**Theory Mid Term**

Question -7

vector<int>merge\_sort(vector<int>a)

{

if(a.size()<=1)

{

return a;

}

int mid=a.size()/2;

vector<int>b;

vector<int>c;

for(int i=0; i<mid; i++)

{

b.push\_back(a[i]);

}

for(int i=mid; i<a.size(); i++)

{

c.push\_back(a[i]);

}

vector<int>sorted\_b=merge\_sort(b);

vector<int>sorted\_c=merge\_sort(c);

Recursion Call Tree Diagram:

**[1.3.5.6]** [6, 5, 1, 3, 8, 7, 10, 9] **[7,8,9,10]**

**Return**

**[5,6]** [6,5,1,3] **[1,3]** **[7,8]** [8,7,10,9] **[9,10]**

**[6]**  [6,5] **[5]** **[1]** [1,3] **[3]** **[8]** [8,7] **[7]** **[9]** [9,10] **[10]**

[6] [5] [1] [3] [8] [7] [10] [9]

{4,3,2,5,1}

{4,3,2,5} { }

{4,3,2]} { }

{4,3} { }

{ } { }

{4,3,**2**,5,1}

{**4**,3,} {**5**,1 }

{ } {3} {} {1}

Assume that, n=9.

Value of i The number of rotation of the loop

1. 1
2. 1
3. 1
4. 0
5. 0
6. 0
7. 0
8. 0
9. 0

Total number of rotation is (1+1+1)=3 . That is, √9.

The time complexity of the above code segment is : **O(√9)**

|  |  |
| --- | --- |
| Operation | Time Complexity |
| inserting an element at the beginning | O(1) |
| inserting an element at any index | O(n) |
| deleting an element at the beginning | O(1) |
| deleting an element at any index of | O(n) |

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | **Best case** | **Average case** | **Worst case** |
| Bubble sort | O(n) | O(n2) | O(n2) |
| Insertion sort | O(n) | O(n2) | O(n2) |
| Merge sort | O(nlogn) | O(nlogn) | O(nlogn) |

|  |  |
| --- | --- |
| **Linked List** | **Array** |
| Linked List is an ordered collection of elements of the same type where each element is linked to the next using pointers. | An array is a collection of elements of a similar data type. |
| Random access to linked list is not possible. Elements must be accessed sequentially. | Array elements can be accessed randomly using array index. |