**INTRODUCTION**

A linked list is a linear data structure. Where data not stored sequentially inside the computer memory but they are link with each other by the help of address. It is very commonly used linear data structure which consists of groups of nodes in a sequence. It is a collection of data element called node. Every node has two parts one is informative and second is pointer or address part which link next element in the list.

A collection of homogeneous elements is called An Array. Array are always stored in consecutive memory location. This makes it easier to calculate the position of each element by simply adding an offset to a base value It can be stored multiple values which can be referenced by a single name.

* Array is a collection of homogenous data type element whereas the linked list is a collection of unordered linked elements known as nodes.
* In array the elements are stored in continuous memory location but in linked list the element can be stored anywhere in the memory.
* Array worked static data structure whereas linked list worked with dynamic data structure.
* Array elements are independent to each other but linked list elements are dependent to each other.
* Array takes more time whereas linked list takes less time.

**COMPARISON**

1. **Linked-List**

A linear Data Structure, linked list is the Collection of different data elements which is not stored in neighboring memory Locations. It contains the nodes which have data and next pair, data stores data element and next stores the reference of next node’s data knows as Pointer.[6]

**Types of Linked-List**

* ***Single Linked List:*** The normal linked list is also called Single Linked List, which is the collection of data elements in the linear way alike the Array.

**DATA**

**NEXT**

**NONE**

**START**

**A**

**C**

**B**

* ***Circular Linked List*:** The linked list that forms the circle by connecting all the nodes. In this linked list the last pointer stores the reference of the first data element so that the cycle is continued until it is stopped.

**START**

**A**

**C**

**B**

* ***Doubly Linked List*:** The linked list that contain one more pointer known as previous pointer. So the linked list contain two pointers in a node, first pointer stores the reference of the data of next node and second pointer stores the reference of the data of previous node.

**NONE**

**PREV**

**NEXT**

**NONE**

**START**

**A**

**C**

**B**

**Basic Operation in Linked List**

* ***Traverse*:** All elements of linked list is displayed one by one.
* ***Insertion*:** Add a node at beginning or between two node or at the End of the linked list.
* ***Deletion*:** Delete a node of the linked list

**Implementation of Algorithms:**

Some of the algorithms of linked list are implemented in python is given below;

**Traverse**

Traverse is defined as the visiting or touching the data element in the linked list or in any data structure and it also display all the elements of the linked list separately.

**Code:**

**class** node:  
 **def** \_\_init\_\_(self,data):  
 self.data=data  
 self.next=**None**;  
**class** LinkedList:  
 **def** \_\_init\_\_(self):  
 self.start=**None**;  
 **def** traVerse(self):  
 **if** self.start== **None**:  
 print(**"Empty Linked list"**)  
 **else**:  
 tr=self.start  
 **while** tr!=**None**:  
 print(tr.data,end=**""**)  
 tr=tr.next  
list1=LinkedList()  
list1.traVerse()

**Deletion**

Deletion is defined as the rearranging of the elements after removing the elements already present in the linked list or in any other data structure.

**Code:**

**class** node:  
 **def** \_\_init\_\_(self,data):  
 self.data=data  
 self.next=**None**;  
**class** LinkedList:  
 **def** \_\_init\_\_(self):  
 self.start=**None**;  
 **def** Delete(self):  
 **if** self.start==**None**:  
 print(**"Empty Linked list"**)  
 **else**:  
 self.start=self.start.next  
list1=LinkedList()  
list1.Delete()

**Insertion**

Insertion is defined as the adding of data elements in the start or in between or in the End of the linked list or in any other data structure.

**Code:**

**class** node:  
 **def** \_\_init\_\_(self,data):  
 self.data=data  
 self.next=**None**;  
**class** LinkedList:  
 **def** \_\_init\_\_(self):  
 self.start=**None**;  
 **def** insert(self,value):  
 Nnode=node(value)  
 **if** self.start==**None**:  
 self.start=Nnode  
 **else**:  
 tr=self.start  
 **while** tr.next!=**None**:  
 tr=tr.next  
 tr.next=Nnode  
list1=LinkedList()  
list1.insert(11)

**Time Complexity:**

In linked list to access and search the element we have to traverse node to node till the End. So, the Time complexity of Access and Search is O(n).

In insertion the prev node is re-assign to next node in the linked list to insert the new element in the linked list. So, the time Complexity of Insertion is O(1).

In Deletion the prev node is re-assign to next node in the linked list to deletes an element in the linked list. So, the time Complexity of Deletion is O(1).

“Worse case” is same as the above case.[7]

**Space Complexity**

In linked list the space complexity is O(n) for single as well as for doubly linked list so that Doubly linked list has the advantage over single which is that the traverse is performed in both forwards as well as backwards direction but the memory is same as the single use.

The Space complexity of linked list is O(n).

**Applications:**

***Computer World*:** Linked list is used in the Stacks and queues implementation, and in the implementation of the advance data structures (Fibonacci Heap). This is also used in free blocks, preforming Arithmetic operations on large integers, maintaining Directories and Sparse Matrices representation.

***Real World*:** Linked list is used in Image Viewer to next and pervious the image, Music player to next and pervious the song and also web browser to next and previous the web page.[8]

**Advantages**

They are strong data structure can grow or shorten during program implementation.

Effective memory consumption: memory is not pre assigned where it is required and unallocated when no coating is needed.

Insertion and deletion are unchallenging and well organized: provides adaptability in insertion of data at specified place and deletion from stated position.

**Disadvantages**

Supplementary Memory: If more the fields more memory will be required.

Entrance to a random data item is bit awkward and also time absorbing.

Difficult to sort the elements of the linked list.

1. **Array**

An array is the simplest data structure; it is a sequent collection of elements of similar data type and it stores a fixed number of data items in a consecutive memory location. Indexes are used to assess their data Items in array. The index range of an array of size N starts from 0 to N-1.

Index

10 20 30 40 50 60

0 1 2 3 4 5

Value

Array length = 6

**Types of Array**

The Followings are the Types of Array

1- One dimensional array

2- Multi-dimensional array

1- One dimensional array

A one-dimensional array is a sequential collection of elements (often called array elements) that can be accessed specifically by specifying the position of the elements with their index values.

2- Multi-dimensional array

A multidimensional array has multiple indexes of each element in the array. The most generally used multidimensional array is the two-dimensional array or 2-D Array, it is also known as a table or matrix. A two-dimensional array has two indexes of Each element in the array. 2D array is identify by the notation (row, column)

**Basic Operations in Array**

* ***Traverse*** − it will show all the elements in array one by one
* ***Insertion*** − it will add element in array of given Index
* ***Deletion*** − Deletes an element at the given index.

**Implementation of Algorithms:**

**Traverse**

Traversal is completed by starting with the primary element of the array and getting to the last. Traversal operation are often utilized in counting the array elements, printing the values stored in an array, updating the prevailing values or summation all the element values. If a user wants to try to do similar calculation on each element of the array, it is going to use traversal operation.

**Code:**

**import** array **as** arr

y=arr.array(**"i"**,([1,2,36,5,8,56)]))

**for** i **in** range(6):

y[i]=v

print(v)

**Deletion**

Deletion operation is use to removing a present element from the array and re-forming all elements of an array.

**Code:**

**import** array **as** arr  
y=arr.array(**"i"**,([22,26,25,25,6,26]))  
n=len(y)  
**for** i **in** range(n):  
 v=int(input(**"Enter the vlaues"**+str(i)))  
 y[i]=v  
print(y)  
j=n-1  
pos=2  
item=7  
**while** j<pos:  
 y[j-1]= y[j]  
 j=j+1  
y[pos]=item  
n=n-1  
print(y)

**Insertion**

Insert operation is use to insert one or more data items into an array. Based on the need a new data item can be inserted at the any given index of array or beginning, end.

**Code:**

**import** array **as** arr  
y=arr.array(**"i"**,([22,26,25,25,6,26]))  
n=len(y)  
**for** i **in** range(n):  
 v=int(input(**"Enter the vlaues"**+str(i)))  
 y[i]=v  
print(y)  
j=n-1  
pos=2  
item=7  
**while** j>=pos:  
 y[j+1]= y[j]  
 j=j-1  
y[pos]=item  
n=n+1  
print(y)

**Applications**

Arrays are implemented in database records.

Arrays can be used for CPU planning.

Arrays can be used for sorting purpose. Multiple sorting methods like Insertion sort, Selection sort Bubble sort, etc. use arrays to store and sort elements easily.

We can implement Adjacency list implementation of graph uses vectors using arrays.

Data structures such as map, heap, balanced binary trees and set use binary search tree and which uses are implemented using arrays.

Arrays are used to preserve multiple variables with the same name.it helps to maintain large data

stack and queues are also implemented by array

We can use array to implementvectors and lists.

A real time application of array is if we want to store the contacts numbers on our phone, it will simply store all our contacts in an array.