ASSOCIATION OF ARRAY, LINKED LIST, BINARY AND LINEAR SEARCH

Nancy Bansal[1], Parul Choudhary[2]

[1]B.E Scholar , Department of CSE, KITE, Raipur [C.G]

[2]Asst. Proessor , Department of CSE, KITE, Raipur [C.G]

Email-id: bansalnancy22@gmail.com, cparul2605@gmail.com

ABSTRACT

For storing large amount of data, computer system is generally used according to different searching techniques. Important issue is to make easy searching fast with efficient storage. In this paper, we study performance of array and linked list data structure and searching techniques like binary and linear search. Searching an element from the list is the elementary aspects in software world. There are various numbers of algorithms are developed for searching an element among which linear search and binary search are the most popular algorithms. We have made efforts to implement on various data structures and also to explain their algorithms. At last evaluate all algorithms with the help of bar graph

Keywords: Array, *linked list, binary and linked list.*

1.Introduction

We mostly perform three steps to perform various operations on data take input, process it and get back output. For example if we want to search location of particular city on goggle map, we simply give initial and final point as input, also while logging in to your mail; we give your mail id and password. Likewise, in last step, application gives us output in terms of site open or mail is open.

Computers can store, retrieve and process large amount of data. Data Structure is an arrangement of data in computer's memory in such a way that it could make the data quickly available to the processor for calculations. Data Structure tells us how the data will be stored?? And what operations will be performed on it?? Data Structure can be classified in two categories – *linear structures and hierarchical structures.* Arrays and linked lists are linear structures while tree and graphs are hierarchical structures.

Arrays are statically implemented data structures. The size of array is fixed and cannot be altered at runtime. We can store similar types of data either be a integer, character or float data type. In array insertion and deletion of elements are situation dependent. Elements are stored in a consecutive memory location.

Linked lists are dynamically implemented data structures. Memory is allocated whenever required and can be de-allocated when it is no longer needed. It is a special list of some data elements linked to one another. They linked with each other through POINTER who points another element. In linked list insertion and deletion are easier and efficient.

In array and linked list if we want to check whether element is present or not then we use searching technique. The searching would be successful if and only if given element is found. Searching helps to get location of element in array. Other case if element would not found then we called that searching is unsuccessful. For searching the elements in array there are two most popular methods – Linear search and Binary search.

In Linear Search, we can access elements of an array one by one sequentially. It will start searching element from leftmost element and go till the end of the array and compare each element with the searched element and if element is matched with searched element it will return the position of the searched element and if element is not matched it will return element is not present in array.

In Binary Search, we compare the particular element by comparing it with the middle most element of the array. If a match occurs then the middle position of array is returned. If middle element is greater than then the element, then the element is searched in the sub-array to the left of the middle element. Otherwise, the element is searched in the sub-array to the right of the middle element. The process continues until the size of the sub-array reduced to zero.

2.Description

A] ARRAY : Array contains a group of elements. These elements are of the same data type like integer, float, character, or string. On Array we can perform various types of operations such as Insertion, Deletion, Updating, Traverse, Sorting, or Searching. Each element is identified by a unique index number. Instead of declaring individual variables such as A1, A2, A3, A4… A100. We can declare one array name as A and use A[1], A[2], A[3], A[4],….,A[100] to represent individual variables. All arrays consist of contiguous memory location. Lowest index number represent first element and highest index number represent last element of array.

Syntax –

i] One Dimensional Array : *data_type variable_name[ARRAY_SIZE];*

This is the Syntax of One Dimension Array. The ARRAY_SIZE must be a integer constant and greater than zero and data_type must be valid.

Example – We have to store 10 values of integer type in array with variable_name amount like- int amount[10];

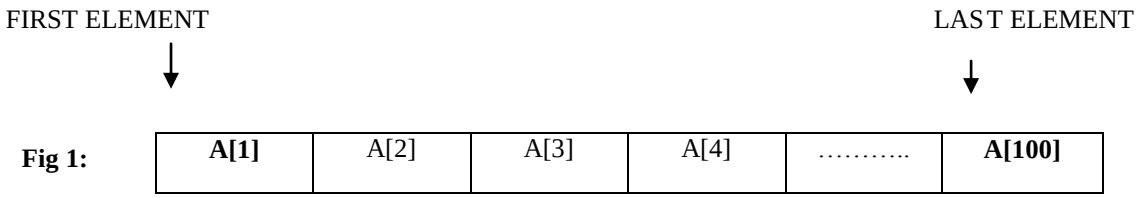

Fig 1:

Diagrammatical Representation of One-Dimension Array.

Algorithm : PRINT(A)

Step 1: START

Step 2: [INITIALIZE ARRAY] A[j] WITH VALUES 1,2,3,4,5,6

Step 3: PRINT “DISPLAYING VALUE IN ARRAY”

Step 4: FOR j = 1 to 6

Step 5: PRINT A[j]

Step 6: END OF FOR LOOP

Step 7: PRINT “DONE”

Step 8: END

Explanation : In this Algorithm in Step 2 we initialize array of name A[j] with their values, and in Step5 we print all values of array using FOR loop

ii] Two Dimensional Array: data_type variable_name[X][Y];

This is the Syntax of Two Dimension Array. It can be considered as table where [X] represent the no. of row in table and [Y] represent the no. of column in table.

Example – We have to make a table of 3rows and 3columns : int table[3][3];

	Column 0	Column 1	Column 2
Row 0	**table[0][0]**	**table[0][1]**	**table[0][2]**
Row 1	**table[1][0]**	**table[1][1]**	**table[1][2]**
Row 2	**table[2][0]**	**table[2][1]**	**table[2][2]**

Table 1: Diagrammatical Representation of Two-Dimension Array.

***B*] LINKED LISTS:** If the memory allocation is done before the execution of any program, then it is fixed and cannot be changed further. Therefore, We have to take an alternative way to allocate memory only when is needed. There is a Unique Data Structure known as Linked Lists, which provides more storage and flexible system. Linked lists are dynamically implemented data structures. Memory is allocated whenever required and can be de-allocated when it is no longer needed. The logical ordering represented by having each element pointing to the next element. Every element is called **NODE,** it has 2 parts, information which is stored in **INFO** and **POINTER** which points the next element.

The basic operations operations of Linked List are Creation, Insertion, Searching, Deletion, Traversing, Concatenation, Display.

Types of Linked Lists: Basically, they are of 4 types –

1] SINGLY LINKED LIST:

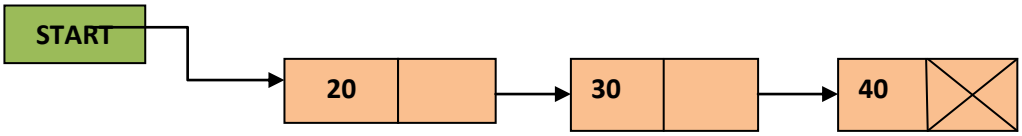

Fig 2: Diametrical Representation of Single Linked List.

2] DOUBLY LINKED LIST:

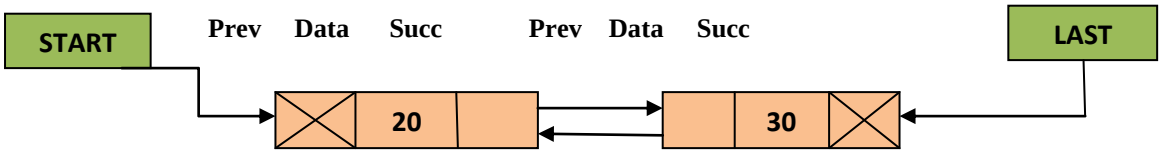

Fig 3: Diametrical Representation of Doubly Linked List.

3] CIRCULAR LINKED LIST:

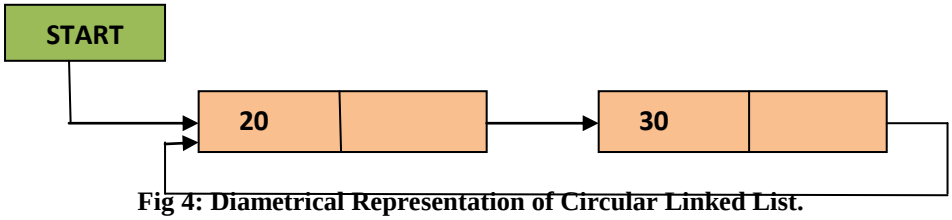

Fig 4: Diametrical Representation of Circular Linked List.

4] CIRCULAR DOUBLY LINKED LIST:

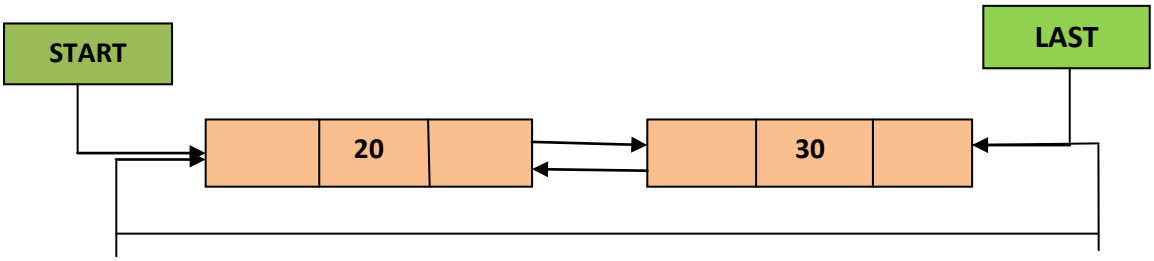

Fig 5: Diametrical Representation of Circular Doubly Linked List.

Algorithm : INSERT(PTR, ELEMENT)

Step 1: [Check Overflow]

if PTR = NULL, then Print Overflow EXIT

else PTR = (Node *) malloc (size of (node)). [END if]

Step 2: Set PTR → INFO = ELEMENT

Step 3: Set PTR →NEXT

Step 4: END

Explanation — In Step1 we first of all check overflow and if **PTR** is NULL then we print Overflow and if **PTR** is not NULL then we create new node and enter **ELEMENT** to that node.

C] LINEAR SEARCH: Linear Search is a search in which we access each element one by one sequentially. This search is also known as **SEQUENTIAL SEARCH.** It uses LOOP to step through starting point of an array till the ending point of an array. It compares each element with the value which we are searching, and stop its searching when we found the element or we reached to the end point of an array. In Worst Case we may search or scan whole array and in the Average Case we may have to search or scan half of the array, So, for Worst and Average Case our complexity will be **O(n)** and **O(n/2)** respectively. If the desired element is present in the 1^{st} position or we have to scan less than half of an array then that case considered as a Best Case.

Advantages of Linear Search:

1 It is a simple and easy to understand.

2 It is very easy to implement.

3 It is not necessary that data stored in any particular order in array.

Disadvantages of Linear Search:

1 In average case may be n/2 comparisons will be made.

2 In worst case n comparisons will be made.

3 Time Complexity → O(n)

Algorithm: LINEAR_SEARCH(ARR, N, VALUE)

Step 1: [INITIALIZE] Set position : = -1

[Initialize] set J:= 1
Repeat step 4 while J<=N
If arr[J] = VALUE
Set position := J
Print position
Go to step6
[End If]
Set J := J+1
[End loop]
Step 2: if position = -1
Print value is not in array
[End if]
Step 3: exit

Explanation: In this algorithm first of all we initialize the **POSITION** with -1 and we take a variable **J** for running a loop. The loop will run till the value is not found or we reached to the end of the array.

D] BINARY SEARCH: In Binary Search we use **Divide and Conquer technique**. For this search array must be arranged in a particular manner. In other words we can say that array must be sorted. In this search we divide array from middle to two parts and compare the desired element with the middle element. If the middle element of array is larger than the desired element then we search desired element in 1^{st} half of the array or if the middle element is smaller than the desired /element then we search desired element in 2^{nd} half of the array. This Search is also known as **HALF – INTERVAL SEARCH.** The Worst and Average Case Complexity is **O(logn)**, and the Best case Complexity will be **O(1)**. If we find our desired element in minimum iterations then that is considered as Best case and we found desired element in maximum iterations then that is considered as Worst case. Its Time Complexity is **$O(\log_2 n)$.**

Advantages of Binary Search:

1 Binary Search is faster.
2 The average no. of comparison is less.
3 It is easy to implement.

Disadvantages of Binary Search:

1 Array must be sorted in a particular manner.
2 It works only in Array.

Algorithm: BINARY_SEARCH(lb, ub, Value, A, LOC)
Step 1: [INITIALIZE SEGMENT VARIABLES]
Set F := lb, B := ub, C := int((F+B))/2
Step 2: Repeat Steps 3 and 4 while F <= B and A[C]!= Value
Step 3: If Value < A[mid], then
Set B := C-1

Else

Set F := C+1

[End of If]

Step 4: Set C := int((F+B))/2

[End of Step2 loop]

Step 5: If A[C] = Value then

Set LOC := C

Else

Set LOC:= NULL

[End of If]

Step 6:END

Explanation: **A** is a sorted array with lower bound **lb** and upper bound **ub,** and **Value** is a given item which is to be searched. The variables **F, B** and **C** denote the beginning, end and middle location of a segment of element of **A.**

2.Comparative Analysis

I] Array and Linked Lists -

S.No.	COMPONENTS	ARRAY	LINKED LISTS
1.	BASIC	It is a consistent set of a fixed number of data elements.	It is an ordered set of consisting a variable number of data elements.
2.	SIZE	Size is fixed and specified during declaration.	Size is not fixed and it can grow and shrink during execution.
3.	ALLOCATION	Location is assigned during Compile time.	Location is assigned during Run time.
4.	STORING ORDER OF ELEMENTS	Elements are Stored Consecutively.	Elements are Stored Randomly.
5.	ACCESS ELEMENTS	Directly or Randomly can access elements.	Sequentially can access elements.
6.	TYPES	One Dimension and Multi Dimension (2-D) arrays.	Singly linked list, Doubly linked list, Circular linked list, and Circular

			Doubly linked lists.
7.	SEARCH TECHNIQUE	Both Binary and Linear Search.	Linear Search only.
8.	INSERTION AND DELETION	Relatively slow because of shifting of elements is required.	Easy, Fast and Effective as compare to Array.
9.	UTILIZATION OF MEMORY	Utilization of memory is ineffective.	Utilization of memory is efficient.
10.	REQUIRED MEMORY	Requirement of memory is less.	Requirement of memory is more.

Table 2: Comparison of Array and Linked Lists.

II] Linear Search And Binary Search –

S.No.	COMPONENTS	LINEAR SEARCH	BINARY SEARCH
1.	DEFINITION	In this search we search sequentially.	In this search we divide array into two sub array.
2.	BEST CASE COMPLEXITY	If the desired element is First element of the array then complexity will be O(1).	If desired element is in middle of the array then its complexity will be O(1).
3.	TIME COMPLEXITY	O(N)	$O(\log_2 N)$
4.	IMPLEMENTED	Both Array and Linked list.	Direct implementation on linked list is not possible.
5.	TYPE OF ALGORITHM	Iterative algorithm.	Divide and Conquer algorithm.
6.	PRE-	No such requirement	Array must be sorted in particular manner.

	REQUIREMENT OF ARRAY		
7.	LINE OF CODE	Less	More
8.	INSERTION	We can easily insert element at the end of list.	Require Processing to insert element into its place to maintain sorted list.
9.	USEFULLNESS	Easy to use because no need of ordered elements.	Tricky to use because elements mush be arranged in an order.
10.	SPEED	Speed of linear search is slow as compared to binary search.	Speed of binary search is fast as compared to linear search.

Table 3: Comparison of Linear Search and Binary Search.

4. Conclusion

In this paper we try to distinguish all standard algorithms of array, linked list, linear search and binary search. With the help of algorithms we see step by step instruction of searching and flow chart indicates diagrammatical representation of searching techniques. Array and linked lists are the type of data structures in their structure, accessing methods, memory requirement and utilization. And have particular advantage and disadvantage over its implementation. So through this paper one can be used it as per their requirement. Linear and Binary search is the searching techniques, which can be as per their requirement.

5.Acknowledgement

The writing of this paper was a enormous task for which lot of help required. I had a fine support and I Want to THANK to very special person in my life My Family and My teachers who always appreciate me. My sincere thanks go to my guide Ms Parul Choudhary for his inspiration and guidance in writing this paper. We would like to spread knowledge of Array, Linked Lists, Linear Search and Binary Search.

6.Reference

[1]. Reema Thareja, “Data Structure Using C” [Second Edition]

[2]. Adam Drozdek, “Data Structure and Algorithm Using C++” [Fourth Edition]

[3]. Sapinder, Ritu, Arvinder Singh, “Analysis of Linear and Binary Search Algorithm” [Volume I and II]

[4]. .JeanPaul Tremblay, Paul G. Sorenson, P. G. Sorenson,“An Introduction to Data Structures with Applications”[Mcgraw Hill Computer Science Series]

