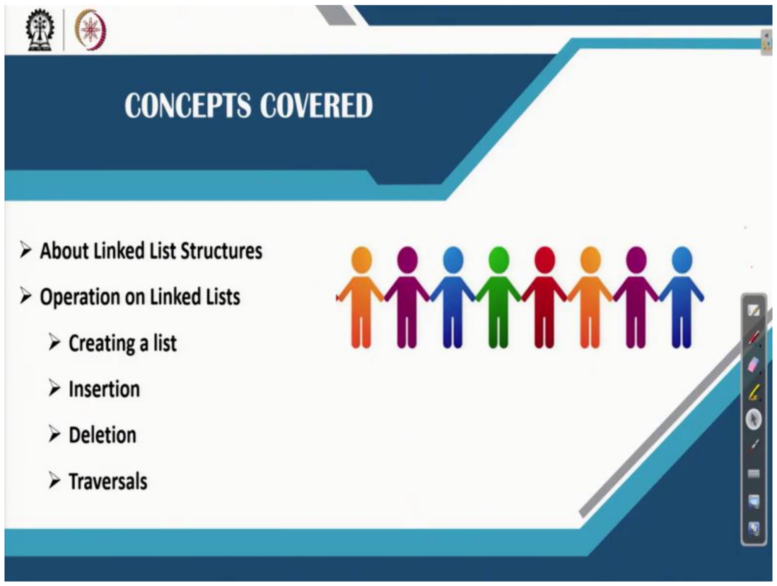
Data Structures and Algorithms Using Java
Professor. Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture No. 16
Linked List Data Structure (Part - 1)

Till time we have covered 4 modules and each modules contains the different topics like say, genetic programming, collection framework, arrays and then array data structure like. So, our next module it is the fifth module. In this module we will study another new data structure. This is very popular data structure which has many significance so far the real life application development is concerned it is called the linked list and in this module first we will try to understand the concept of these data structures from the theoretical point of view.

And then we go for a programming practice of this, then how this linked list can be program. That means, you can write your own program to implement this structure, linked list structure. And finally, we will see how the Java Collection Framework supports to implement the linked list. So, the 3 aspects. Now today, in this lecture, particularly, we will discuss about how the linked list data structure is there, which is very popular one data structure for the computer science students actually.

(Refer Slide Time: 1:47)



CONCEPTS COVERED

- About Linked List Structures
- Operation on Linked Lists
  - Creating a list
  - Insertion
  - Deletion
  - Traversals

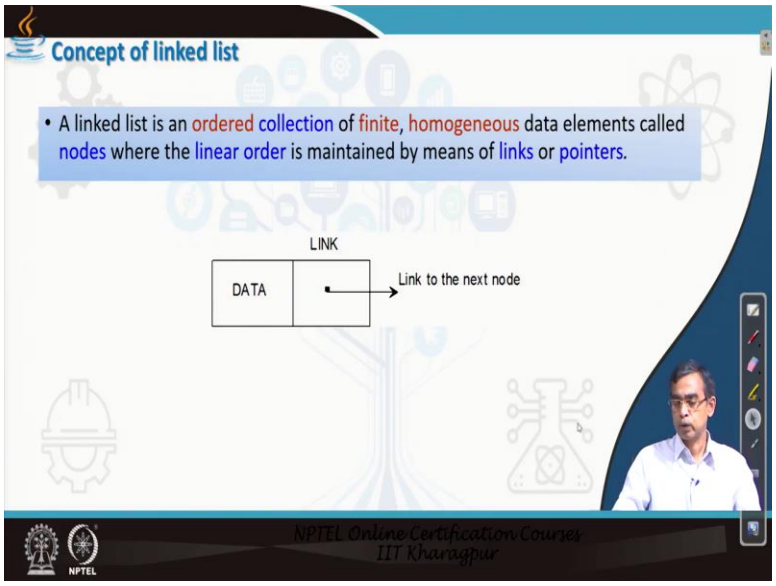
Now, in this, in this video lectures, will try to cover basic about the linked list structures and then the different operations on it. Operations like user operations like creating a list, the insertion operation, deletion operation, and finally the traversal is basically visiting all the elements those are there in the linked lists. So, this is basically our today's agenda. And now, let us first learn about the linked list structure.

Like array, linked list is also linear structure. Linear structure means it basically maintain an order that is why it is called an ordered data structure or is called a sequential data structures, but there is a different compared to array structure, the linked list is, the difference is that this linked list in case of array all elements are sequential, but in a continuous location that means in memory they are placed one after another in the memories.

But in this linked list structure, the elements are not necessary to be any continuous location in the memory, they are in random, wherever it is there. So, one good advantage of the structure compared to array is that, if you want to store say, 1000 elements and for which is the contiguous memory location is not possible, then we will not be able to work with these array.

On the other hand, linked list structure as the memory is in a distributed ways, so, here and there wherever, it is possible even the memory at one chunk if it is not possible to accommodate 100 but all together in different locations if you see then you can do it. So, this is the one way of that is ordered elements, but it is not in this contiguous location like that array it is there.

(Refer Slide Time: 3:44)



Concept of linked list

- A linked list is an ordered collection of finite, homogeneous data elements called nodes where the linear order is maintained by means of links or pointers.

Now, that is why the different the definition the basic or fundamental definition of a linked list is that it is an ordered collection like array, it is a finite. Of course, you cannot include infinite number of element. Although like array, you can contains any number of elements, but unlike array again it can grow automatically is a dynamically.

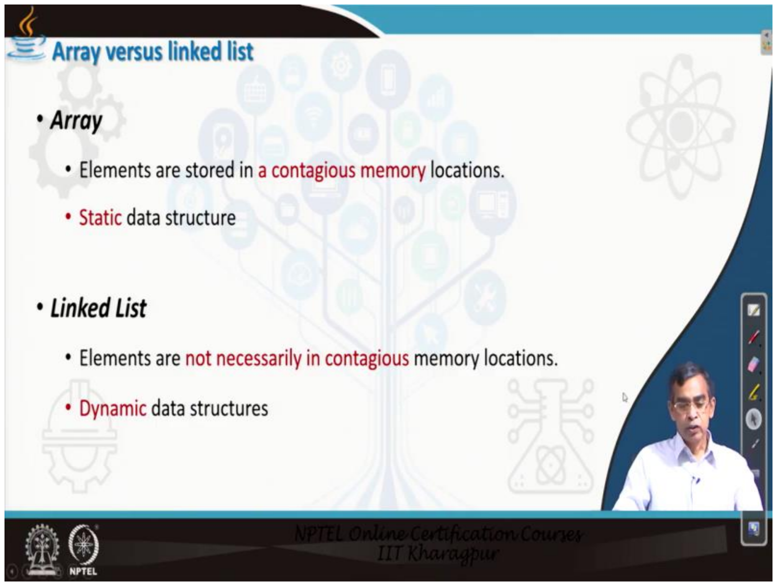
So, this is the one good advantage compared to the array and it is homogeneous means if you can define a linked list it should store only one type of elements in it, whether its entire elements integers, all integers or all strings are all any type user defined whatever it is there. And then, so, these are the basically the definition is a fundamental definition a basic definition of a linked list structure is an ordered collection of finite homogeneous elements.

Now, here few more terminologies that is required. All elements is stored in the form of nodes. So, nodes is an important concept in linked list structure. Now, here as an example, we say a node usually has the two fields. The one field, first field is the data itself which data you want to store. So, if you want to store integer or string you have to define that it is integer or string like this one and this is called a data field.

That mean actual content that you want to store. And then the next field is basically link, link to the next node. The idea is that if you want to store say 10, 15 and 20 then so in the first node, it will store 10, another node 15 and then third node 20. Now, the first node which stored 10, it basically links to the next node namely 15. So it basically these fields stored the address of the next node where the next element is stored and so on.

And at the end, the this link filed should be null, null means it does not point to any other nodes indicating that this is the end node. So, this is the concept that is followed there in the linked list structure and the concept that is, is called a concept of node. Node has the two field,s the data fields and link field, link is basically addressed to the next node.

(Refer Slide Time: 6:22)



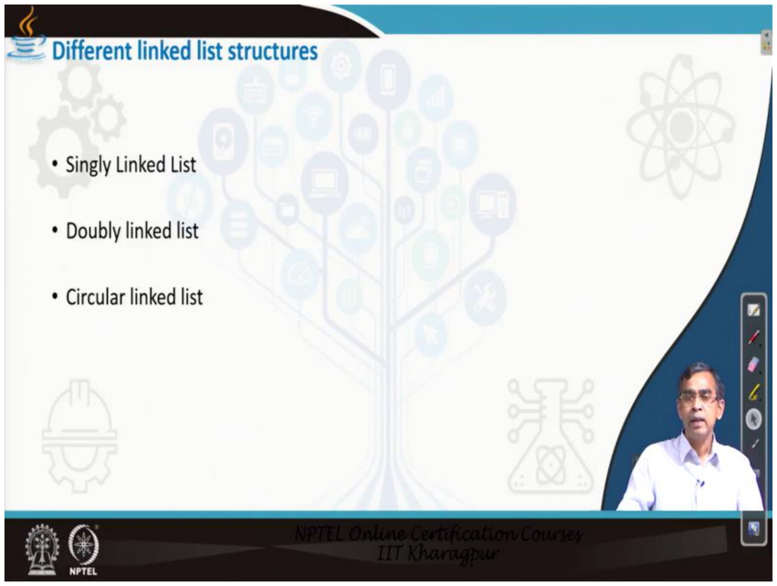
Array versus linked list

- *Array*
  - Elements are stored in a contagious memory locations.
  - Static data structure
- *Linked List*
  - Elements are not necessarily in contagious memory locations.
  - Dynamic data structures

Now, I have already mentioned about the difference between the arrays and the linked list, the fundamental difference is that array is a static type whereas, linked list is dynamic and array, the all elements stored in a continues memory location whereas linked list not necessarily otherwise they are same, they are ordered, they are homogeneous whatever it is.

Now, so, this is basically array, linked list and we have learned about the basic concept of linked lists and difference between the array and linked list. Now, we will see the different types of linked list are there, different types of linked list means depending on how it can store the elements in it.

(Refer Slide Time: 7:09)



Different linked list structures

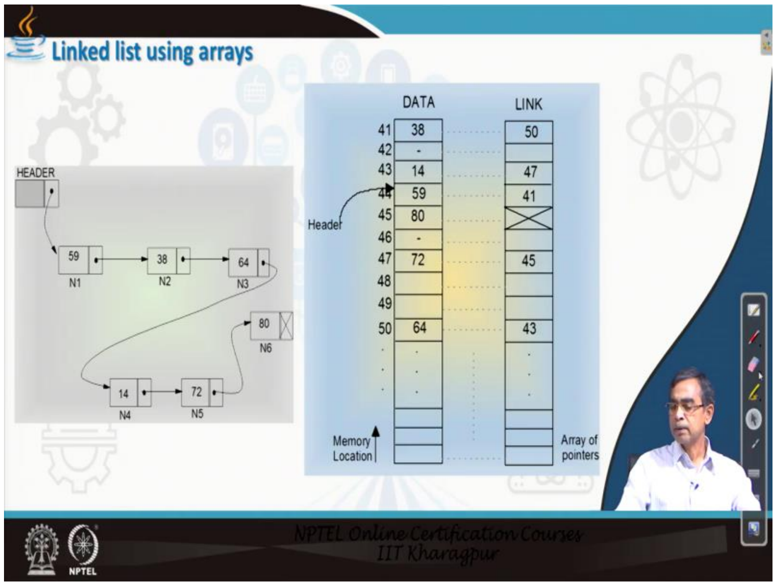
- Singly Linked List
- Doubly linked list
- Circular linked list

The types are basically 3 or there are 3 types actually singly linked list, doubly linked list and then circular linked lists. So, there are mainly 3 different types, we will discuss all these types one by one.

Now, let us first concentrate on singly linked list structure. Now, the name implies that it basically contains only one link. One link means it will store it will basically point to only one node, doubly linked list means if a node points to two nodes more than double again triple nodes also you can but it is not required unless and it will mix complexity or it will invite complexity in the concept.

So, usually singly linked list and doubly linked list. Singly linked list these basically point to a single node or next node only one link, doubly linked list means it basically point 2 nodes, 2 nodes.

(Refer Slide Time: 8:11)



Now, a single linked list typically look like this. So, here is a typical view of a linked list structure as you can see, in this example, as you see 6 nodes are there, each node contain their own data like, this is the one data field this is the data content so, 59, 38, 64 and you see there is an order is the, ordering is there.

So, here ordering that in N1, N2, N3, N4, N5 and everything, but, they are not necessarily in continuous; they are here and there. So, it basically in memory view it looked like this. So, 59 is the starting 1. So, this is our starting element that is the first node actually and after these things the 41 is basically indicates a link, so here the link or address of the memory element while the next element next data is stored there.

So, 59 is the data and it 41 indicates that next node is next node means which is the next element after 59. So, 41 so, go to the 41 is 38 then 38 50 that means, link is 50 So, go to the 50, you can go 64 and so on so on. So, here you can see in memory we have to store two informations actually. The first information about the data itself and second information about that the location of the next data actually and in addition to these so, data and link, there is 1 more extra node is used, this node is used called header node.

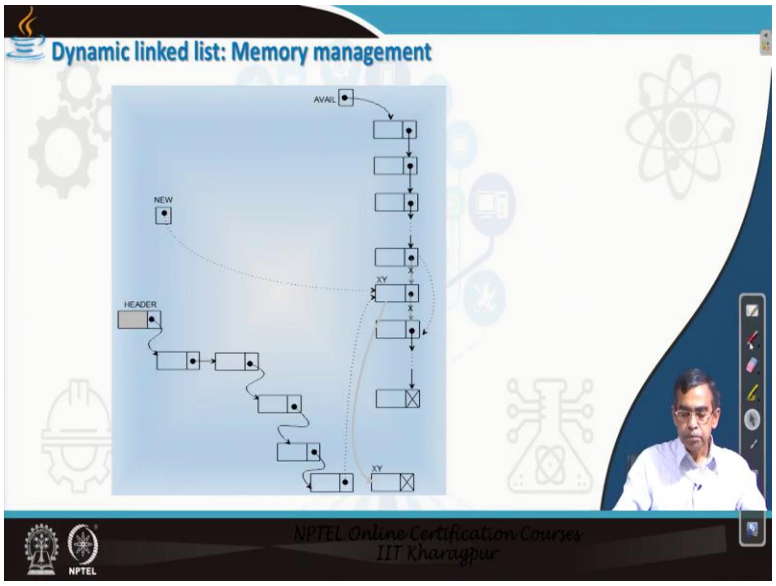
So, header node is basically a dummy node. It does not include any data, but it basically indicates a link to the start node. So, if this is empty or null, this means the link of the header node is null, this indicates that list is empty. Otherwise if it is not null it points to elements, then non-empty linked list. So, this way is a concept that is followed in the data structure is that header node, header node points the first element and then each each node will contains that data of its own and point it it points to that next node.

So, that concept is there and again you know the last node whose link it is again null, so null is pictorially shown like this. So, this indicates that is empty that means, this is that end node, so, there is no link, no more elements there. So, this is a concept that I have illustrated to you to understand about the linked list, singly linked list structure actually, both the conceptually, pictorially it shows look like this and physically.

So, this is basically the logical view and this is basically the physical view of the storing of elements using linked list structures. I can still or in other way show that a linked list essentially two arrays actually. One array content the data and another array contains the pointers, but actually in actual implementation of programming you need not to maintain 2 arrays, in can even single arrays, but here objects or class can be defined, that mean, node type can be defined.

Now we will discuss about how a node can be defined in a, programming issue will be discussed next video lectures, but today we will discuss about the theoretical point of this one.

(Refer Slide Time: 11:50)


Dynamic linked list: Memory management
AVAIL
NEW
HEADER
XY
XY
NPTEL Online Certification Courses
IIT Kharagpur

Now, so, this is the singly linked list and then different memory management that can be done very easily. Like say you want to insert a element insert an element into the existing list how you can do it. So, here is the idea. So, there are 2 actually, there is a pool of memory from where the element needs to be.

Because I told you that linked list is a dynamic structure it grows automatically. Now, so, if there is a pool, pool means the memory available memory is there. So, this basically the memory bank we can say from where the element will be collected and then that element will be inserted into this 1.

So, suppose this is our existing linked list going on. So, this is the at any instant the current structure of the linked list and if you want to add 1 element into the list how you can do that. So, first you see that whether the memory is available to accommodate your new data, it is available so, then the memory manager return that this is an element for you it can give it so this basically newly allocated node for you.

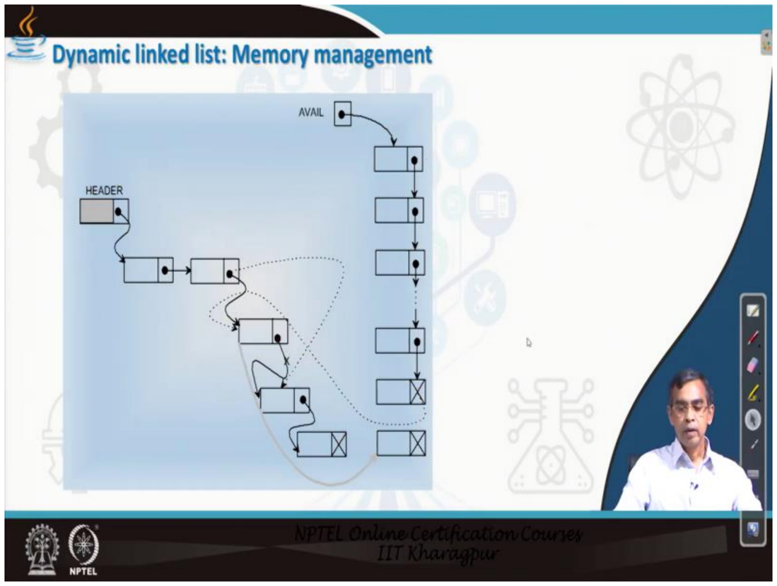
And then you can just simply set the pointers which was initially empty to this linked list here and these can be make null, and then this basically automatically will be added into this one and then the pointer of this memory will be connected to these ones. So, this way memory management, the bank will maintain its memory after giving the memory to you and this one.

But here 1 important thing that you should consider the pointer management or that means link setting and everything is an issue, here for example, we first get this so, this is basically available node and then set this one, then we can do one thing we can just change this pointer the the node which point this one will be saved by this one.

So, this pointer is there. And finally, you can make it there. So, this is the first one. Then the second is this one. And then finally, third pointer sitting is there if you do not follow in different order or not in other words, then you may come into a wrong way of adding element into the sum. So those things are very important. Anyway, those things we will discuss while we do the programming and that can be taken care.

Now, let us come to, so, this is memory management, memory management regarding adding element into a linked list. Now again returning an element into the linked list also can be done in the same way and, and here is our idea about how you can return if you feel that this memory needs to be returned to the memory bank, how can do that?

(Refer Slide Time: 14:51)



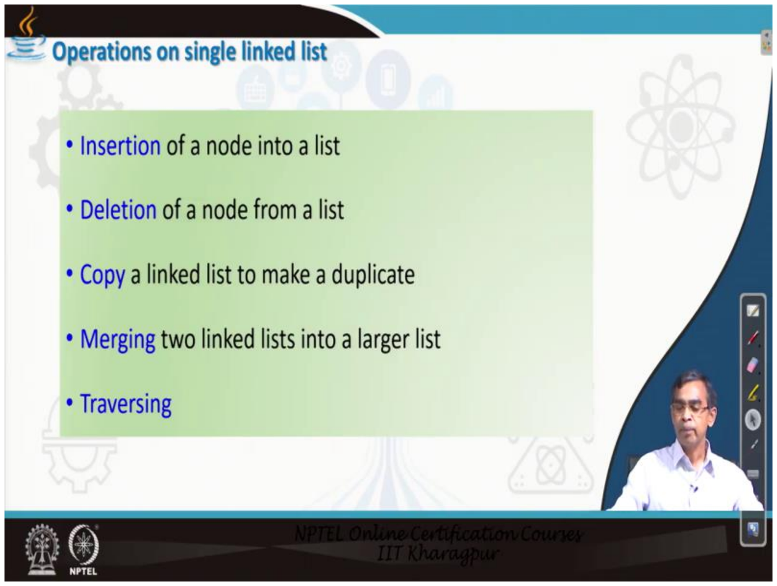
Dynamic linked list: Memory management

So, here again, here is one example that how you can return the memory. So, this is support existing structure that you want, that you are having and at any moment if you feel that this element needs to be returned to the bank, so, how you can do it? So, you know that this is the memory that needs to be returned. So, and this is a memory bank and memory bank is the last is the there.

So, first you can see it the pointer because it is initially empty. So, you can set this point to this, which basically the node to be returned and then this can be made null, but before making null, we see that, this pointer should be changed to this one because it is returned and then (())(15:41). So, the first this links needs to be set, then second link to be done and then third is to be done. And this way you can easily return memory return a node into that pool of memory.

Anyway, so, these are the very efficient way those are the memory return, memory allocation can be done and using linked lists structure that is why linked list is very popular and very fundamental. It is used in any system software, any operating system, implementation or any application implementation it is there. Now, let us come to the basic operations that we can do using singly linked list.

(Refer Slide Time: 16:24)



Operations on single linked list

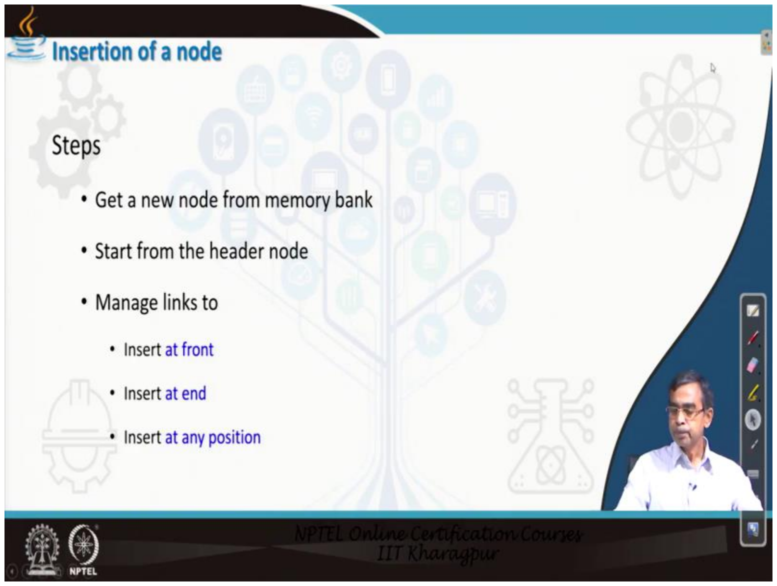
- Insertion of a node into a list
- Deletion of a node from a list
- Copy a linked list to make a duplicate
- Merging two linked lists into a larger list
- Traversing

NPTEL Online Certification Courses
IIT Kharagpur

Operations, are insertion, how a new element can be inserted, we have already studied about basic idea that how a new node can be added into the linked list, existing list. Deletion operation basically removing how you can return if a node is no more required, return to the memory so that other program can use it. Copy it is just like a copy means do making a duplicate, merging two list are there if you want to merge the two lists into a singly linked list.

And finally traversing is a very trivial operation so that you can visit the entire list. Visit means like either you can print or you can perform some operations on it like this one. Now, let us see how those operations can be implemented or can be carried out theoretically. The programming issues we will be discussing the next class.

(Refer Slide Time: 17:12)



Insertion of a node

Steps

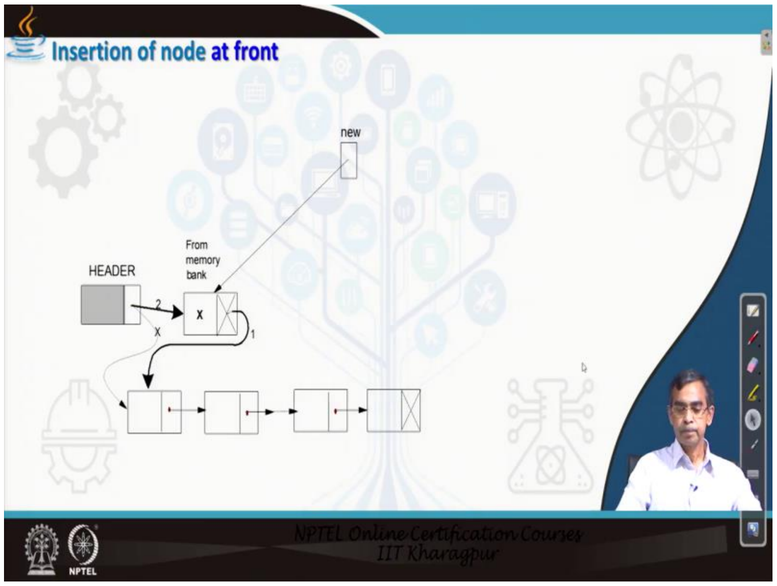
- Get a new node from memory bank
- Start from the header node
- Manage links to
  - Insert at front
  - Insert at end
  - Insert at any position

We will discuss about the insertion operation, insertion operation that we have discussed in one example in one slides in this just a few minutes back is a very simple one but they are insertion can be taken place in many different situations, I have mentioned here 3 situations, because depending on the situation your operation will be a little bit different actually.

So, the 3 situations or 3 cases like insertion at front, insertion at end and insertion at any position. Here at any position means, after a certain element, you can specify that we want to insert after that element. So, these are the basic operations and now, the 3 major steps that needs to be followed the first step is basically you have to first allocate a new node to add new elements.

Now, here your allocation may be failed because memory may be exhausted, there is no more memory available in the bank, memory bank. So, in that case it will fail. Otherwise, you can go ahead then start from header node to go whether you want to insert that front or end or position and depending on the situation or cases, you have to just manage the links. So, learning of the different cases of insertion at different situation is basically managing the different links that way. So, we will see exactly how the different cases of insertion operation can be carried out.

(Refer Slide Time: 18:50)



Insertion of node at front

Now, let us first discuss about insertion at front. Now, so, from the memory bank, it basically suppose new is available new nodes. So, these basically returned from which memory location the next element will be obtained. So, new basically return, the memory manager returns node for you and the address of this is a new.

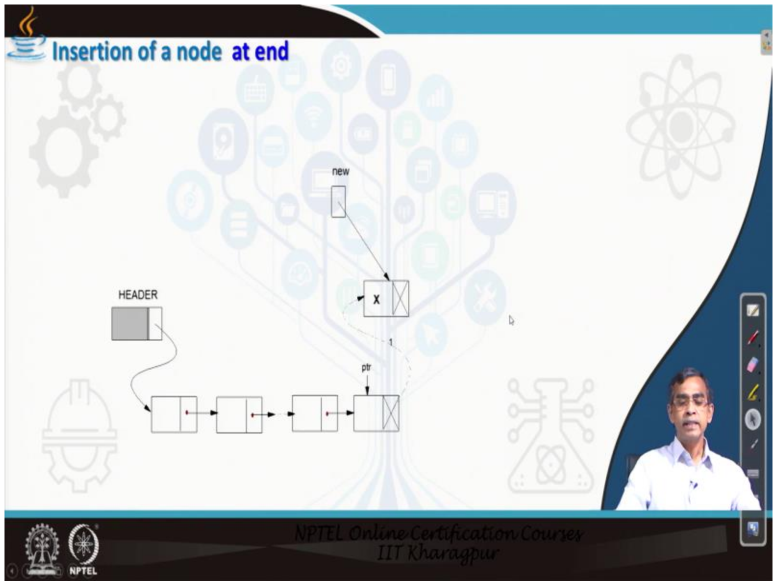
So, new is basically address of the newly available node. Once you have it, then you have to go for a little bit careful link management. So, you should start from the header as a viewer because header is right and you want to insert at font that mean, the first element, insertion as a first element actually, so, you have to insert here. So, say suppose, this is the new node that is given to you.

The first you do is that you just initially when the node is supplied to you, from the memory manager whose point this link field is null. So, you have to save this link field, this is a first one. How you can set it which one? Now, you have to go which is the front element if it is there it may be null. So, there may be situation of course, if it is null, then basically null will be there, anyway. So, you just go to the header and from the header know that which is the first element.

So, this basically is a first element. This means that the pointer or link of this pair, it will be saved by the link of this 1. So, this means the header points to this node and this is basically will be set to this one. So, this is basically linking. So, this is the way that you can, so, this this link will point to the first node, which basically header points it actually. So, this is the first step. So, once we set this, set this, our next is basically so, now, what we have done?

So, this basically set it and then header should point this element to this one. So, we know that this is the address of this next load. So, this point will be, this link will be set by the address of the new node that is given to you. So, this is the second link, so, only 2 links are to be managed, first is basically link of that newly available node by that first node that is point by the header and then second is basically just set the link of the header by the address of the newly available nodes. So, these 2 links are to be managed in order to perform the insertion operation at front.

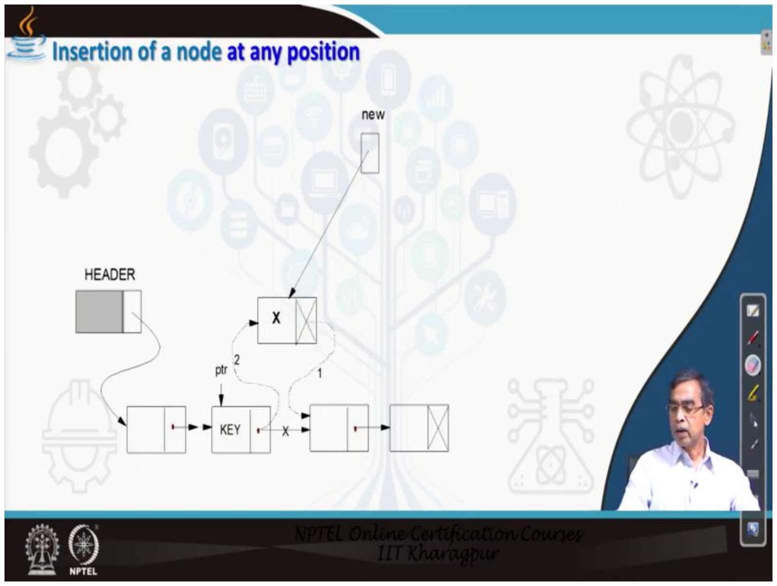
(Refer Slide Time: 21:56)


Insertion of a node at end
new
X
HEADER
ptr
1
NPTEL Online Certification Courses
IIT Kharagpur
NPTEL

Now, next, 'at end' like front, at end is also very easy, what is the idea, idea again first let us, this is basically the newly available node which is given to you by memory manager and this is a new node actually, the address of the new node is new. Now, you have to go to the end. So, this is basically address or basically link or location of the last node and let it be the ptr, ptr is basically location of the last node.

Now, initially this link is empty. So, adding the new node at the end is very simple because you have to set this link by the new, address of the new node. So, this is only one link needs to be managed and another is that the setting that this should be make null, indicating that the new node which is inserted at the end become the last node. So, only 1 link needs to be managed in case of insertion of an element into at the end. Now, let us see insertion at any position.

(Refer Slide Time: 23:18)

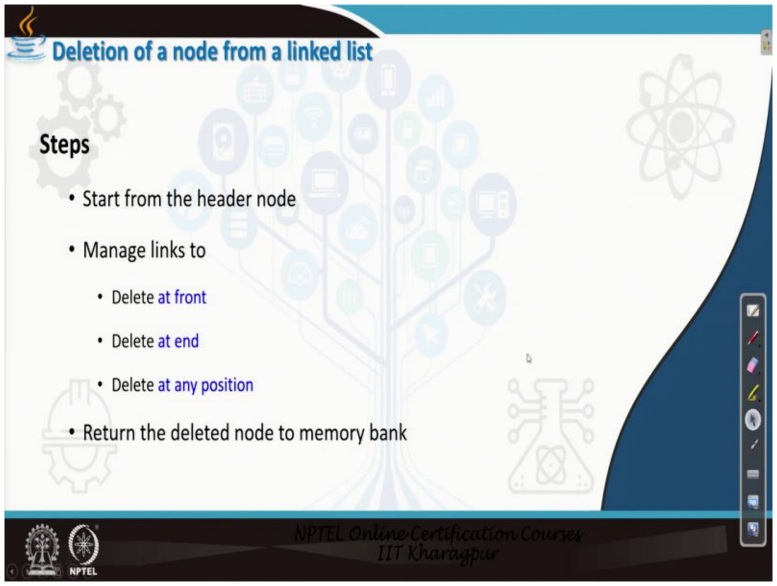

Insertion of a node at any position
new
HEADER
X
ptr
2
1
KEY
X
NPTEL Online Certification Courses
IIT Kharagpur
NPTEL

They are also only 2 links are to be managed. Now, let us consider this is the new node that is returned by the memory manager. Now, you have to insert after a particular element let us say, this element. So, starting from the linked list header, we have to come to this node and this is a pointer that it points after which node we have to edit. So, that is fine. In addition to you have to have this another pointer of the next also because this is required.

Now, what is the link management that is required is that so, this is a new node initially whose link is null. So, you have to set this link by this link. So, it is basically set this one that mean, this link should be initialized with the address of this node, fine, this is done. Next is basically the link of these fields should be set by the new node. So, this link which initially points to this one, we just initially we just set this link field by the address of the field.

So, this is a second link that needs to be managed. So, by this process you can see you can loss the link from this node this node, rather you can set the link from this node and from this node, so, the new node is added at this position like this. So, this basically it is there. So, this is a concept that insertion at any position can be done. So, this is all about the 3 cases of insertion. Now, next come to the deletion operation. Like insertion, deletion operation is a matter of link management.

(Refer Slide Time: 25:14)



Deletion of a node from a linked list

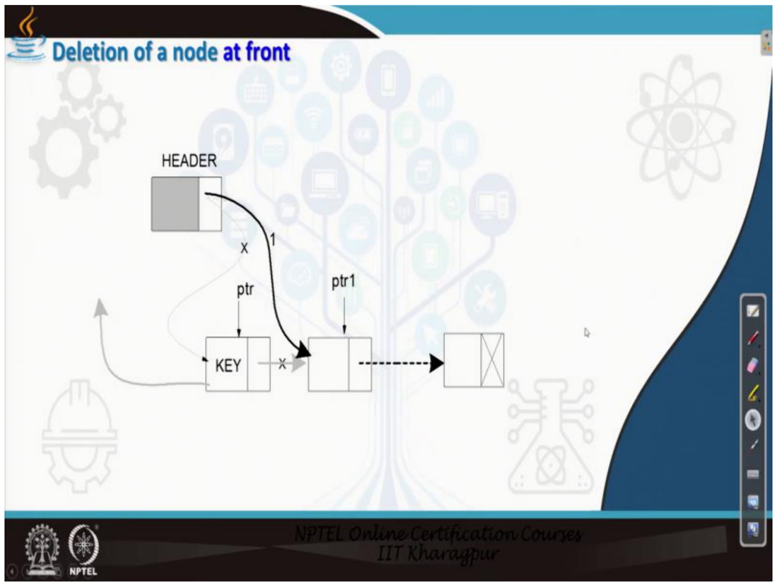
Steps

- Start from the header node
- Manage links to
  - Delete at front
  - Delete at end
  - Delete at any position
- Return the deleted node to memory bank

NPTEL Online Certification Courses
IIT Kharagpur

Now, like again insertion operation, deletion can be according to 3 different cases means, deletion at front, deletion at end and deletion at any position. Again 3 steps that needs to be followed, first you have to start from the header node, where to insert, if it if front be there, if it is end, come to the end node and if it at any position you search that where you have to go if you find a position then you can edit, and then finally return to the memory bank the node that is deleted. Now, let us see the example that you can do it of 3 different cases here.

(Refer Slide Time: 25:57)



Deletion of a node at front

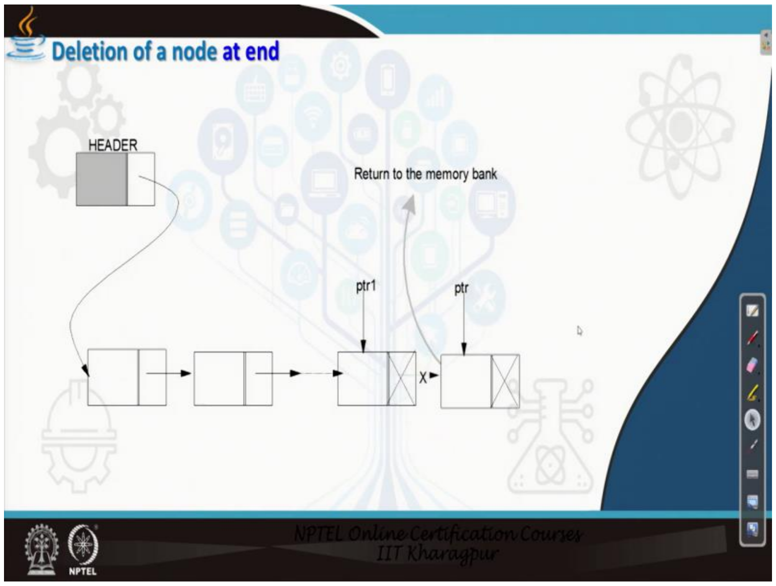
NPTEL Online Certification Courses
IIT Kharagpur

First we will elaborate about insert, deletion of a node at front. Now, here is the this is basically your initial linked list that is given to you and we want to delete this node that is our objective. So, what you can do is, first starting with header we have to go to the first node and let this is basically address of this node ptr is basically address of the first node.

Then what do you do is, first what link needs to be managed, first is basically we can have the address of that next to next that means second node, first node this one. So, this node is needs to be tracked first so that we can have it then the header node should points to this node, so, header node will point to this node. By this process, the link from header node to the first node is lost. So, this basically lost them.

Next, basically we have to make this pointer is null. So, that is all. Then what happens is that this goes this is become freely available and the next linked list is basically like this one and we can return this node to the memory bank. So, this is a procedure that we can follow only 1 link needs to be managed. I have been setting and other is basically setting the null. So, this is about the in deletion of a node at the front.

(Refer Slide Time: 27:48)

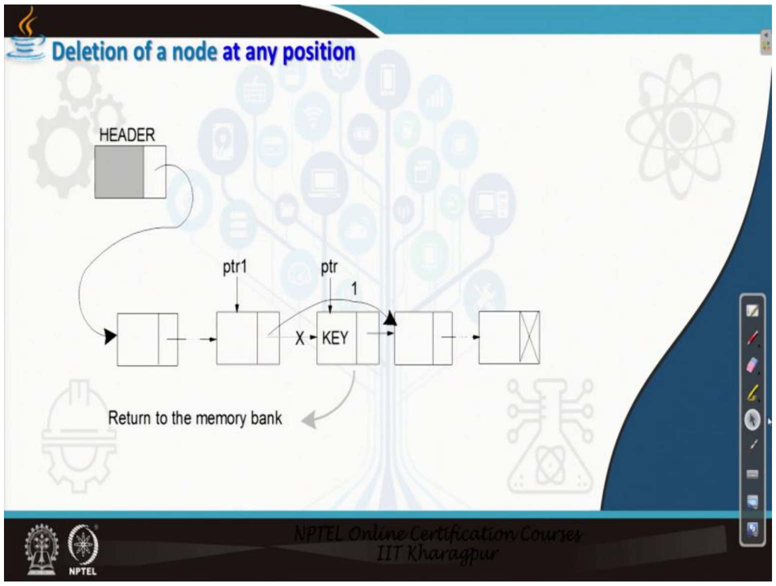

Deletion of a node at end

Now we will discuss about deletion of node at end it is again similar to the font. It is very simple. So, so initially given this is basically your least as a whole and you want to delete this node which is the end node that is basically the case. Now what actually you want to do is that, you have to come to the last node so, this is the pointer here and before the last node, this node needs to be also keep a track.

So, like the front we have to keep track of the header the next, next node like this one. Front node, next node here again n node and before the n node, this one. So, ptr and ptr are the 2 pointer needs to be stored there. Then the deletion can be accomplished by setting the link only. So, first we it is already null we do not have to set anything, only what you have to do is that which nodes is the ptr 1 is basically linked field of this to be set as it a null, that is all.

And then this is basically. So, now your link is like this one and this basically is a free you can return this free to the memory bank. So, this is a case of operation deletion. In this case and our next operation is basically deletion at any position.

(Refer Slide Time: 29:14)


Deletion of a node at any position
HEADER
ptr1
ptr
1
X
KEY
Return to the memory bank
NPTEL Online Certification Courses
IIT Kharagpur

Now, this operation again initially this is the list that is given to you and you want to make free this node this node having key that you want to get and what you want to do is that this node you have a track.

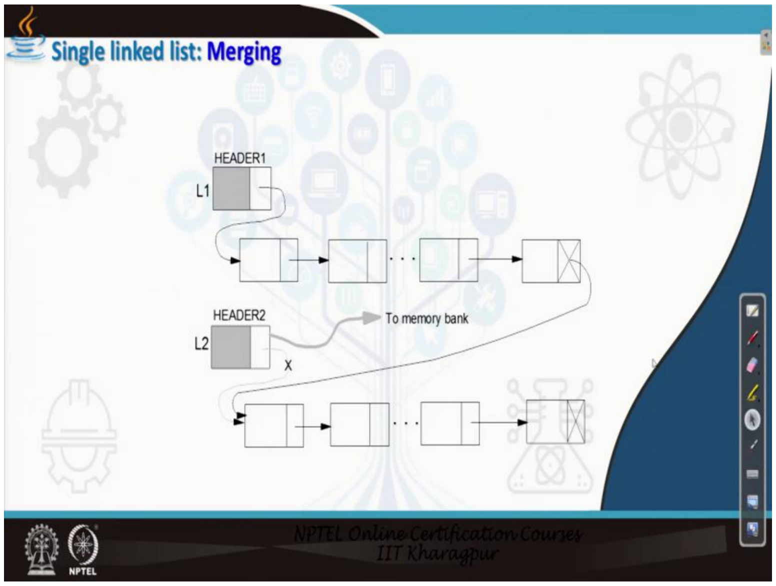
So, here you have to keep a track of before this node this node is there, so, 2 pointers needs to be maintained here. One is the node which needs to be removed and the nodes which has to be removed just before this node and you note that this node address needs not to be tracked. So, these 2 pointers have to be tracked first, after visiting, after traversing from there you can come here.

So, that can be recorded. Now, what, what modification of the link that we should do is just simply we know actually knowing this node, we know the pointer of this node because this one. So, this is the address which basically map the next node we should set this address by this one. So, these basically the link that we that means, we can change the address of this node address or the link, the field, link field of this node will be set by the address of this node.

So, this can be done and by doing this , you basically lost this information, that means this node does not indicate this one, point this one. Next is this pointer that is free and we have to make this basically is a null and then return. So, these become free and you can return to the memory bank. So, this is the idea about deletion at any position. Again we can see only you have to keep track of the different nodes that needs to be taken into consideration while we are doing deletion operation and then managing the links.

So in most of the cases as we see, in case of deletion, only 1 link needs to be saved and then another is basically the node that needs to be returned, this linked to me marked as a null, that is all. So, these are the basic operation that you can do so far the deletion operation is concerned.

(Refer Slide Time: 31:50)

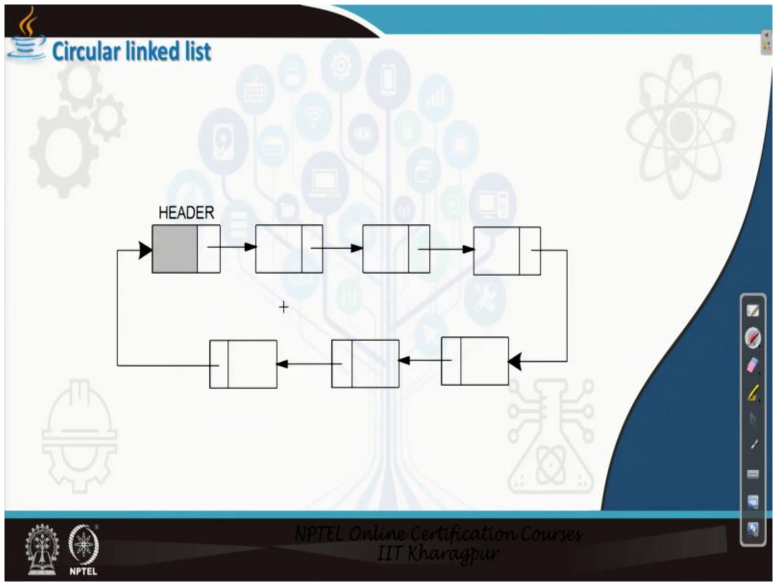


Single linked list: Merging

Merging operation is quite simple. If you are familiar to delete, merging operation is there; for example here, 2 list, this is one list and this is another list is given to you and we want to merge this list after this list, then here the what operation you can do is that from this header, we know that this is the pointer.

So, what you can do this is the null field we will just simply set this pointer to here. This way you can lost this pointer, so empty and then return this header because this header is not required to memory bank and then this way, what you can do is, we can just do the merging of that to a list. So, it is quite simple only 1 link needs to be managed here. So, this is called a merging operation. Now, let us come to the circular linked list.

(Refer Slide Time: 32:46)

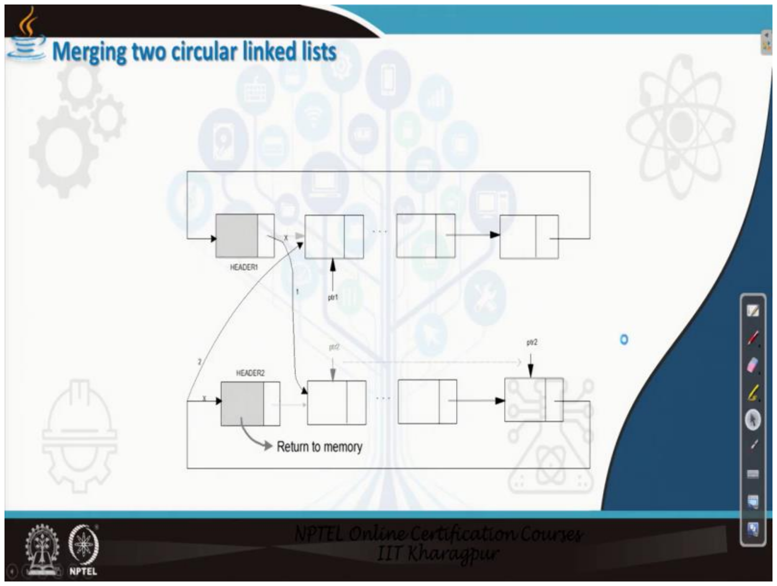


Circular linked list

What exactly the circular linked list looks like. Now, in case of singly linked lists that we have learned about if it is singly linked list look like then that last node or the node which is at the end having the field, link field is empty or null. But in case of, in case of circular linked list as we see it basically the last node, the last node basically indicates to the hidden node. Now, if you can traverse it and you can just took a starting from the header in case of singly linked list we will go to the only last node, but in case of circular linked list, starting from the header you can reach to the header again that is the difference between the singly linked list and circular linked list.

In case of singly linked list, you can never return to the header because it basically gives you on way movement but in case of circular linked list, you can come back to the header and again repeat the traversing like this one. In many situation and it is useful, but application is a different issue. But I just discussing about concept of this list. So, this is the concept of circular linked list.

(Refer Slide Time: 34:05)

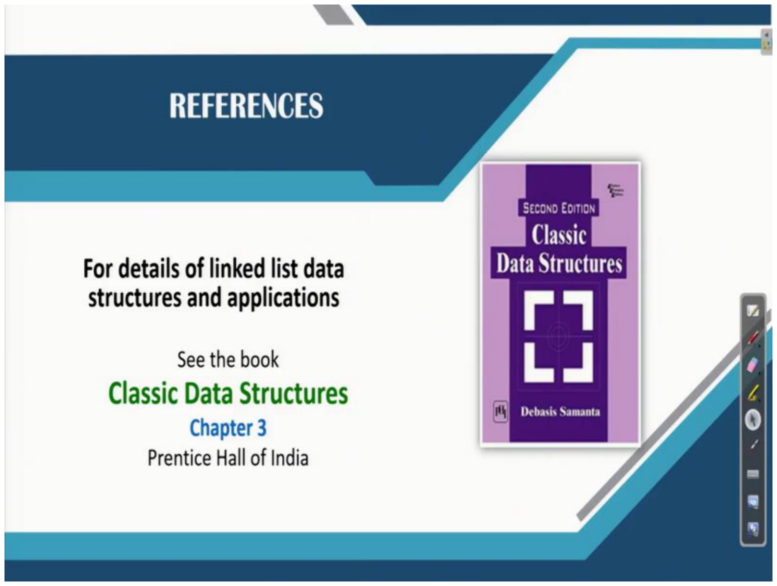

Merging two circular linked lists

And merging of the 2 circular linked list again simple like the singly linked list merging is there. Now here is an example. So, this is a 1 circular linked list and this is another circular linked list, I want to merge 2 circular linked list into 1 circular linked list. So, how we can do that? So, here basically we can go, we can know that which is the last node by this one. So, somehow we can have this is the last node of the circular linked list and this is the last node of this circular linked list.

Last node means the node which points to the header is the last node; here the node which points here is the last node. Now setting is very simple. So, first what we should do each we can set this pointer to this one. So, this way it comes there. So, this way it basically comes to this and then and then said this pointer to this 1. So, this way just, two sitting is there. Now, there are opposite way also you can do we can we can merge this list before this list the opposite way the link can be managed.

So, this is basically merely managing the link and then you can do after the merging the last node can be returned. So, this is the merging of circular linked list.

(Refer Slide Time: 35:34)



REFERENCES

For details of linked list data structures and applications

See the book

Classic Data Structures

Chapter 3

Prentice Hall of India

Second Edition
Classic
Data Structures
Debasis Samanta

Now, we have discussed about singly linked list and discussed about the different operations on it namely, insertion and deletion and circular linked list is also the concept is now we have learned about. If you want to learn many more things about these linked lists structures, I should suggest you to go to this book you can find it there and details of operation and everything will be there application of different structure will be there.

In our next video lectures we will discussed that other linked list structure namely doubly linked list and the different operations of it. So, this is the topics for today. Thank you!