Branch : CSE/IT Batch : Hinglish

WEEKLY TEST – 03

Subject : Algorithm

Topic : Design Strategy

Maximum Marks 15

Q.1 to 5 Carry ONE Mark Each

[MCQ]

1. Assume that there are 4n sorted list of size $\left(\frac{n}{2}\right)$, then what is the time complexity of merging them into single sorted list

(a) $\theta(n^2 \log n)$ (b) $(\log n)$
(c) $\theta(n\log n)$ (d) $\theta(n^2)$

[MSQ]

2. Assume that, quick sort implementation is used to sort an array in ascending order after the first partition step has been completed, the contents of the array are in the following order.

50 30 40 20 80 90 120 100 140 110 160

Which of the following elements could be a pivot element?

(a) 40 (b) 30
(c) 80 (d) 160

[NAT]

3. Assume that a binary search is used to search for a particular key within a sorted array of 256 values then what is the maximum number of key comparisons, the program. Would require to make before finding the key _____(if the key is present ,round of to two decimal.)

[MSQ]

4. Choose the correct statements from the following statements

(a) Binary search in an unsorted array will take O(nlogn) time in worst case.
(b) Searching for an element in an unsorted array will take O(n) time in the worst case.
(c) Binary search on a sorted linked list takes O(n) time in worst case.
(d) Applying binary search on sorted linked list takes O(logn) time in worst case.

[MCQ]

5. What is the auxiliary space complexity of merge sort

(a) O(1) (b) O(logn)
(c) O(n) (d) O(nlog)

Q.6 to 10 Carry TWO Mark Each

[NAT]

6. Generally, merge is a divide and conquer technique which can also be implemented in a recursive manner. If there are 200 elements in an array then find the exact number of merge sort function calls which will perform the recursive call_____.

[MSQ]

7. Consider the following code segment

(Note: n is power of 2 and base of log is 2)

```
int a, b, n;
for (a = 2; a < 2^logn; a++)
    {
      for (b = 2; b< = a; b++)
        {
        if(b%i = = 0)
        printf ("GATEWALLAH");
        }
    }
```

Which of the following statements is/are true?

(a) If f(n) is the number of times "GATEWALLAH" printed in terms of n, then f(n) is equivalent to n
(b) If f(n) is the number of times "GATEWALLA" is printed in terms of n then f(n) is equivalent to n – 1
(c) Time complexity of the given code is $\theta(n)$
(d) Time complexity of the given code is $\theta(n^2)$

[MCQ]

8. Consider a variation of merge sort in which we divide the list into 3 sorted sub lists of equal size, recursively sorting each list, and then merging the three lists to get the final sorted list.

What is the recurrence relation that is required for the number of comparisons used by this algorithm in worst case?

(NOTE: Assume that the number of elements to be sorted is a power of 3 so that all of the divisions are into three sub lists workout evenly)

(a) $T(n) = 3T(n/3) + n - 1$

(b) $T(n) = 2T(n/2) + n - 1$

(c) $T(n) = 6T(n/3) + n - 1$

(d) None of these

[MCQ]

9. Consider a list which contains 2^n sorted lists each of size n and is merged using merge sort, then what is the tightest upper bound worst case complexity?

(a) $O(n^2 2^n)$ (b) $O(2^n \log n)$

(c) $O(n.2^n)$ (d) None of these

[MCQ]

10. Let's suppose you are given an array in which, the few elements in the beginning are present in ascending order and remaining elements are in descending order, then what is the complexity of most efficient algorithm to find the maximum value of this array?

(a) $\theta(n \log n)$ (b) $\theta(n)$

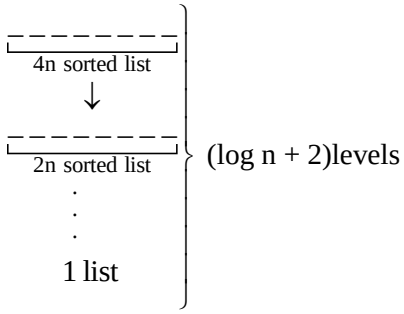
(c) $O(1)$ (d) $\theta(\log n)$

Answer Key

1. **(a)**
2. **(c, d)**
3. **(8 to 8)**
4. **(a, b, c)**
5. **(c)**
6. **(398 to 398)**
7. **(b, d)**
8. **(a)**
9. **(a)**
10. **(d)**

Hints and Solutions

1. (a)



Merging time complexity at each level = $4n \times \frac{n}{2} = 2n$

$\therefore\ 2n^2 (\log n + 2) = \theta (n^2 \log n)$

2. (c, d)

All the elements before 80 are less than 80 and elements after 80 are greater than so 80 could be a pivot element.

All elements before 160 are less than 160, s0 160 could also be a pivot element.

$\therefore$ option (c, d) are correct.

3. (8 to 8)

The maximum number of comparisons in a binary search on sorted array of 256 elements are $\lceil \log_2 n \rceil$, where n is the number of elements.

$\therefore \lceil \log_2 256 \rceil, = 8$

$\therefore$ 8 number of comparisons are required (This include the case where elements not present in the array)

4. (a, b, c)

(a) Yes, Binary search in an unsorted array will take O(nlogn) time in worst case.

(b) Yes, Searching an elements in an unsorted array will take O(n) time in the worst case.

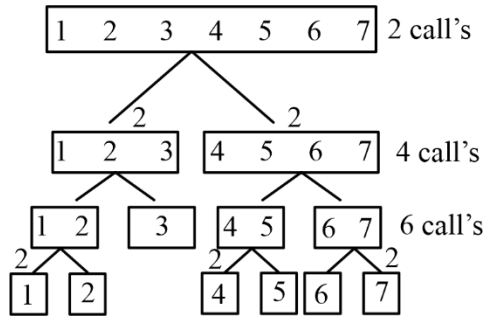
(c) Yes, Binary search on an sorted linked list takes O(n) time in worst case.

5. (c)

An additional space of O(n) is required in order to merge two sorted arrays. Thus, merge sort is not an in-place sorting algorithm.

6. (398 to 398)

n=7



For n = 1 → 0 calls

n = 2 → 2 calls

n = 3 → 4 calls

n = 4 → 6 calls

$\therefore$ generalized formula is 2(n–1) calls

So, for n = 200,

2(200 – 1) = 2 (199) = 398

7. (b, d)

As we can see that,

Whenever "b = i" in inner loop, "GATEWALLAH" will be printed, so "GATEWALLAH" will be printed energy time once for i = 2 to i = n ie,.. (n – 1) times

f(n) = (n – 1)

Also, for (i = 2) inner loop will execute for 2 times

for (i = 3) inner loop will execute for 3 times.

for (i = n) inner loop will execute for n times.

So, total complexity of given code is

$2 + 3 + 4 \ldots\ldots n \Rightarrow \frac{n(n+1)}{2} - 1 = \theta(n^2)$

$\therefore$ option b, d are correct.

8. (a)

let T(n) be the time of sorting the list using merge sort.

Given that list has to be divided into 3 equal parts and perform sorting in each sub lists

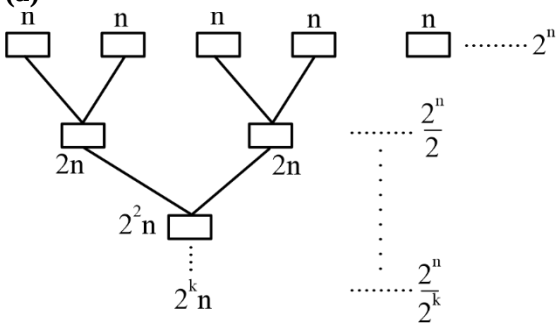
So, $T(n) = T(n/3) + T(n/3) + T(n/3)$

After performing sorting, we need to copy the list again into the original list which will take $(n - 1)$ units or work.

$T(n) = 3T(n/3) + n - 1$

$\therefore$ option (a) is correct.

9. (a)



Cost of level 0 $2^n \times n$

Cost of level 1 is $\frac{2^n}{2} \times 2n = 2^n \times n$

$\therefore$ cost of every level is $2^n \times n$

Total number of levels = $k + 1$ $\{2^0$ to $2^k\}$

Overall cost = $2^n \times n\,[k + 1]$

Here $\frac{2^n}{2^k} = 1$

$2^k = 2^n$

Taking log on both sides

$k = n$

$\therefore$ overall cost = $2^n \times n\,(n + 1)$

$= 2^n \times n^2 + 2^n \times n = O(n^2 \times 2^n)$

$\therefore$ (a) is correct option.

10. (d)

Apply binary search

If a[mid] > a [right & a[mid] > a[left]

Then max = a[mid]

if a[mid] > a [right] & a [mid] < a [left] → binary search (start, mid -1)

if a[mid] <a[right] & a[mid] > a[left] → binary search (mid+1, last)