Conceptual Discussion on Operations of Array: Traversal, Insertion & Deletion

PANKAJ KUMAR GUPTA
Assistant Professor & Head, BCA Department
Durga Prasad Baljeet Singh (PG) College, Anoopshahr
District Bulandshahr (Uttar Pradesh)

DR. P. K. TYAGI
Associate Professor & Head, Dept.of Statistics
Durga Prasad Baljeet Singh (PG) College, Anoopshahr
District Bulandshahr (Uttar Pradesh)

Abstract:
This Paper describes full discussion about theoperations of Array Data Structures like: Traversal, Insertion, Deletion. Discussions about Algorithms and Procedures of same above defined operations also made in this paper.

Keywords: *Array, Homogenous, Contiguous, Linear, Primary, Lower Bound, Upper Bound, SDA, Insert, Delete, Traverse, Sorted, Unsorted, Complexity, Worst Case, Best Case, Average Case.*

1. Meaning of array

Computer Application we define Array as: it is a Linear Data Structure. It is also termed as a collection of homogenous data elements. All the elements of an array share common name and stored on contiguous locations. It is also categorized into derived data types.

1.1 Example

1. A collection of rollnumbers of 50 students(int).
2. A collection of marks of 50 students in 6 subjects for a class(float).
3. A collection of names of 50 students(char).

2. Categories of Arrays

Arrays are categorized into 3 categories that are defined as follows:

1. Single Dimensional Arrays.
2. Two Dimensional Arrays.
3. Multidimensional Arrays.

3. Operations of Array Data Structures

Various operations are performed on Array Data Structure, that are as follows:
Here Discussion is made on Single Dimensional Array.

1. Traversal
2. Insertion
3. Deletion
4. Searching
5. Sorting
6. Merging

But Traverse, Insert & Delete operations are discussed in this paper.

3.1 Traversal

Traversal is a process of visiting each and every node of a list in systematic manner. That means to go through each and every item of the list at least once.

3.1.1 Algorithm TRAVERSEARRAY(A,N,LB,UB)

/* Here TRAVERSEARRAY is an algorithm which is used to traverse all the elements of Array A with N elements. LB is Lower bound & UB is the Upper bound of Array. */
Step-1 Start
Step-2 set K=LB.
Step-3 Repeat steps 4 and 5 while K<=UB
Step-4 PROCESS A[K].
Step-5 set K=K+1.
Step-6 Exit.

Since anybody goes through all elements one by one to traverse the array of size N, So the Complexity of traverse operation will be O(N).

3.1.2 Insertion

Insertion is a process of adding one or more items in the given list. If an array of elements is given and an item is said to add at specific location. If array is already full then insertion operation results "Overflow". To insert the item into array, all the elements from that location are shifted to the right side of the Array. When we insert an item in an array, there may be two cases: 1. Array is in unsorted order 2. Array is in sorted order. First of all, we will discuss about insertion in unsorted array then about insertion in sorted order.

3.1.3 Case 1: if Array is in Unsorted Order

Specific location,on which item is to be inserted, may in:
1.Beginning of the Array
2.Middle anywhere
3.Last of the Array
Here Lower bound of index is considered to be as 0.
Array A is given as A[5]={1,2,3,4}

1	2	3	4	
0	1	2	3	4

Item to be inserted is 55
Location is:

At Beginning (That is before element 1): New array will be like:

55	1	2	3	4
0	1	2	3	4

Since all elements are shifted by one position in this case to store a new item, so complexity for inserting at beginning in the array of size N will be O(N). it will be the Worst Case of Complexity.

In Middle anywhere (Loc=1 as index)
(That is before element 2) New array will be like:

1	55	2	3	4
0	1	2	3	4

Since, first of all, location will be searched before that item is to be inserted and all elements after that item are shifted by one position in this case to store a new item, so complexity for inserting before any specified location in the array of size N will be O(N). it will be the Average Case of Complexity.

At Last: (That is after element 4) New array will be like:

1	2	3	4	55
0	1	2	3	4

Since no element is shifted in this case to store a new item, so complexity for inserting at last in the array of size N will be O(1). it will be the Best Case of Complexity.

3.1.4 Algorithm INSERTARRAY (A, N, ITEM, LOC)

/* Here INSERTARRAY is an algorithm which is used to insert an ITEM on the Location LOC in an Array A with N elements. Lower bound of A is considered to be 0. */
Step-1 Start
Step-2 set J=N.
Step-3 Repeat steps 4 and 5 while J>=LOC
Step-4 set A[J]=A[J-1].
Step-5 set J=J-1.
Step-6 set A[LOC]:=ITEM.
Step-7 set N:=N+1.
Step-8 Exit.

3.2 Case 2: if Array is in Sorted Order

An item is to be inserted before a specific element, in this case item that is to be inserted is compared with last element and if item is less than the comparing element, we shift element to one position right side and same procedure is repeated again & again until comparing element is less than or equal to the item, if item is greater than or equal to the comparing element, then we replace the element which is located after the comparing element with item. Here Lower bound of index is considered to be as 0.
Array A is given as A[5]={11,22,33,44}

11	22	33	44	
0	1	2	3	4

Item to be inserted is 30
In this case 30 is compared with 44 & since 44 > 30, 44 is shifted to index 4 from index 3.
New array will be like:

11	22	33	44	44
0	1	2	3	4

Now 30 is compared with 33 & since 33 > 30, 33 is shifted to index 3 from index 2.
New array will be like:

11	22	33	33	44
0	1	2	3	4

Now 30 is compared with 22 & since 22 < 30, element 33 on index 2 will be replaced with 30.
New array will be like:

11	22	30	33	44
0	1	2	3	4

If item is smaller than all elements of array then all elements will be shifted one position to right (Worst Case), so Complexity for inserting an item in sorted array of size N will be O(N).

If item is greater than all elements of array then no element will be shifted (Best Case), so Complexity for inserting an item in sorted array of size N will be O(1).

In another case except previous cases (Average Case) Complexity for inserting an item in sorted array of size N will be O(N).

3.2.1 Algorithm INSERTARRAY (A, N, ITEM, LOC)

/* Here INSERTARRAY is an algorithm which is used to insert an ITEM in asorted Array A with N elements. Lower bound of A is considered to be 0. */
Step-1 Start
Step-2 set J=N.
Step-3 Repeat steps 4 and 5 while K>= 0&
A[J] > ITEM.
Step-4 set A[J]=A[J-1].
Step-5 set J=J-1.
Step-6 set A[J+1]:=ITEM.
Step-7 set N:=N+1.
Step-8 Exit.

3.2.2 Deletion

Deletion is a process of removing one or more items from given list. If an array of elements is given and an item is said to remove from specific location. If array is already empty then deletion operation results "Underflow". To delete the item from array, all the elements from that location are shifted from right to the left side of the Array in this way the element that is to be deleted is overwritten by next element right to the same location.When we delete an item from an array: there may be two cases: 1. Array is in unsorted order 2.Array is in sorted order. First of all we will discuss about deleting an item from unsorted array then about deleting an item from sorted order.

3.2.3 Case 1: if Array is in Unsorted Order

Specific location, on which item is to be deleted, may in:

1. Beginning of the Array
2. Middle anywhere
3. Last of the Array

Here Lower bound is considered to be as 0.
Array A is given as A[5]={1,2,3,4,5}

1	2	3	4	5
0	1	2	3	4

Item to be deleted is 1 on location 0 (i.e. Beginning)
New array will be like:

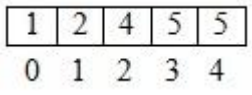
2	3	4	5	5
0	1	2	3	4

Now Size of Array will be decreased by 1 (i.e. 5-1=4).

2	3	4	5	~~5~~
0	1	2	3	4

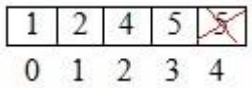
Since all elements are shifted by one position in this case to delete an item, so complexity for deleting an item from beginning of the array of size N will be O(N). it will be the Worst Case of Complexity.

Item to be deleted is 3 on location 2 (i.e. Middle anywhere)
New array will be like:



d

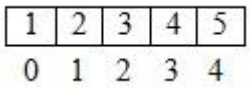
Now Size of Array will be decreased by 1 (i.e. 5-1=4).



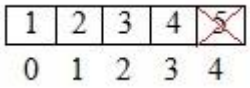
Since, first of all, location will be searched before that item is to be deleted and all elements after that item are shifted by one position in this case to delete an item, so complexity for deleting any specified location in the array of size N will be O(N). It will be the Average Case of Complexity.

Item to be deleted is 5 on location 4 (i.e. from Last)

New array will be like:



Now Size of Array will be decreased by 1 (i.e. 5-1=4).



Since no element is shifted in this case to delete an item, so complexity for deleting an item from last position of the array of size N will be O(1). It will be the Best Case of Complexity. In all situations we decrease the size of array by 1 so that last element is removed from Array.

3.3 Algorithm DELETEARRAY(A,N,ITEM,LOC)

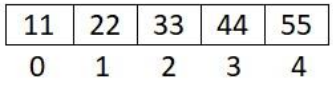
/*Here DELETEARRAY is an algorithm which is used to delete an ITEM from the Location LOC in an Array A with N elements. Lower bound of A is considered to be 0. */

Step-1 Start
Step-2 set J:=LOC.
Step-3 set ITEM:=A[LOC].
Step-4 Repeat steps 4 and 5 while J<N
Step-5 set A[J]:=A[J+1].
Step-6 set J:=J+1.
Step-7 set N:=N-1.
Step-8 Exit.

3.3.1 Case 2: if Array is in Sorted Order

An item is to be deleted from an Array of Size N, in this case item that is to be deleted is compared with the first element and if item is less than the comparing element, we shift on next element to compare and repeat this procedure until the element is found. If item is found, we overwrite this element with the next to this found element and all the elements except last are also overwritten with next to that elements. If item is not found then no element will be deleted.Here Lower bound of index is considered to be as 0.

Array A is given as A[5]={11,22,33,44,55}



Item to be deleted is 33

In this case 33 is compared with 11& since 11 < 33, then we shift from 11 to 22, due to same condition we shift from 22 to 33, on comparing 33 (i.e. deleting element) with 33 (on index 2) we found both are equal so 33 (on index 2) will be overwritten by next element (i.e. 44).
New array will be like
:

11	22	44	44	55
0	1	2	3	4

Now 44 (on index 3) will be overwritten by next element (i.e. 55).
New array will be like:

11	22	44	55	55
0	1	2	3	4

Now Size of Array will be decreased by 1 (i.e. 5-1=4).
New array will be like:

11	22	44	55	~~55~~
0	1	2	3	4

If item is found on last place or may be item (i.e. to be deleted) greater than all elements of array then all elements will be compared to the deleting element to delete the item(Worst Case), so Complexity for deleting an item from sorted array of size N will be O(N).

If item is found on first place than same element is deleted and no further comparison is required (Best Case), so Complexity for deleting an item from sorted array of size N will be O(1).

In another case except previous cases (Average Case) Complexity for deleting an item from sorted array of size N will be O(N).

3.4 Algorithm DELETEARRAY(A,N,ITEM,LOC)

/*Here DELETEARRAY is an algorithm which is used to delete an ITEM from the Sorted Array A with N elements. Lower bound of A is considered to be 0. */

Step-1 Start
Step-2 set J:=0.
Step-3 Repeat steps 4 while J<N &
A[J] <= ITEM
Step-4 set J:=J+1.
Step-5 set J:=J-1.
Step-6 set ITEM:=A[J].
Step-7 Repeat steps 8 and 9 while J<N
Step-8 set A[J]:=A[J+1].
Step-9 set J:=J+1.
Step-10 set N:=N-1.
Step-11 Exit.

4. Conclusion

An array can also be termed as a collection of homogeneous values. Anybody can treat an array as a single object by referring to it through a variable. Variable name is succeeded by square brackets (that is [ ] symbols) But you can also treat the components of the array as if they are themselves variables. Elements of same type as array can be inserted and deleted in arrays at any place also. Complexity of Worst and Average case in insertion in unsorted array is same and is different from Best Case.

Complexity of Worst and Average case in deletion in unsorted array is same and is different from Best Case. Complexity of Worst and Average case in insertion in sorted array is same and is different from Best Case. Complexity of Worst and Average case in deletion in sorted array is same and is different from Best Case.

References

1. Computer Science with C++ By SumitaAroraby SumitaArora.
2. Let Us C by YashavantKanetkar
3. Sartajsahni, "Data structures, algorithms and applications in C++", University press.
4. Seymour Lipschutz. "Theory and problems of data structures", Tata Mcgraw hill international editions".
5. The C Programming Language by Brian W. Kernighan / Dennis Ritchie
6. www.en.wikipedia.org/wiki/Array.
7. www.geeksforgeeks.org
8. www.w3schools.com