Sorting Questions

Sorting MCQ Question 1

View this Question Online >

Let P be a quicksort program to sort numbers in ascending order using the first element as the pivot. Let t_1 and t_2 be the number of comparison made by P for the inputs [1 2 3 4 5] and [4 1 5 3 2] respectively. Which one of the following holds?

1. $t_1 = 5$
2. $t_1 < t_2$
3. $t_1 > t_2$
4. $t_1 = t_2$

Answer (Detailed Solution Below)

Option 3 : $t_1 > t_2$

Sorting MCQ Question 1 Detailed Solution

Concept:

If the array is sorting is increasing or non-increasing order, choosing either first or last element given the worst-case time complexity in Quick Sort.

Explanation

In every step of quick sort, numbers are divided as per the following recurrence.

$T(n) = T(n-1) + O(n)$

Average case time complexity (t2) = O(logn)

Worst case time complexity (t1) = $O(n^2)$

Since t1 has the worst case ∴ t1 > t2

Important Point:

Number of comparisons will vary based on implementation

Always in ascending or descending array (worst case) number of comparisons is higher

Sorting MCQ Question 2

View this Question Online >

Suppose P, Q, R, S, T are sorted sequences having lengths 20, 24, 30, 35, 50 respectively. They are to be merged into a single sequence by merging together two sequences at a time. The number of comparisons that will be needed in the worst case by the optimal algorithm for doing this is ____.

Answer (Detailed Solution Below) **358**

Sorting MCQ Question 2 Detailed Solution

Given sequence: 20, 24, 30, 35, 50

Ascending order: 20, 24, 30, 35, 50

Merge(20, 24) → new size (44)

Number of comparisons = 20 + 24 − 1 = 43

Sequence: 30, 35, 44, 50

Merge(30, 35) → new size (65)

Number of comparisons = 30 + 35 − 1 = 64

Sequence: 44, 50, 65

Merge(44, 50) → new size (94)

Number of comparisons = 44 + 50 − 1 = 93

Sequence: 65, 94

Merge(65, 94) → new size (150)

Number of comparisons = 65 + 94 − 1 = 158

Therefore, total number of comparisons, 43 + 64 + 93 + 158 = 358

Sorting MCQ Question 3

View this Question Online >

Assume that a mergesort algorithm in the worst case takes 30 seconds for an input of size 64.
Which of the following most closely approximates the maximum input size of a problem that can be solved in 6 minutes?

1. 256
2. 512
3. 1024
4. 2048

Answer (Detailed Solution Below)

Option 2 : 512

Sorting MCQ Question 3 Detailed Solution

Concept:

Merge sort algorithm worst case time complexity is O (nlogn)

Data:

T_1 (n) =30 seconds, n_1 = 64

T_2 (n) = 6 minutes

Formula:

$T(n) = n\log_2 n$

Calculation:

$30 = c \times 64 \log_2(64)$

$30 = c \times 64 \log_2(2^6)$

$30 = c \times 64 \times 6$

$$c = \frac{5}{64}$$

$6 \times 30 = c\, n_2 \log_2 n_2$

$$6 \times 60 = \frac{5}{64} \times n_2 \log_2 n_2$$

∴ n_2 = 512

The maximum input size of a problem that can be solved in 6 minutes is 512

Sorting MCQ Question 4

View this Question Online >

An array of 25 distinct elements is to be sorted using quick sort. Assume that the pivot element is chosen uniformly at random. The probability that the pivot element gets placed in the worst possible location in the first round of partitioning (rounded off to 2 decimal places) is ________.

Answer (Detailed Solution Below) **0.08**

Since the given array is sorted

Let given array be **1**, 2, 3, 4, 5, 6, 7, ... 25

Probability of choosing each element is $\frac{1}{25}$

Worst case for quick sort will only we arise if 1 is chosen or 25 is chosen as pivot element

Assume 1 is chosen as a pivot element

After first round of partitioning, pivot will be in its same position (1^{st} position), this gives rise to worst case in quick sort since the complexity will be $O(n^2)$.

Assume 25 is chosen as a pivot element

After first round of partitioning, pivot will be in its same position (last position), this gives rise to worst case, in quick sort since the complexity will be $O(n^2)$.

$P(X = 1) = \frac{1}{25} = 0.04$

and $P(X = 50) = \frac{1}{25} = 0.04$

$P(X = 1 \text{ or } X = 25)$

$= 0.04 + 0.04$

= 0.08

Sorting MCQ Question 5

View this Question Online >

Consider a complete binary tree where the left and the right subtrees of the root are max-heaps. The lower bound for the number of operations to convert the tree to a heap is

1. Ω (log n)
2. Ω (n)

3. Ω (n log n)

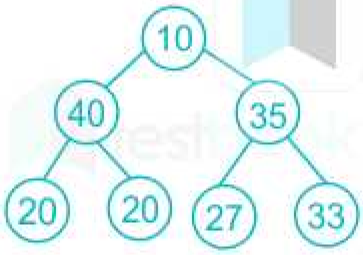
4. Ω (n^2)

Answer (Detailed Solution Below)

Option 1 : Ω (log n)

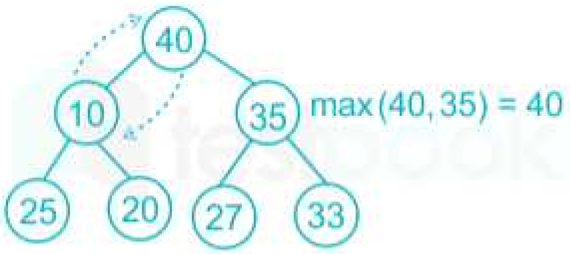
Sorting MCQ Question 5 Detailed Solution

Consider complete binary tree where left and right subtrees of the root are max-heaps given below

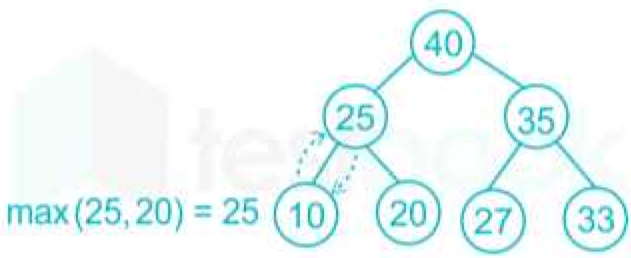


To convert the tree into a heap which is possible by calling MAX-HEAPIFY at root. MAX- HEAPIFY operation takes time as the height of the tree. i.e. if we have n elements in the tree then log(n) is the height of the tree.

Step 1: Swap 10 and 40



Step 2: swap 10 and 25



The above tree is a MAX-HEAP

To convert this into a max heap it takes only 2 swap and 2 comparison which is nothing but the height of the tree. So, log(n) time is required to convert the tree into a heap.

Sorting MCQ Question 6

View this Question Online >

The number of swappings needed to sort the numbers 8, 22, 7, 9, 31, 5, 13 in ascending order, using bubble sort is

1. 11
2. 12
3. 13
4. 10

Answer (Detailed Solution Below)

Option 4 : 10

Sorting MCQ Question 6 Detailed Solution

Bubble sort repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted.

Array elements: 8, 22, 7, 9, 31, 5, 13

1^{st} pass = 8, **7**, **9**, **22**, 5, **13**, 31

4 swaps

2^{nd} pass = **7**, 8, 9, **5**, **13**, 22, 31

3 swaps

3^{rd} pass = 7, 8, **5**, 9, 13, 22, 31

1 swap

4^{th} pass = 7, **5**, 8, 9, 13, 22, 31

1 swap

5^{th} pass = **5**, 7, 8, 9, 13, 22, 31

1 swap

Since array is sorted after 5^{th} pass

∴ no further swaps possible

Total number of swaps = 4 + 3 + 1 + 1 + 1 = 10

Sorting MCQ Question 7 View this Question Online >

A complete binary min-heap is made by including each integer in [1,1023] exactly once. The depth of a node in the heap is the length of the path from the root of the heap to that node. Thus, the root is at depth 0. The maximum depth at which integer 9 can appear is______.

Answer (Detailed Solution Below) **8**

Sorting MCQ Question 7 Detailed Solution

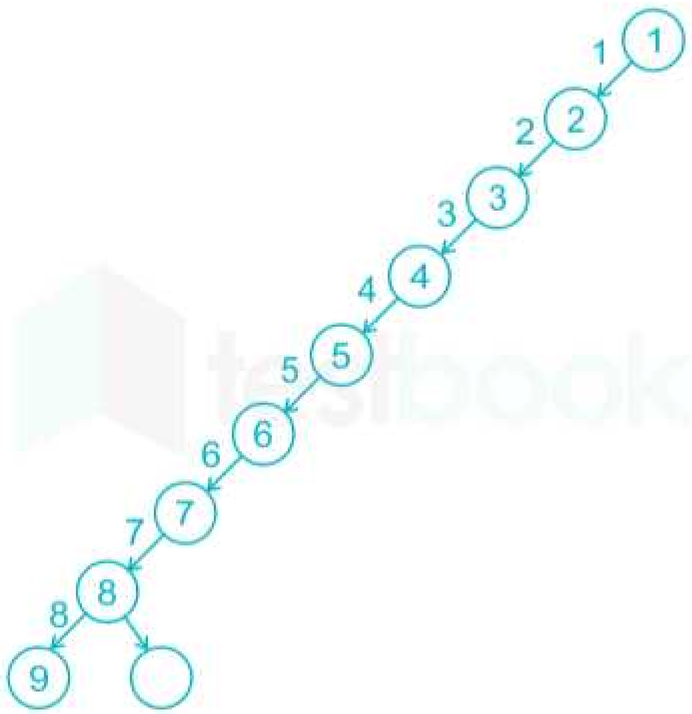
Concept:

Min- heap is a binary tree such that data contained in each nodes is less than (or equal to) the data in that node's children.

We have to consider the integers in the range [1, 1023].

It is also given that root is at depth 0. As it is a min heap so integer 1 has to be at root only.

Tree will become:



Now, once we place 1 – 9 then remaining elements can be placed easily to fill the min-heap.

Here, the maximum depth at which integer 9 can appear is 8.

The number of possible min-heaps containing each value from {1, 2, 3, 4, 5, 6, 7} exactly once is _____.

Answer (Detailed Solution Below) **80**

Sorting MCQ Question 8 Detailed Solution

case 1: 2 and 3 are on 2nd level.

so, 4,5,6 and 7 can occupy any of 4 positions in the 3rd level as all are less than 2 and 3. So, 4! arrangements. And 2 and 3 can be arranged in 2 ways in 2nd level.

2*24=48

case 2**:** 2 and 4 are on 2nd level

Now, 3 can only be below 2 and not 4 as it is MIN heap. So, 2 cases are possible 3XXX or X3XX, which gives 3!+3!=12 arrangements. And 2 and 4 can be arranged in 2 ways in 2nd level.

2*(6+6)=24

case 3**:** 2 and 5 are on 2nd level

Now, 3 and 4 can only be below 2 and not 5 as it is MIN heap. So, 2 arrangements are possible for 3 and 4 and similarly 2 for 6 and 7. And 2 and 5 can be arranged in 2 ways in 2nd level.

2*(2+2)=8

Sorting MCQ Question 9

View this Question Online >

A list of n strings, each of length n, is sorted into lexicographic order using merge - sort algorithm. The worst case running time of this computation is:

1. O(n log n)
2. $O(n^2 \log n)$
3. $O(n^2 + \log n)$
4. $O(n^3)$

Answer (Detailed Solution Below)

Option 2 : $O(n^2 \log n)$

Sorting MCQ Question 9 Detailed Solution

The correct answer is **"option 2".**

CONCEPT:

The **Recurrence relation** for the **number of comparisons** needed to **sort** an array of **n integers** is:

$T(n) = 2T(n/2) + n$

$= O(n\log_2 n)$

EXPLANATION:

Consider **n strings** of **length n** in place of **each integer**, since **each integer takes O(1) time** to sort then,

Complexity to sort one string of size n - O(n)

Complexity to sort n strings of size n - $O(n * n\log_2 n) = O(n^2\log_2 n)$

Hence, worst case running time is $O(n^2\log_2 n)$.

Sorting MCQ Question 10

View this Question Online >

Consider the following array.

23	32	45	69	72	73	89	97

Which algorithm out of the following options uses the least number of comparisons (among the array elements) to sort above array in ascending order?

1. Insertion sort
2. Selection sort
3. Quicksort using the last element as pivot
4. Merge sort

Answer (Detailed Solution Below)

Option 1 : Insertion sort

Sorting MCQ Question 10 Detailed Solution

Insertion sort:

In Insertion sort, the best-case takes Θ (n) time, the best case of insertion sort is when elements are sorted in ascending order. In that case, the number of comparisons will be **n - 1 = 8 - 1 = 7**

It is the least number of comparisons (among the array elements) to sort the above array in ascending order:

The number of swaps needed is zero.

Additional Information

In Insertion sort, the worst-case takes Θ (n^2) time, the worst case of insertion sort is when elements are sorted in reverse order. In that case the number of comparisons will be like:

$$\sum_{p=1}^{N-1} p = 1 + 2 + 3 + \ldots . + N - 1 = \frac{N(N-1)}{2} - 1$$

This will give Θ (n^2) time complexity.