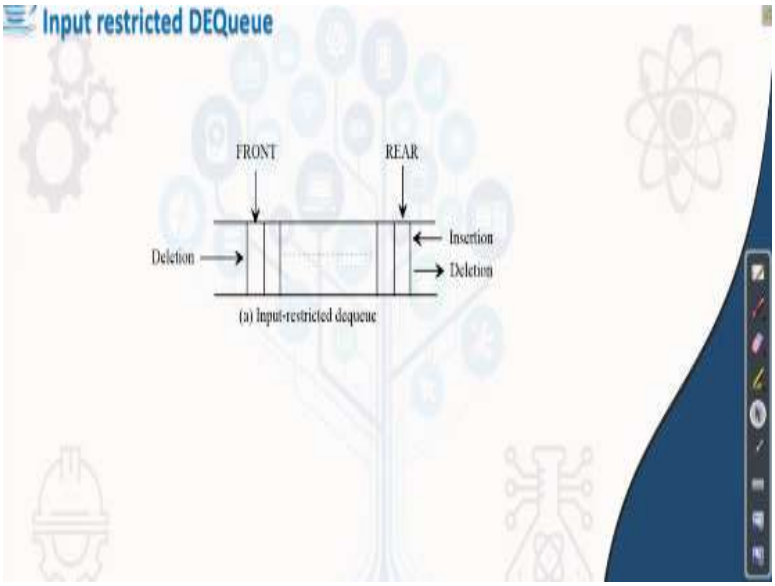


The concept double ended queue also it is called deck coming into the concept is that now in case of ordinary queue that we have learned slower the insertion can be taken place at rear position and deletion at the front position, but double ended queue is a special type of queue where both insertion and deletion can be taken place from either end from front and rear end. So if it is like this then it is called a double ended queue.

And in this picture as you can say this is the rear end where both the insertion can be taken place here and these are front and here is there. Now there may be some reason some situation where you have to have the insertion and deletion at the both end that is why it is there. It look little bit chaos, but it is actually not chaotic things in that situation because in many application it takes place.

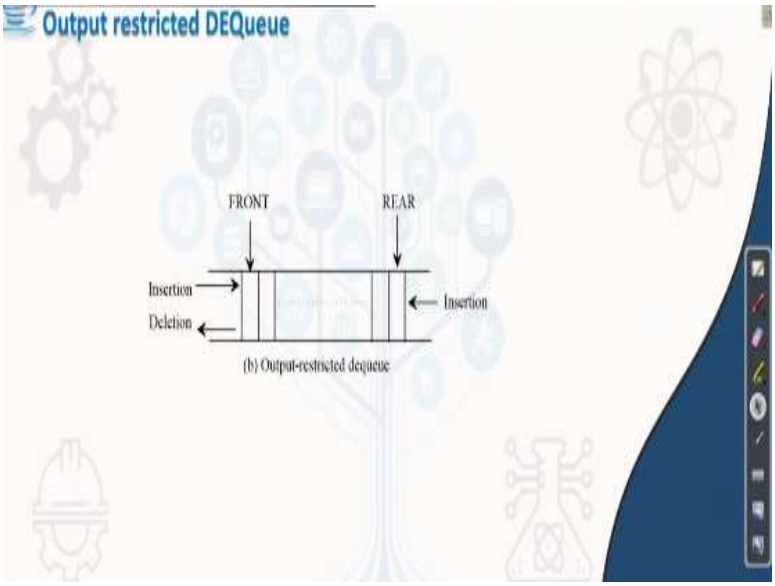
But although it is really chaotic I understand that it is chaotic so in order to make it little bit stringent, so there is computer science concept it is there let us restrict to the insertion deletion okay let us allow it, but not that both the operation at both the end.

(Refer Slide Time: 27:19)



So this way we can have some restriction the restriction can be in 2 ways one is at input restriction and then output restriction. If you restrict the input then it is called the input restriction that input restriction means we do not allow to do the input at the front only deletion will be there and then at the rear end both insertion, deletion can take place. So if it is like then it is called the input restricted dequeue.

(Refer Slide Time: 27:50)



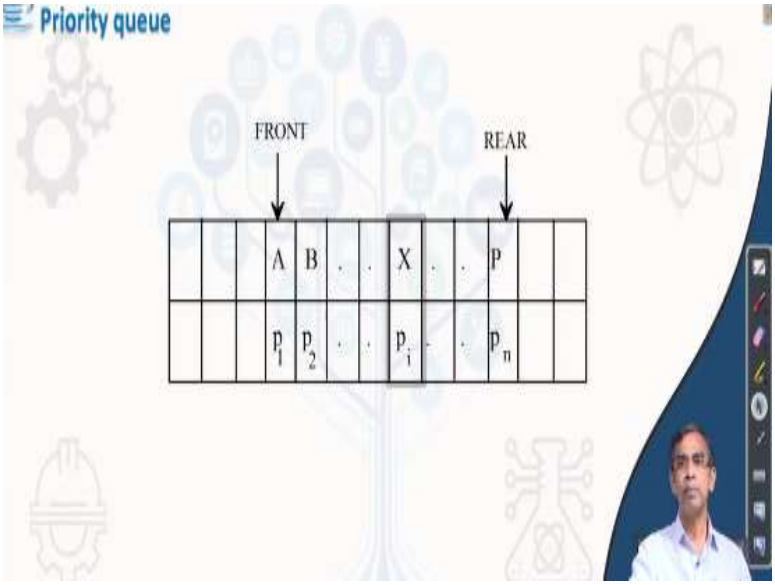
On the other hand there is output restricted dequeue as the name implies you are allowed to do insertion only at the rear end. However, you can do insertion deletion at the both end. So if you do so then it is called the output restricted dequeue. Now obviously there are certain rational behind this. I do not want to discuss exactly why all those input restricted dequeue and output restricted dequeue. For all those things you can have the detail discussion in several textbooks on the subject you can find it.

(Refer Slide Time: 28:26)

Anyway other type of queue is called the priority queue. Now priority queue is also like random queue sort of thing because here in this queue this queue structure policy will be

allowed to remove from any location of the queue actually. Now if you do it then it is the priority queue.

(Refer Slide Time: 28:48)

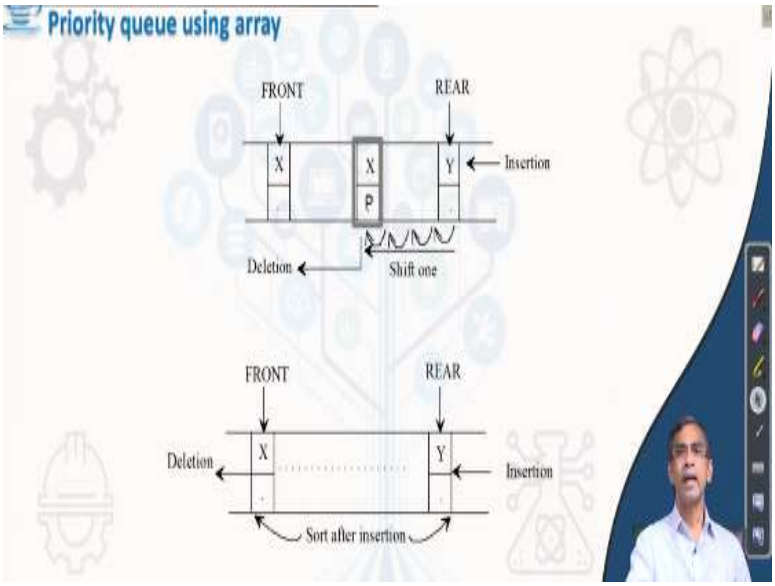


Now again the problem is that in which location you will be allowed to remove an item. So for each item there will be a priority to be assigned. So the item which needs to be removed not necessarily from the rear pointer, but it is from anywhere in between front and rear depending on this priority value that means highest value of the priority will be removed first so this is the concept.

And it is little bit if you see in that stricter sense or stringent sense that FIFO structure will be violated because it is not strictly in the first in first out manner actually, but it is in a priority manner. Again, if the two elements are having the same priority and they are of highest priority then which elements are to be removed. So this element should be removed again first in and first order.

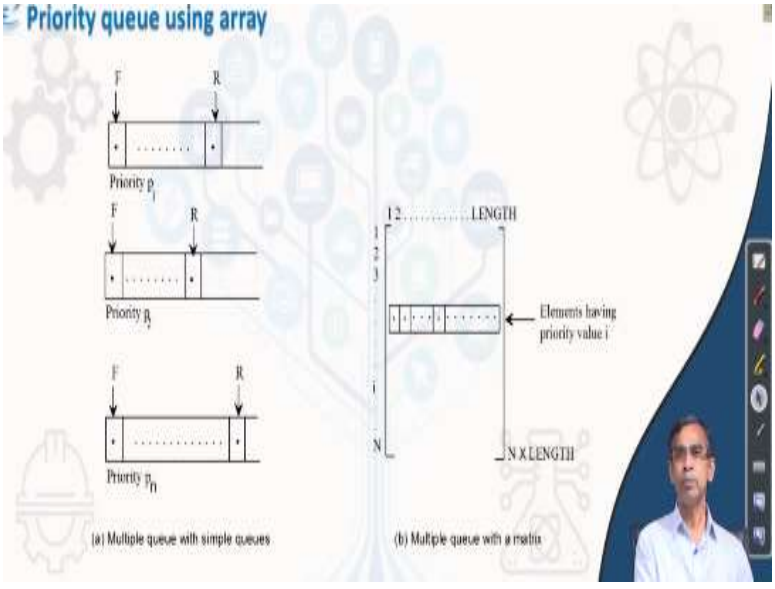
So if the two things are of the same priority, but one is who came first that will be that means which element is basically in the rear most will be removed first actually. So this concept is called the priority queue. So, priority queue as you know implementation of enqueue and dequeue operation become complex, but if your logic is clear and then all those things will not be a difficult things it is there.

(Refer Slide Time: 30:07)



And for priority queue implementation if you use array and everything once you remove one element from the array as it is from in between intermediate intra-element we can say then lot of movement have to be followed. If you remove it or if you add one element according to the priority add it so lot of movement it is there. So that is why array based representation to implement priority queue is not a good idea linked list may be a good idea.

(Refer Slide Time: 30:37)

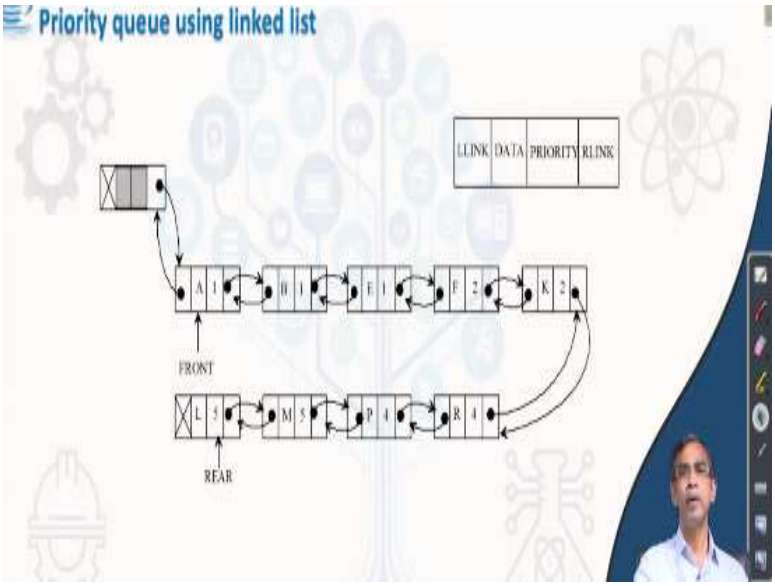


But there is a better idea that is available using matrix representation. What is matrix? Matrix basically maintains a number of arrays actually. So each row basically maintains the priority for each. So here one to end different priority. So, if an item reaches to a queue and which

having priority I so it will hit to the Ith array and each array again maintain rear and front indicating which thing is there.

So this is one good solution that matrix representation of priority queue is good, but it has limitation is that sometimes an array can exhausted and if it is an exhausted and a particular element of a having a particular priority queue for which that array is exhausted you will not be able to add it so this has limitation.

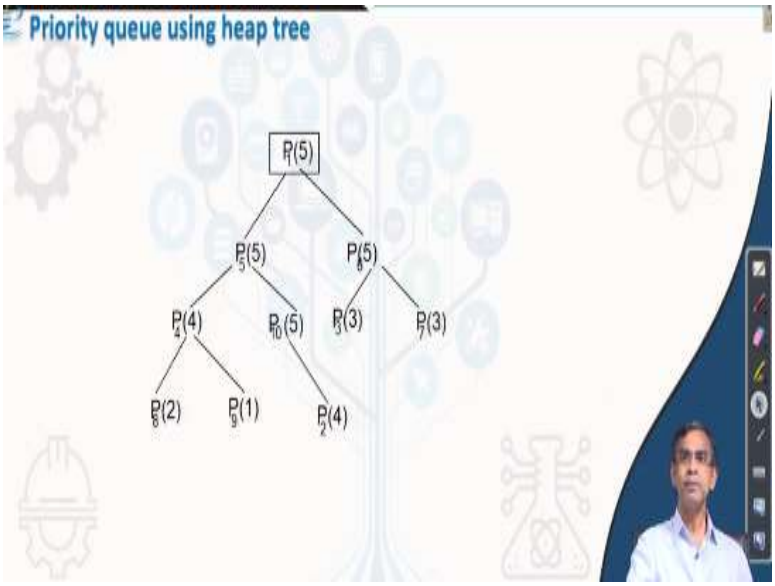
(Refer Slide Time: 31:34)



Now there is a better idea obviously using linked list because it dynamically grows it can add, but it can add according to the position that means whenever you want to okay remove can be rear and front, but here priority whenever you want to insert, insert at the end and that insertion in the order that since that according to the priority value it will be inserted. So it is basically element will be inserted in the sorted order of their priority value.

So this means that those are the elements which are the highest priority they will be towards the front position like. So front to rear if you sort it according to the descending order of the priority value it will work for you to implement priority queue.

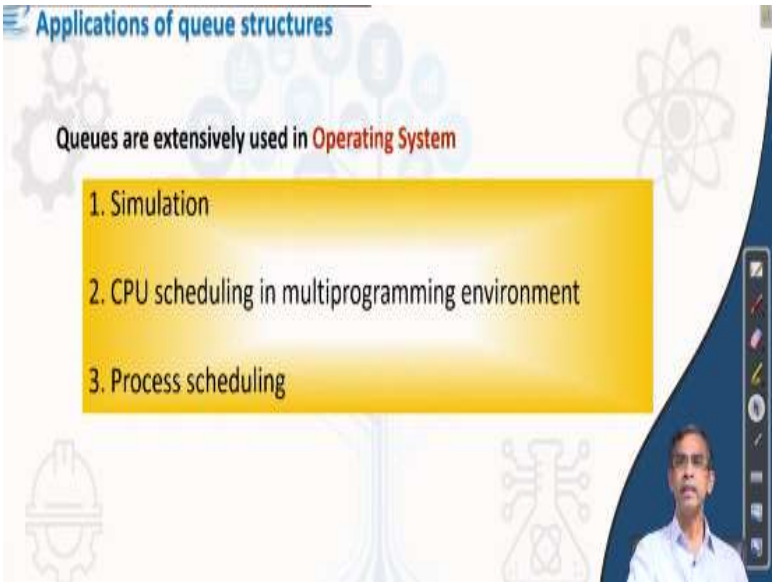
(Refer Slide Time: 32:25)



Now there is a better one structure which we will discuss in details while will discuss another data structure called tree. So using tree based representation a priority queue can be implemented as this concept will be discussed in details later on. So, I just want to skip it for time being I do not want to discuss in detail here, but I want to say that priority queue is basically is a concept of heap and heap can be stored in an array actually.

So, a priority queue means heap or priority queue is basically array based representation, but it is again priority based queue implementation where an element will be removed or added according to the priority from the queue that will be discussed in detail later on.

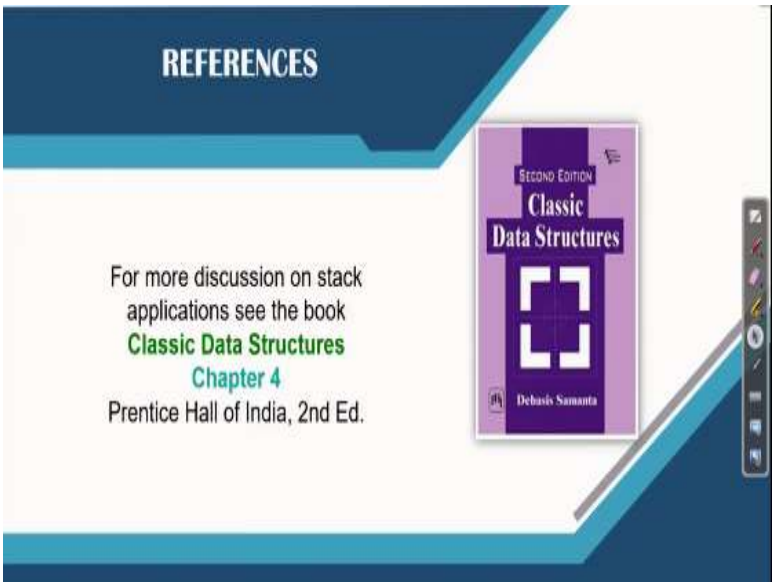
(Refer Slide Time: 33:10)



Now application of queue let us come to the application I will not discuss the details about the applications here. But there are many applications those are basically for system level programming if you want to write program for compiler or operating system or game design or some other window programming and everything or client server computing there you have to follow queue structure like say simulation which is very much common in mathematical research or research oriented study there you can use queue.

CPU scheduling which are required for client server programming or distributed computing or multithreading you can use process scheduling again this is multithreading programming or concurrent programming in client server environment you can use it.

(Refer Slide Time: 33:58)



And all those discussion really very advanced level so I do not want to discuss in this course also beyond this course there you can have more idea about this queue structure in details because I could not cover so many things in details because the topic is too exhaustive. So I should suggest you I recommend you to go through this book where a detailed discussion about the queue structure you can find in chapter 4 and you can learn more from there.

And in the next video, we will discuss about how a queue can be programmed using JAVA because JAVA programming can be little bit hands-on for you to have the details idea about it. So that will be the next video lecture we should discuss. Thank you very much.