Data Structure

Lecture 1

Data structure is a systematic way of manipulating data. A data structure structures data. Usually more than one piece of data and we have to provide legal operations on the data and the data can also be joined together. That is it can be collection. Basically data structure talks about two aspects.

One aspect is data organization and other is manipulating data. In fact manipulation of data, how we want to manipulate the data and how do we organize the data. Results in different data structure that we going to talk about. All of you are some of you have done computer programming and they you have used programming languages and there are some data types that are called primitive data types. For example and this holds a single piece of data in java are may be in 'C' most programming languages built in data types called int means integer, char means character ,long stand for long integer and Boolean means Boolean values. There are some Legal operations which is stress for this built in data types like for example for integer: +,-,*, /. Because of the coming of object oriented programming languages abstract data types are get a lot of importance .We talk about data types we also talk about abstract data types. Abstract data types are like a black box. An abstract data type (ADT) is a data type together with the operations, whose properties are specified user of the abstract data type does not have to know what particular implementation used for implementing that particular data type.

When we look from data to data structures actually the machine store only binary data. So that what we can see the figure and then you have primitive data types. For example 23 which is an integer 3.1415 which is a decimal and 'A' is a character and then you have some basic data structures like arrays and structures used to store the data types and then have a high level data structure like stack, queue and list. Stack, queue and list are data structures that define so that does particular operations these types of data structures. We talk about data to data structures actually as we all of you know data stored in a machine in the form of binary bits and that's what machine level data storage and then we have primitive data types that are provided most programming languages like integers, floating point 3.145 and character like A and then have basic data structures that are also provided in programming languages like arrays and structures in addition to this we have high level data structure which are special type of list and structures like stack, queue and list. When you talk about data structures there are two types of data structures one is linear data structure and another one is non-linear data

structures.

Linear structures

Array is common linear data structure. Most important concept of array is that the size of array is fixed. Linked list is a variable size. Then you have a stack is very common data structures. It is a restricted form of list where we can add and delete from one point on top of the list. Then we have queue add to back and remove from front. These are ordinary queue. Priority queue add anywhere remove the highest priority. Other important types of data structures which is very necessary for many application of computer science and one such as a hash table .The hash table commonly used for compilers , storing of variables names compilers and so on.

Hash tables: unordered list which use a 'hash function' to insert and delete. A tree is a very important data structures. It is unlike most of the data structure since it is a non-linear data structure because it has a branching structure. Graph: Graph is a more linear type of data structure than the Tree. Tree has no loops. Graph has less stringent connection conditions there can be loops in that. When we talk about data structures talk about the data organizations then addition to the data organizations what manipulation you can do with this data types its very important concept. Necessarily to some of the operations you normally do in data structures one is creating an empty data structures. Create an empty list. Give a list we have Determine whether a list is empty. Determine the number of items on a list. Add an item at a given position in a list. Remove the item at a given position in a list. Remove all the items from a list. Now actually what are the actions doing have a half a do depends on the application. Some applications just we do look at only addition and deletion. Certain application you do have to find a list is empty. So operation and combination of operations depends on a particular application that you want to a data structure to have and another important operations are data structures.

Another important operation of data structure is to get the item from a given position in a list. Accessing on elements in a list very important operation so in a sorting and searching in any application. For example counseling that you have in Anna University and certain other application. Sorting of the ranks and searching for a particular student then list is a very important operation and mostly all these occupied searching and sorting frequently used operations of data structure. Some time you need the capacity of the list data structure what is the total count of the elements in list. Some time you have needed to modify, update, delete and insert in list. Generally you say that for most applications. This type of modification is not accessed common as action. If we look at the data

structure you can also think of a matrix as a data structures. Linked list as a data structures, a tree and network as a data structures. How this is implemented in how this is represented.

LIST:

This is the fundamental data structure in computer science and this processing is considered as one of the most important data structures. Infract has been the inspiration for the particular type of programming and language called Lisp. List of integers is the most widely used kind of list and that can also be set or collection of instance.

Examples of list are 0,+1,-1,+2,-2,+3,-3 is an example of integer list. We can also have the Days of week = { s,m,t,w,th,f,sa} and sometimes it is not necessary that all elements need to be same type. Instances may or may not be related. {Apple, chair, 2,5.2, red, green, jack}. There is no connection between elements of the list back but tree still that also a list. In many ways list can be classified. But most important classification is unordered and ordered list. Unordered list means insertion list are not in any particular order. Sorted order it could be an alphabetical, numerically but the way in unordered list. Ordered lists are very similar to alphabetical list of employee names. E.g.: The rank of student exam and so on. Difficult example of ordered list. These ordered lists keep items in a specific order and so most manipulation depends on one this order list. When ever an item added to a list. That has to be placed in correct order and similarly when you remove an element from the list. Still have to remain in sorted order.

Linear list:

Example of linear list. It is a sequence of elements. There is something called as a first element and last element and if you take an every element it has a previous and next element is obliviously there is no element before first another is no element in after the last element. So for an example you can represent list.

Linear (or ordered lists)

It could be a linear list or ordered list of the form of each row to e0,e1,e2,..en-1 and assigning that n is finite. And size of the list is obliviously n. e0 is the zero. Here if we look e0 is the front e0 is the zeroth element of the list. En-1 is the last element and ei always precedes ei+1 succeeds en. So there is definite relation between elements of the list. Especially in the case of order list. Example of linear list is Student in some particular course. Jack, Jill, tree, henry, Mary and Judy. Please note that this is not an ordered list.

Linear list examples

Students in comp 2210 = (jack, Jill, able, Hendry, Mary, Judy).

OPERATIONS ON LIST:

First create an empty list. We go and look at each operation in detail as we go long. So create an empty list. Determine whether a list is empty. Get the item at a given position is a list that is normally called as retrieval. Determine the no.of items on a list that is count the item of list. Then given an element finite index of an element. Then traverse the list you want to find the all element in the list. Add as item at a given position in a list. Remove the item at a given position in a list. Remove all the items from a list. You can thing a may such operation depending on an application. But which are discussed some of the very common operation do on any list. So first we look at create an empty list. First look at a create (list). Create (list)

In description is creates a new empty list and the inputs are either it can be no inputs are and i want to create an integer list. Types of elements are list integers are character list or character and output as an only empty list no specific output is given. If we look at this result is a new creating an empty list? You have specified in size. You may have specific the type and you may have want to specify something else also.
Is empty () .You have found whether list is empty or not. So input is obliviously in the list. Output is it a Boolean output. True if one list is empty and gives false if the list is not empty and the result is after still list is remain unchanged. Important operation of the list or linear list is getting an index. Get an element the input is getting in element which has specific an index. L=(a,b,c,d,e), this an example list if you give get(0). Remember that list is start from 0 to n-1. When you say get(0) you get a. when you say get(2) you get c and so on. If you give index that are not valid like -1 you get an error and 9 though it is a valid integer. It is not a index of the list. Maximum of the index list is e0 to e4. Here so that gives an error. The index is given an index want to get an item from the list. So the input is an index. The item to be got that an index. Output is return to the items at the specified index and it throws an exception if index is not valid. But here result for list is unchanged. Throws an exception if index is out of range. The result is remaining in unchanged. Retrieval means which we just discuss. Suppose i give this and i want to retrieval the element called dog. I don't know for examples are given index is one. It retrieval dog for me. List at the top and the bottom do not change. Only for that particular index taking are data element and showing it the use. Size and indexof: Size means determine list size. For example if given L is a,b,c,d,e the size of the list is 5. Important one is index of the element. Here unlike the previous one. When for example given a list a,b,c,d,e given the index of d is 2. Given the index of a is 0. The index of z is -1. Because is not there. The index of d is actually here 3.

Traverse

Traverse is finding all the going to list and get out the elements in particular order. When you say get first you gave an input. Get first (list) - returns first element of it list. Get next (list) returns next element if it exists. Both functions will return null if there is no data at all. Otherwise calling get next in a loop we will get one by one all elements in the list. Linear list operation first one is inserting. You adding given an index add an element and adding an element into the array. So function answer an item to list and the pre condition is list has to be existing. The list should not be full. Give a more explanation of full as we go long. And assuming the no duplicates are allowed. If the item is already in the list does not allowing insertion the element. The post condition is after insertion operation will be in the list.

Suppose if we have list L is a,b,c,d,e,f,g and give an add(0,h) 0 is which are index to be add. L is element to be added. Then this will be a result. L=h{a,b,c,d,e,f,g). Please note the index of a,b,c,d,e,f,g has increase by 1. Value to be provided before the insert an add operation the index of a is 0. After the add operation index of a is 1. Similarly, that when we have want to add to the front of the list. Suppose if you want to add in the second position 0,1, & 2. So here add (2,h) now the index of c,d,e,f will not change or increase by 1. Again complete elements after insertion operation shifted or moved by one place. Suppose if you give add (10,h) the 10 does not exist in the element . That the no of element is <10. Then will get an error and given an (-6, h) also will get an error.

Insert list, data -> new list

Let look at insertion operation. I want to insert an element which is called 25. And i want to insert it. Add the second position. So what happen in the initial list is having 10,20,30 and data want to add 25. And it has been told to the i want to added after 25. After the insertion operation i get 10,20. 25 & 30. Now have be 25 is added to a list and get the new list. Next is removing.

Specifications (Remove index)

Remove an element from the list given an index. Remove an element from the list. Input which an item which you want to remove. And index of an item. And then if an item out of range. If an item is valid has to be removed. An all item in the list will have moved position ok decreased by 1. You will see the example so this is about previous saw about removal next is deleting. Remove and return an element with given index. Remove an element delete the element whose key matches the item's key. List there are some pre condition are delete. First if you have to a list and the key member of an item has been initialized and there are assuming again no duplicate if you find an item finally 1. So item's key will be nothing to only one element. Post condition is delete is key element matching is

key element. So suppose if have a,b,c,d,e,f,g and is given remove 2. Please remember index is 0,1,2,3,4,5. so remove 2 means if will return c. And list will change. Now a,b,c,d,e,f,g and now what has happened is d,e,f and g all the index is d,e,f, and g has increased by 1. If you take an example of this particular list blue, green, red and yellow. And you want to delete an element which called red what will happen after the deletion operation. The list is been blue, green and yellow only the output given red because the red has been deleted. Other operation is remove all. Removes all the elements in the list. Input is none and output is none after you finish the operation the output comes out to be empty. Example for this kind is phone book data phone records. The normal operation you want to do on the list add to an empty list, remove an entry, and look up someone's phone number by name. Name will be given we have to search for that particular name and number. We have to find the list of all the phone number for one entry at a time this like a traversal and get the number of entries is like a count. Add, remove, retrieve, traverse and get the number of items. Here the details are normally in the phone book entries are stored in alphabetical order. Another is shopping list to do list schedule. Normally there are some order and when we talk about items in a list they of the same type. They will be either integer or they will be names of people or they will be name and phone number and so on. And the variation of list include what order you have. For the same list you have alphabetical order and numerical order and what are the types of element in the list and what are the common operations you want to do. Depending upon the application it may vary. What can we can do with the linear list is we can delete element, insert element, find an element and we can traverse a list and so on. You can sort by numerically alphabetically you can do any type of sorting.

Classification of list

When you look at this a specifically linear list and think of 2 types of following classification. Where you have general and restricted list? General list can be unordered and order. Restricted list can FIFO will talk about the queue. And LIFO talk list in detail. Which is the stack? Up to now where not look at the implementation at all. We generally look at as what is the list and how the list can be how the list looks like and be the list of numbers. It can be names can be list of telephone numbers and so on. And we look at what are the operations you can do. All the operation that any application of which user list you what but then if you what to actually represent in this computer. You have to implement using some particular method of implementation.

IMPLEMENTATION OF LIST:

Different ways in which list can be implemented. One is arrays one is linked list and we can also implement list using doubly linked list. Array: how to choose the implementation of whether do you want an array implementation a list implementation depends upon operation you want do to the list. And nothing else and what are the operations you want to do the list depends upon the applications of the list. You have decide you want to do array implementation or linked list implementation depending upon the applications and application will be desired what operation. Operation decided on the application. First we look at the array implementation before look how do to array implementation letters look at what type of operation in list mode easy by the array implementation. Search is easy if you use the array. Traversal is easy use arrays. Insert and delete is not easy. It is fairly operation in an application better to avoid array. One of the major problem in array implementation is before you start the implementation you do not the application is not know how much storage want to an array. For an array, before start an implementation. Ok 100. I want to allocate 100 locations for the particular implementation in the list. But the problem is application can you guess it. To large in a number waste in a memory. So this is the problem of using array. But has array is one advantage it because if the list which is implementation using an array. But has array is one advantage it because if the list which is implemented using an array it mainly using for searching. For traversal then the array implementation of the list is best. So you can also implement list using linked list and slight confusion among the students as what is the list and what is the linked list. Actually list is nothing but list of names, numbers and so on. Nothing to do with the implementation list can be implemented using different data structures like arrays, link list and so on. Linked list data structures is used for implementation is make of the difference in the data structures which we can implement either an array or link list. Linked list not only used for represent the list. Linked list implementation can be used for queue, stack, trees and so on. This is the difference between list and array.

Implementation list using arrays. Array is set of pairs the pair is an index and a value is a data structures. Array is a sequence of indexed items for each index there is a value associated with that particular index suppose you say array one. Array one will a value A, array 5 will have a particular value c. This is a pair that is index along with the value. One of important concepts in array is the length array is fixed when the array is created and the array is consecutive memory location is allocated to the array that is physically as well as logically contiguous memory. Actually what happen when a array is created is sequence of blocks are allocated dynamically to store the value of the array.

Example: we are assuming that two address spaces are used to store content 1228 as 0 and 1230 as 1 so this 01234 if you take it as list it is stored in contiguous

memory location. That is the basic aspect of arrays. Normally when you talk about the array you have in this is ranging from the lower bound. Normally 0 in most programming languages through a higher bound n-1 and n represents total no of elements stored in the array. Any item in the array can be accessed using the index if you say a1 it will be access the first element a5 it will access the fifth element. If you say 1 you will get the element you can access it directly. That why you say the time clock complexity is of the order of one because if you cay one you get immediately go and get the element in the first place and a1000 suppose the array of size thousand. The operation that you do obviously the same the you going represent the list in the array means operations are create initialize then the length property to find the length of the array then insertion deletion and then searching. Searching for the element from the array.

There are many application that uses list and which use arrays for list implementation. One of it is polynomial. Polynomial is defined by list of coefficient and exponents and the degree of polynomial is the largest exponent. This particular polynomial can be represented using list. If you look at link list implementation this is how the link list will be I will talk about this in detail later. One important different between list implementation using array and list implementation using link list is while in array the elements of the list are logically contiguous and physically contiguous and when you use link list for implementation of the list the elements of the list are logically still contiguous but physically need not to be contiguous for example bag, cat, sat and vat. Bag can stored at thousandth location and cat can be stored at three hundredth location sat can be stored at four fifty location and vat can be stored at some 600 location no continuity but still bag, cat, sat and vat are logically connected because of the links between the nodes if you look at the link list implementation. Link list is the chain of elements. If you look at each node of elements or each element it consist of two parts data part and the link part and the link part job is to point to the next element or logically next element in the list. Here a link list will normally have a head pointer which is called a p head that pointed to the first node in the list and each node in the list will have date and link data will store element and link will be the address of the next element and if you have an empty link list it will just represented by the p head.

You can have a node of different types for example a node with one date field. The element and the data stored is number the link is the address if you look at the next one the node with three data field name id and some grdpts are whatever you can any number of data fields. But normally only one link if it is this type of link list and that is structure like name, address and phone and the link can also be represented as structures there are many ways in which link list are implemented we are looking at one we are defining two structure one is list head

some implementations of link list do no separate out this list head bt in this particular implementation we are talking about list head. List head will also have the count of the number of elements as well as a link field so if you look at the left hand side you have list count integer and head is a pointer. Pointer means address. That is the structure heads structures you have a node structure where you have data which is the data type and link which is again a hundreds this is data here we have assumed that data is one field it can have more than one.