Data Structure

Lecture 3

STACKS

Stacks are a very important data structure and are necessary for implementing the logic behind many programmers. So let us look at start at the end of this lesson. we objective of this lesson this to introduce the concept of this stacks and then look at the operations of the stack and then the applications of the stacks these are thinks we going to talk about in this lesson before we go to on to the complete aspect of stacks as you know there are stacks every day in our life like stack of shirt, stack of books and stack of boxes and so on. As you see the idea is that normally when you take out the first shirt you take give be the top shirt of this stack and so on. So these are otherwise (57 sec) fall down and this catch in very carefully the same concept in applied to stacks in data structure. So this is the typical application of stack there will have stacks of book and stacks of coins at the same think represented in computer stack. Computer stack has gone to number of object set of placed one on top of other in a sense so another common linear data structure that I already talk to you about is stacks one you already seen its list and in one way the stack is nothing but the list but getter lot of restriction of how the list is used, then it became a stacks with the stack we have restricted to the access to the list to one end we take out the element from the stack using the pop element and we push element in to the stack using push operations. So the result of this restriction is that the list pile one on top of the other so basically talk about stacks we talk about when we what is call the last in first out policy. To get to the bottom item, we must first remove all the items above it another words so the behaviour is sometimes described as this is the policy we that talk about which is called is last in first out. So whenever you think of stacks you must think of it has last in first out whatever you put in last will be taken out first are so it's called a LIFO and this is because I already told you that last item put in the stack the item removed first. So elements can be added and removed only at one end this end will call as the top of the stack with this stack the top item is always the last item to enter the stack and it is always the first item to leave the stack because no other items can be removed or added on to the stack without affecting the top element so let's look at some terminology above the stack so the depth of stack is the number of elements it continues so suppose you have ten elements this stack it's deep ten and you say that there are the depth is stack is ten and normally as obviously an empty stack is get a depth is zero. So why to be used a stack. Stacks provide a very efficient storage structure for manipulating data that is last in first out policy and stacks are often used as temporary storage to speculate the solution of a problem. So stack are restricted linear list as already told you of where

additions and detections are made only to the top and what happened is because of this specialization are restrictions that we do on the stack we can implement the stack using link list and we can also implement the stack using arrays and we think of ways in which in efficiency of the implementation can be improved. So as before you just seen basically what I stack data structure is you must remember that stack data structure is that restricted types of list and its follows the last in first out policy all access to the stack is to through the pointer call pop and through the top you can instead element and delete elements and the last item which has to be which has been instead the first item that is to be removed.

Now let us look at all the operations that you can do on stacks. So first operations the do on the stacks .what is called a max item. Max that is nothing but the maximum number of items that would be on a stack. Now before you going to details about this that what the meaning of max item depends on the implementation of the stack so if it is arrays implementation. You already discuss array implementation they will be maximum size of the array and that will be the maximum number of elements you can put in the stack but how many elements has actually we put is the max item and item type can vary it can single and simple integer are we can a structure and we can as we saw list san be any data item they others are make empty that is an operation that make a stack completely and you can also have a Boolean operation call is empty that is to test whether the stack is empty or not similarly you can have Boolean is full the test whether the stack is full this actually make sense only you can talking about array implementation and next you have push (X,S) this is to add push an element X to the top of a stack S and then this is the function as this is the adds X to the top of the stack and the pre-condition is that in order to push in a element to the stack. The stack should not be full and the post condition is at after the push operation the element X becomes the top elements of the stack next you have pop S this is nothing but up to delete element an a stack it's call pop the top most elements of from the stack and the functions of obviously its remove the element stack and return the item and the pre-condition has for any deletion operation for that can be matters this stack should next be a empty it has to be initialize and it has cannot be empty if it's empty the pop operation is become un defined the post condition is that the top element would have remove from the stack and copy of item will be available and the top has been modified then you have something call has top S the stack S has it is hear is axis to the top most element of the stack hear no change occurs to the stack and only the top most element you will get the value of the element that's it . So let us go little more in details about the operation of the stack so you have first the push. Push is add and item and the problem here is that you can over flow suppose you say the stack in contain ten item and it's already full within ten item then you try to added another element eleventh element then the stack is

overflow condition occurred this is the only for the array implementation and for the pop element you remove an element you can have already total you can have a underflow and the stack top is what the top is stack so this is the pop operation you end element and this is the stack pop operation where you take out it so let's take this more in details so you have the take this more in details so you have the data that you want to push on the stack this is the initial stack of the stack. So for example the stack has to element and assuming the stack starts with the top is equal to one here you have the top is equal to two so now when you sit what happened is this the element data is pushed on to like this the top increment is one and the element stack size increases. Now let us assume we want explain the pop operation as you already know it moves data element from top of the stack initially the stack has three element and top now points to three when you do a pop operation there is no questions of which element to delete you can only delete the top most element the way the stack is defined so what you do you give the pop operation the top most element is deleted that is return and top reduces by one so this is the pop operation. Then look at this operation this is the stack top operation you don't have any change in the stack. Stack has three element and also a three element top value here is three here top value is three and only think is you are able to access the element and value is given out please not when I say top is equal to 1, 2, 3 am assuming the array on the stack is which is told in a array and top start from one it's not necessary you can start stack from hundred you can array location hundred and you can also store stack in link list then the meaning of this is different it here is explain it's so you understand what the operation is this so let's look at simple figure here if you look here in this step one you have the empty stack and given a data that you want to push on the stack now what happens here you will note the tope is equal to zero and the stack is empty now incremented top by one and pushing into green into the stack now this is the new condition is the stack is end. Then with that you add another element which is blue here so now top becomes two and belay pushed on the stack this is the condition of the stack after the second push operation then what you do you go to the see the pop given the stack of the stack if I just give pop there is no questions of which element is pop because pop the top most element so what happened after the pop operation blue comes out and this is new condition of the stack where top is been decremented by the one and blue is no longer present in the stack then next let's see if you want added another element that element is push this is similar to at do it here and top increments by one then doing a stack top operation here what happens the state a stack is before and after the operation is same but only think if you can access the element then this operation and this operation is the same you just removed a element and here also you doing pop operation . you see here that is green has been removed that is top most element

from the stack has been remove the stack is empty now if you give pop operation there is error in the condition because the stack has is empty .so let us look at stack as an abstract data type so that means we are not bother about the implementation we going to using this stack so basically for any stack. Whatever the implementation the following at the operation of function that associated with the stack so one is create this means you just create an a empty stack for the implementation then the push you seen in of about the push. Push means added an element top of the stack then you have pop which means pop an element from the top of the stack you don't have to specify the element because only think at pop from the stack is top element then stack top which you already defined that is you are accessing the top element of the stack no changing in the stack that is the top remains in the same number of element in the stack remains the same then you can also have full condition the question mark that we put here is you cannot check a full condition when you do a link list implementation of a stack because you can keep an adding notes dynamically but fill make sense in the stack is implemented as an a array and then empty is when the stack is element that is does not contain any element so this is the stack in empty condition so you can just look at this small data structure of this so you have a external interface you have a data structure and can have the data here so this is the data you want to input in to the stack this is just giving feeling of what is available outside. The outside is when you doing push you have indicate which element it is X. which stack you want to put it and similarly when you defining the stack you have to also know the access point that is top of the stack so one typical application of a stack is the famous parenthesis matching or matching of brackets so what it do here this you don't have a matching parenthesis here so what it do you input when you see the traverses list from this corner when you see the bracket you put it in the stack then you see the other bracket . Put it the stack then you see an a just pass it don't be anything you see plus B don't be anything then you see the closing bracket. So you look at this stack and C whether whatever you have top of the stack matches this if it matches you removed that so now you left with is one bracket one that we put here then you see a divide sign don't be anything just pass it C. don't be anything then you finished you come to the end of the expression when you come to the end of the expression if it was matching expression. You should have add the stack is empty unfortunately now the stack is contain the single bracket so you know that the equation has not brackets have not been matched if you look at this same think you can do it surffency you see that this you will put on the stack. Then here you will you put on the stack again are left with closing bracket that is not matching so here are matching we a put in this on the stack then you putting a open parenthesis and the stack then you have a matching closing parenthesis which will cancel each other that will left with in

open parenthesis here you will left with the closing parenthesis but however you do that always you should have the stack is empty then only it's matched now in this example you take on simple example there is only one type of bracket you can have any type you can have square bracket and check the corresponding matching. So you can actually do parenthesis matching automatically using the stack data structure for any type of complex expression so as we already saw the stack ADT can have several implementations. Implementations may be different but implementation details are be hidden and the interface function have to be the same and most application programmes should not see any difference just because using array implementation arm using a stack implementation. So let us first look at the stack implementation using an array so stack is stored as an array when you stored stack is an array immediately it means that the number of element has to be allocated that is how many element this stack in cold so I can here constantly look at the stack as we done before so I have defined something called as defined stack array and then have a something called as stack max. Stack max is maximum number of elements that I can store is physical array and top tells be the top element you can actually not have the count also the doesn't matter but stack array, stack max, and top has to be defined. If you want to define a stack using array so implementing stack using array. It's a simple implementation. The size of the stack must be determined when a stack object is declared space is wasted if we use less elements suppose you have a stack you declared it have hundred elements and usually the stack within ten and fifteen then you wasting the space and we cannot push the more element and then the array can hold. So this is the typical definition of stack of integers in an array implementation. So what you doing here is we are declaring the array call stack array and then this is the number of element at count then stack max this is equivalent to what we saw before for this . We are defining the structure so this is the maximum element and this top is the index of the top element. So all of them are integers in this case so first step here is creating a stack so you will say the create a stack with maximum elements ok so if the stack is equal to null check as stack is empty you return a null otherwise you going to now you going to allocate the stack. So first what we do you initialize the top of the stack to be minas one and count is equal to zero remember this is stack which is empty. So doesn't have anything. So you can defined the top of many way in this implementation have a initialize. The stack to be minas one and stack count be equal to zero that is same and then stack max has been initialize to the maximum. Number of elements that we want the stack hold and the stack array that is the complete array defined as allocate maximum element into size of the element size. So if stacks stack array is equal to null. Then that means free all free the stacks always return the null so it done here this you creates at a stack which called stack array which has top is equal to minas one

count is equal to zero and the stack max equal maximum number of element. So next let as look at the push operation so the push stack you have when you remember when I told you told that you have a push stack operation you have two think as to define one is the stack array which is the array in which you want to push into the next is data element that we want to push so first for any push operation infect this is two five any in session operation you have first check whether there is place to instant the element in other words. So for that we are doing a stack check full stack check. So here what you doing is if the count of the number of elements is equal to the maximum number of elements is stack can hold then we are returning zero let us it is over flow and the number of elements in this stack equal to the maximum number you can hold and you cannot add any more elements otherwise when you come out this it means that you have to place the hold elements but in case what you do is you increment the count so the number of elements in the stack increases by one you increment pop by one because now want to create to a place pointed at to this top, there you want to inset the element and then what you do you with array at the position calls top you inset the elements that you want to inset and you return one . just this indicate that the push operation has been success if it already full then you did have return zero which means that there is the problem there is an error and this condition the error is a overfull. So this is how you do please note here we are first incrementing talk to create place and then we are putting the element into the top position so next let look at pop operation again as you know in any deletion operation you should have something to delete so in deletion operation you will get an under flow the same is true for stack. So here what you do you given a stack given data output pointer this is the data you going to output and given a stack you can you did not have the as in input parameter you can gives the stack and you can delete from that but here want to get out what is top on the stack for that's why having the extra parameter some pop operation implementation do not look at this is a separate parameter inside the pop operation. One of the example. We saw people we did not have to parameters for the pop – operation. So here what you do this please remember this is array implementations? So what you do is you first. Check whether stack is empty if it is empty then you cannot do a deletion operation, became you don't know anything to delete. So you getting underflow condition otherwise what you do is you have to take out the them the list of the stack so what you do the access is top element of the stack and take it out is the variable call data output points and decrement the count by one because round in the count number of element is decreased and decrement top also by one because previously whatever top is pointing to is now a empty so end don't want to access is you will decrement all and return one indicate one it is successful operation this is the top operation this function retrieve the data is the top of the stack without

changing the stack so again as we already saw pre-conditions stack is pointer to the stack and post return one its success zero is under flow so have what you do is we have the stack again and this is the output pointer as we did in the pop operation so it the count is greater than zero than only you can do the operation. If it is can then zero you don't have to access at all because these stack in empty you don't have a element so it is next zero that this then element greater than zero that is element so you just take out the elements not take out you just access the element but remember no top changing nothing change is the count does not decrease nothing happens only think is accessing the top element so hear after the top operation this stack remain same is as already told you and returning zero this return zero is if not able to do the operation because of count being not greater than zero which mean that's the stack is this is the condition check whether the stack is empty as next so the pre-condition is stack is pointer to the stack and post condition is if returns one its empty and zero is there is data is this stack so here what you do is again the input the stack and you that's the structure stack. If the count is equal to zero that means is an empty stack. You just Boolean you will return the value that is true value similarly full this function determine the stack is full so fully detained as heat full ok heap full for latter so pre stack is pointed to the stack head note suppose if return one heap for stack is full and zero is heap as room what you do here is this is full stack condition again we have the complete structure now here because its and array implementation then you know the stack is full then the count is equal to these stack max so if count is equal to these stack max so if the count is equal to the maximum element the stack in held then we know that the stack hen full then you return the value now Growable tray – based stack this is the slightly difference diversion of what will apposing to say In a push operation, when the array is full, instead of throwing an exception, we can replace the array with a larger one. How large should the new array be? Incremental strategy: increase the size by a constant C. doubling strategy, double the size, what you mean by here is we docent usually know cannot predict how much the size of stack will be now the problem here is that you have if you put stack is 1000 elements are using one 10 or 15 eliminating wasting are memory resources are then have you put is or 10 elements you against to 20 or 30 elements then you concisely you will getting a stack full condition. So one we doing this is have global array stating what will you do is if you get stack full condition that point you reallocated larger array and then put all the stack elements is to that array so this is not a very simple operation. This is changing of elements from one array to other bus assume that you can do that but two strategists of how the largers array allocated how much of how much should be one think you going to suppose the original stack size is 10 and you a have stack full condition you can increased by constancy size 5 and every time to get a stack full condition keep

increment, increasing it by 5 please not this depends of the explication purely on the application. How you have a doubling stratarge Initially I will put 10 I will act put 10 and then came now will have stack size of 20 again can put 20 and so on. So this is a predict. So depending of the application you do this now we go the next type of implementation plus please not but when I said we going of global array after you stack full condition is array happen when you allocate a larger array that is the lot of completion on work needed transfer this array implementation of next array implementation. So this not to travel operation so this not to say that global array is a solution to the problem of using of array stack operator so next implementation in the link list implementation. So here what you do is you have to understand the concept of an abstract data type again you already seeing this and you need to see understand nested implementation and you need to see some application of link list so every data of course touch a goes also top link list implementation of stack. So this is the typical link list implementation of a stack so here you see that this is conceptual view here the conceptual view of whether you implement of the stack using link list are whether implement the stack using array conceptual view is same but when you look at this now what you have the bottom most element will have the link pointer to be equal to zero and then you have the next element pointing to this element pointing to this, this element pointing to this and count is the number of element in the stack before and top now giving the address of the top most pointer, this link is given the address of the next top most the bottom most element. The address link will be equal to zero this is the link list implementation of the stack. Now let's look at this is no count this is the top this is the before you create a stack now look here you are incremented. This is the a creating a empty stack so what happened this you have count equal to zero and top pointing to null still the stack is empty you just created a stack now let us look at push operation now what happens the count is zero the top most equal to zero now what am doing am creating a node to put green this is the same operation what we saw when we did link list implementation of a stack so what happens after this green I have to change the address of this stack from null to the address of this node and this next here has to be null so next is again a push operation now this has to be now node has to be created and please note that is new node will come here because now this is we comes the bottom most node and the top has point to the most reason node and top most node now what will have to increment the count by two this buy will point to the new node and the next of the new node will pointed to the previous top so this is what happens is the blue has to be removed so now top will point to green again and blue will be removed it's blue will be entered here and blue will be removed from there so if you want to distributed stack what you do this you make a count zero. If you don't have any point of from top address of

top immediately stack not accessible all so disturbed the stack. So here you have two data structure that we using two modes that using one this is stack head other this stack node so this stack will have a count and top which is count an integer and top is a doubt pointer this will have a data type and this will be a link which is again note point this is similar to structure that we used for this but here the operation are restricted I already told you that again this data type can be a single data type or have a large record whatever so the key data structure of this structure that used this CR again a node with key integer data and struck node which is an address and you can also have stack and the stack is represented by a count and the stack node pointing to the top so let us look at operation a stack so when you create a stack you have a pointer to the stack and you have allocate this equal to be size of the stack. So if stack equal to null then you initialize the top and have you going to this is creation of the stack remember and you already seeing here how do you create a stack what do you this to put the count equal to zero and top equal to null that is exactly what you a doing here putting the top equal to null and count is equal to zero and the address of the stack is return this is the allocating the size of the I mean allocating stack to be dynamic allocation of a node.