Algorithms MCQ

1. Merge sort uses which of the following technique to implement sorting?

   a) backtracking
   b) greedy algorithm
   c) divide and conquer
   d) dynamic programming

2. What is the average case time complexity of merge sort?
   a) O(n log n)
   b) O(n2)
   c) O(n2 log n)
   d) O(n log n2)

3. What is the auxiliary space complexity of merge sort?

   a) O(1)
   b) O(log n)
   c) O(n)
   d) O(n log n)

4. Merge sort can be implemented using O(1) auxiliary space.
   a) true
   b) false

5. What is the worst case time complexity of merge sort?

   a) O(n log n)
   b) O(n2)
   c) O(n2 log n)

d) O(n log n2)

6. Which of the following method is used for sorting in merge sort?

a) merging
b) partitioning
c) selection
d) exchanging

7. What will be the best case time complexity of merge sort?

a) O(n log n)
b) O(n2)
c) O(n2 log n)
d) O(n log n2)

8. Which of the following is not a variant of merge sort?

a) in-place merge sort
b) bottom up merge sort
c) top down merge sort
d) linear merge sort

9. Choose the incorrect statement about merge sort from the following?

a) it is a comparison based sort
b) it is an adaptive algorithm
c) it is not an in place algorithm
d) it is stable algorithm

10. Which of the following is not in place sorting algorithm by default?

a) merge sort
b) quick sort
c) heap sort

d) insertion sort

11. Which of the following is not a stable sorting algorithm?

a) Quick sort
b) Cocktail sort
c) Bubble sort
d) Merge sort

12. Which of the following stable sorting algorithm takes the least time when applied to an almost sorted array?

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

13. Merge sort is preferred for arrays over linked lists.

a) true
b) false

14. Which of the following sorting algorithm makes use of merge sort?

a) tim sort
b) intro sort
c) bogo sort
d) quick sort

15. Which of the following sorting algorithm does not use recursion?

a) quick sort
b) merge sort
c) heap sort
d) bottom up merge sort

16. Choose the correct code for merge sort.

a)

```
void merge_sort(int arr[], int left, int right)
{
    if (left > right)
    {
        int mid = (right-left)/2;
        merge_sort(arr, left, mid);
        merge_sort(arr, mid+1, right);
        merge(arr, left, mid, right); //function to merge sorted arrays
    }
}
```

[✓ b)]

```
void merge_sort(int arr[], int left, int right)
{
    if (left < right)
    {
        int mid = left+(right-left)/2;
        merge_sort(arr, left, mid);
        merge_sort(arr, mid+1, right);
        merge(arr, left, mid, right); //function to merge sorted arrays
    }
}
```

c)

```
void merge_sort(int arr[], int left, int right)
{
    if (left < right)
    {
        int mid = left+(right-left)/2;
        merge(arr, left, mid, right); //function to merge sorted arrays
        merge_sort(arr, left, mid);
        merge_sort(arr, mid+1, right);
```

```
   }
}
```

d)

```
void merge_sort(int arr[], int left, int right)
{
   if (left < right)
   {
      int mid = (right-left)/2;
      merge(arr, left, mid, right); //function to merge sorted arrays
      merge_sort(arr, left, mid);
      merge_sort(arr, mid+1, right);
   }
}
```

17. How many passes does an insertion sort algorithm consist of?

a) N
b) N-1
c) N+1
d) N

18. Which of the following algorithm implementations is similar to that of an insertion sort?

a) Binary heap
b) Quick sort
c) Merge sort
d) Radix sort

19. What is the average case running time of an insertion sort algorithm?

a) O(N)
b) O(N log N)
c) O(log N)
d) $O(N^2)$

20. Any algorithm that sorts by exchanging adjacent elements require $O(N^2)$ on average.

a) True
b) False

21. What is the average number of inversions in an array of N distinct numbers?

a) N(N-1)/4
b) N(N+1)/2
c) N(N-1)/2
d) N(N-1)/3

22. What is the running time of an insertion sort algorithm if the input is pre-sorted?

a) O(N2)
b) O(N log N)
c) O(N)
d) O(M log N)

23. What will be the number of passes to sort the elements using insertion sort?

14, 12, 16, 6, 3, 10

a) 6
b) 5
c) 7
d) 1

24. For the following question, how will the array elements look like after second pass?

34, 8, 64, 51, 32, 21

a) 8, 21, 32, 34, 51, 64
b) 8, 32, 34, 51, 64, 21
c) 8, 34, 51, 64, 32, 21
d) 8, 34, 64, 51, 32, 21

25. Which of the following real time examples is based on insertion sort?

a) arranging a pack of playing cards
b) database scenarios and distributes scenarios
c) arranging books on a library shelf
d) real-time systems

26. In C, what are the basic loops required to perform an insertion sort?

a) do- while
b) if else
c) for and while
d) for and if

27. Binary search can be used in an insertion sort algorithm to reduce the number of comparisons.

a) True
b) False

28. Which of the following options contain the correct feature of an insertion sort algorithm?

a) anti-adaptive
b) dependable
c) stable, not in-place

d) stable, adaptive

29. Which of the following sorting algorithms is the fastest for sorting small arrays?

a) Quick sort

b) Insertion sort

c) Shell sort

d) Heap sort

30. For the best case input, the running time of an insertion sort algorithm is?

a) Linear

b) Binary

c) Quadratic

d) Depends on the input

31. Which of the following examples represent the worst case input for an insertion sort?

a) array in sorted order

b) array sorted in reverse order

c) normal unsorted array

d) large array

32. Which of the following is correct with regard to insertion sort?

a) insertion sort is stable and it sorts In-place

b) insertion sort is unstable and it sorts In-place

c) insertion sort is stable and it does not sort In-place

d) insertion sort is unstable and it does not sort In-place

33. Which of the following sorting algorithm is best suited if the elements are already sorted?

a) Heap Sort

b) Quick Sort

c) Insertion Sort

d) Merge Sort

34. The worst case time complexity of insertion sort is O(n^2). What will be the worst case time complexity of insertion sort if the correct position for inserting element is calculated using binary search?

a) O(nlogn)

b) O(n^2)

c) O(n)

d) O(logn)

35. Insertion sort is an example of an incremental algorithm.

a) True

b) False

36. Consider the code given below, which runs insertion sort:

```
void insertionSort(int arr[], int array_size)
{
 int i, j, value;
 for (i = 1; i < array_size; i++)
 {
        value = arr[i];
        j = i;
        while (_______ )
        {
                arr[j] = arr[j − 1];
                j = j − 1;
        }
        arr[j] = value;
 }
}
```

Which condition will correctly implement the while loop?

a) (j > 0) || (arr[j − 1] > value)
b) (j > 0) && (arr[j − 1] > value)
c) (j > 0) && (arr[j + 1] > value)
d) (j > 0) && (arr[j + 1] < value)

37. Which of the following is good for sorting arrays having less than 100 elements?

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

38. Consider an array of length 5, arr[5] = {9,7,4,2,1}. What are the steps of insertions done while running insertion sort on the array?

a) 7 9 4 2 1   4 7 9 2 1   2 4 7 9 1   1 2 4 7 9
b) 9 7 4 1 2   9 7 1 2 4   9 1 2 4 7   1 2 4 7 9
c) 7 4 2 1 9   4 2 1 9 7   2 1 9 7 4   1 9 7 4 2
d) 7 9 4 2 1   2 4 7 9 1   4 7 9 2 1   1 2 4 7 9

39. **Statement 1:** In insertion sort, after m passes through the array, the first m elements are in sorted order.

Statement 2: And these elements are the m smallest elements in the array.

a) Both the statements are true
b) Statement 1 is true but statement 2 is false
c) Statement 1 is false but statement 2 is true
d) Both the statements are false

40. In insertion sort, the average number of comparisons required to place the 7th element into its correct position is ____

a) 9

b) 4

c) 7

d) 14

41. Which of the following is not an exchange sort?

a) Bubble Sort

b) Quick Sort

c) Partition-exchange Sort

d) Insertion Sort

42. Which of the following sorting algorithms is the fastest?

a) Merge sort

b) Quick sort

c) Insertion sort

d) Shell sort

43. Quick sort follows Divide-and-Conquer strategy.

a) True

b) False

44. What is the worst case time complexity of a quick sort algorithm?

a) O(N)

b) O(N log N)

c) $O(N^2)$

d) O(log N)

45. 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

46. 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

47. 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

48. 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)

49. 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

50. Quick sort uses join operation rather than merge operation.

a) true
b) false

51. How many sub arrays does the quick sort algorithm divide the entire array into?

a) one
b) two
c) three
d) four

52. 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

53. 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

54. Quick sort is a ___________

a) greedy algorithm

b) divide and conquer algorithm

c) dynamic programming algorithm

d) backtracking algorithm

55. What is the worst case time complexity of the Quick sort?

a) O(nlogn)

b) O(n)

c) O(n3)

d) O(n^2)

56. 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

57. 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

58. Quick sort is a stable sorting algorithm.

a) True

b) False

59. 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

60. 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

61. 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

62. 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

63. Quick sort is a space-optimised version of ____

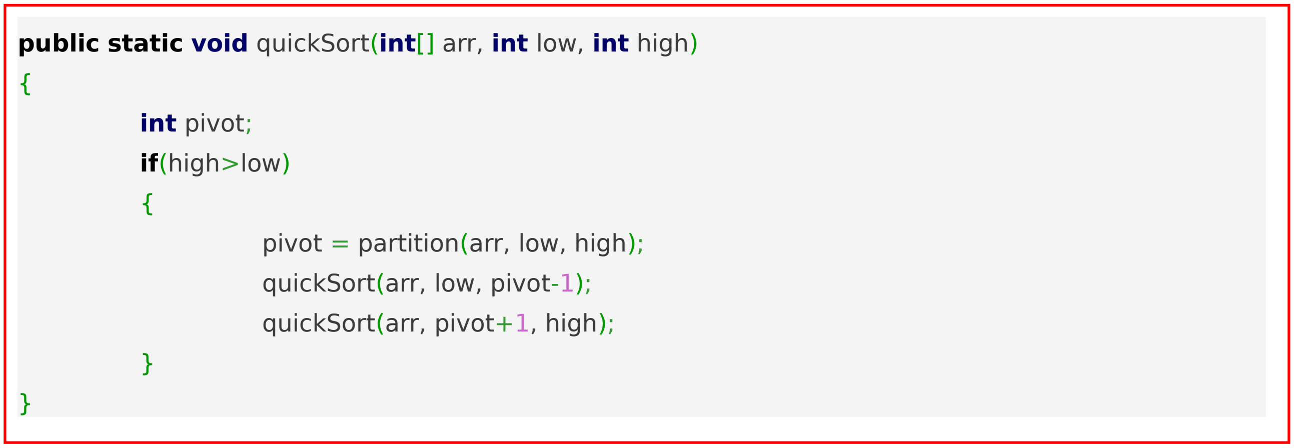
a) Bubble sort
b) Selection sort
c) Insertion sort
d) Binary tree sort

64. QuickSort can be categorized into which of the following?

a) Brute Force technique
b) Divide and conquer
c) Greedy algorithm
d) Dynamic programming

65. 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);
        }
}
```

66. 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

67. 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;
}
```

68. What is the best case complexity of QuickSort?

a) O(nlogn)

b) O(logn)

c) O(n)

d) O(n2)

69. 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

70. What is the average case complexity of QuickSort?

a) O(nlogn)

b) O(logn)

c) O(n)

d) O(n2)

71. 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

72. 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

73. Quick sort uses which of the following algorithm to implement sorting?

a) backtracking

b) greedy algorithm

c) divide and conquer

d) dynamic programming

74. 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

75. 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

76. 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)

77. 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)

78. Quick sort uses which of the following method to implement sorting?

a) merging

b) partitioning

c) selection

d) exchanging

79. Randomized quick sort is an in place sort.

a) true

b) false

80. Randomized quick sort is a stable sort.

a) true

b) false

81. 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)

82. 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.

83. 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]);
}
```

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

a) O(n)

b) O(n log n)

c) O(n^2)

d) O(n2 log n)

85. 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

86. 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

87. 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)

88. 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)

89. Median of three quick sort is an in place sort.

a) true
b) false

90. Median of three quick sort is a stable sort.

a) true
b) false

91. What is the best case time complexity Median of three quick sort?

a) O(log n)
b) O(n log n)
c) O(n2)
d) O(n2 log n)

92. 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;
}
```

93. 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

94. On which algorithm is heap sort based on?

a) Fibonacci heap
b) Binary tree
c) Priority queue
d) FIFO

95. In what time can a binary heap be built?

a) O(N)

b) O(N log N)

c) O(log N)

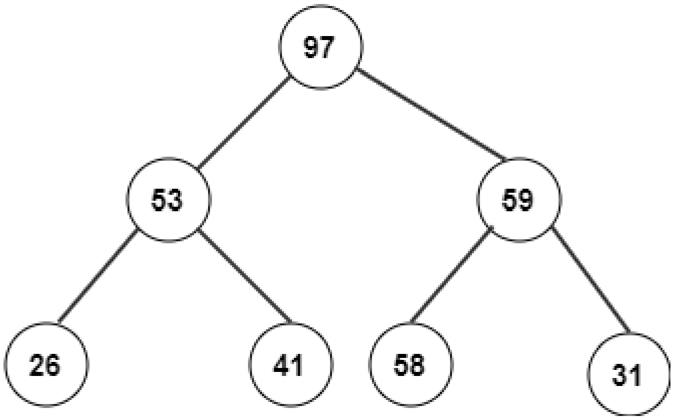
d) O(N2)

96. Heap sort is faster than Shell sort.

a) true

b) false

97. Consider the following heap after buildheap phase. What will be its corresponding array?



a) 26, 53, 41, 97, 58, 59, 31
b) 26, 31, 41, 53, 58, 59, 97
c) 26, 41, 53, 97, 31, 58, 59
d) 97, 53, 59, 26, 41, 58, 31

98. In what position does the array for heap sort contains data?
a) 0
b) 1
c) -1
d) anywhere in the array

99. In heap sort, after deleting the last minimum element, the array will contain elements in?
a) increasing sorting order
b) decreasing sorting order
c) tree inorder

d) tree preorder

100. What is the typical running time of a heap sort algorithm?

a) O(N)
b) O(N log N)
c) O(log N)
d) O(N2)

101. How many arrays are required to perform deletion operation in a heap?

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

102. What is the time taken to perform a delete min operation?

a) O(N)
b) O(N log N)
c) O(log N)
d) O(N2)

103. Heap sort is an extremely stable algorithm.

a) true
b) false

104. What is the average number of comparisons used in a heap sort algorithm?

a) N log N-O(N)
b) O(N log N)-O(N)
c) O(N log N)-1
d) 2N log N + O(N)

105. What is the time taken to copy elements to and from two arrays created for deletion?

a) O(N)

b) O(N log N)

c) O(log N)

d) O(N2)

106. What is the average number of comparisons used to heap sort a random permutation of N distinct items?

a) 2N log N-O(N)

b) 2N log N-O(N log N)

c) 2N log N-O(N log log N)

d) 2N log N-O(log N)

107. Heap sort is an implementation of _______ using a descending priority queue.

a) insertion sort

b) selection sort

c) bubble sort

d) merge sort

108. Which one of the following is false?

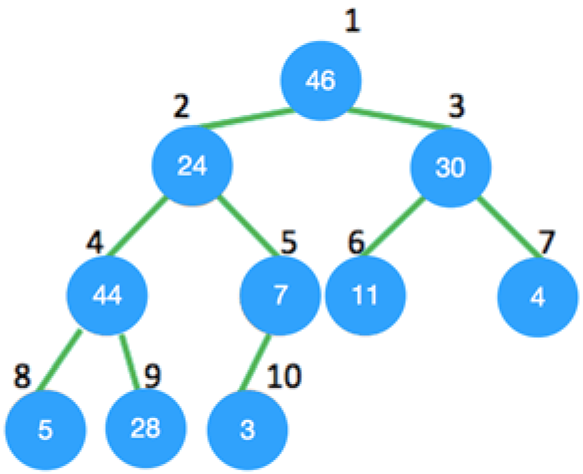
a) Heap sort is an in-place algorithm

b) Heap sort has O(nlogn) average case time complexity

c) Heap sort is stable sort

d) Heap sort is a comparison-based sorting algorithm

109. The essential part of Heap sort is construction of max-heap. Consider the tree shown below, the node 24 violates the max-heap property. Once heapify procedure is applied to it, which position will it be in?



a) 4
b) 5
c) 8
d) 9

110. The descending heap property is ___________

a) A[Parent(i)] = A[i]
b) A[Parent(i)] <= A[i]
c) A[Parent(i)] >= A[i]
d) A[Parent(i)] > 2 * A[i]

111. What is its wort case time complexity of Heap sort?

a) O(nlogn)
b) O(n2logn)

c) O(n2)

d) O(n3)

112. In average case Heap sort is as efficient as the Quick sort.

a) True

b) False

113. Choose the correct option to fill? X so that the code given below implements the Heap sort.

```
#include <stdio.h>
void heapify(int arr[], int n, int i)
{
   int largest = i; // Initialize largest as root
   int l = 2*i + 1; // left = 2*i + 1
   int r = 2*i + 2; // right = 2*i + 2
   if (l < n && arr[l] > arr[largest])
      largest = l;
   if (r < n && arr[r] > arr[largest])
      largest = r;
   if (largest != i)
   {
      swap(arr[i], arr[largest]);
      heapify(arr, n, largest);
   }
}
void heapSort(int arr[], int n)
{
   for (int i = n / 2 - 1; i >= 0; i--)
      heapify(arr, n, i);
   for (int i=n-1; i>=0; i--)
   {
      X;
      heapify(arr, i, 0);
   }
}
void printArray(int arr[], int n)
{
   for (int i=0; i<n; ++i)
      printf(“%d”,arr[i]);
   printf(“\n”);
}
int main()
{
   int arr[] = {12, 11, 13, 5, 6, 7};
   int n = sizeof(arr)/sizeof(arr[0]);
   heapSort(arr, n);
   printf(“Sorted array is \n");
   printArray(arr, n);
}
```

a) swap(arr[0], arr[n])

b) swap(arr[i], arr[n])

c) swap(arr[0], arr[i])

d) swap(arr[i], arr[2*i])

114. Which one of the following is a variation of Heap sort?

a) Comb sort

b) Smooth sort

c) Binary tree sort

d) Shell sort

115. Introsort algorithm is combination of _____________

a) Quick sort and Heap sort

b) Quick sort and Shell sort

c) Heap sort and Merge sort

d) Heap sort and insertion sort

116. How many elements can be sorted in O(logn) time using Heap sort?

a) O(1)

b) O(n/2)

c) O(logn/log(logn))

d) O(logn)

117. Master's theorem is used for?

a) solving recurrences

b) solving iterative relations

c) analysing loops

d) calculating the time complexity of any code

118. How many cases are there under Master's theorem?

a) 2

b) 3

c) 4
d) 5

119. What is the result of the recurrences which fall under first case of Master’s theorem (let the recurrence be given by T(n)=aT(n/b)+f(n) and f(n)=n^c?

a) T(n) = O(n^log$_b$a)
b) T(n) = O(nc log n)
c) T(n) = O(f(n))
d) T(n) = O(n^2)

120. What is the result of the recurrences which fall under second case of Master’s theorem (let the recurrence be given by T(n)=aT(n/b) +f(n) and f(n)=n^c?

a) T(n) = O(nlogba)
b) T(n) = O(n^c log n)
c) T(n) = O(f(n))
d) T(n) = O(n^2)

121. What is the result of the recurrences which fall under third case of Master’s theorem (let the recurrence be given by T(n)=aT(n/b)+f(n) and f(n)=n^c?

a) T(n) = O(nlog$_b$a)
b) T(n) = O(n^c log n)
c) T(n) = O(f(n))
d) T(n) = O(n^2)

122. We can solve any recurrence by using Master’s theorem.

a) true
b) false

123. Under what case of Master's theorem will the recurrence relation of merge sort fall?

a) 1
b) 2
c) 3
d) It cannot be solved using master's theorem

124. Under what case of Master's theorem will the recurrence relation of stooge sort fall?

a) 1
b) 2
c) 3
d) It cannot be solved using master's theorem

125. Which case of master's theorem can be extended further?

a) 1
b) 2
c) 3
d) No case can be extended

126. What is the result of the recurrences which fall under the extended second case of Master's theorem (let the recurrence be given by $T(n)=aT(n/b)+f(n)$ and $f(n)=n^c(\log n)^k$?

a) $T(n) = O(n\log_b a)$
b) $T(n) = O(nc \log n)$
c) $T(n)= O(n^c (\log n)^{k+1}$
d) $T(n) = O(n^2)$

127. Under what case of Master's theorem will the recurrence relation of binary search fall?

a) 1

b) 2

c) 3

d) It cannot be solved using master's theorem

128. Solve the following recurrence using Master's theorem.

$T(n) = 4T(n/2) + n^2$

a) T(n) = O(n)

b) T(n) = O(log n)

c) $T(n) = O(n^2 \log n)$

d) T(n) = O(n2)

129. Solve the following recurrence using Master's theorem.

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

a) $T(n) = O(n^2)$

b) $T(n) = O(n^2 \log n)$

c) $T(n) = O(2^n)$

d) cannot be solved

130. Solve the following recurrence using Master's theorem.

T(n) = 16T (n/4) + n

a) T(n) = O(n)

b) T(n) = O(log n)

c) $T(n) = O(n^2 \log n)$

d) $T(n) = O(n^2)$

131. Solve the following recurrence using Master's theorem.

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

a) T(n) = O(n)

b) T(n) = O(log n)

c) T(n) = O(n^2log n)

d) cannot be solved using master's theorem

132. Solve the following recurrence using Master's theorem.

T(n) = 0.7 T (n/2) + 1/n

a) T(n) = O(n)

b) T(n) = O(log n)

c) T(n) = O(n2log n)

d) cannot be solved using master's theorem

133. Solve the following recurrence using Master's theorem.

T(n) = 4 T (n/2) + n!

a) T(n) = O(n!)

b) T(n) = O(n! log n)

c) T(n) = O(n^2log n)

d) cannot be solved using master's theorem

134. Solve the following recurrence using Master's theorem.

T(n) = 4T (n/4) + n log n

a) T(n) = O(n (log n)^2)

b) T(n) = O(n log n)

c) T(n) = O(n^2log n)

d) cannot be solved using master's theorem

135. What will be the recurrence relation of the following code?

```
Int sum(int n)
{
  If(n==1)
     return 1;
  else
     return n+sum(n-1);
}
```

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

b) T(n) = T(n-1) + n

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

d) T(n) = T(n/2) + O(1)

136. What will be the recurrence relation of the following code?

```
int xpowy(int x, int n)
if (n==0) return 1;
if (n==1) return x;
if ((n % 2) == 0)
return xpowy(x*x, n/2);
else
return xpowy(x*x, n/2) * x;
```

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

b) T(n) = T(n-1) + n

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

d) T(n) = T(n/2) + O(1)

137. What will be the time complexity of the following code?

```
int xpowy(int x, int n)
{
   if (n==0)
      return 1;
   if (n==1)
      return x;
   if ((n % 2) == 0)
      return xpowy(x*x, n/2);
   else
      return xpowy(x*x, n/2) * x;
}
```

a) O(log n)

b) O(n)

c) O(n log n)

d) O(n^2)

138. Recursion is a method in which the solution of a problem depends on _____

a) Larger instances of different problems
b) Larger instances of the same problem
c) Smaller instances of the same problem
d) Smaller instances of different problems

139. Which of the following problems can't be solved using recursion?

a) Factorial of a number
b) Nth fibonacci number
c) Length of a string
d) Problems without base case

140. Recursion is similar to which of the following?

a) Switch Case
b) Loop
c) If-else
d) if elif else

141. In recursion, the condition for which the function will stop calling itself is _____________

a) Best case
b) Worst case
c) Base case
d) There is no such condition

142. Which of the following statements is true?

a) Recursion is always better than iteration
b) Recursion uses more memory compared to iteration
c) Recursion uses less memory compared to iteration
d) Iteration is always better and simpler than recursion

143. What will happen when the below code snippet is executed?

```
void my_recursive_function()
{
  my_recursive_function();
}
int main()
{
  my_recursive_function();
  return 0;
}
```

a) The code will be executed successfully and no output will be generated
b) The code will be executed successfully and random output will be generated
c) The code will show a compile time error
d) The code will run for some time and stop when the stack overflows

144. What is the output of the following code?

```
void my_recursive_function(int n)
{
   if(n == 0)
   return;
   printf("%d ",n);
   my_recursive_function(n-1);
}
int main()
{
   my_recursive_function(10);
   return 0;
}
```

a) 10
b) 1
c) 10 9 8 … 1 0
d) 10 9 8 … 1

145. What is the base case for the following code?

```
void my_recursive_function(int n)
{
    if(n == 0)
    return;
    printf("%d ",n);
    my_recursive_function(n-1);
}
int main()
{
    my_recursive_function(10);
    return 0;
}
```

a) return
b) printf(“%d “, n)
c) if(n == 0)
d) my_recursive_function(n-1)

146. How many times is the recursive function called, when the following code is executed?

```
void my_recursive_function(int n)
{
    if(n == 0)
    return;
    printf("%d ",n);
    my_recursive_function(n-1);
}
int main()
{
    my_recursive_function(10);
    return 0;
}
```

a) 9
b) 10
c) 11
d) 12

147. What does the following recursive code do?

```
void my_recursive_function(int n)
```

```
{
    if(n == 0)
    return;
    my_recursive_function(n-1);
    printf("%d ",n);
}
int main()
{
    my_recursive_function(10);
    return 0;
}
```

a) Prints the numbers from 10 to 1
b) Prints the numbers from 10 to 0
c) Prints the numbers from 1 to 10
d) Prints the numbers from 0 to 10

148. What will be the output of the following code?

```
int cnt=0;
void my_recursive_function(int n)
{
    if(n == 0)
    return;
    cnt++;
    my_recursive_function(n/10);
}
int main()
{
    my_recursive_function(123456789);
    printf("%d",cnt);
    return 0;
}
```

a) 123456789
b) 10
c) 0
d) 9

149. What will be the output of the following code?

```
void my_recursive_function(int n)
{
    if(n == 0)
    {
        printf("False");
                return;
    }
    if(n == 1)
    {
        printf("True");
        return;
    }
    if(n%2==0)
    my_recursive_function(n/2);
    else
    {
        printf("False");
        return;
    }

}
int main()
{
    my_recursive_function(100);
    return 0;
}
```

a) True

b) False

150. What is the output of the following code?

```
int cnt = 0;
void my_recursive_function(char *s, int i)
{
    if(s[i] == '\0')
      return;
    if(s[i] == 'a' || s[i] == 'e' || s[i] == 'i' || s[i] == 'o' || s[i] == 'u')
    cnt++;
    my_recursive_function(s,i+1);
}
int main()
{
    my_recursive_function("thisisrecursion",0);
    printf("%d",cnt);
    return 0;
}
```

a) 6

b) 9

c) 5

d) 10

151. What is the output of the following code?

```c
void my_recursive_function(int *arr, int val, int idx, int len)
{
   if(idx == len)
   {
       printf("-1");
       return ;
   }
   if(arr[idx] == val)
   {
       printf("%d",idx);
       return;
   }
   my_recursive_function(arr,val,idx+1,len);
}
int main()
{
    int array[10] = {7, 6, 4, 3, 2, 1, 9, 5, 0, 8};
    int value = 2;
    int len = 10;
    my_recursive_function(array, value, 0, len);
    return 0;
}
```

a) 3
b) 4
c) 5
d) 6

152. Where is linear searching used?

a) When the list has only a few elements

b) When performing a single search in an unordered list
c) Used all the time
d) When the list has only a few elements and When performing a single search in an unordered list

153. What is the best case for linear search?

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

154. What is the worst case for linear search?

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

155. What is the best case and worst case complexity of ordered linear search?

a) O(nlogn), O(logn)
b) O(logn), O(nlogn)
c) O(n), O(1)
d) O(1), O(n)

156. Which of the following is a disadvantage of linear search?

a) Requires more space

b) Greater time complexities compared to other searching algorithms

c) Not easy to understand

d) Not easy to implement

157. Select the code snippet which performs unordered linear search iteratively?

a)

```
int unorderedLinearSearch(int arr[], int size, int data)
{
    int index;
    for(int i = 0; i < size; i++)
    {
        if(arr[i] == data)
        {
            index = i;
            break;
        }
    }
    return index;
}
```

b)

```
int unorderedLinearSearch(int arr[], int size, int data)
{
    int index;
    for(int i = 0; i < size; i++)
    {
        if(arr[i] == data)
        {
            break;
        }
    }
    return index;
}
```

c)

```
int unorderedLinearSearch(int arr[], int size, int data)
{
    int index;
    for(int i = 0; i <= size; i++)
    {
        if(arr[i] == data)
        {
            index = i;
            break;
        }
    }
    return index;
}
```

d)

```
int unorderedLinearSearch(int arr[], int size, int data)
{
    int index;
    for(int i = 0; i < size-1; i++)
    {
        if(arr[i] == data)
        {
```

```
      index = i;
      break;
    }
  }
  return index;
}
```

158. Select the code snippet which performs ordered linear search iteratively?

a)

```
public int linearSearch(int arr[],int key,int size)
{
    int index = -1;
            int i = 0;
    while(size > 0)
    {
        if(data[i] == key)
        {
                      index = i;
        }
        if(data[i] > key))
        {
                     index = i;
          break;
        }
        i++;
    }
    return index;
}
```

b)

```
public int linearSearch(int arr[],int key,int size)
{
       int index = -1;
            int i = 0;
       while(size > 0)
       {
             if(data[i] == key)
             {
                     index = i;
             }
             if(data[i] > key))
             {
                  break;
             }
             i++;
       }
       return index;
}
```

c)

```
public int linearSearch(int arr[],int key,int size)
{
    int index = -1;
            int i = 0;
    while(size > 0)
    {
        if(data[i] == key)
        {
                   break;
        }
        if(data[i] > key))
        {
          index = i;
        }
        i++;
     }
     return index;
}
```

d)

```
public int linearSearch(int arr[],int key,int size)
{
    int index = -1;
            int i = 0;
    while(size > 0)
    {
        if(data[i] == key)
        {
                      break;
        }
        if(data[i] > key))
        {
          break;
          index=i;
        }
        i++;
     }
     return index;
}
```

159. Choose the code snippet which uses recursion for linear search.

a)

```
public void linSearch(int[] arr, int first, int last, int key)
{
           if(first == last)
     {
                      System.out.print("-1");
           }
           else
     {
                      if(arr[first] == key)
          {
                                 System.out.print(first);
                      }
                      else
          {
                                 linSearch(arr, first+1, last, key);
                      }
           }
}
```

b)

```
      public void linSearch(int[] arr, int first, int last, int key)
      {
                   if(first == last)
                {
                            System.out.print("-1");
                   }
                   else
                {
                            if(arr[first] == key)
```

```
                        {
                                       System.out.print(first);
                             }
                             else
                        {
                                       linSearch(arr, first+1, last-1, key);
                             }
                   }
      }
```

c)

```
public void linSearch(int[] arr, int first, int last, int key)
{
          if(first == last)
    {
                    System.out.print("-1");
          }
          else
    {
                    if(arr[first] == key)
        {
                              System.out.print(last);
                    }
                    else
        {
                              linSearch(arr, first+1, last, key);
                    }
          }
}
```

d)

```
public void linSearch(int[] arr, int first, int last, int key)
{
          if(first == last)
    {
                    System.out.print("-1");
          }
          else
    {
                    if(arr[first] == key)
        {
                              System.out.print(first);
                    }
                    else
        {
                              linSearch(arr, first+1, last+1, key);
                    }
          }
}
```

160. What does the following piece of code do?

```
for (int i = 0; i < arr.length-1; i++)
{
    for (int j = i+1; j < arr.length; j++)
    {
        if( (arr[i].equals(arr[j])) && (i != j) )
        {
            System.out.println(arr[i]);
        }
    }
}
```

a) Print the duplicate elements in the array
b) Print the element with maximum frequency
c) Print the unique elements in the array
d) Prints the element with minimum frequnecy

161. Select the code snippet which prints the element with maximum frequency.

a)

```
public int findPopular(int[] a)
{
          if (a == null || a.length == 0)
                   return 0;
          Arrays.sort(a);
          int previous = a[0];
          int popular = a[0];
          int count = 1;
          int maxCount = 1;
          for (int i = 1; i < a.length; i++)
        {
                   if (a[i] == previous)
                   count++;
                   else
                {
                             if (count > maxCount)
                        {
                                      popular = a[i-1];
                                      maxCount = count;
                             }
                   previous = a[i];
                   count = 1;
                   }
          }
          return count > maxCount ? a[a.length-1] : popular;
}
```

b)

```
public int findPopular(int[] a)
{
          if (a == null || a.length == 0)
                   return 0;
          Arrays.sort(a);
          int previous = a[0];
          int popular = a[0];
          int count = 1;
          int maxCount = 1;
          for (int i = 1; i < a.length; i++)
    {
                   if (a[i] == previous)
                             count++;
                   else
         {
                             if (count > maxCount)
             {
                                      popular = a[i];
                                      maxCount = count;
                             }
                             previous = a[i];
                             count = 1;
                   }
          }
          return count > maxCount ? a[a.length-1] : popular;
}
```

c)

```
public int findPopular(int[] a)
{
    if (a == null || a.length == 0)
        return 0;
    Arrays.sort(a);
    int previous = a[0];
    int popular = a[0];
    int count = 1;
    int maxCount = 1;
    for (int i = 1; i < a.length; i++)
  {
        if (a[i+1] == previous)
            count++;
        else
    {
            if (count > maxCount)
     {
                popular = a[i-1];
                maxCount = count;
            }
            previous = a[i];
            count = 1;
        }
    }
    return count > maxCount ? a[a.length-1] : popular;
}
```

d)

```
public int findPopular(int[] a)
{
    if (a == null || a.length == 0)
        return 0;
    Arrays.sort(a);
    int previous = a[0];
    int popular = a[0];
    int count = 1;
    int maxCount = 1;
    for (int i = 1; i < a.length; i++)
  {
        if (a[i+2] == previous)
            count++;
        else
    {
            if (count > maxCount)
     {
                popular = a[i-1];
                maxCount = count;
            }
            previous = a[i];
            count = 1;
        }
    }
    return count > maxCount ? a[a.length-1] : popular;
}
```

162. Is there any difference in the speed of execution between linear serach(recursive) vs linear search(lterative)?

a) Both execute at same speed
b) Linear search(recursive) is faster
c) Linear search(Iterative) is faster
d) Cant be said

163. Is the space consumed by the linear search(recursive) and linear search(iterative) same?

a) No, recursive algorithm consumes more space
b) No, recursive algorithm consumes less space
c) Yes
d) Nothing can be said

164. What is the worst case runtime of linear search(recursive) algorithm?

a) O(n)
b) O(logn)
c) O(n2)
d) O(nx)

165. Linear search(recursive) algorithm used in _____________

a) When the size of the dataset is low
b) When the size of the dataset is large
c) When the dataset is unordered
d) Never used

166. The array is as follows: 1,2,3,6,8,10. At what time the element 6 is found? (By using linear search(recursive) algorithm)

a) 4th call
b) 3rd call
c) 6th call
d) 5th call

167. The array is as follows: 1,2,3,6,8,10. Given that the number 17 is to be searched. At which call it tells that there's no such element? (By using linear search(recursive) algorithm)

a) 7th call
b) 9th call
c) 17th call
d) The function calls itself infinite number of times

168. What is the best case runtime of linear search(recursive) algorithm on an ordered set of elements?

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

169. Can linear search recursive algorithm and binary search recursive algorithm be performed on an unordered list?

a) Binary search can’t be used
b) Linear search can’t be used
c) Both cannot be used
d) Both can be used

170. 10. What is the recurrence relation for the linear search recursive algorithm?

a) T(n-2)+c
b) 2T(n-1)+c
c) T(n-1)+c
d) T(n+1)+c

171. Which of the following code snippet performs linear search recursively?

a)

```
        for(i=0;i<n;i++)
         {
                   if(a[i]==key)
                   printf("element found");
         }
```

b)

```
        LinearSearch(int[] a, n,key)
         {
                   if(n<1)
                   return False
                   if(a[n]==key)
                   return True
                   else
                   LinearSearch(a,n-1,key)
         }
```

c)

```
        LinearSearch(int[] a, n,key)
         {
                   if(n<1)
                   return True
                   if(a[n]==key)
                   return False
                   else
                   LinearSearch(a,n-1,key)
         }
```

d)

```
        LinearSearch(int[] a, n,key)
         {
                   if(n<1)
                   return False
                   if(a[n]==key)
                   return True
                   else
                   LinearSearch(a,n+1,key)
         }
```

172. What is the advantage of recursive approach than an iterative approach?

a) Consumes less memory
b) Less code and easy to implement
c) Consumes more memory
d) More code has to be written

173. Given an input arr = {2,5,7,99,899}; key = 899; What is the level of recursion?

a) 5
b) 2
c) 3
d) 4

174. Given an array arr = {45,77,89,90,94,99,100} and key = 99; what are the mid values(corresponding array elements) in the first and second levels of recursion?

a) 90 and 99
b) 90 and 94
c) 89 and 99
d) 89 and 94

175. What is the worst case complexity of binary search using recursion?

a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n2)

176. What is the average case time complexity of binary search using recursion?

a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n2)

177. Which of the following is not an application of binary search?

a) To find the lower/upper bound in an ordered sequence
b) Union of intervals
c) Debugging
d) To search in unordered list

178. Binary Search can be categorized into which of the following?

a) Brute Force technique
b) Divide and conquer
c) Greedy algorithm
d) Dynamic programming

179. Given an array arr = {5,6,77,88,99} and key = 88; How many iterations are done until the element is found?

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

180. Given an array arr = {45,77,89,90,94,99,100} and key = 100; What are the mid values(corresponding array elements) generated in the first and second iterations?

a) 90 and 99
b) 90 and 100
c) 89 and 94
d) 94 and 99

181. What is the time complexity of binary search with iteration?

a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n2)

182. Choose the appropriate code that does binary search using recursion

a)

```
public static int recursive(int arr[], int low, int high, int key)
{
		int mid = low + (high - low)/2;
		if(arr[mid] == key)
		{
			return mid;
		}
```

```
        else if(arr[mid] < key)
        {
                return recursive(arr,mid+1,high,key);
        }
        else
        {
                return recursive(arr,low,mid-1,key);
        }
}
```

b)

```
public static int recursive(int arr[], int low, int high, int key)
{
        int mid = low + (high + low)/2;
        if(arr[mid] == key)
        {
                return mid;
        }
        else if(arr[mid] < key)
        {
                return recursive(arr,mid-1,high,key);
        }
        else
        {
                return recursive(arr,low,mid+1,key);
        }
}
```

c)

```
public static int recursive(int arr[], int low, int high, int key)
{
        int mid = low + (high - low)/2;
        if(arr[mid] == key)
        {
                return mid;
        }
        else if(arr[mid] < key)
        {
                return recursive(arr,mid,high,key);
        }
        else
        {
                return recursive(arr,low,mid-1,key);
        }
}
```

d)

```
public static int recursive(int arr[], int low, int high, int key)
{
        int mid = low + ((high - low)/2)+1;
        if(arr[mid] == key)
        {
                return mid;
        }
        else if(arr[mid] < key)
        {
                return recursive(arr,mid,high,key);
        }
        else
        {
                return recursive(arr,low,mid-1,key);
        }
}
```

183. Choose among the following code for an iterative binary search.

a)

```
public static int iterative(int arr[], int key)
{
        int low = 0;
        int mid = 0;
        int high = arr.length-1;
        while(low <= high)
        {
                mid = low + (high + low)/2;
                if(arr[mid] == key)
                {
                        return mid;
                }
                else if(arr[mid] < key)
                {
                        low = mid - 1;
                }
                else
                {
                        high = mid + 1;
                }
        }
        return -1;
}
```

b)

```
public static int iterative(int arr[], int key)
{
        int low = 0;
        int mid = 0;
        int high = arr.length-1;
        while(low <= high)
        {
                mid = low + (high - low)/2;
                if(arr[mid] == key)
                {
                        return mid;
                }
                else if(arr[mid] < key)
                {
                        low = mid + 1;
                }
                else
        {
            high = mid - 1;
        }
    }
    return -1;
}
```

c)

```
public static int iterative(int arr[], int key)
{
    int low = 0;
    int mid = 0;
    int high = arr.length-1;
    while(low <= high)
    {
        mid = low + (high + low)/2;
        if(arr[mid] == key)
        {
            return mid;
        }
        else if(arr[mid] < key)
        {
            low = mid + 1;
        }
        else
        {
            high = mid - 1;
        }
    }
    return -1;
}
```

d)

```
public static int iterative(int arr[], int key)
{
    int low = 0;
    int mid = 0;
    int high = arr.length-1;
    while(low <= high)
    {
        mid = low + (high - low)/2;
        if(arr[mid] == key)
        {
            return mid;
        }
        else if(arr[mid] < key)
        {
            low = mid - 1;
        }
        else
        {
            high = mid + 1;
        }
    }
```

```
            return -1;
}
```

184. Depth First Search is equivalent to which of the traversal in the Binary Trees?

a) Pre-order Traversal

b) Post-order Traversal

c) Level-order Traversal

d) In-order Traversal

185. Time Complexity of DFS is? (V – number of vertices, E – number of edges)

a) O(V + E)

b) O(V)

c) O(E)

d) O(V*E)

186. The Data structure used in standard implementation of Breadth First Search is?

a) Stack

b) Queue

c) Linked List

d) Tree

187. The Depth First Search traversal of a graph will result into?

a) Linked List
b) Tree
c) Graph with back edges
d) Array

188. A person wants to visit some places. He starts from a vertex and then wants to visit every vertex till it finishes from one vertex, backtracks and then explore other vertex from same vertex. What algorithm he should use?

a) Depth First Search
b) Breadth First Search
c) Trim's algorithm
d) Kruskal's Algorithm

189. Which of the following is not an application of Depth First Search?

a) For generating topological sort of a graph
b) For generating Strongly Connected Components of a directed graph
c) Detecting cycles in the graph
d) Peer to Peer Networks

190. When the Depth First Search of a graph is unique?

a) When the graph is a Binary Tree
b) When the graph is a Linked List

c) When the graph is a n-ary Tree

d) When the graph is a ternary Tree

191. Regarding implementation of Depth First Search using stacks, what is the maximum distance between two nodes present in the stack? (considering each edge length 1)

a) Can be anything

b) 0

c) At most 1

d) Insufficient Information

192. In Depth First Search, how many times a node is visited?

a) Once

b) Twice

c) Equivalent to number of indegree of the node

d) Thrice

193. Which of the following data structure is used to implement DFS?

a) linked list

b) tree

c) stack

d) queue

194. Which of the following traversal in a binary tree is similar to depth first traversal?

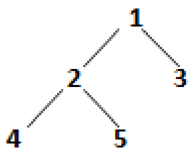
a) level order

b) post order

c) pre order

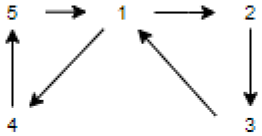
d) in order

195. What will be the result of depth first traversal in the following tree?



a) 4 2 5 1 3
b) 1 2 4 5 3
c) 4 5 2 3 1
d) 1 2 3 4 5

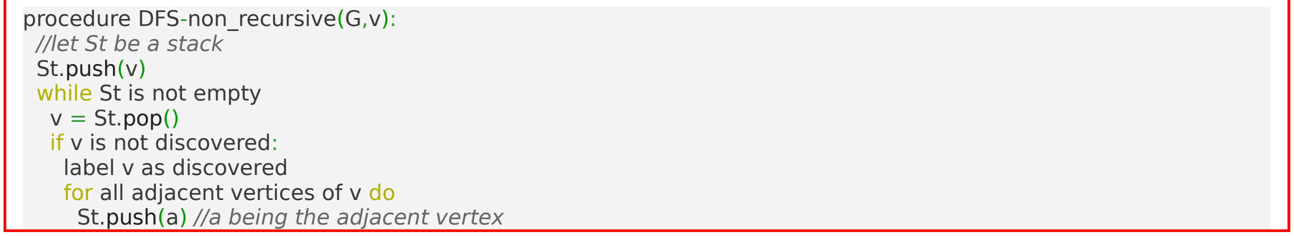
196. Which of the following is a possible result of depth first traversal of the given graph(consider 1 to be source element)?



a) 1 2 3 4 5
b) 1 2 3 1 4 5
c) 1 4 5 3 2
d) 1 4 5 1 2 3

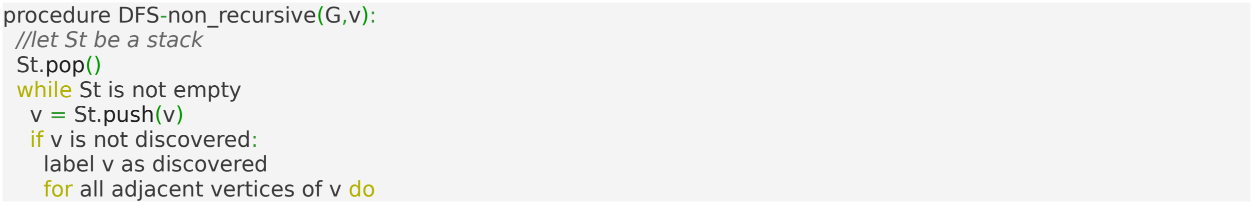
197. Which of the following represent the correct pseudo code for non recursive DFS algorithm?

a)



```
procedure DFS-non_recursive(G,v):
  //let St be a stack
  St.push(v)
  while St is not empty
    v = St.pop()
    if v is not discovered:
      label v as discovered
      for all adjacent vertices of v do
        St.push(a) //a being the adjacent vertex
```

b)



```
procedure DFS-non_recursive(G,v):
  //let St be a stack
  St.pop()
  while St is not empty
    v = St.push(v)
    if v is not discovered:
      label v as discovered
      for all adjacent vertices of v do
    St.push(a) //a being the adjacent vertex
```

c)

```
procedure DFS-non_recursive(G,v):
 //let St be a stack
 St.push(v)
 while St is not empty
  v = St.pop()
  if v is not discovered:
   label v as discovered
   for all adjacent vertices of v do
    St.push(v)
```

d)

```
procedure DFS-non_recursive(G,v):
 //let St be a stack
 St.pop(v)
 while St is not empty
  v = St.pop()
  if v is not discovered:
   label v as discovered
   for all adjacent vertices of v do
    St.push(a) //a being the adjacent vertex
```

198. What will be the time complexity of the iterative depth first traversal code(V=no. of vertices E=no.of edges)?

a) O(V+E)
b) O(V)
c) O(E)
d) O(V*E)

199. Which of the following functions correctly represent iterative DFS?

a)

```
void DFS(int s)
{
    vector<bool> discovered(V, true);
    stack<int> st;
    st.push(s);
    while (!st.empty())
    {
        s = st.top();
        st.pop();
        if (!discovered[s])
        {
            cout << s << " ";
            discovered[s] = true;
        }
        for (auto i = adjacent[s].begin(); i != adjacent[s].end(); ++i)
            if (!discovered[*i])
                st.push(*i);
    }
}
```

b)

```
void DFS(int s)
{
    vector<bool> discovered(V, false);
    stack<int> st;
    st.push(s);
    while (!st.empty())
    {
        s = st.top();
        st.pop();
        if (!discovered[s])
        {
            cout << s << " ";
            discovered[s] = true;
        }
        for (auto i = adjacent[s].begin(); i != adjacent[s].end(); ++i)
            if (!discovered[*i])
                st.push(*i);
    }
}
```

c)

```
void DFS(int s)
{
    vector<bool> discovered(V, false);
    stack<int> st;
    st.push(s);
    while (!st.empty())
    {
        st.pop();
        s = st.top();
        if (!discovered[s])
        {
            cout << s << " ";
            discovered[s] = true;
        }
        for (auto i = adjacent[s].begin(); i != adjacent[s].end(); ++i)
            if (!discovered[*i])
                st.push(*i);
    }
}
```

d)

```
void DFS(int s)
{
    vector<bool> discovered(V, false);
    stack<int> st;
    st.push(s);
    while (!st.empty())
    {
        s = st.top();
        st.pop();
        if (!discovered[s])
        {
            cout << s << " ";
            discovered[s] = false;
        }
        for (auto i = adjacent[s].begin(); i != adjacent[s].end(); ++i)
            if (discovered[*i])
                st.push(*i);
    }
}
```

200. What is the space complexity of standard DFS(V: no. of vertices E: no. of edges)?

a) O(V+E)

b) O(V)

c) O(E)

d) O(V*E)

201. Which of the following data structure is used to implement BFS?

a) linked list

b) tree

c) stack

d) queue

202. Choose the incorrect statement about DFS and BFS from the following?

a) BFS is equivalent to level order traversal in trees

b) DFS is equivalent to post order traversal in trees

c) DFS and BFS code has the same time complexity

d) BFS is implemented using queue

203. Breadth First Search is equivalent to which of the traversal in the Binary Trees?

a) Pre-order Traversal

b) Post-order Traversal

c) Level-order Traversal

d) In-order Traversal

204. Time Complexity of Breadth First Search is? (V – number of vertices, E – number of edges)
a) O(V + E)
b) O(V)
c) O(E)
d) O(V*E)

205. The Data structure used in standard implementation of Breadth First Search is?
a) Stack
b) Queue
c) Linked List
d) Tree

206. The Breadth First Search traversal of a graph will result into?
a) Linked List
b) Tree
c) Graph with back edges
d) Arrays

207. A person wants to visit some places. He starts from a vertex and then wants to visit every place connected to this vertex and so on. What algorithm he should use?
a) Depth First Search
b) Breadth First Search
c) Trim’s algorithm
d) Kruskal’s algorithm

208. Which of the following is not an application of Breadth First Search?
a) Finding shortest path between two nodes
b) Finding bipartiteness of a graph
c) GPS navigation system
d) Path Finding

209. When the Breadth First Search of a graph is unique?

a) When the graph is a Binary Tree
b) When the graph is a Linked List
c) When the graph is a n-ary Tree
d) When the graph is a Ternary Tree

210. Regarding implementation of Breadth First Search using queues, what is the maximum distance between two nodes present in the queue? (considering each edge length 1)

a) Can be anything
b) 0
c) At most 1
d) Insufficient Information

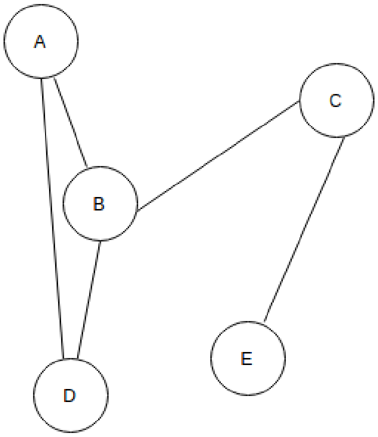
211. In BFS, how many times a node is visited?

a) Once
b) Twice
c) Equivalent to number of indegree of the node
d) Thrice

212. Which of the following statements for a simple graph is correct?

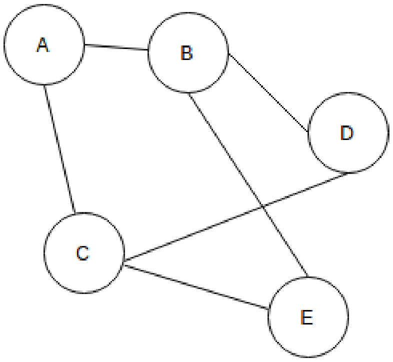
a) Every path is a trail
b) Every trail is a path
c) Every trail is a path as well as every path is a trail
d) Path and trail have no relation

213. In the given graph identify the cut vertices.



a) B and E
b) C and D
c) A and E
d) C and B

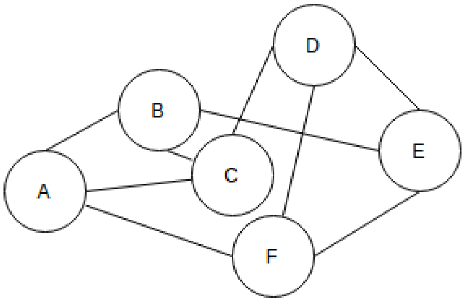
214. For the given graph(G), which of the following statements is true?



a) G is a complete graph
b) G is not a connected graph
c) The vertex connectivity of the graph is 2
d) The edge connectivity of the graph is 1

215. What is the number of edges present in a complete graph having n vertices?
a) (n*(n+1))/2
b) (n*(n-1))/2
c) n
d) Information given is insufficient

216. The given Graph is regular.



a) True
b) False

217. In a simple graph, the number of edges is equal to twice the sum of the degrees of the vertices.

a) True

b) False

218. A connected planar graph having 6 vertices, 7 edges contains _______________ regions.

a) 15

b) 3

c) 1

d) 11

219. If a simple graph G, contains n vertices and m edges, the number of edges in the Graph G'(Complement of G) is ____________

a) (n*n-n-2*m)/2

b) (n*n+n+2*m)/2

c) (n*n-n-2*m)/2

d) (n*n-n+2*m)/2

220. Which of the following properties does a simple graph not hold?

a) Must be connected

b) Must be unweighted

c) Must have no loops or multiple edges

d) Must have no multiple edges

221. What is the maximum number of edges in a bipartite graph having 10 vertices?

a) 24

b) 21

c) 25

d) 16

222. Which of the following is true?

a) A graph may contain no edges and many vertices

b) A graph may contain many edges and no vertices

c) A graph may contain no edges and no vertices

d) A graph may contain no vertices and many edges

223. For a given graph G having v vertices and e edges which is connected and has no cycles, which of the following statements is true?

a) v=e
b) v = e+1
c) v + 1 = e
d) v = e-1

224. For which of the following combinations of the degrees of vertices would the connected graph be eulerian?

a) 1,2,3
b) 2,3,4
c) 2,4,5
d) 1,3,5

225. A graph with all vertices having equal degree is known as a ___________

a) Multi Graph
b) Regular Graph
c) Simple Graph
d) Complete Graph

226. Which of the following ways can be used to represent a graph?

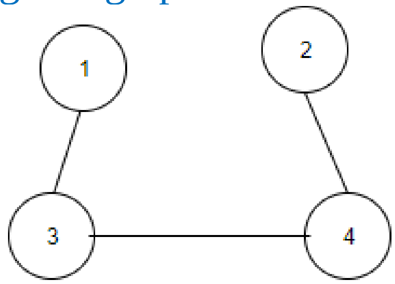
a) Adjacency List and Adjacency Matrix
b) Incidence Matrix
c) Adjacency List, Adjacency Matrix as well as Incidence Matrix
d) No way to represent

227. The number of elements in the adjacency matrix of a graph having 7 vertices is _______

a) 7
b) 14
c) 36

d) 49

228. What would be the number of zeros in the adjacency matrix of the given graph?



a) 10

b) 6

c) 16

d) 0

229. Adjacency matrix of all graphs are symmetric.

a) False

b) True

230. The time complexity to calculate the number of edges in a graph whose information in stored in form of an adjacency matrix is _____

a) O(V)

b) O(E2)

c) O(E)

d) O(V^2)

231. For the adjacency matrix of a directed graph the row sum is the _______ degree and the column sum is the _______ degree.

a) in, out

b) out, in

c) in, total

d) total, out

232. What is the maximum number of possible non zero values in an adjacency matrix of a simple graph with n vertices?

a) (n*(n-1))/2

b) (n*(n+1))/2

c) n*(n-1)

d) n*(n+1)

233. On which of the following statements does the time complexity of checking if an edge exists between two particular vertices is not, depends?

a) Depends on the number of edges

b) Depends on the number of vertices

c) Is independent of both the number of edges and vertices

d) It depends on both the number of edges and vertices

234. In the given connected graph G, what is the value of rad(G) and diam(G)?

a) 2, 3

b) 3, 2

c) 2, 2

d) 3, 3

235. Which of these adjacency matrices represents a simple graph?

a) [[1, 0, 0], [0, 1, 0], [0, 1, 1]]

b) [[1, 1, 1], [1, 1, 1], [1, 1, 1]]

c) [[0, 0, 1], [0, 0, 0], [0, 0, 1]]

d) [[0, 0, 1], [1, 0, 1], [1, 0, 0]]

236. Given an adjacency matrix A = [[0, 1, 1], [1, 0, 1], [1, 1, 0]], The total no. of ways in which every vertex can walk to itself using 2 edges is ____

a) 2
b) 4
c) 6
d) 8

237. If A[x+3][y+5] represents an adjacency matrix, which of these could be the value of x and y.

a) x=5, y=3
b) x=3, y=5
c) x=3, y=3
d) x=5, y=5

238. Two directed graphs(G and H) are isomorphic if and only if A=PBP-1, where P and A are adjacency matrices of G and H respectively.

a) True
b) False

239. Given the following program, what will be the 3rd number that'd get printed in the output sequence for the given input?

```
#include <bits/stdc++.h>
using namespace std;
int cur=0;
int G[10][10];
bool visited[10];
deque <int> q;

void fun(int n);

int main()
{
        int num=0;
        int n;
        cin>>n;

        for(int i=0;i<n;i++)
```

```
         for(int j=0;j<n;j++)
         cin>>G[i][j];

         for(int i=0;i<n;i++)
    visited[i]=false;

    fun(n);
         return 0;
}

void fun(int n)
{
         cout<<cur<<" ";
         visited[cur]=true;
         q.push_back(cur);

         do
    {
                  for(int j=0;j<n;j++)
        {
                    if(G[cur][j]==1 && !visited[j])
          {
                      q.push_back(j);
                      cout<<j<<" ";
                      visited[j]=true;
               }

         }

                  q.pop_front();
                  if(!q.empty())
                  cur=q.front();
          }while(!q.empty());
}
```

Input Sequence:-

```
9
0 1 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0
0 0 0 1 1 1 0 0 1
0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 1 0
0 0 1 0 0 0 1 0 0
0 0 0 0 0 1 0 1 1
0 0 0 0 1 0 1 0 0
1 0 1 0 0 0 1 0 0
```

a) 2
b) 6
c) 8
d) 4

240. For which type of graph, the given program won’t run infinitely? The Input would be in the form of an adjacency Matrix and n is its dimension (1<n<10).

```
#include <bits/stdc++.h>
using namespace std;
int G[10][10];
void fun(int n);

int main()
{
        int num=0;
        int n;
        cin>>n;
        for(int i=0;i<n;i++)
                for(int j=0;j<n;j++)
                cin>>G[i][j];
        fun(n);
        return 0;
}

void fun(int n)
{
        for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)
        if(G[i][j]==1)
        j--;
}
```

a) All Fully Connected Graphs
b) All Empty Graphs
c) All Bipartite Graphs
d) All simple graphs

241. Given the following adjacency matrix of a graph(G) determine the number of components in the G.

```
[0 1 1 0 0 0],
[1 0 1 0 0 0],
[1 1 0 0 0 0],
[0 0 0 0 1 0],
[0 0 0 1 0 0],
[0 0 0 0 0 0].
```

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

242. Incidence matrix and Adjacency matrix of a graph will always have same dimensions?

a) True

b) False

243. The column sum in an incidence matrix for a simple graph is ___________

a) depends on number of edges

b) always greater than 2

c) equal to 2

d) equal to the number of edges

244. What are the dimensions of an incidence matrix?

a) Number of edges*number of edges

b) Number of edges*number of vertices

c) Number of vertices*number of vertices

d) Number of edges * (1⁄2 * number of vertices)

245. The column sum in an incidence matrix for a directed graph having no self loop is ___________

a) 0

b) 1

c) 2

d) equal to the number of edges

246. Time complexity to check if an edge exists between two vertices would be ____________

a) O(V*V)

b) O(V+E)

c) O(1)

d) O(E)

247. The graphs G1 and G2 with their incidences matrices given are Isomorphic.

	e1	e2	e3	e4	e5	e6
v1	1	0	0	0	0	0
v2	1	1	0	0	0	1
v3	0	1	1	0	1	0
v4	0	0	1	1	0	0
v5	0	0	0	1	1	1

	e1	e2	e3	e4	e5	e6
v1	0	0	1	0	0	0
v2	1	0	1	0	1	0
v3	1	1	0	1	0	0
v4	0	1	0	0	0	1
v5	0	0	0	1	1	1

a) True

b) False

248. If a connected Graph (G) contains n vertices what would be the rank of its incidence matrix?

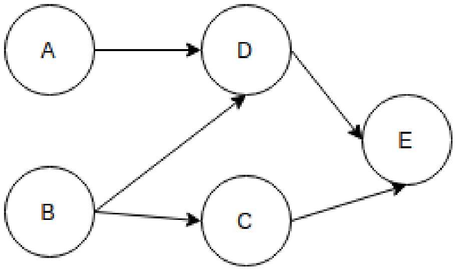
a) n-1

b) values greater than n are possible

c) values less than n-1 are possible

d) insufficient Information is given

249. In the following DAG find out the number of required Stacks in order to represent it in a Graph Structured Stack.



a) 1

b) 2

c) 3

d) 4

250. A Graph Structured Stack is a _____________

a) Undirected Graph
b) Directed Graph
c) Directed Acyclic Graph
d) Regular Graph

251. If a Graph Structured Stack contains {1,2,3,4} {1,5,3,4} {1,6,7,4} and {8,9,7,4}, what would be the source and sink vertices of the DAC?

a) Source – 1, 8 Sink – 7,4
b) Source – 1 Sink – 8,4
c) Source – 1, 8 Sink – 4
d) Source – 4, Sink – 1,8

252. Graph Structured Stack finds its application in _____________

a) Bogo Sort
b) Tomita's Algorithm
c) Todd–Coxeter algorithm
d) Heap Sort

253. 12. If in a DAG N sink vertices and M source vertices exists, then the number of possible stacks in the Graph Structured Stack representation would come out to be N*M.

a) True
b) False

254. Space complexity for an adjacency list of an undirected graph having large values of V (vertices) and E (edges) is ___________

a) O(E)
b) O(V*V)
c) O(E+V)
d) O(V)

255. For some sparse graph an adjacency list is more space efficient against an adjacency matrix.

a) True
b) False

256. Time complexity to find if there is an edge between 2 particular vertices is _________

a) O(V)
b) O(E)
c) O(1)
d) O(V+E)

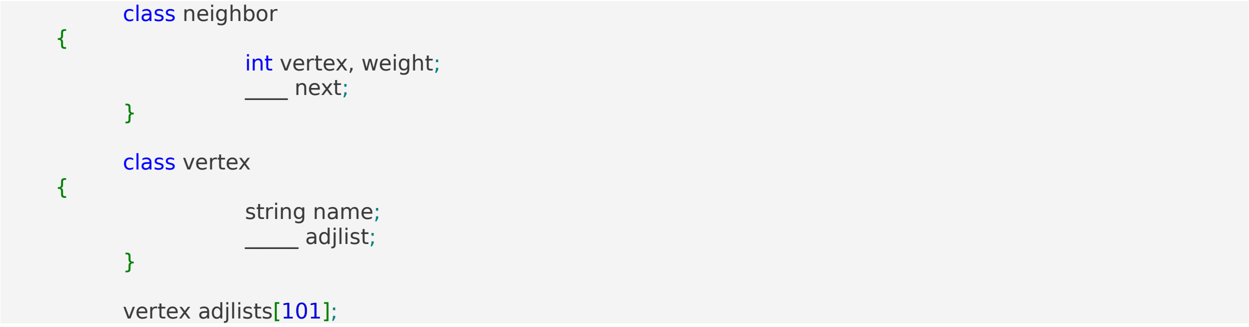
257. For the given conditions, which of the following is in the correct order of increasing space requirement?

i) Undirected, no weight
ii) Directed, no weight
iii) Directed, weighted
iv) Undirected, weighted
a) ii iii i iv
b) i iii ii iv
c) iv iii i ii
d) i ii iii iv

258. Space complexity for an adjacency list of an undirected graph having large values of V (vertices) and E (edges) is ___________

a) O(V)
b) O(E*E)
c) O(E)
d) O(E+V)

259. Complete the given snippet of code for the adjacency list representation of a weighted directed graph.



```
        class neighbor
  {
                        int vertex, weight;
                        ____ next;
        }

        class vertex
  {
                        string name;
                        _____ adjlist;
        }

        vertex adjlists[101];
```

a) vertex, vertex
b) neighbor, vertex
c) neighbor, neighbor
d) vertex, neighbor

260. In which case adjacency list is preferred in front of an adjacency matrix?

a) Dense graph
b) Sparse graph
c) Adjacency list is always preferred
d) Complete graph

261. To create an adjacency list C++'s map container can be used.

a) True
b) False

262. What would be the time complexity of the following function which adds an edge between two vertices i and j, with some weight 'weigh' to the graph having V vertices?

```
vector<int> adjacent[15] ;
vector<int> weight[15];
void addEdge(int i,int j,int weigh)
{
            adjacent[a].push_back(i);
            adjacent[b].push_back(j);
            weight[a].push_back(weigh);
            weight[b].push_back(weigh);
}
```

a) O(1)
b) O(V)
c) O(V*V)
d) O(log V)

263. What would be the time complexity of the BFS traversal of a graph with n vertices and n1.25 edges?

a) O(n)
b) O($n^{1.25}$)
c) O($n^{2.25}$)
d) O(n*n)

264. The number of possible undirected graphs which may have self loops but no multiple edges and have n vertices is ________

a) $2^{((n*(n-1))/2)}$
b) $2^{((n*(n+1))/2)}$
c) $2^{((n-1)*(n-1))/2)}$
d) $2^{((n*n)/2)}$

265. Given a plane graph, G having 2 connected component, having 6 vertices, 7 edges and 4 regions. What will be the number of connected components?

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

266. Number of vertices with odd degrees in a graph having a eulerian walk is ________

a) 0
b) Can’t be predicted

c) 2
d) either 0 or 2

267. How many of the following statements are correct?
i) All cyclic graphs are complete graphs.
ii) All complete graphs are cyclic graphs.
iii) All paths are bipartite.
iv) All cyclic graphs are bipartite.
v) There are cyclic graphs which are complete.
a) 1
b) 2
c) 3
d) 4

268. All paths and cyclic graphs are bipartite graphs.
a) True
b) False

269. What is the number of vertices of degree 2 in a path graph having n vertices,here n>2.
a) n-2
b) n
c) 2
d) 0

270. All trees with n vertices consists of n-1 edges.
a) True
b) False

271. Which of the following graphs are isomorphic to each other?

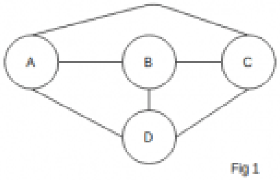


Fig 1

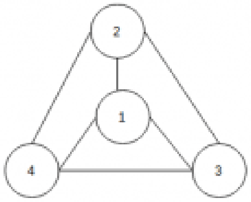


Fig 2

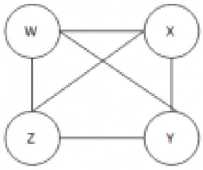
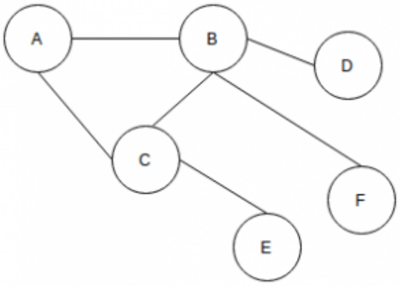


Fig 3

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

272. In the given graph which edge should be removed to make it a Bipartite Graph?



a) A-C
b) B-E
c) C-D
d) D-E

273. What would the time complexity to check if an undirected graph with V vertices and E edges is Bipartite or not given its adjacency matrix?
a) O(E*E)
b) O(V*V)
c) O(E)
d) O(V)

274. Dijkstra's Algorithm will work for both negative and positive weights?

a) True

b) False

275. A graph having an edge from each vertex to every other vertex is called a ___________

a) Tightly Connected

b) Strongly Connected

c) Weakly Connected

d) Loosely Connected

276. What is the number of unlabeled simple directed graph that can be made with 1 or 2 vertices?

a) 2

b) 4

c) 5

d) 9

277. Floyd Warshall Algorithm used to solve the shortest path problem has a time complexity of __________

a) O(V*V)

b) O(V*V*V)

c) O(E*V)

d) O(E*E)

278. All Graphs have unique representation on paper.

a) True

b) False

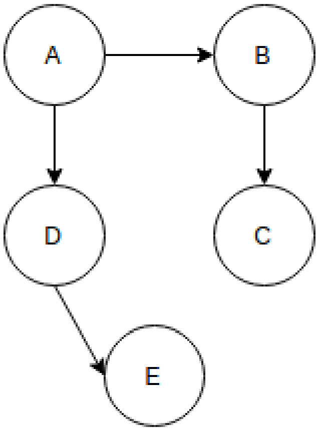
279. Assuming value of every weight to be greater than 10, in which of the following cases the shortest path of a directed weighted graph from 2 vertices u and v will never change?

a) add all values by 10
b) subtract 10 from all the values
c) multiply all values by 10
d) in both the cases of multiplying and adding by 10

280. What is the maximum possible number of edges in a directed graph with no self loops having 8 vertices?

a) 28
b) 64
c) 256
d) 56

281. What would be the DFS traversal of the given Graph?



a) ABCED
b) AEDCB
c) EDCBA
d) ADECB

282. What would be the value of the distance matrix, after the execution of the given code?

```
#include <bits/stdc++.h>
```

```
#define INF 1000000
int graph[V][V] = {   {0,   7,  INF, 4},
                      {INF, 0,   13, INF},
                      {INF, INF, 0,   12},
                      {INF, INF, INF, 0}
                  };

int distance[V][V], i, j, k;

for (i = 0; i < V; i++)
        for (j = 0; j < V; j++)
                  distance[i][j] = graph[i][j];

for (k = 0; k < V; k++)
                  for (i = 0; i < V; i++)
                  for (j = 0; j < V; j++)
                {
                                    if (distance[i][k] + distance[k][j] < distance[i][j])
                                    distance[i][j] = distance[i][k] + distance[k][j];

                           return 0;
                }
```

a)

```
{
    {0,   7,  INF, 4},
    {INF, 0,   13, INF},
    {INF, INF, 0,   12},
    {INF, INF, INF, 0}
};
```

b)

```
{
    {0,   7,  20, 24},
    {INF, 0,   13, 25},
    {INF, INF, 0,   12},
    {INF, INF, INF, 0}
};
```

c)

```
{
    {0,   INF,  20, 24},
    {INF, INF,   13, 25},
    {INF, INF, 0,   12},
    {INF, INF, INF, 0}
    {INF, 0,   13, 25},
    {INF, INF, 0,   12},
    {24, INF, INF, 0}
};
```

d) None of the mentioned

283. What is the maximum number of edges present in a simple directed graph with 7 vertices if there exists no cycles in the graph?

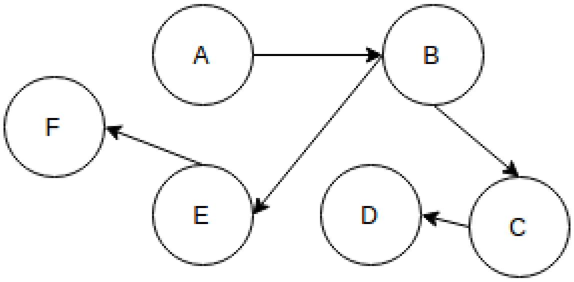
a) 21

b) 7

c) 6
d) 49

284. Every Directed Acyclic Graph has at least one sink vertex.
a) True
b) False

285. Which of the following is not a topological sorting of the given graph?



a) A B C D E F
b) A B F E D C
c) A B E C F D
d) A B C D F E

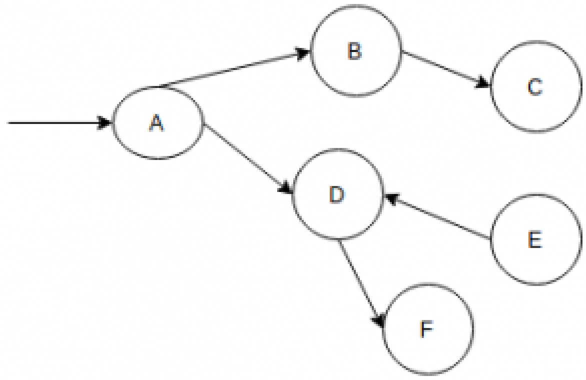
286. With V(greater than 1) vertices, how many edges at most can a Directed Acyclic Graph possess?
a) (V*(V-1))/2
b) (V*(V+1))/2
c) (V+1)C2
d) (V-1)C2

287. The topological sorting of any DAG can be done in ________ time.
a) cubic
b) quadratic
c) linear
d) logarithmic

288. If there are more than 1 topological sorting of a DAG is possible, which of the following is true.

a) Many Hamiltonian paths are possible
b) No Hamiltonian path is possible
c) Exactly 1 Hamiltonian path is possible
d) Given information is insufficient to comment anything

289. What sequence would the BFS traversal of the given graph yield?



a) A F D B C E
b) C B A F E D
c) A B D C E F
d) E F D C B A

290. What would be the output of the following C++ program if the given input is

```
0 0 0 1 1
0 0 0 0 1
0 0 0 1 0
1 0 1 0 0
1 1 0 0 0

#include <bits/stdc++.h>
using namespace std;
bool visited[5];
int G[5][5];
```

```
void fun(int i)
{
        cout<<i<<" ";
        visited[i]=true;
        for(int j=0;j<5;j++)
                if(!visited[j]&&G[i][j]==1)
                        fun(j);
}

int main()
{
        for(int i=0;i<5;i++)
                for(int j=0;j<5;j++)
                        cin>>G[i][j];

        for(int i=0;i<5;i++)
                visited[i]=0;

        fun(0);
                return 0;
}
```

a) 0 2 3 1 4
b) 0 3 2 4 1
c) 0 2 3 4 1
d) 0 3 2 1 4

291. Which of the given statement is true?

a) All the Cyclic Directed Graphs have topological sortings
b) All the Acyclic Directed Graphs have topological sortings
c) All Directed Graphs have topological sortings
d) All the cyclic directed graphs hace non topological sortings

292. For any two different vertices u and v of an Acyclic Directed Graph if v is reachable from u, u is also reachable from v?

a) True
b) False

293. What is the value of the sum of the minimum in-degree and maximum out-degree of an Directed Acyclic Graph?

a) Depends on a Graph
b) Will always be zero
c) Will always be greater than zero
d) May be zero or greater than zero

294. If $f(x) = (x^3 – 1) / (3x + 1)$ then $f(x)$ is?

a) $O(x^2)$
b) $O(x)$

c) $O(x^2 / 3)$
d) $O(1)$

295. If $f(x) = 3x^2 + x^3\log x$, then f(x) is?
a) $O(x^2)$
b) $O(x^3)$
c) $O(x)$
d) $O(1)$

296. The big-O notation for $f(n) = (n\log n + n^2)(n^3 + 2)$ is?
a) $O(n^2)$
b) $O(3^n)$
c) $O(n^4)$
d) $O(n^5)$

297. The big-theta notation for function $f(n) = 2n^3 + n – 1$ is?
a) n
b) n^2
c) n^3
d) n^4

298. The big-theta notation for $f(n) = n\log(n^2 + 1) + n^2\log n$ is?
a) $n^2\log n$
b) n^2
c) logn
d) $n\log(n^2)$

299. The big-omega notation for $f(x, y) = x^5y^3 + x^4y^4 + x^3y^5$ is?
a) x^5y^3
b) x^5y^5
c) x^3y^3
d) x^4y^4

300. If $f_1(x)$ is $O(g(x))$ and $f_2(x)$ is $o(g(x))$, then $f_1(x) + f_2(x)$ is?

a) $O(g(x))$
b) $o(g(x))$
c) $O(g(x)) + o(g(x))$
d) None of the mentioned

301. The little-o notation for $f(x) = x\log x$ is?

a) x
b) x^3
c) x^2
d) $x\log x$

302. The big-O notation for $f(n) = 2\log(n!) + (n^2 + 1)\log n$ is?

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

303. The big-O notation for $f(x) = 5\log x$ is?

a) 1
b) x
c) x^2
d) x^3

304. The big-Omega notation for $f(x) = 2x^4 + x^2 – 4$ is?

a) x^2
b) x^3
c) x
d) x^4

305. Which of the following case does not exist in complexity theory?

a) Best case
b) Worst case
c) Average case
d) Null case

306. The complexity of linear search algorithm is _________

a) O(n)
b) O(log n)
c) O(n^2)
d) O(n log n)

307. The complexity of Binary search algorithm is _________

a) O(n)
b) O(log)
c) O(n^2)
d) O(n log n)

308. The complexity of merge sort algorithm is _________

a) O(n)
b) O(log n)
c) O(n^2)
d) O(n log n)

309. The Worst case occur in linear search algorithm when _________

a) Item is somewhere in the middle of the array
b) Item is not in the array at all
c) Item is the last element in the array
d) Item is the last element in the array or is not there at all

310. The worst case complexity for insertion sort is _________

a) O(n)
b) O(log n)
c) O(n^2)
d) O(n log n)

311. The complexity of Fibonacci series is _________

a) O(2^n)

b) O(log n)
c) O(n^2)
d) O(n log n)

312. The worst case occurs in quick sort when __________
a) Pivot is the median of the array
b) Pivot is the smallest element
c) Pivot is the middle element
d) None of the mentioned

313. The worst case complexity of quick sort is __________
a) O(n)
b) O(log n)
c) O(n^2)
d) O(n log n)

314. Which is used to measure the Time complexity of an algorithm Big O notation?
a) describes limiting behaviour of the function
b) characterises a function based on growth of function
c) upper bound on growth rate of the function
d) all of the mentioned

315. If for an algorithm time complexity is given by O(1) then the complexity of it is ____________
a) constant
b) polynomial
c) exponential
d) none of the mentioned

316. If for an algorithm time complexity is given by O($\log_2 n$) then complexity will be ___________

a) constant
b) polynomial
c) exponential
d) none of the mentioned

317. If for an algorithm time complexity is given by O(n) then the complexity of it is ____________
a) constant
b) linear
c) exponential
d) none of the mentioned

318. If for an algorithm time complexity is given by $O(n^2)$ then complexity will ____________
a) constant
b) quadratic
c) exponential
d) none of the mentioned

319. If for an algorithm time complexity is given by $O((^3/_2)^n)$ then complexity will be ____________
a) constant
b) quardratic
c) exponential
d) none of the mentioned

320. The time complexity of binary search is given by ____________
a) constant
b) quardratic
c) exponential
d) none of the mentioned

321. The time complexity of the linear search is given by ____________
a) $O(\log_2 n)$
b) O(1)
c) exponential

d) none of the mentioned

322. Which algorithm is better for sorting between bubble sort and quicksort?
a) bubble sort
b) quick sort
c) both are equally good
d) none of the mentioned

323. Time complexity of the binary search algorithm is constant.
a) True
b) False