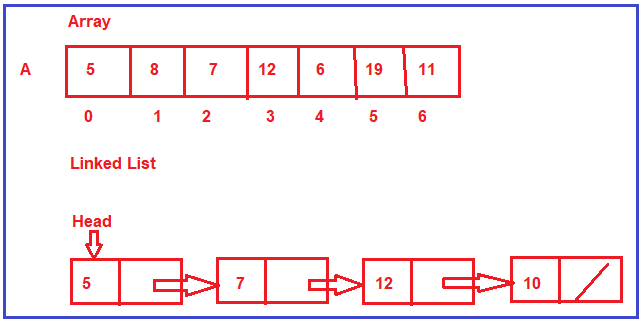
**Physical vs Logical Data Structure:**

In this article, we are just going to give an **Introduction to Logical vs Physical Data structure**. In our previous article, we discussed [**how a program utilized the main memory**](https://dotnettutorials.net/lesson/introduction-to-data-structure-and-algorithm/) by dividing the memory into sections like stack and heap and we also discussed what is static and dynamic memory allocation. Data Structures are categorized as physical data structures and logical data structures.

**Physical Data Structure:**

The Array and LinkedList are the two physical data structures. We can have more physical data structures by taking the combination of these two data structures i.e. array and linked list.

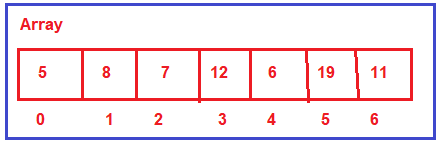


**Why we are calling array and linked list as physical data structures?**

We call array and linked list as the physical data structure because these two data structures decide or define **how the memory is organized or how the memory is allocated**. So, let us look at them one by one

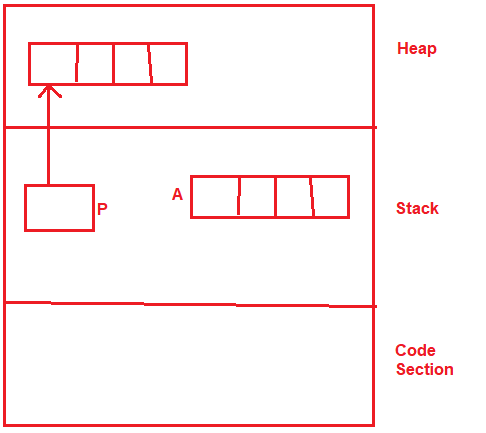
**Array:**

The Array is directly supported by programming languages such as C, C++, C#, Java, etc. The Array is a collection of contiguous memory locations i.e. all these locations are side by side. If we have an array for seven integers, then all these places for seven integers are together. They are in one place. The array will have a fixed size. Once it is created with some size, then that size cannot be increased or decreased. So, it is a fixed size i.e. the size of the array is static.



**Where the array can be created?**

The array can be created either inside the stack or it can be created inside the heap. We can have a pointer, pointing to the array created on the heap. So, the array can be created either inside the stack or inside the heap.

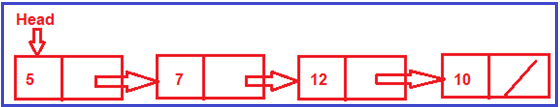


**When to use the array data structure?**

We need to use the array data structure when we are sure what is the maximum number of elements that we are going to store. That means if we know the length of the list then we need to go for the array data structure.

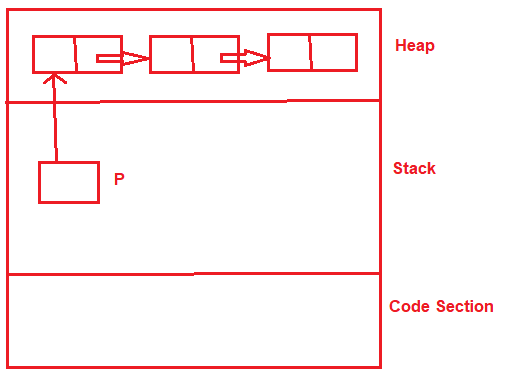
**Linked list:**

The Linked List data structure is a completely dynamic data structure. It is a collection of nodes where each node contains data and a link to the next node. The length of the linked list can grow and reduce dynamically. So, it is having a variable length. As per our requirement, you can go on and adding more and more nodes and add more elements or you can reduce the size.



**Where the linked list can be created?**

The linked list is always created in the heap i.e. collection of nodes is created always in heap and can be accessed using a pointer and that pointer is created inside the stack. So, a linked list is always created in the heap.



**When to use the Linked List data structure?**

We need to go with the linked list data structure when we are not sure what is the maximum number of elements that we are going to store. That means if we don’t know the length of the list then we need to go for the linked list data structure.

These two data structures are physical data structures because they define how the memory should be organized for storing the elements or data. These are more related to memory. I have just introduced these two data structures as there are separate topics in our course for array and linked list where we will discuss array and linked list in detail. Now, let us move on to the next type of data structure that is the Logical Data Structure.

**Logical Data Structure:**

The following is a list of logical data structures.

1. **Stack**
2. **Queue**
3. **Tree**
4. **Graph**
5. **Hash Table**

Note: We will discuss each of the above logical data structures in detail in their respective chapter.

**Difference between the logical data structure and the physical data structure:**

**logical data structures define how the operations are to be performed.**

Physical data structures are actually meant for storing the data in the memory. Then on the stored data or values, we may be performing some operations like inserting more values or deleting existing values or searching for the values, and many more operations. Now, the question is, how you want to utilize those values? How you will be performing insertion and deletion? What is the discipline that you are going to follow? That discipline is defined by logical data structures i.e. stack, queues, trees, graphs, and hash table.

Stack and Queue are Linear Data Structure. Tress and Graphs are non-linear data structures. The Hash table may be a linear or tabular data structure.

Stack works on the discipline of LIFO i.e. Last in First Out. Queue works on the discipline of FIFO i.e. First in First Out. The trees are a non-linear data structure and they will be organized in a hierarchy. The graph is a collection of nodes and the links between the nodes. These data structures are actually used in applications and algorithms.

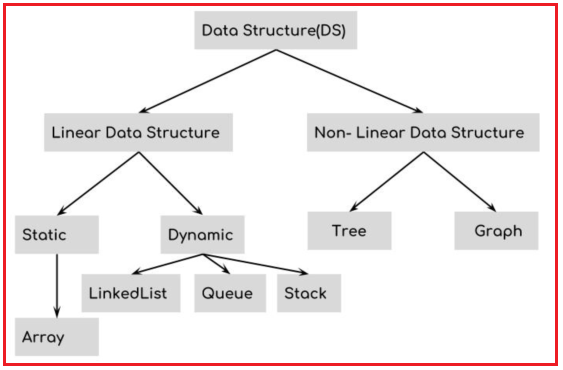
The most important point that you need to remember is for implementing the logical data structures (Stack, Queue, Trees, Graphs, Hash Tables) we either use an array or linked list or a combination of array and linked list physical data structures. So, that is all we have given the introduction of various types of data structures. This was just the introduction to give us awareness.

**Types of Data Structure:**

Data Structure classified mainly into two types.

1. **Linear Data Structure:** A Linear data structure have data elements arranged in a sequential manner and each member element is connected to its previous and next element. Example: Array, LinkedList, stacks, queue, etc.
2. **Non-Linear Data Structures:** A non-linear data structure has no set sequence of connecting all its elements and each element can have multiple paths to connect to other elements. Example: Trees, Graphs.

For a better understanding of the classification of data structure please have a look at the following diagram.



The conclusion of this article is, we wanted to differentiate different types of data structures that are the physical data structure (arrays and linked lists) and logical data structures (stack, queue, trees, graphs, and hash table). The logical data structure is implemented using the physical data structures either using array and linked lists.

**Difference between Static and Dynamic Memory Allocation:**

**Memory Allocation:** Memory allocation is a process by which computer programs and services are assigned with physical or virtual memory space. The memory allocation is done either before or at the time of program execution. There are two types of memory allocations: 

1. [Compile-time or Static Memory Allocation](https://www.geeksforgeeks.org/difference-between-static-allocation-and-heap-allocation/)
2. [Run-time or Dynamic Memory Allocation](https://www.geeksforgeeks.org/what-is-dynamic-memory-allocation/)

**Static Memory Allocation:** Static Memory is allocated for declared variables by the compiler. The address can be found using the [*address of*](https://www.geeksforgeeks.org/address-function-c-cpp/) operator and can be assigned to a pointer. The memory is allocated during compile time.

**Dynamic Memory Allocation:** Memory allocation done at the time of execution (run time) is known as **dynamic memory allocation**. Functions [calloc() and malloc()](https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/) support allocating dynamic memory. In the Dynamic allocation of memory space is allocated by using these functions when the value is returned by functions and assigned to pointer variables.

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Static Memory Allocation** | **Dynamic Memory Allocation** |
| 1 | In the static memory allocation, variables get allocated permanently. | In the Dynamic memory allocation, variables get allocated only if your program unit gets active. |
| 2 | Static Memory Allocation is done before program execution. | Dynamic Memory Allocation is done during program execution. |
| 3 | It uses [stack](http://www.geeksforgeeks.org/stack-data-structure/) for managing the static allocation of memory | It uses [heap](https://www.geeksforgeeks.org/heap-data-structure/) for managing the dynamic allocation of memory |
| 4 | It is less efficient | It is more efficient |
| 5 | In Static Memory Allocation, there is no memory re-usability | In Dynamic Memory Allocation, there is memory re-usability and memory can be freed when not required |
| 6 | In static memory allocation, once the memory is allocated, the memory size can not change. | In dynamic memory allocation, when memory is allocated the memory size can be changed. |
| 7 | In this memory allocation scheme, we cannot reuse the unused memory. | This allows reusing the memory. The user can allocate more memory when required. Also, the user can release the memory when the user needs it. |
| 8 | In this memory allocation scheme, execution is faster than dynamic memory allocation. | In this memory allocation scheme, execution is slower than static memory allocation. |
| 9 | In this memory is allocated at compile time. | In this memory is allocated at run time. |
| 10 | In this allocated memory remains from start to end of the program. | In this allocated memory can be released at any time during the program. |
| 11 | **Example:** This static memory allocation is generally used for [array](https://www.geeksforgeeks.org/introduction-to-arrays/). | **Example:** This dynamic memory allocation is generally used for [linked list](http://www.geeksforgeeks.org/data-structures/linked-list/). |

## ****Abstract Data Type (ADT)****

Let us first understand what is data type and then we will learn abstract data type.

##### ****What does it mean by data type?****

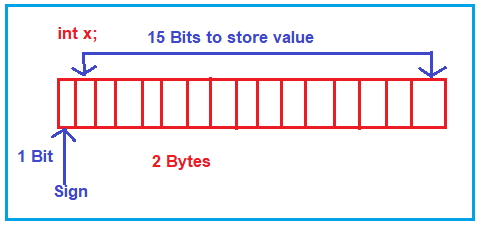
A data type is defined as

1. **Representation of data and**
2. **Operations on data,**

So, a data type is defined in two terms i.e. first thing is how the data is represented or how you are storing the data. The second thing is what are the operations that we allow on the data.

So, if we take an example of integer data type in C or C++ languages, then if we assume that integer data type takes 2 bytes in C or C++, once we declare any variable of type integer in C or C++, then we get the memory of 2-bytes that is 16-bits.

An integer type of data is stored in 2-bytes together as a single value or single data. In which 1-bit is reserved as sign bit that allow both positive, as well as negative number and rest of the 15-bits, are allowed for storing data that is any number. This is how an integer is represented inside the memory in 2-bytes. For better understanding, please have a look at the following image.



So, this is the first thing we have discussed i.e. the representation of data. Now let us understand what all operations can be performed on the integer data type.

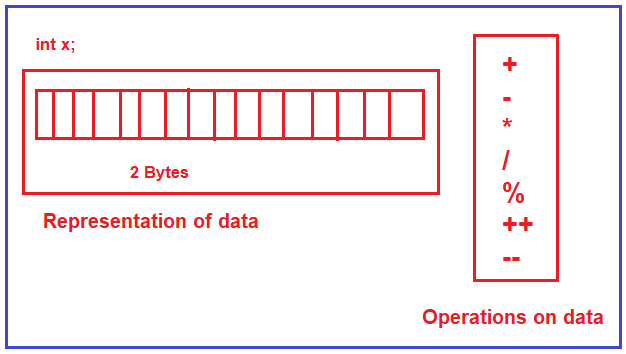
##### ****What are the operations allowed on integer type data in C, C++ language?****

Arithmetic operations are allowed on integer type data, that is **+, -, \*, / and %**. Apart from these, relational operation i.e. >, <, >=, <= as well as increment and decrement operators that is ‘++’ or ‘- -‘ are allowed. These are some of the sets of operations that are allowed on the integer data type.

So, if we talk about data type, any data type in any language then that data type will have its representations and the set of operations that can be performed on the data. When we are learning any programming language, when we learn any data type, we learn mostly about its operations and sometimes we go into detail and also understand its representation.

##### ****What is Abstract?****

**The abstract means hiding internal details.** For better understanding please have a look at the following image which shows the representation and operations that can be performed on integer data. For performing the operations (+, -, \*, /, %, ++, –) do we really need to know, how they are performed in the binary from inside the main memory? The answer is No. We are only concerned about declaring a variable and using it by performing those operations. So, we need not know the internal details of how these operations are performed. These things are hidden from us. So, we can call them Abstract i.e. without knowing internal details we can use it. Internal details are Abstract for us.



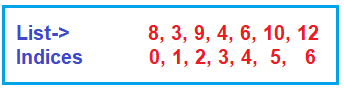
The above is the example of a primitive data type I have taken and explain to you the meaning of data type and the meaning of abstract. This is not an abstract data type, that is a primitive data type. But first I try to explain to you the meaning of data type and abstract and I hope you understand this. Let us proceed further and try to understand the Abstract Data Type.

##### ****Why is Abstract Data Type introduced and what is the meaning of Abstract Data Type?****

Abstract Data Type concept is related to Object-Oriented Programming languages. When the Object programming languages being started used in software Development then using the classes, we can define our own data types. That is Abstract, that is without knowing internal details we can use them. Let us take an example of a list and define it as Abstract Data Type i.e. list of elements or collection of elements.

**List-> 8, 3, 9, 4, 6, 10, 12**

So, here we have taken the list of elements. Now, I can give the indices either starting from 0 or 1 that depends on my requirement. Here, I will start indices from 0 onwards.



Now, I want this list to be used in my program. Now the question is how I can represent a list? What are the things that I have to store for representing the list? We need the following things.

1. Space for storing the elements.
2. The second thing is, we need to have some capacity of a list,
3. The third is, inside that capacity how many elements already we have in the list, that is length of list or size of a list

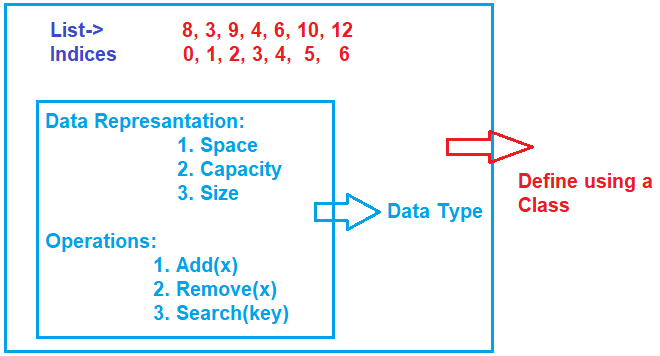
For representing a list, we need three things. Space for storing the elements and its capacity that is the maximum capacity and its size which is the number of elements it is having. So, for the representation of this, we have two options. We can use either array or linked list in a program. So, it can be done using any of these methods (array and linked list). Then let us look at operations on a list.

##### ****What are the operations that we perform on a list?****

The operations that can be performed on a list are as follows. Here we are adding some of the operations.

1. **Add(x)**: To add more elements to the list
2. **Remove(x):** Remove an element from the list
3. **Search(key):** To search a particular element from the list

So, when we have two things, data representations and operations on the data together, then it becomes a data type. Now we can put the above data representation and operations together and we can define a class in C++ or in any other Object-Oriented Programming language. For better understanding, please have a look at the following image.



##### ****How we will be storing this list of elements?****

Either Array or Linked List. Whatever is used this is going to work perfectly and we’ll be performing the set operations.

How the presentation is done, I need not bother about it. When the class is written, I can just create the object of the class and use it. How internally things are working, I need not be a worry. That’s, what is Abstract.

So, the concept of ADT defines the data and the operations on the data together and let it be used as a data type by hiding all the internal details. This concept of ADT is very common in C++. So, we can say that when we write any class in C++ which has the data presentation and operations together it defines an ADT.

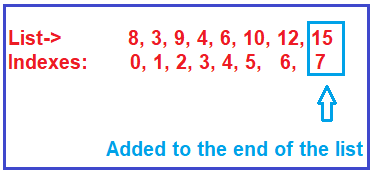
In this course, we are going to learn about various data structures like an array, linked list, stack, queues, graphs, trees, Hash Tables, and all these things we will try to represent them as ADT. So, I will be seeing the code for C language as well as for C++ and then we will see how ADT is implemented. We will be defining all these data structures as ADT (Abstract Data Type) through C++.

##### ****Operations on a list:****

We will look at the operations on a list one by one.

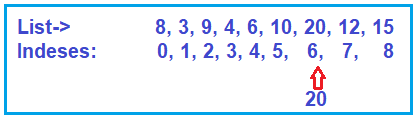
###### **Add(Element) / Append (Element)**

This is used for adding an element to the end of a list. For example, if we want to add 15, then it should be added to the list at the index position 7. This add can also be called as append an element



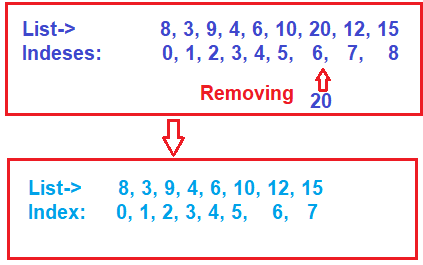
###### **Add(Index, Element) / Insert(Index, Element)**

This is used for adding an element at a given index. For example, if we want to add 20 at index 6. But in index position 6, 12 is there. Then we should shift the elements and make a free space for 20. It means we should move 15 to the next place i.e. index 8 and we should bring 12 to index 7 and then we should insert 20 in index 6. So, if we want to insert any element at a given index then we have to shift the elements. This can also be called as insert at a given index.



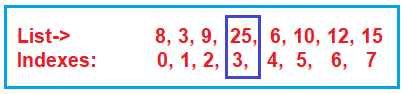
##### ****Remove(index)****

This is used for removing an element and the element which you want to remove, you must give that element index number. For example, if we want to remove 20 i.e. index 6, and when we remove 20 that place will be vacant in the list. Then we have to shift the rest of the elements. So, we will be having a total of 8 elements index from 0 to 7, removing one element rest of the elements are still part of the list.



###### **Set(index, element) / Replace(index, element)**

This is used for changing an element at a given index. Suppose you want to change an element at index 3 to 25. The set can also be called replace, which is replacing an element at a given index with a new element.



###### **Get(index)**

If you want to know the element at a given index then this method you need to use. For example, if you want to know what is there at index 5. Then it will return the value of 10. So, knowing an element from a given index.

###### **Search(key) / Contains(key)**

This is used for searching an element with a given key i.e. searching an element in a list. For example, if you want to search for element 9, yes, it is found there at index 2. The result of the search is we get the index. We know element 9, we want to search where it is in the list, it is at index 2. The search is also called as contains. So, we want to know whether the key element is there in the list or not i.e. the list contains that element or not.

###### **Sort()**

This is used for sorting the list i.e. it is used to arrange all the elements in some.

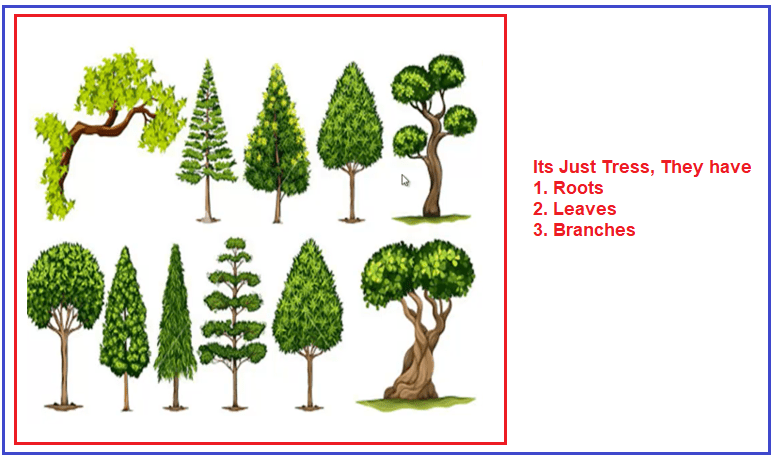
These are the few operations on a list. There are other operations that we can perform such as reverse the list and when there is more than one list, you can combine them or you can merge them or you can split a list. So, a lot of other operations you can perform on a list. As a part of this ADT article, we have taken an example of a list and we have shown, how it is represented and what are the operations.

##### ****Understanding Abstract Data Type (ADT):****

In today’s world, there are more than 500 to 600 known data structures. Some of them are showing in the below diagram. Nothing about it, you have a situation and you have to go and choose the best data structure suitable for your needs.



And it’s really very difficult to remember each of these data structures. So, if you can go and categories them or you can think in an abstract way that would be great. For example, if have trees i.e. you can have a mango tree, you can have an orange tree, or you can have a coconut tree, and so on. If you think in an abstract way, this is just trees. They have roots, they have leaves and they have branches. For better understanding, please have a look at the below diagram.

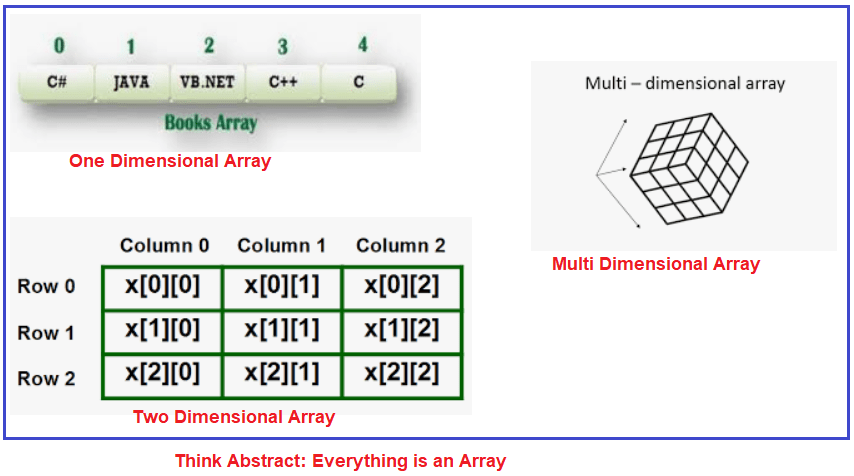


In abstract thinking, we don’t have to go for each implementation and you know abstractly what you want to do. So, from a data structure point of view, instead of remembering all those 500 or 600 data structures, it would be great, if we group them into categories or abstract them into categories.

##### ****Example: Array****

Let us think about the array. An array is one of the data structures. In array we have the following types:

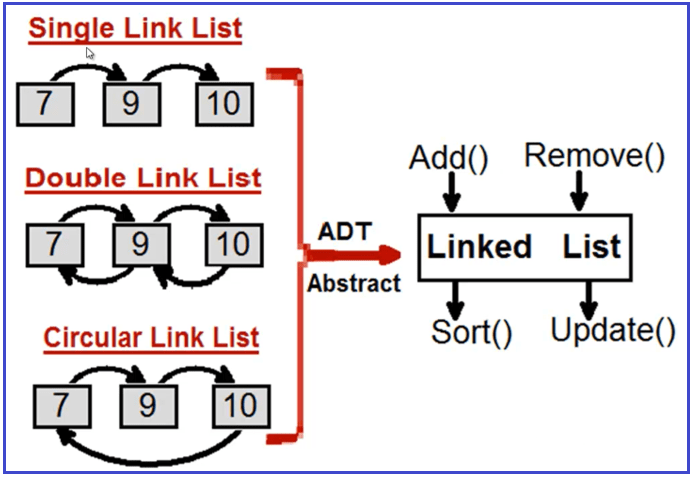
1. One dimensional array
2. Two-dimensional array
3. Multi-Dimensional Array



Instead of remembering a one-dimension array, two-dimension array, multi-dimension array, think just abstract, it is an array. It can have values; it can have size. So, when you think about the abstract, then you don’t need to get into the implementation of each data structure.

##### ****Example: Linked List****

Link List is another data structure. And we have a Single, Double, and Circular Linked List. For better understanding please have a look at the below image.



So instead of remembering Single, Double, and Circular Linked List, think about it in an abstract way. It is just a Linked List that has Add, Remove, Sort, and Update method. So, if you think like this, then those 500 or 600 data structures just come under 5 or 6 ADT (Abstract Data Type).

## ****Time and Space Complexity:****

In this article, I am going to discuss **Time and Space Complexity** with Examples. Please read our previous article where we discussed [**Abstract Data Type (ADT)**](https://dotnettutorials.net/lesson/abstract-data-type/) in detail. Time and Space Complexity is a very important topic and sometimes it is difficult for the students to understand even though it is not that difficult. It is not that complex even though the word complexity is used, it is very simple. First, we will learn about time complexity and then we will learn about space complexity.

##### ****Time Complexity:****

In daily life when we do any work or any task, we want to know how much time it takes for performing that particular task. For example, it is a one-hour task or a one-day task, that amount of time required depending on the work that we have to do. So, usually in our daily life, we measure the time based on the work that we have to do.

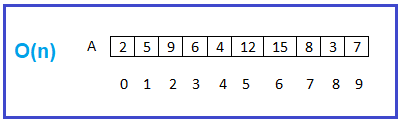
Nowadays, we are using machines to do our work i.e. computers to do our work. We want to know how much time the machine takes for doing the same task. For example, if a person is making one bread in 15-20 minutes, then if we are using a machine for making that bread, then how much time the machine takes? We are interested in that one. If the machine is taking 4-5 hours then it is better to do it manually.

So, how much time a machine takes is very important for us. We use computers for problem-solving. What type of problem-solving? The work that we used to do using pen and paper, that same work we want our computer to do.

So, computers are used for performing computation tasks and computation also needs time. We want to measure how much time a machine will take. Actually, that depends on the process or the procedure for completing that task. The time complexity basically depends on the procedure that we are adopting. Now we are going to discuss variable examples and will see what procedure takes what amount of time.

##### ****Array:****

Let us start with the array. The following is an array with some size and some elements are there. The example shown below has 10 elements. But the question is will it be always 10 elements or depends on the problem? The answer is depending on the problem. So, how many elements are there in the array, let say n number of elements. We don’t know how many elements will be there maybe 10, 10000, or 10000000. We don’t know, so we can say n elements (n means some number of elements). That number of elements may start from 1 and goes up to infinity. We don’t define infinity, so we can say up to the maximum number or whatever that number we can imagine we take that one.

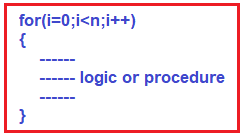


Now what we want to do with all elements? We want to add all of them and so we have to go through all of them one by one and adding them. So how much time it will take? It depends on the number of elements. As there is n number of elements, so the time taken will be n.

The next thing we want to do is we want to search for a particular number i.e. whether element 12 is there or not. Search for it, and it is there in the list. Again, we want to search for element 21, is it there or not? Check for all, No, 21 is not there. At most, how much time is it taking? It depends on the number of elements that we have to compare. So how many elements are there, n elements are there. So, what is the time taken? N time is taken.

It means in a list if you have some n elements and if you are going through all of them just once, then the time is n and we represent this n as a degree. So, we can say order(n) i.e. O(n). Usually use the term degree and order are the same thing). There are other terms like O, Ω, Ɵ that we will discuss at the end of the course because till then you will be having a good understanding of time complexities. So, throughout this course, we’ll be using term order. When we know what are asymptotic notations, like O, Ω, Ɵ then we will understand how to use them.

The next very important thing, if you want to access all the elements of the list i.e. from the array, what is the code that you have to write? You have to write a for loop as shown below. Within the body whatever you want to do, you can. For example, you can write the code for search something or add all of them or count the number of elements, or finding the maximum element. So, whatever you want to do that logic or procedure comes there.



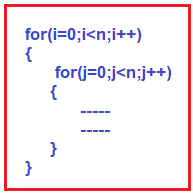
The above for loop is taking us through all those elements. As there is n number of elements so what is the time? It is order(n). For finding the time complexity either you can measure the time based on the work that you are doing or else from the program code you can also find the time complexity. If there is a loop going through 0 to the last element means it is n i.e. it is taking an order(n) time. Whatever it is there inside the for loop, it will repeat the same for n times. So, we can analyze the time based on the procedure as well as based on the code.

The most confusing thing is when the code is given, we get confused about how to analyze this one. So, actually, what the code is doing, you do that work and based on the work you analyze it. It’s very simple. If you don’t want to understand what the code is doing, then it is a very difficult task. So, if a for loop is used means there is a chance that time is order(n).

##### ****Next Situation:****

Now, let us move to the next situation. In that list, I want to compare each element with all other elements. For example, for the 1st element we are comparing or processing with all other elements, and again for the second element, we are processing it with all other elements and so on. This type of processing may be required when we are comparing for sorting purposes.

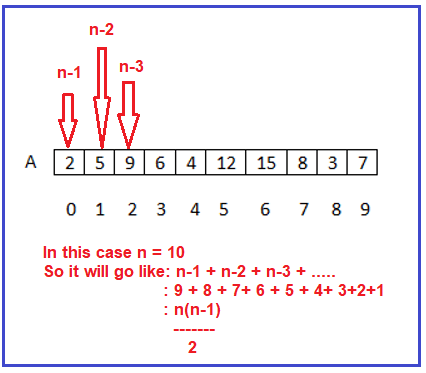
In this case, n elements are being processed but how many times? For each element n elements are processed, this will be n2. We can say order(n2). Now, in the same case, we will see one more thing. For processing like this, how the code should look like? The code should have a nested for loop as shown below.



So, when we have two nested ‘for’ loops, it is n2.

##### ****The third situation for the same array:****

Suppose being on the first element, we want to process the rest of the elements. i.e. n-1 elements. Being on the second element I am processing the rest of the element but not the first element i.e. n-2 element. Similarly, being on the third element we are processing the rest of the elements but not with the 1st and 2nd element i.e. n-3 elements, and so on. For better understanding, please have a look at the below code.

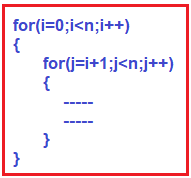


**So, 1+2+3+4+…. +n-3+n-2+n-1**

**=n(n-1)/2**

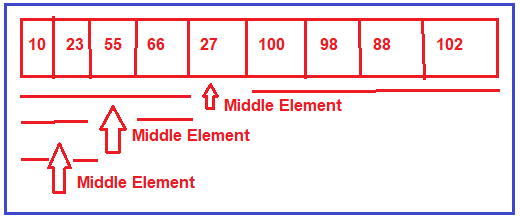
**=(n2-n)2**

So, the degree of this polynomial is n2 i.e. O(n2). Let us see how to write the same thing using code as shown below.

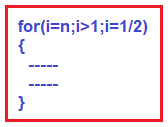


##### ****Fourth Situation:****

Suppose first of all we are processing the middle element. Then again either on the left or on the right side of the middle element, we process the middle element and so on. In this case, we are not processing the entire list, we are processing half of the list, again half of the list, and again half of the list and that process is always dividing the list by two. For better understanding, please have a look at the below image.

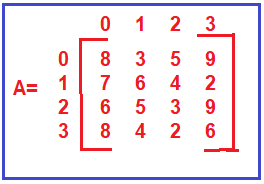


When something is successively divided until it reaches one, that is represented as log and we are dividing it 2 and the number of elements is n log of n elements. So, the time complexity is **log2n**. The code for the above cases is given below.



##### ****Matrix:****

Matrix is having how many elements. The matrix showing in the below image is a 4\*4 dimensions matrix. Total how many elements? If dimensions are n\*n then total n2 elements.



When we are processing upon a matrix then it will require n2 amount of time if we are processing all the elements. If we say we are just processing a row then a row is having n elements so the order is O(n). Similarly, if we are processing a column again it is having n elements so the time will be O(n). If we are processing all elements then it is n2. Now, we will show the code for processing a matrix of elements. We need two nested ‘for’ loops as shown below.

