Quick Sort

1. Which of the following sorting algorithms is the fastest?
a) Merge sort
b) **Quick sort**
c) Insertion sort
d) Shell sort
Answer: b
Explanation: Quick sort is the fastest known sorting algorithm because of its highly optimized inner loop.

2. Quick sort follows Divide-and-Conquer strategy.
a) True
b) False
Answer: a
Explanation: In quick sort, the array is divided into sub-arrays and then it is sorted (divide-and-conquer strategy).

3. What is the worst case time complexity of a quick sort algorithm?
a) O(N)
b) O(N log N)
c) O(N2)
d) O(log N)
Answer: c
Explanation: The worst case performance of a quick sort algorithm is mathematically found to be O(N2).

4. Which of the following methods is the most effective for picking the pivot element?
a) first element
b) last element
c) median-of-three partitioning
d) random element
Answer: c

Explanation: Median-of-three partitioning is the best method for choosing an appropriate pivot element. Picking a first, last or random element as a pivot is not much effective.

5. Find the pivot element from the given input using median-of-three partitioning method.
8, 1, 4, 9, 6, 3, 5, 2, 7, 0.
a) 8
b) 7
c) 9
d) 6
Answer: d
Explanation: Left element=8, right element=0,
Centre=[position(left+right)/2]=6.

6. Which is the safest method to choose a pivot element?
a) choosing a random element as pivot
b) choosing the first element as pivot
c) choosing the last element as pivot
d) median-of-three partitioning method
Answer: a
Explanation: This is the safest method to choose the pivot element since it is very unlikely that a random pivot would consistently provide a poor partition.

7. What is the average running time of a quick sort algorithm?
a) O(N2)
b) O(N)
c) O(N log N)
d) O(log N)
Answer: c
Explanation: The best case and average case analysis of a quick sort algorithm are mathematically found to be O(N log N)

8. Which of the following sorting algorithms is used along with quick sort to sort the sub arrays?
a) Merge sort
b) Shell sort
c) Insertion sort
d) Bubble sort
Answer: c
Explanation: Insertion sort is used along with quick sort to sort the sub arrays.
It is used only at the end.

9. Quick sort uses join operation rather than merge operation.
a) true
b) false
Answer: a
Explanation: Quick sort uses join operation since join is a faster operation than merge

10. How many sub arrays does the quick sort algorithm divide the entire array into?
a) one
b) two
c) three
d) four
Answer: b
Explanation: The entire array is divided into two partitions, 1st sub array containing elements less than the pivot element and 2nd sub array containing elements greater than the pivot element.

11. Which is the worst method of choosing a pivot element?
a) first element as pivot
b) last element as pivot
c) median-of-three partitioning
d) random element as pivot
Answer: a
Explanation: Choosing the first element as pivot is the worst method because if the input is pre-sorted or in reverse order, then the pivot provides a poor partition.

12. Which among the following is the best cut-off range to perform insertion sort within a quick sort?
a) N=0-5
b) N=5-20
c) N=20-30
d) N>30
Answer: b A good cut-off range is anywhere between N=5 and N=20 to avoid nasty degenerate cases.

13. Quick Sort is a _____
a) greedy algorithm
b) divide and conquer algorithm
c) dynamic programming algorithm
d) backtracking algorithm
Answer: b
Explanation: Quick sort is a divide and conquer algorithm. Quick sort first partitions a large array into two smaller sub-arrays. And then recursively sorts the sub-arrays.

14. What is the worst case time complexity of the Quick sort?
a) O(nlogn)
b) O(n)
c) O(n3)
d) O(n2)
Answer: d
Explanation: The worst case running time for Quick sort is O(n2). In Quick sort, the worst case behaviour occurs when the partitioning routine produces two sub-arrays one with n – 1 element and other with 0 elements.

15. Apply Quick sort on a given sequence 7 11 14 6 9 4 3 12. What is the sequence after first phase, pivot is first element?

a) 6 4 3 7 11 9 14 12

b) 6 3 4 7 9 14 11 12

c) 7 6 14 11 9 4 3 12

d) 7 6 4 3 9 14 11 12

Explanation: Let's apply Quick sort on the given sequence,

For first phase, pivot = 7

```
7    11    14    6    9    4    3    12
     i                              j
7    11    14    6    9    4    3    12
     i                         j
7    3     14    6    9    4    11   12
          i               j
7    3     4     6    9    14   11   12
               i    j
7    3     4     6    9    14   11   12
               j    i
6    3     4     7    9    14   11   12
```

16. The best case behaviour occurs for quick sort is, if partition splits the array of size n into _____

a) n/2 : (n/2) – 1

b) n/2 : n/3

c) n/4 : 3n/2

d) n/4 : 3n/4

Answer: a

Explanation: The best case analysis of quick sort occurs when the partition splits the array into two subarrays, each of size no more than n/2 since one is of size n/2 and one of size (n/2) – 1.

17.Quick sort is a stable sorting algorithm.
a) True
b) False
Answer: b
Explanation: In stable sorting algorithm the records with equal keys appear in the same order in the sorted sequence as they appear in the input unsorted sequence. Quick sort does not preserve the relative order of equal sort items. Therefore, Quick sort is not a stable sort.

18. Consider the Quick sort algorithm in which the partitioning procedure splits elements into two sub-arrays and each sub-array contains at least one-fourth of the elements. Let T(n) be the number of comparisons required to sort array of n elements. Then T(n)<=?
a) T(n) <= 2 T(n/4) + cn
b) T(n) <= T(n/4) + T(3n/4) + cn
c) T(n) <= 2 T(3n/4) + cn
d) T(n) <= T(n/3) + T(3n/4) + cn
Answer: b
Explanation: If there are n/4 elements in one sub-array then T(n/4) comparisons are needed for this sub-array. And T(3n/4) comparison are required for the rest 4n/5 elements, and cn is time required for finding the pivot. If there are more than n/4 elements in one sub-array then other sub-array will have less than 3n/4 elements and time complexity will be less than T(n/4) + T(3n/4) + cn.

19. Consider the Quick sort algorithm which sorts elements in ascending order using the first element as pivot. Then which of the following input sequence will require a maximum number of comparisons when this algorithm is applied on it?
a) 22 25 56 67 89
b) 52 25 76 67 89
c) 22 25 76 67 50
d) 52 25 89 67 76
Ans: a
Explanation: If the input sequence is already sorted then worst case behaviour occurs for the Quick sort algorithm which use the first element as pivot. Therefore, the input sequence given in 22 25 56 67 89 will require a maximum number of comparisons.

20. A machine needs a minimum of 200 sec to sort 1000 elements by Quick sort. The minimum time needed to sort 200 elements will be approximately _____
a) 60.2 sec
b) 45.54 sec
c) 31.11 sec
d) 20 sec
Ans:c
Explanation: The Quick sort requires nlog2n comparisons in best case, where n is size of input array. So, 1000 * log21000 ≈ 9000 comparisons are required to sort 1000 elements, which takes 200 sec. To sort 200 elements minimum of 200 * log2200 ≈ 1400 comparisons are required. This will take 200 * 1400 / 9000 ≈ 31.11 sec.

21.Which one of the following sorting algorithm is best suited to sort an array of 1 million elements?
a) Bubble sort
b) Insertion sort
c) Merge sort
d) Quick sort
Answer: d
Explanation: The Quick sort is best suited to sort the array of 1 million elements. The practical implementations of Quick sort use randomised version. In practice randomised Quick sort algorithms rarely shows worst case behaviour and is almost always O(nlogn). And Quick sort requires little additional space and exhibits good cache locality.

22. Quick sort is a space-optimised version of ____
a) Bubble sort
b) Selection sort
c) Insertion sort
d) Binary tree sort
Answer: d
Explanation: Quick sort is a space-optimised version of the binary tree sort. In binary sort tree, the elements are inserted sequentially into the binary search tree and Quick sort organises elements into a tree that is implied by the recursive calls.

23. Quick Sort can be categorized into which of the following?
a) Brute Force technique
b) Divide and conquer
c) Greedy algorithm
d) Dynamic programming
Answer: b
Explanation: First you divide(partition) the array based on the pivot element and sort accordingly.

24. Select the appropriate recursive call for QuickSort.(arr is the array, low is the starting index and high is the ending index of the array, partition returns the pivot element, we will see the code for partition very soon)
a)

```
public static void quickSort(int[] arr, int low, int high)
{
        int pivot;
        if(high>low)
        {
                pivot = partition(arr, low, high);
                quickSort(arr, low, pivot-1);
                quickSort(arr, pivot+1, high);
        }
}
```

b)

```
public static void quickSort(int[] arr, int low, int high)
{
        int pivot;
        if(high<low)
        {
                pivot = partition(arr, low, high);
                quickSort(arr, low, pivot-1);
                quickSort(arr, pivot+1, high);
        }
}
```

c)


```
public static void quickSort(int[] arr, int low, int high)
{
        int pivot;
        if(high>low)
        {
                pivot = partition(arr, low, high);
                quickSort(arr, low, pivot);
                quickSort(arr, pivot, high);
```

```
        }
}
```

d)

```
public static void quickSort(int[] arr, int low, int high)
{
        int pivot;
        if(high>low)
        {
                pivot = partition(arr, low, high);
                quickSort(arr, low, pivot);
                quickSort(arr, pivot+2, high);
        }
}
```

Answer: a

Explanation: Based on the pivot returned by the call to partition, recursive calls to quickSort sort the given array.

25.What is the worst case complexity of QuickSort?

a) O(nlogn)

b) O(logn)

c) O(n)

d) O(n2)

Answer: d

Explanation: When the input array is already sorted.

26. What is a randomized QuickSort?

a) The leftmost element is chosen as the pivot

b) The rightmost element is chosen as the pivot

c) Any element in the array is chosen as the pivot

d) A random number is generated which is used as the pivot

Answer: c

Explanation: QuickSort is randomized by placing the input data in the randomized fashion in the array or by choosing a random element in the array as a pivot.

27.Which of the following code performs the partition operation in QuickSort?

a)

```
private static int partition(int[] arr, int low, int high)
{
        int left, right, pivot_item = arr[low];
        left = low;
        right = high;
        while(left > right)
        {
                while(arr[left] <= pivot_item)
                {
                        left++;
                }
                while(arr[right] > pivot_item)
                {
                        right--;
                }
                if(left < right)
                {
                        swap(arr, left, right);
                }
        }
        arr[low] = arr[right];
        arr[right] = pivot_item;
        return right;
}
```

b)

```
private static int partition(int[] arr, int low, int high)
{
        int left, right, pivot_item = arr[low];
        left = low;
        right = high;
        while(left <= right)
        {
                while(arr[left] <= pivot_item)
                {
                        left++;
                }
                while(arr[right] > pivot_item)
                {
                        right--;
                }
                if(left < right)
                {
                        swap(arr, left, right);
                }
        }
```

```
        arr[low] = arr[right];
        arr[right] = pivot_item;
        return right;
}
```

c)

```
private static int partition(int[] arr, int low, int high)
{
        int left, right, pivot_item = arr[low];
        left = low;
        right = high;
        while(left <= right)
        {
                while(arr[left] > pivot_item)
                {
                        left++;
                }
                while(arr[right] <= pivot_item)
                {
                        right--;
                }
                if(left < right)
                {
                        swap(arr, left, right);
                }
        }
        arr[low] = arr[right];
        arr[right] = pivot_item;
        return right;
}
```

d)

```
private static int partition(int[] arr, int low, int high)
{
        int left, right, pivot_item = arr[low];
        left = low;
        right = high;
        while(left > right)
        {
                while(arr[left] > pivot_item)
                {
                        left++;
                }
                while(arr[right] <= pivot_item)
                {
                        right--;
                }
                if(left < right)
                {
```

```
                        swap(arr, left, right);
                }
        }
        arr[low] = arr[right];
        arr[right] = pivot_item;
        return right;
}
```

Ans:B Explanation: The array is partitioned such that the elements left to the pivot are lesser than the pivot while the elements right of the pivot are greater than the pivot.

28.What is the best case complexity of QuickSort?

a) O(nlogn)

b) O(logn)

c) O(n)

d) O(n2)

Answer: a

Explanation: The array is partitioned into equal halves, using the Divide and Conquer master theorem, the complexity is found to be O(nlogn).

29. The given array is arr = {2,3,4,1,6}. What are the pivots that are returned as a result of subsequent partitioning?

a) 1 and 3

b) 3 and 1

c) 2 and 6

d) 6 and 2

Answer: a

Explanation: The call to partition returns 1 and 3 as the pivot elements.

30. What is the average case complexity of QuickSort?

a) O(nlogn)

b) O(logn)

c) O(n)

d) O(n2)

Answer: a

Explanation: The position of partition(split) is unknown, hence all(n) possibilities are considered, the average is found by adding all and dividing by n.

31. The given array is arr = {2,6,1}. What are the pivots that are returned as a result of subsequent partitioning?
a) 1 and 6
b) 6 and 1
c) 2 and 6
d) 1
Ans: d
Explanation: There is only one pivot with which the array will be sorted, the pivot is 1.

32. Which of the following is not true about QuickSort?
a) in-place algorithm
b) pivot position can be changed
c) adaptive sorting algorithm
d) can be implemented as a stable sort
Answer: b
Explanation: Once a pivot is chosen, its position is finalized in the sorted array, it cannot be modified.

33. Quick sort uses which of the following algorithm to implement sorting?
a) backtracking
b) greedy algorithm
c) divide and conquer
d) dynamic programming
Answer: c
Explanation: Quick sort uses the technique of divide and conquer in order to sort a given array. It divides the array into two parts about the pivot and then apply a quick sort to both the parts.

34. What is a randomized quick sort?
a) quick sort with random partitions
b) quick sort with random choice of pivot
c) quick sort with random output
d) quick sort with random input
Answer: b
Explanation: Randomized quick sort chooses a random element as a pivot. It is done so as to avoid the worst case of quick sort in which the input array is already sorted

35. What is the purpose of using randomized quick sort over standard quick sort?
a) so as to avoid worst case time complexity
b) so as to avoid worst case space complexity
c) to improve accuracy of output
d) to improve average case time complexity
Answer: a
Explanation: Randomized quick sort helps in avoiding the worst case time complexity of O(n2) which occurs in case when the input array is already sorted. However the average case and best case time complexities remain unaltered.

36. What is the auxiliary space complexity of randomized quick sort?
a) O(1)
b) O(n)
c) O(log n)
d) O(n log n)
Answer: c
Explanation: Auxiliary space complexity of randomized quick sort is O(log n) which is used for storing call stack formed due to recursion. Note that the algorithms with space complexity as O(log n) also qualifies as in place algorithms as the value of log n is close to 1.

37.What is the average time complexity of randomized quick sort?
a) O(n log n)
b) O(n2)
c) O(n2 log n)
d) O(n log n2)
Answer: a
Explanation: The average case time complexity of randomized quick sort is same as that of standard quick sort as randomized quick sort only helps in preventing the worst case. It is equal to O(n log n).

38. Quick sort uses which of the following method to implement sorting?
a) merging
b) partitioning
c) selection
d) exchanging
Answer: b
Explanation: Quick sort makes partitions of the input array about the pivot in order to implement sorting. Thus its method of sorting is called partitioning.

39. Randomized quick sort is an in place sort.
a) true
b) false
Answer: a
Explanation: In-place algorithms requires constant or very less auxiliary space. Quick sort qualifies as an in place sorting algorithm as it has a very low auxiliary space requirement of O(log n).

40. Randomized quick sort is a stable sort.
a) true
b) false
Answer: b
Explanation: Randomized quick sort like standard quick sort is also not a stable sorting algorithm. It is because the elements with the same values are not guaranteed to appear in the same relative order in the output sorted array.

41. What is the best case time complexity randomized quick sort?
a) O(log n)
b) O(n log n)
c) O(n2)
d) O(n2 log n)
Answer: b
Explanation: Best case time complexity is given in the case when there is equal partitioning of the array about the pivot. It is given by the relation T(n) = 2T(n/2) + n which gives the result O(n log n).

42. Which of the following is incorrect about randomized quicksort?
a) it has the same time complexity as standard quick sort
b) it has the same space complexity as standard quick sort
c) it is an in-place sorting algorithm
d) it cannot have a time complexity of O(n2) in any case.
Answer: d
Explanation: Randomized quick sort prevents the worst case complexity of O(n2) in most of the cases. But in some rare cases the time complexity can become O(n2). The probability of such a case is however very low.

43.Which of the following function chooses a random index as pivot.

a)

```
void partition_random(int arr[], int low, int high)
{
    srand(time(NULL));
    int random = low + rand() % (high - low);
    swap(arr[random], arr[high]);
}
```

b)

```
void partition_random(int arr[], int low, int high)
{
    srand(time(NULL));
    int random = high + rand() % (high - low);
    swap(arr[random], arr[high]);
}
```

c)

```
void partition_random(int arr[], int low, int high)
{
    srand(1);
    int random = low + rand() % (high - low);
    swap(arr[random], arr[high]);
}
```

d)

```
void partition_random(int arr[], int low, int high)
{
    srand(time(NULL));
    int random = low + rand() % (high - low-1);
    swap(arr[low], arr[high]);
}
```

Answer: a

Explanation: For generating unique random numbers every time we use srand(time(NULL)). Then after generating the random index we swap the value of element at the random index with the element at last index.

44. What is the worst case time complexity of randomized quicksort?

a) O(n)

b) O(n log n)

c) O(n2)

d) O(n2 log n)

Answer: c

Explanation: Randomized quicksort prevents the worst case complexity of O(n2) in most of the cases. But in some rare cases the time complexity can become O(n2). The probability of such a case is however very low.

45. What is the median of three techniques in quick sort?
a) quick sort with random partitions
b) quick sort with random choice of pivot
c) choosing median element as pivot
d) choosing median of first, last and middle element as pivot
Answer: d
Explanation: In the median of three technique the median of first, last and middle element is chosen as the pivot. It is done so as to avoid the worst case of quick sort in which the time complexity shoots to O(n2)

46. What is the purpose of using a median of three quick sort over standard quick sort?
a) so as to avoid worst case time complexity
b) so as to avoid worst case space complexity
c) to improve accuracy of output
d) to improve average case time complexity
Answer: a
Explanation: Median of three quick sort helps in avoiding the worst case time complexity of O(n2) which occurs in case when the input array is already sorted. However, the average case and best case time complexities remain unaltered.

47.What is the auxiliary space complexity of a median of three quick sort?
a) O(1)
b) O(n)
c) O(log n)
d) O(n log n)
Answer: c
Explanation: Auxiliary space complexity of median of three quick sort is O(log n) which is used for storing call stack formed due to recursion. Note that the algorithms with space complexity as O(log n) also qualifies as in place algorithms as the value of log n is close to 1.

48. What is the average time complexity of the median of three quick sort?
a) O(n log n)
b) O(n2)
c) O(n2 log n)
d) O(n log n2)
Answer: a
Explanation: The average case time complexity of median of three quick sort is the same as that of a standard quick sort as randomized quick sort only helps in preventing the worst case. It is equal to O(n log n).

49.Quick sort uses which of the following method to implement sorting?
a) merging
b) partitioning
c) selection
d) exchanging
Answer: b
Explanation: Quick sort makes partitions of the input array about the pivot in order to implement sorting. Thus its method of sorting is called partitioning.

50. Median of three quick sort is an in place sort.
a) true
b) false
View Answer
Answer: a
Explanation: In-place algorithms require constant or very less auxiliary space. Median of three quick sort qualifies as an in-place sorting algorithm. It has a very low auxiliary space requirement of O(log n).

51. Median of three quick sort is a stable sort.
a) true
b) false
View Answer
Answer: b
Explanation: Median of three quick sort like standard quick sort is also not a stable sorting algorithm. It is because the elements with the same values are not guaranteed to appear in the same relative order in the output sorted array.

52. What is the best case time complexity Median of three quick sort?
a) O(log n)
b) O(n log n)
c) $O(n^2)$
d) $O(n^2 \log n)$
Answer: b
Explanation: Best case time complexity is given in the case when there is equal partitioning of the array about the pivot. It is given by the relation $T(n) = 2T(n/2) + n$ which gives the result O(n log n).

53 Which of the following function chooses a random index as the pivot?
a)

```
int Median(arr, left, right)
{
    int mid;
    mid = (left + right)/2
    if (arr[right] < arr[left]);
        Swap(arr, left, right); //to swap arr[left],arr[right]
    if (arr[mid] < arr[left]);
        Swap(arr, mid, left);//to swap arr[left],arr[mid]
    if (arr[right] < arr[mid]);
        Swap(arr, right, mid);// to swap arr[right],arr[mid]
    return mid;
}
```

b)

```
int Median(arr, left, right)
{
    int mid;
    mid = (left + right)/2
    if (arr[right] > arr[left]);
        Swap(arr, left, right); //to swap arr[left],arr[right]
    if (arr[mid] < arr[left]);
        Swap(arr, mid, left);//to swap arr[left],arr[mid]
    if (arr[right] < arr[mid]);
        Swap(arr, right, mid);// to swap arr[right],arr[mid]
    return mid;
}
```

c)

```
int Median(arr, left, right)
```

```
{
    int mid;
    mid = (left + right)/2
    if (arr[left] < arr[right]);
        Swap(arr, left, right); //to swap arr[left],arr[right]
    if (arr[left] < arr[mid]);
        Swap(arr, mid, left);//to swap arr[left],arr[mid]
    if (arr[right] < arr[mid]);
        Swap(arr, right, mid);// to swap arr[right],arr[mid]
    return mid;
}
```

d)

```
int Median(arr, left, right)
{
    int mid;
    mid = (left + right)/2
    if (arr[right] < arr[left]);
        Swap(arr, left, right); //to swap arr[left],arr[right]
    if (arr[left] < arr[mid]);
        Swap(arr, mid, left);//to swap arr[left],arr[mid]
    if (arr[mid] < arr[right]);
        Swap(arr, right, mid);// to swap arr[right],arr[mid]
    return mid;
}
```

Answer: a
Explanation: In the median of three quick sort the median of first, last and middle element is chosen. Then the chosen element is taken as a pivot. This helps in avoiding the worst case of $O(n^2)$.

54. What will be the pivot for the array arr={8,2,4,9} for making the first partition when a median of three quick sort is implemented?

a) 8

b) 2

c) 4

d) 9

Answer: a

Explanation: While making the first partition the first, last and middle elements will be 8, 9 and 2 respectively. Thus the median element will be 8.

55. What is recurrence for worst case of QuickSort and what is the time complexity in Worst case?

a) Recurrence is T(n) = T(n-2) + O(n) and time complexity is O(n^2)
b) Recurrence is T(n) = T(n-1) + O(n) and time complexity is O(n^2)
c) Recurrence is T(n) = 2T(n/2) + O(n) and time complexity is O(nLogn)
d) Recurrence is T(n) = T(n/10) + T(9n/10) + O(n) and time complexity is O(nLogn)

56. Suppose we have an O(n) time algorithm that finds the median of an unsorted array. Now consider a QuickSort implementation where we first find the median using the above algorithm, then use the median as a pivot. What will be the worst-case time complexity of this modified QuickSort?

a) O(n^2 Logn)
b) O(n^2)
c) O(n Logn Logn)
d) O(nLogn)

Ans: If we use the median as a pivot element, then the recurrence for all cases becomes T(n) = 2T(n/2) + O(n)
The above recurrence can be solved using Master method. It falls in case 2 of the master method.
So, the worst-case time complexity of this modified QuickSort is O(nLogn).

57. Which of the following is not a stable sorting algorithm in its typical implementation.

a) Insertion Sort
b) Merge Sort
c) Quick Sort
d) Bubble Sort

Ans: Quick Sort is not a stable sorting algorithm in its typical implementation.

58.Which of the following sorting algorithms in its typical implementation gives best performance when applied on an array which is sorted or almost sorted (maximum 1 or two elements are misplaced).

a) Quick Sort
b) Heap Sort
c) Merge Sort
d) Insertion Sort

Explanation

Insertion sort takes linear time when input array is sorted or almost sorted (maximum 1 or 2 elements a re misplaced). All other sorting algorithms mentioned above will take more than linear time in their typi cal implementation.

59. Given an unsorted array. The array has this property that every element in the array is at most k distance from its position in a sorted array where k is a positive integer smaller than the size of an array. Which sorting algorithm can be easily modified for sorting this array and what is the obtainable time complexity?

Insertion Sort with time complexity O(kn)
Heap Sort with time complexity O(nLogk)
Quick Sort with time complexity O(kLogk)
Merge Sort with time complexity O(kLogk)

Explanation

We can perform this in O(nlogK) time using heaps:
First, create a min-heap with first k+1 elements. Now, we are sure that the smallest element will be in thi s K+1 element. Now, remove the smallest element from the min-heap(which is the root) and put it in the result array. Next, insert another element from the unsorted array into the mean-heap, now, the second s mallest element will be in this..extract it from the mean-heap and continue this until no more elements ar e in the unsorted array. Next, use a simple heap sort for the remaining elements.
Time Complexity:
O(k) to build the initial min-heap
O((n-k)logk) for remaining elements.
Thus we get O(nlogk). Hence, option B is correct.

60. Suppose we are sorting an array of eight integers using quicksort, and we have just finished the first partitioning with the array looking like this:

```
2 5 1 7 9 12 11 10
```

Which statement is correct?

a) The pivot could be either the 7 or the 9.
b) The pivot could be the 7, but it is not the 9
c) The pivot is not the 7, but it could be the 9
d) Neither the 7 nor the 9 is the pivot.

Explanation

7 and 9 both are at their correct positions (as in a sorted array). Also, all elements on left of 7 and 9 are smaller than 7 and 9 respectively and on right are greater than 7 and 9 respectively.

61. You have to sort 1 GB of data with only 100 MB of available main memory. Which sorting technique will be most appropriate?

a) Heap sort
b) Merge sort
c) Quick sort
d) Insertion sort

Explanation

The data can be sorted using external sorting which uses merging technique. This can be done as follo ws: 1. Divide the data into 10 groups each of size 100. 2. Sort each group and write them to disk. 3. Loa d 10 items from each group into main memory. 4. Output the smallest item from the main memory to disk. Load the next item from the group whose item was chosen. 5. Loop step #4 until all items are not outputted. The step 3-5 is called as merging technique.

62. In quick sort, for sorting n elements, the (n/4)th smallest element is selected as a pivot using an O(n) time algorithm. What is the worst-case time complexity of the quick sort?
(A) $theta\ theta$ (n)
(B) $theta\ theta$ (n*log(n))
(C) $theta\ theta$ (n^2)
(D) $theta\ theta$ ($n^{2 \log n}$)

A
B
C
D

The recursion expression becomes: T(n) = T(n/4) + T(3n/4) + cn After solving the above recursion, we g et $theta\ theta$ (n*log(n)).

63. Consider the Quicksort algorithm. Suppose there is a procedure for finding a pivot element that splits the list into two sub-lists each of which contains at least one-fifth of the elements. Let T(n) be the number of comparisons required to sort n elements. Then

T(n) <= 2T(n/5) + n
T(n) <= T(n/5) + T(4n/5) + n
T(n) <= 2T(4n/5) + n
T(n) <= 2T(n/2) + n

Explanation

For the case where n/5 elements are in one subset, T(n/5) comparisons are needed for the first subset w ith n/5 elements, T(4n/5) is for the rest of 4n/5 elements, and n is for finding the pivot. If there are more than n/5 elements in one set then other sets will have less than 4n/5 elements and time complexity will be less than T(n/5) + T(4n/5) + n because the recursion tree will be more balanced.

64. Which sorting algorithms is most efficient to sort string consisting of ASCII characters?

a) Quick sort
b) Heap sort
c) Merge sort
d) Counting sort

Explanation

Counting sort algorithm is efficient when range of data to be sorted is fixed. In the above question, the range is from 0 to 255(ASCII range). Counting sort uses an extra constant space proportional to range o f data.

65. Which of the following is true about merge sort?

a) Merge Sort works better than quick sort if data is accessed from slow sequential memory.
b) Merge Sort is stable sort by nature
c) Merge sort outperforms heap sort in most of the practical situations.
d) All of the above.

Ans: Merge Sort satisfies all the three options given above.

66.Which one of the following in place sorting algorithms needs the minimum number of swaps?

a) Quick sort
b) Insertion sort
c) Selection sort
d) Heap sort

67. Suppose we have a O(n) time algorithm that finds median of an unsorted array. Now consider a QuickSort implementation where we first find median using the above algorithm, then use median as pivot. What will be the worst case time complexity of this modified QuickSort.

a) O(n^2 Logn)
b) O(n^2)
c) O(n Logn Logn)
d) O(nLogn)

Explanation:
When we choose median as pivot element then after the partition algorithm it will go in the middle of the array having half of the elements to left the left and half in the right. Thus after partition algorithm the array will be divided into two equal parts of n/2 elements each. Hence the resultant recurrence relation would be- T(n) = O(n) (for selecting median) + O(n) (for partition) + T(n/2) + T(n/2) T(n) = O(n) + 2T(n/2) We can solve the above recurrence relation using master method Reference: https://en.wikipedia.org/wiki/Master_theorem Same as http://geeksquiz.com/algorithms-searching-and-sorting-question-3/ This solution is contributed by Parul Sharma.

68. A machine needs a minimum of 100 sec to sort 1000 names by quick sort. The minimum time needed to sort 100 names will be approximately

a) 50.2 sec
b) 6.7 sec
c) 72.7 sec
d) 11.2 sec

Explanation
Best case time complexity of quick sort to take the minimum time to sort names = O(n log n)
let t1 = time taken to sort 1000 names = 100 sec;
n1=1000 names
let t2 = time taken to sort 100 names;
n2= 100 names

t1/t2 = n1*log(n1) / n2*log(n2)
100/t2 = 1000*log(1000) / 100*log(100)
100/t2 = 10*log(1000) / log(100)
100/t2 = 10*3 / 2
t2 = 20/3 = 6.66

69. In quick sort, for sorting n elements, the (n/4)th smallest element is selected as a pivot using an O(n) time algorithm. What is the worst-case time complexity of the quick sort?
(A) θ(n)
(B) θ(n*log(n))
(C) θ(n^2)
(D) θ(n^2 log n)

Explanation
The recursion expression becomes T(n) = T(n/4) + T(3n/4) + cn
After solving the above recursion, we get θ(n*log(n)).

70. Consider the Quicksort algorithm. Suppose there is a procedure for finding a pivot element which splits the list into two sub-lists each of which contains at least one-fifth of the elements. Let T(n) be the number of comparisons required to sort n elements. Then

a. $T(n) <= 2T(n/5) + n$
b. $T(n) <= T(n/5) + T(4n/5) + n$
c. $T(n) <= 2T(4n/5) + n$
d. $T(n) <= 2T(n/2) + n$

71.Let P be a QuickSort Program to sort numbers in ascending order using the first element as pivot. Let t1 and t2 be the number of comparisons made by P for the inputs {1, 2, 3, 4, 5} and {4, 1, 5, 3, 2} respectively. Which one of the following holds?

a) t1 = 5
b) t1 < t2
c) t1 > t2
d) t1 = t2

When first element or last element is chosen as pivot, Quick sort's worst case occurs for the sorted arrays. In every step of quick sort, numbers are divided as per the following recurrence. T(n) = T(n-1) + O(n)

72. You have an array of n elements. Suppose you implement quick sort by always choosing the central element of the array as the pivot. Then the tightest upper bound for the worst case performance is

a) $O(n^2)$
b) O(n*log(n))
c) Theta(n*log(n))
d) $O(n^3)$

Explanation

For any input, there are some permutations for which worst case will be $O(n^2)$. In some cases, choosing t he middle element minimizes the chances of encountering $O(n^2)$, but in worst case it can go to $O(n^2)$. Whi chever element we take as Pivot, either first or middle, worst case will be $O(n^2)$ since Pivot is fixed in pos ition. While choosing a random pivot minimizes the chances of encountering worst case i.e. $O(n^2)$. Refer t his article on Quick Sort.

73. Randomized quicksort is an extension of quicksort where the pivot is chosen randomly. What is the worst case complexity of sorting n numbers using randomized quicksort?

a) O(n)
b) O(n*log(n))
c) $O(n^2)$
d) O(n!)

Explanation

If all elements of the given array are same then that is the worst case for the randomised quicksort. And time complexity of worst case of quicksort is $O(n^2)$ that is proven already. So, option (C) is correct.

74.Which of the following changes to typical QuickSort improves its performance on average and are generally done in practice.

1) Randomly picking up to make worst case less likely to occur.
2) Calling insertion sort for small sized arrays to reduce recursive calls.
3) QuickSort is tail recursive, so tail call optimizations can be done.
4) A linear time median searching algorithm is used to pick the median, so that the worst case time reduces to O(nLogn)

a) 1 and 2
b) 2, 3, and 4
c) 1, 2 and 3
d) 2, 3 and 4

Explanation

The 4th optimization is generally not used, it reduces the worst case time complexity to O(nLogn), but th e hidden constants are very high.

75. Which one of the following is the recurrence equation for the worst case time complexity of the Quicksort algorithm for sorting n(≥ 2) numbers? In the recurrence equations given in the options below, c is a constant.

a) T(n) = 2T (n/2) + cn
b) T(n) = T(n – 1) + T(0) + cn
c) T(n) = 2T (n – 2) + cn
d) T(n) = T(n/2) + cn

Explanation

In the worst case, the chosen pivot is always placed at a corner position and recursive call is made for the following:

a) For subarray on left of pivot which is of size n -1 in worst case.
b) For subarray on right of pivot which is of size 0 in worst case.

76. Consider a list of recursive algorithms and a list of recurrence relations as shown below. Each recurr ence relation corresponds to exactly one algorithm and is used to derive the time complexity of the algori thm.

Recursive Algorithm		**Recurrence Relation**	
P.	Binary search	I.	T(n) = T(n-k) + T(k) + cn
Q.	Merge sort	II.	T(n) = 2T(n-1) + 1
R.	Quick sort	III.	T(n) = 2T(n/2) + cn
S.	Tower of Hanoi	IV.	T(n) = T(n/2) + 1

a) P-II, Q-III, R-IV, S-I
b) P-IV, Q-III, R-I, S-II
c) P-III, Q-II, R-IV, S-I
d) P-IV, Q-II, R-I, S-III

Explanation

These are examples of some standard algorithms whose Merge Sort: T(n) = 2T(n/2) + Θ(n). It falls in case 2 as c is 1 and $\text{Log}_b a$] is also 1 and the solution is Θ(n Logn) *//time complexity can be evaluated us ing Master Method* Binary Search: T(n) = T(n/2) + Θ(1). It also falls in case 2 as c is 0 and $\text{Log}_b a$ is also 0 a nd the solution is Θ(Logn) *//time complexity can be evaluated using Master Method* Quick Sort : Time take n by QuickSort in general can be written as T(n) = T(k) + T(n-k-1) +*thetatheta*(n) Tower of Hanoi : T(n) = 2T(n-1) + 1 Read More at : http://www.geeksforgeeks.org/analysis-algorithm-set-4-master-method-so lving-recurrences/

77. Which of the following sorting algorithms has the minimum running time complexity in the best and average case?

a) Insertion sort, Quick sort
b) Quick sort, Quick sort
c) Quick sort, Insertion sort
d) Insertion sort, Insertion sort

Explaination
Insertion sort has a best case complexity of O(n), if the array is already sorted while it has an average case complexity of O(n2) Quick sort has a best case complexity of O(n log n), while it has an average case complexity of O(n log n) also. So, option (A) is correct.

78. Which is the correct order of the following algorithms with respect to their time Complexity in the best case ?

a) Merge sort > Quick sort >Insertion sort > selection sort
b) insertion sort < Quick sort < Merge sort < selection sort
c) Merge sort > selection sort > quick sort > insertion sort
d) Merge sort > Quick sort > selection sort > insertion sort

Explanation
In best case,
Quick sort: O (nlogn)
Merge sort: O (nlogn)
Insertion sort: O (n)
Selection sort: O (n^2)

79. What is the best sorting algorithm to use for the elements in array are more than 1 million in general?

a) Merge sort.
b) Bubble sort
c) Quick sort.
d) Insertion sort.

Explanation
Most practical implementations of Quick Sort use randomized version. The randomized version has expected time complexity of O(nLogn). The worst case is possible in randomized version also, but worst case doesn't occur for a particular pattern (like sorted array) and randomized Quick Sort works well in practice. Quick Sort is also a cache friendly sorting algorithm as it has good locality of reference when used for arrays. Quick Sort is also tail recursive, therefore tail call optimizations is done.

80. Quicksort is run on two inputs shown below to sort in ascending order taking the first element as pivot,
(i) 1, 2, 3,....., n
(ii) n, n-1, n-2,....., 2, 1
Let C1 and C2 be the number of comparisons made for the inputs (i) and (ii) respectively. Then,

a) C1 < C2
b) C1 > C2
c) C1 = C2
d) We cannot say anything for arbitrary n

Explanation
Quick Sort performs in the worst case when input is sorted in either ascending order or descending orde r. So, quick sort will return equal number of comparisons for both given inputs. Option (C) is correct.