Answer Script

Question No. 01
1. Write down the time complexity of Bubble sort, Insertion sort and Merge sort. **10**
Answer No. 01
Bubble Sort: Average Time Complexity: O(n^2) Best Time Complexity: O(n) Worst Time Complexity: O(n^2) **Insertion Sort:** Average Time Complexity: O(n^2) Best Time Complexity: O(n) Worst Time Complexity: O(n^2) **Merge Sort:** Average Time Complexity: O(nlogn) Best Time Complexity: O(nlogn) Worst Time Complexity: O(nlogn)

Question No. 02

2. Write two differences between array and linked-list. Why do we need a head/root node in a linked list? **10**

Answer No. 02

The difference between Array and Linked list:

Array	**Linked List**
An array is a collection of elements of a similar data type.	A linked list is a collection of objects known as a node where node consists of two parts data and address.
Array elements store in a contiguous memory location.	Linked list elements can be stored anywhere in the memory or randomly stored.

A singly linked list is a collection of nodes where each node stores two pieces of information, some value and a pointer to the next node.

In a singly linked list, there is a head or root node where the linked list starts.The first node is called the head. It points to the first node of the list and helps us access every other element in the list.

If we want to access every element in the linked list we need to know the head. If we insert value in any index in the link list, delete any value in the linked list, reverse the link list we must need head. If you don't know the head of the linked list then we don't access every element in the linked list. We can't insert any value in the linked list, can't delete or reverse. That's why we need the head node in the linked list.

Question No. 03

3. What is the basic idea behind the binary search algorithm, and how does it differ from linear search? What is the requirement for an array to be suitable for binary search? **10**

Answer No. 03

Binary Search is a searching algorithm used in a sorted array by repeatedly dividing the search interval in half. The basic idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(Log n).

Linear search is a search that finds an element in the list by searching the element sequentially until the element is found in the list. On the other hand, a binary search is a search that finds the middle element in the list recursively until the middle element is matched with a searched element.

The linear search uses an iterative approach to find the element, so it is also known as a sequential approach. In contrast, the binary search calculates the middle element of the array, so it uses the divide and conquer approach.

In a linear search, the elements don't need to be arranged in sorted order. The pre-condition for the binary search is that the elements must be arranged in a sorted order.

A Binary Search is a sorting algorithm that is used to search an element in a sorted array. A binary search technique works only on a sorted array, so an array must be sorted to apply binary search on the array.
So, for a binary search array must need sorted.

Question No. 04

4. What is the time complexity of inserting an element at the beginning of a singly linked list? What is the time complexity of inserting an element at any index of a singly linked list? What is the time complexity of deleting an element at the beginning of a singly linked list? What is the time complexity of deleting an element at any index of a singly linked list? **10**

Answer No. 04

The time complexity of inserting an element at the beginning of a single linked list: O(1).
The time complexity of inserting an element at any index of a single linked list: O(n).
The time complexity of deleting an element at the beginning of a singly linked list: O(1).
The time complexity of deleting an element at any index of a singly linked list: O(n).

Question No. 05

5. Suppose we have an array of 5 integer numbers and a singly linked list of 5 integer numbers. Here the singly linked list of 5 integer numbers takes twice as much memory compared to array. Why is that? Give a proper explanation. **10**

Answer No. 05

Memory consumption is more in Linked Lists when compared to arrays. Because each node contains a pointer in the linked list and it requires extra memory.

As the elements in an array store in one contiguous block of memory, so the array is of fixed size. Suppose we have an array of size 5, and the array consists of 5 elements.
Memory space = 5*4 = 20 bytes

Where 5 is the number of elements in an array and 4 is the number of bytes of an integer type.

In the case of linked list, If the data is of integer type, then total memory occupied by one node is 8 bytes. 4 bytes for data and 4 bytes for pointer variable. If the linked list consists of 5 elements, then the memory space occupied by the linked list would be:
Memory space = 8*5 = 40 bytes

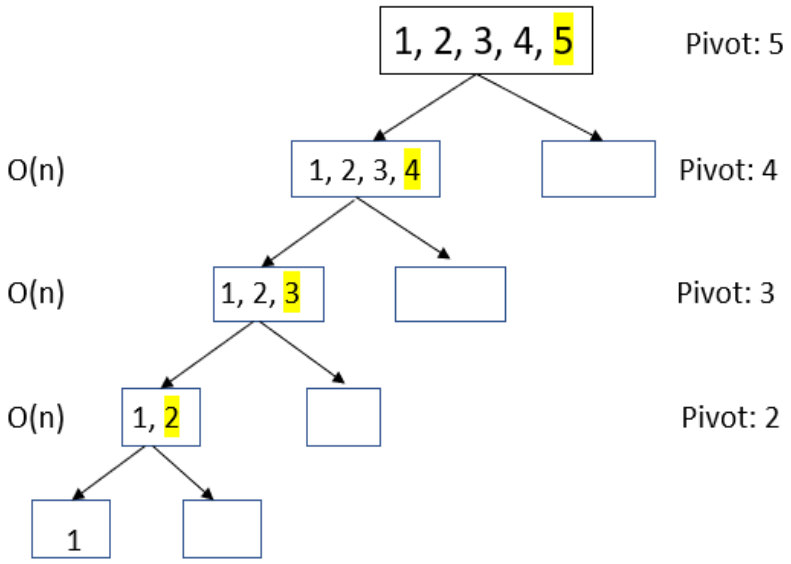
That's why a linked list takes twice as much memory compared to array.

Question No. 06

6. Derive the worst case and average case time complexity of Quick Sort with proper figures. **10**

Answer No. 06

Worst Case:

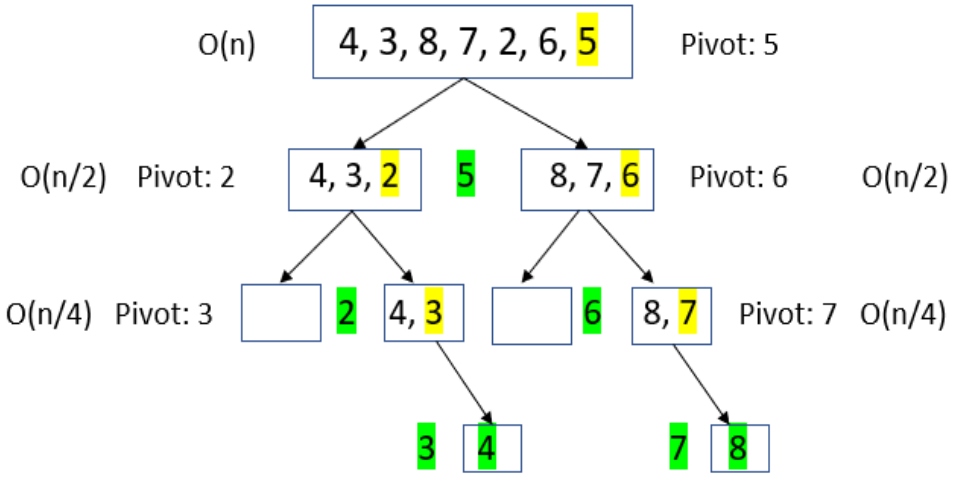


Let, array = {1, 2, 3, 4, 5}
Here, this array has 5 elements.
n = 5
here 5 levels.

1st level we take the last element pivot = 5. For this level time complexity O(n).
2nd level we take pivot = 4. For this level time complexity O(n).
3rd level we take pivot = 3. For this level time complexity O(n).
4th level we take pivot = 2. For this level time complexity O(n).
Last level we take pivot = 1. For this level time complexity O(n).

We have n level. Every level time complexity O(n).
Time Complexity O(n . n). Here 1st n means n level and 2nd n means every level time complexity.
So, total time complexity in worst case O(n^2).

Average Case:



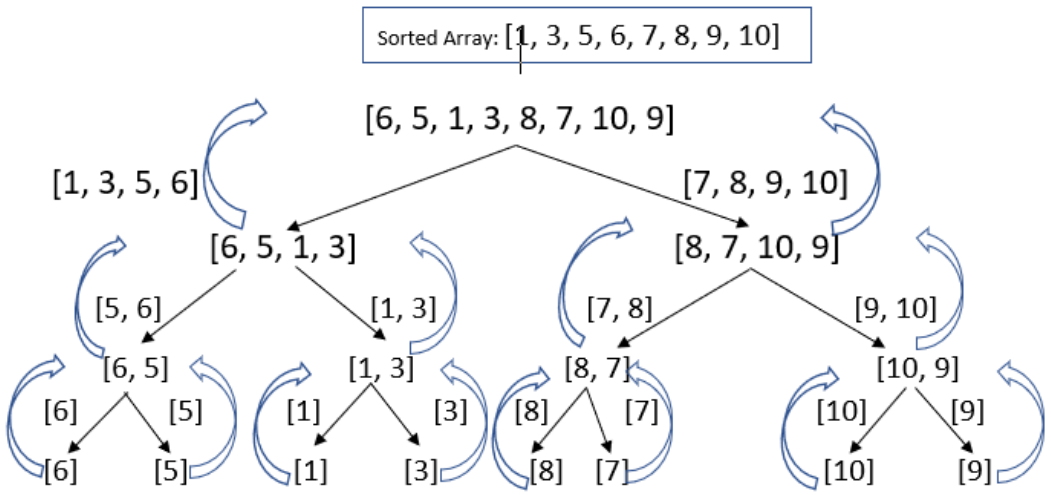
Let, array = {4, 3, 8, 7, 2, 6, 5}
n = 7
1st level we take the last element pivot. Pivot = 5. For this level time complexity O(n).
Then at the next level, we divide this array into 2 parts. Less than equal pivot elements sit on the left side and greater than pivot elements sit on the right side. For this level time complexity O(n/2).
Now on the left side we take pivot = 2 and right side take pivot = 6.
Then at the next level again divide this array into 2 parts. For the left side take pivot = 2 and for the right side take pivot = 6.
Less than equal pivot elements sit on the left side and greater than pivot elements sit on the right side. For this level time complexity O(n/4).
Now last take pivot = 3 and pivot = 7.
Less than equal pivot elements sit on the left side and greater than pivot elements sit on the right side. Here only 1 element. For this level time complexity O(1).

$n/2^k = 1$
$\Rightarrow n = 2^k$
$\Rightarrow \log n = \log 2^k$
$\Rightarrow k = \log n$
So, total time complexity: O(nlogn).

Question No. 07

7. Draw the recursion call tree with return values for Merge Sort in the array [6, 5, 1, 3, 8, 7, 10, 9] **10**

Answer No. 07



Question No. 08

8. Write down the time complexity with proper explanation of the following code segment. **10**

```
for (int i = 1; i * i <= n; i++) {
    if (n % i == 0) {
        cout<< i <<"\n";
        cout<< (n/i) <<"\n";
    }
}
```

Answer No. 08

For this code time complexity is O(sqrt(n)).

The outer loop runs from i=1 to i*i <= n, which means it will run sqrt(n) times.
If n = 10.
Then,
For i = 1:
When i = 1,the condition is true. The condition is i*i<=n. The condition is true (1*1<=10). Then check the if statement. 10 divide by 1 then remainder goes to 0. So this if statement is true. And print i's value 1 and 10 (10/1=10) and i values increase by 1.

For i = 2:
When i = 2, the condition is true. The condition is i*i<=n. The condition is true (2*2<=10). Then check the if statement. 10 divide by 2 then remainder goes to 0. So this if statement is true. And print i's value 2 and 5 (10/2=5) and i values increase by 1.

For i = 3:
When i = 3, the condition is true. The condition is i*i<=n. The condition is true (3*3<=10). Then check the if statement. 10 divide by 3 then remainder goes to 1. So this if statement is false. It doesn't print the inside cout value.

For i = 4:
When i = 4, the condition is false. The condition is i*i<=n. The condition is false (4*4<=10). So, it breaks the for loop.

For n = 10 the for loop works sqrt(n) times. So, we can say for loop time complexity O(sqrt(n)).

Inside the if statement checks if n divides by i and remainder goes to 0 then this statement works. This if statement time complexity O(1).

Inside of the if statement both cout time complexity O(1). Cause both cout just print a value.
Therefore, the overall time complexity of the code segment is O(sqrt(n)) as it is dominated by the sqrt(n) iterations of the outer loop.

Question No. 09

9. Suppose you are working in a project where you need to do many random memory accesses and binary search. Array vs Linked list which is more suitable in this case? Why? **10**

Answer No. 09

Array is more suitable in this case.

An array is a data structure that stores a fixed-size sequential collection of elements of the same type. It can be used to store and organize data in a structured way. Each element in an array is identified by an index, which is an integer value that starts from 0 and goes up to the size of the array minus 1. The elements in an array can be accessed and modified by their index.

A linked list is a data structure that consists of a sequence of elements, each containing a reference or pointer to the next element in the list. It is a dynamic data structure, which means that elements can be added or removed from the list at any time.

For binary search, an array is generally considered to be more suitable than a linked list.

One of the main advantages of an array is that elements can be accessed in constant time by their index, which is not the case with linked lists. In a linked list, elements are accessed by following a chain of pointers, which takes linear time. Thus, for binary search, we need to access the middle element quickly and repeatedly, array is more suitable.

Additionally, arrays have a fixed size, which means that their memory usage is predictable and can be allocated in a contiguous block, whereas linked lists have a dynamic size and use more memory due to the overhead of the pointers.

So, in summary, arrays are more suitable for binary search because they allow for faster element access, have predictable memory usage, and are cache-friendly.

Question No. 10
10. Alice is using a singly linked list to implement undo-redo functionality in a text editor. Bob advised Alice to use a doubly linked list in this scenario. Which approach seems more suitable to you? Give a proper explanation. **10**
Answer No. 10
A doubly linked list would be more suitable for implementing undo-redo functionality in a text editor. A single linked list is a type of linked list where each element or node in the list contains a reference or pointer to the next element in the list. The last element in the list has a reference to null. A doubly linked list is a type of linked list where each element or node in the list contains a reference or pointer to the next element and the previous element in the list. The first element in the list, called the head, has a reference to the next element, but no reference to the previous element, and the last element, called the tail, has a reference to the previous element but no reference to the next element. A doubly linked list is more suitable in this scenario. A singly linked list allows for a single traversal, either forward or backward, whereas a doubly linked list allows for both directions of traversal. This gives Alice the flexibility to revert through her actions in any order, which is necessary for undo-redo functionality. Additionally, the doubly linked list provides Alice with the convenience of quickly locating the most recently added node, which is beneficial for efficient implementation.