Data structure
Lecture -2

Arrays

We are going to look at two implementation of list. Two important implementation of list. One is array and another is using linked list. So first let us look at implementation of list using arrays. Before we are going to that we have to know what an arrays? Array is usually associated with pair of values an index and a value.

An array is a sequence of indexed items. For example when I say the first element it is the index is one. And then the element is also fix as five are something other. If I say 10, A[10] it means that the 10th index is 10 and the element value of the whatever the element there will come out. So associated with each index there is a value, and normally an important constraint of arrays is that the length of the array has to be fixed. Before the array is created and it is implemented using consecutive memory. Now one important thing that we have do known is that when we represent list using arrays, the list is logically continuous as well as physically continuous in other words the elements of the array are stored in consecutive memory location. When an array is created consecutive memory blocks is allocated dynamically to store the values of the element of the array. So if we take an example. Let us assume that the array consist of 0,1,2,3,4. You will see that, here we assume that 2 address location are used represent one element of the array. So 1 to 2 it's contains 0. 1,2,3,0 contains 1. The index will be 1,2,3,4,5. This is contiguous memory location that allocated. Normally, we talk about an array. The indices an array range from the lower index bound normally 0 most programming languages. Through the higher index bound n-1. It means that you can represent N element using the particular array.

Any items of the array can be access directly by just giving its index. So when you say A[1] you can access the first element. When you say A[5], you can access the fifth element and time complexity of accessing an element is 0 of the order of 1, which means error if its 1000th the element array. If you say A[1000] you can access directly to the 1000th element. So operation you do create and initialize the length is a find the number of elements in the array and insertion & deletion. So given an array like a left to right. You can insert of a value and a particular location. If necessary we have to shift the elements. Insertion and deletion is most time consuming and inefficient operations. When you represent list using an array. Now search on the other hand is the most efficient operation that we do.

When you represent the list using an array. List in finding an any particular element from the array from the list represented using an array. So array is a basic data structure and it can store any type of an element. It can be integer array. It can be character array, an array consisting of nodes which is the structure. An important application of array is the representation of polynomial. Polynomial is consist of an exponent, degree of the polynomial is largest exponent. So this is particular polynomial you can represent the array. For example 3 has one data element and 20 as other data element and +2 is other, +5 is other ad so on.

Linked List 1

Another representation of list, please remember talking about list. And how do an implement the list?

The first type saw as an array. The second type is a linked list. Now if you look here. This is typical representation of list. Which is represented to a linked list. Now you see that bat, cat, sat, vat is a list and logically they are next to each other or sequential. But, when you use a linked list to represent a list the memory are physically they or not adjacent to each other, they are only logically adjacent to each other. For example, bat can be stored a 1000 location cat can be stored that 200 location. For example, sat at 600 location and vat at 2000. In spite of that because of this link points that address of the next element and because of this they become logically adjacent to each other. Though they or not physically adjacent to each other. To linked list is a chain of elements. And now every node in a linked list has two parts. At least 2 parts. One is the data and other is the link. So linked list can be represented using a PHead, which is the pointer to the head of the list. Of the first element of the list and if you have an empty linked list then the PHead link pointed to a null value and then node element of the linked list can have just one data field. Which is the example given here can have 3 data fields which is the example given here. Where you have name, id gridPts and whatever it has always in the linked list where see up to now it has a single link field or address field and here we see that node is represented to a structure, name, address, and phone and again have a link field. There are many ways in which linked list can be represented in one the look at we look at defining a linked list using 2 structure. One is head structure, one is data node structure. The head structure need not to be represented there are other implementation do not use a separate structure for representing the head node. We are assuming that they are using a separate structure. So you have head structure and data node structure. The head structure will consist of a count of the no. of element in the node and pointed to the first node in the list. The node consists of data and point to the next element of the list. So if you represent this the head structure consist of count is which is an integer and head which is a pointer. In other words in address and

here you have a data which is a data type. Please note this data can have a more than one element it have a structure it can anything and link which is a pointed to the next element. So these are two structures that are used to define type of the element in the linked list as we have represent the linked list. So this is the method in which you represent this. And this is just representation of what seen previously. If you look here, this is typical linked list, which has been created using the structure given here. So here you have the head which pointed to the first node in the list and the count use the no. of element in the list which is 2 and this is a head node. These are data nodes and this consist of the data and the pointed to the next node and this is the data pointed to the next node. This will be null. If the last node in the linked list. This is the representation of the head and count and this is representation, if it is a name. Here the previous one. Put it as the data. If it is the name list. You have the same structure except that. The data will be represented by a character and here you have a person. You can also represented using a person which is the data. Pointed to the data and node which is a represented pointed to the link. Now will have to look at some operations that we have as for as linked list is concern. One is that you have to find out that linked list is empty and another things is because we talking about the linked list. We have to find allocate memory for the node and that is done by this malloc structure.

This malloc structure will take the node and will allocate memory location for the data node, when it is created. So the operation we do on linked list implementation of list is same as we do on any list. So you have to create a list. You have add a node, you can delete a node, you can find a node, and can a traverse a list. Please note that in the linked list representation the addition of the node at the beginning, middle or end. It slightly different and similarly deletion beginning, middle or end in the slightly different because we have represented something called a head node. This is a representation of how to create a node. So you will have a pointer and you can create a data here. Have a pointer to the null. It can be represented by a null, which doesn't have pointed to the next node or you have pointed to the next node. Now please note here that here is creating a two node list. The structure for this. Creating a linked list with two nodes. So first you have defining two nodes. 1st and 2nd in the first node allocating space for the first node you allocating space for the second node and then what you doing is, you are assuming that this is a first node and this is a second node. So you are the second node. The link of the second node is set equal to null. The data of the second node is set equal to 20 and the data of the first node is set equal to 10 and data of the next is made to point to this. How do you get the address of this, this is available here, when you are doing the malloc. So this is a not generalized implementation.

Implementation of hard coding implementation of how to create the two node list. Now please note that when you have a linked list like this, only you can enter the linked list or access the linked list is throw this the pointer or the pointer has to have an address of the first node. Then only you can access the linked list implementation of the list. So now let us look at how to create a list. Please note that we have said that we had separate structure call head, head node, which is consist of the count the no. of element in the node and pointer to the first node of the element. So before you start the count is undefined and head there is no address pointer. Now I want to create a list. So what I do is I put list head equal to null. That is I set this equal null and I set the count to be equal zero. Because this is an empty list. So after the creation operation. This is the output because count is a becoming equal zero and the head set to be null. Please note that, the no. of element in the list at this pointer still zero and you just create the head node. In the other words, the creation operation you just setting of the head node. So this is an implementation of the head node. So you have the data this pnode, is the pointer to the next element.

So you put the pointer equal to null and you can also defined something call empty. You what do check whether the list is empty also can be done. Now you want to add an element to an empty list, so before the add operation you already create the empty list. So this is what you will have so this is what you saw in the last slide also. And now you want to add this particular node, data node, into this list. This is the data. This is pNew is nothing but the address of this node. This is an undefined as f node. How do you add this node to this list? So, this what you have seen. So the first step is that you set the link of this to be equal to link of this. It so happen because it is an empty list into which you adding this actually will be set to null. Than you have to change the count of the head node one. Because you will adding one element and the next step you are doing this actually change the link to the pointer to this. So that is the next step, this link, which is the point of the head node of the new node. Which is given by list head equal to this address, which is we know pNew. So what is happen is the link field of the has been set equal to link field of this which is null and the head has been made to point to pNew and this is the state of the list and the count to be one. Because now it contains one node and this what happens of the add operation. So this link, this link been set here. So this is how you added the pNew. Please don't think of this link. Just pNew give the address of this node. So this is how add 1 to an empty list. Ok. Now we look at adding at the beginning of the list. This is also similar to adding a node at the one node to empty list. That is this is the list before you do adding at the beginning. This is the node data node. That you want to add to the list. When you want to add here. So this is what the status is. Before you do

the add operation. Now, what you have do first is the link field of this has to be set equal to the address of this. What you done is. You have set the address of this node to be equal to you have set the link of this node. To be equal to this and the set the ink of this node to be equal to be this. If you reverse the operations here. You want to get the answer. You will what happen trouble because I do not know the address of this well. The address of the available. The address of the available only in this. Initially changed this and put the equal to zero. I have loss the address of this. You cannot do insertion. If you have to initially necessary in this order. First you have what you do set the link of this point to this and then change the head. To point to the first node. So now the list is like this head node 39,75. So this is have add an element to be front of the list. So this is just show you that. When you doing an insertion you are just inserting like this and you not changing to link, while when you are doing insertion of list. Using arrays, we were actually are changing the index of the no. of elements. So suppose you had add an element in an array in the 5th position. And an array consist of 1500 element. Then from the 6th position upto 1500 change the index of all the elements.

Linked List 2

Now we are going to see addition of the node in the middle of the list. So, before the add operation. You are assuming that the linked list is like this which consists of a 2 nodes. 39 and 75. PHead is pointing to the first node. Now what we want to do. Yes, we want to add and element 52. Here, that is we want to list look like this now. In order to do that you need to know the address of this node. So that is what is put as previous? So now the node as change after the addition. Now you have 3 nodes. This doesn't' change. Now what you are doing is this pNew's link. Has to be changed are a link has to be fixed at this 75 or the address of 75. How do we that address of 75 nothing but the link of previous. So now you are setting. pNew's link nothing but P previous link. The address that you set here. The address that you add here. So that is the first step and then this P previous link has to be set equal to this address of pNew. So that the next step. What happen here is first, the pPre link this link is set. Link is equivalent to these step. This step equivalent to this step. That is pNew link is set equal to pPre link. pPre link was actually that address of this node so that is the link and then what you doing? This link now you are changing. You use already the changing this is point to pNew. That is this step. So this link correspond to this step and this link correspond to this step. Please note again. You cannot interchange. If you interchange this you loss this address. That is very important next is add an end of the list. We are not again go to do. But here what happen is. This is node where you want to add. And this is the address of the previous node. But here that the disadvantage of the just not been told here. We assuming that pPre is available.

pPre will not available please note already told you. That can you access the list only from the head nodes. So if you want to find p previous is that you keep on searching the list, till you find a node who is address link field has null.

Then you know come to the end of the list and you note that address as P free and then what you do? You set the change the link this point to pNew. And then make p new link is equal to null. But this is one disadvantage of representing list using linked list. Because what happen that if you want to find the end of the list. Assume that this is only 3 nodes found easily. Suppose if you want to insert at end of the node in 1000. Element list representing using a linked list. You have to search from right here and go upto 1000th element. Before and find the address of the 1000 the element. Before you can do this addition. Addition is end of the list is not very convenient or efficient. If you represent the list in this way and other way is you can insert after a particular position. Insert a node with data into the list pointed to after that node. So you are assuming that you know the address of what is trying do is. You want to insert after these you given the address just insert after it is. That is another way of doing is that is what is told here. And this is where you insert. This is what you already seen in the previous test may be insert you first set point this. This is what is initially there. You remove this link. You don't remove the link, actually. What do you do this? You address know. So set the link do this and make this link into this after inset it. This is already treated. Just the program for that. So up to known you seen creation of empty list, you seen addition to empty list you seen addition to front of the list, you seen addition of the middle of the list and then seen addition after that particular node whose address is given. This is fairly strait forward suppose you had to insert an element in sorted order. Let us 5,10,15,25. And now I want to insert 18 is sorted order. Then again the have to traversal from the beginning. Find out whether the 15 is store the value and then do insertion in middle. So that is also little bit of DTS of as for as insertion to list using a linked list.

Now let us look at deletion again deletion you can delete at the first node. Please note this a head node. And this is the list that I have. Now I want to delete the first node that is want to delete 39. Now what I do. I have to make this point to this that all. So what happen is. The head, the link this head now point to this node. And count has to be change to 2. And this location that is the node contain 39, can be recycled or put back into a store of node available for insertion. So this is the deletion. So actually what happen is when you delete to node change a link there is no link to this node. So this in other words are also called as dangling reference. This was deletion of the first node. Now the advantage here was you know the address of the first node. Which is given by this head. But in suppose

you have a general case of deletion. Where you want to delete. Let us say this node, then I need the address of this node if I want to delete the node. Because what happen is have to set this I need address of this to find out and set this link point do this. So, again as we did in the case of Insertion what we doing is we are assuming that we know the address may be conveniently but actually find out the address you have to traversal so, what happen this is Ppre the address of the previous node. So this link may to point to link of this node. Because these link was point it to zero. Then only step for deletion and this is the recycled node. Which is no longer in the picture. So this is for searching. So you can search for a target node. So you can 5,10, 15,20, 95, 100 and if you can search for the less than first node. You can search for greater than last node and you can search for particular node. So if you search less than 5 you won't find it. If you search greater than 15 you can find it. If you target 15 to 25 own to find it. So you can search like that. So this is again deletion just make it clear. So you have the node and you want to delete. So you want to delete 10. Delete it and put there. So if you want to delete the node other than the node. Deleting the first node. This is a special case. As you told before you can insert or you can access a list we representing linked list using only a node and since this node have now the node has been remove you have to set this node to be 50 are the address of this node. Otherwise you cannot access it. The node that's only that is has been set to 50 and if you want delete here 50. Set the delete here and this been removed. This is removed of this. So here what you have node is equal to next. Node not next equal. So suppose you want to delete the node after the next so you put node of next is equal to node of next. Which is set to address of next. So that is this step and then free this 50. That is free 10. traverseList. Just you go the through the list pointing to one of the other and till you reach the null. So after this function call pointer value will be null. Because only you after reach a null this end of the list. Some time you want to get the first then the pass position will be point to the first element and otherwise it will point to the next element because of link. So you can other operation like swapping elements. Insert first, last, delete before, last and so on. There are lot of variation you do for insertion, deletion also.