Searching and Sorting
Assignment -#6

1. Which of the following searching algorithm is fastest?
   a) binary search
   b) linear search
   c) jump search
   d) More than one of the above
   e) None of the above

2. Where is linear searching used?
   a) Used all the time
   b) When the list has only a few elements
   c) When performing a single search in an unordered list
   d) More than one of the above
   e) None of the above

3. How can Jump Search be improved?
   a) Step size should be other than sqrt(n)
   b) Cannot be improved
   c) Begin from the kth item, where k is the step size
   d) More than one of the above
   e) None of the above

4. Which of the following searching algorithm is used with exponential sort after finding the appropriate range?
   a) Jump search
   b) Linear search
   c) Binary search
   d) More than one of the above
   e) None of the above

5. Which of the following searching algorithm is fastest when the input array is not sorted but has uniformly distributed values?
   a) linear search
   b) jump search
   c) binary search
   d) More than one of the above
   e) None of the above

6. In which of the cases uniform binary search fails compared to binary search?
   a) Complexity of code
   b) Many searches will be performed on several arrays of the same length
   c) Many searches will be performed on the same array
   d) More than one of the above
   e) None of the above

7. Interpolation search is a variation of?
   a) Exponential search
   b) Linear search
   c) Binary search
   d) More than one of the above
   e) None of the above

8. Which of the following is not an application of binary search?
   a) To search in unordered list
   b) Union of intervals
   c) To find the lower/upper bound in an ordered sequence
   d) More than one of the above
   e) None of the above

9. Which of the following step is taken after finding an element having value greater than the element being searched?
   a) binary search takes place in the forward direction
   b) linear search takes place in the forward direction
   c) linear search takes place in the backward direction
   d) More than one of the above
   e) None of the above

10. Which of the following is not an advantage of Fibonacci Search?
   a) When the element being searched for has a non uniform access storage
   b) It can be applied efficiently on unsorted arrays
   c) Can be used for large arrays which do not fit in the CPU cache or in the RAM
   d) More than one of the above
   e) None of the above

11. Is there any difference in the speed of execution between linear serach(recursive) vs linear search(Iterative)?
   a) Both execute at same speed
   b) Linear search(recursive) is faster
   c) Linear search(Iterative) is faster
   d) More than one of the above
   e) None of the above

12. 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) More than one of the above
e) None of the above

13. What is the worst case runtime of linear search(recursive) algorithm?
a) O(n)
b) O(logn)
c) $O(n^2)$
d) More than one of the above
e) None of the above

14. 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) More than one of the above
e) None of the above

15. 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) More than one of the above
e) None of the above

16. 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) More than one of the above
e) None of the above

17. 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) More than one of the above
e) None of the above

18. 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) More than one of the above
e) None of the above

19. 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 than one of the above
e) None of the above

20. Binary Search can be categorized into which of the following?
a) Brute Force technique
b) Divide and conquer
c) Greedy algorithm
d) More than one of the above
e) None of the above

21. 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) More than one of the above
e) None of the above

22. What is the best time complexity of bubble sort(optimised)?
a) N^2
b) NlogN
c) N
d) More than one of the above
e) None of the above

23. 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) Insertion Sort
d) More than one of the above
e) None of the above

24. Which of the following sorting algorithms has the lowest worst-case complexity?
   a) Merge Sort
   b) Bubble Sort
   c) Quick Sort
   d) More than one of the above
   e) None of the above

25. 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) More than one of the above
   e) None of the above

26. 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) More than one of the above
   e) None of the above

27. The worst case running times of Insertion sort, Merge sort and Quick sort, respectively, are:
   a) Θ(n log n), Θ(n log n) and Θ(n2)
   b) Θ(n2), Θ(n2) and Θ(n Log n)
   c) $\Theta(n^2)$, Θ(n log n) and $\Theta(n^2)$
   d) More than one of the above
   e) None of the above

28. A sorting technique is called stable if:
   a) It takes O(n*log(n)) time
   b) It maintains the relative order of occurrence of non-distinct elements
   c) It uses divide and conquer paradigm
   d) More than one of the above
   e) None of the above

29. 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) More than one of the above
   e) None of the above

30. Which of the below given sorting techniques has highest best-case runtime complexity.
   a) Quick sort
   b) Selection sort
   c) Insertion sort
   d) More than one of the above
   e) None of the above

31. Which of the following is true for computation time in insertion, deletion and finding maximum and minimum element in a sorted array ?
   a) Insertion – 0(1), Deletion – 0(1), Maximum – 0(1), Minimum – 0(1)
   b) Insertion – 0(1), Deletion – 0(1), Maximum – 0(n), Minimum – 0(n)
   c) Insertion – 0(n), Deletion – 0(n), Maximum – 0(1), Minimum – 0(1)
   d) More than one of the above
   e) None of the above

32. Selection sort algorithm design technique is an example of
   a) Greedy method
   b) Divide-and-conquer
   c) Dynamic Programming
   d) More than one of the above
   e) None of the above

33. The average case and worst case complexities for Merge sort algorithm are
   a) (n2), O (n2)
   b) (n2), O (n log2n)
   c) (n log2n), O (n log2n)
   d) More than one of the above
   e) None of the above

34. Which of the following is not a sorting algorithm?
   a) Bubble Sort
   b) Insertion sort
   c) Long Sort
   d) More than one of the above
   e) None of the above

35. ..... is the method used by card sorter.
   a) Radix sort
   b) Insertion
   c) Heap
   d) More than one of the above
   e) None of the above

Q 36: Which of the following is a comparison-based sorting algorithm?

a) Bucket sort
b) Radix sort
c) Merge sort
d) Counting sort
e) More than one of the above

Solution with Explanation

Answer1: a) binary search
Explanation: Binary search has the least time complexity (equal to log n) out of the given searching algorithms. This makes binary search preferable in most cases.

Answer2: d) More than one of the above (b and c)
Explanation: It is practical to implement linear search in the situations mentioned in When the list has only a few elements and When performing a single search in an unordered list, but for larger elements the complexity becomes larger and it makes sense to sort the list and employ binary search or hashing.

Answer3: c) Begin from the kth item, where k is the step size
Explanation: This gives a very slight improvement as you are skipping the first k elements.

Answer4: c) Binary search
Explanation: In exponential search, we first find a range where the required elements should be present in the array. Then we apply binary search in this range.

Answer5: a) linear search
Explanation: Out of the given options linear search is the only searching algorithm which can be applied to arrays which are not sorted. It has a time complexity of O(n) in the worst case.

Answer6: a) Complexity of code
Explanation: Uniform Binary Search is an optimized version of Binary Search used for repeated searches on arrays of the same size. Instead of computing midpoints during each search, midpoints are precomputed and stored in a lookup table. This eliminates the need for arithmetic operations during search, improving efficiency. The algorithm maintains an index modified by the lookup table, making it faster than traditional binary search.

Answer7: c Binary search
Explanation: Interpolation search is a variation of binary search which gives the best result when the array has uniformly distributed values. Interpolation search goes to different positions depending on the value being searched whereas binary search always goes to the middle element.

Answer8: a) To search in unordered list
Explanation: In Binary search, the elements in the list should be sorted. It is applicable only for ordered list. Hence Binary search in unordered list is not an application.

Answer9: a) binary search takes place in the forward direction
Explanation: First an element having value greater than the element being searched is found. After this linear search is performed in a backward direction.

Answer10: b) It can be applied efficiently on unsorted arrays
Explanation: Fibonacci Search is a comparison-based algorithm that works on sorted arrays. It is faster than Binary Search for accessing nearby elements and large arrays.
Similarities with Binary Search: Works on sorted arrays. Both are Divide and Conquer algorithms. Time complexity: O(log n).
Key Differences: Fibonacci Search divides the array into unequal parts. It uses addition/subtraction instead of division. It examines closer elements in each step, making it efficient for large arrays.
It requires a sorted array to function properly.

Answer11: c Linear search(Iterative) is faster
Explanation: The Iterative algorithm is faster than the latter as recursive algorithm has overheads like calling function and registering stacks repeatedly.

Answer12: a) No, recursive algorithm consumes more space
Explanation: The recursive algorithm consumes more space as it involves the usage the stack space(calls the function numerous times).

Answer13: a) O(n)
Explanation: In the worst case scenario, there might be a need of calling the stack n times. Therfore O(n).

Answer14: d) When the size of the dataset is low and unordered
Explanation: It is used when the size of the dataset is low as its runtime is O(n) which is more when compared to the binary search O(logn).

Answer15: a) 7^{th} cell
Explanation: The function calls itself till the element is found. But at the 7th call it terminates as goes outside the array.

Answer16: a) O(1)
Explanation: The best case occurs when the given element to be found is at the first position. Therefore O(1) is the correct answer.

Answer17: a) Binary search can't be used
Explanation: As binary search requires comparison, it is required that the list be ordered. Whereas this doesn't matter for linear search.

Answer18: c) T(n-1)+c
Explanation: After each call in the recursive algorithm, the size of n is reduced by 1. Therefore the optimal solution is T(n-1)+c.

Answer19: b) Less code and easy to implement
Explanation: A recursive approach is easier to understand and contains fewer lines of code.

Answer20: b) Divide and conquer - Binary search uses a divide and conquer strategy.

Answer21: b) Recurrence is T(n) = T(n-1) + O(n) and time complexity is O(n^2)
Explanation: The worst case of QuickSort occurs when the picked pivot is always one of the corner elements in sorted array. In worst case, QuickSort recursively calls one subproblem with size 0 and other subproblem with size (n-1). So recurrence is T(n) = T(n-1) + T(0) + O(n) The above expression can be rewritten as T(n) = T(n-1) + O(n).

Answer22: c) n
Explanation: The bubble sort is at its best if the input data is sorted. i.e. If the input data is sorted in the same order as expected output. This can be achieved by using one boolean variable. The boolean variable is used to check whether the values are swapped at least once in the inner loop.

Answer23: c) Insertion Sort
Explanation: Insertion sort takes linear time when input array is sorted or almost sorted (maximum 1 or 2 elements are misplaced). All other sorting algorithms mentioned above will take more than linear time in their typical implementation.

Answer24. A) Merge sort
Explanation: Worst-case complexities for the above sorting algorithms are as follows:
Merge Sort: O(n*log(n))
Bubble Sort: $O(n^2)$
Quick Sort: $O(n^2)$
Selection Sort: $O(n^2)$

Answer25: d) More than one of the above
Explanation: Merge Sort satisfies all the three options given above.

Answer26: c) $O(n^2)$
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.

Answer27: c) $\Theta(n^2)$, $\Theta(n \log n)$ and $\Theta(n^2)$
Explanation: Insertion Sort: Time complexity: $\Theta(n^2)$. Recursive relation: $T(n) = T(n-1) + \Theta(n)$.
The outer loop selects elements, and the inner loop finds the right position and shifts larger elements.
Merge Sort: Time complexity: $\Theta(n \log n)$. Recursive relation: $T(n) = 2T(n/2) + \Theta(n)$. The array is divided into two halves, sorted recursively, and merged.
Quick Sort: Time complexity: $\Theta(n^2)$ (worst case). Recursive relation: $T(n) = T(n-1) + \Theta(n)$. In the worst case (e.g., sorted array with corner pivots), the partitioning is unbalanced.

Answer28: b) It maintains the relative order of occurrence of non-distinct elements
Explanation: The correct option is B it maintains the relative order of occurrence of non-distinct elements. A sorting algorithm is called stable if it keeps elements with equal keys in the same relative order in the output as they were in the input.

Answer29: b) insertion sort < Quick sort < Merge sort < selection sort
Explanation: In best case,
Quick sort: O (nlogn)
Merge sort: O (nlogn)
Insertion sort: O (n)
Selection sort: O (n^2)

Answer30: b) Selection sort
Explanation: Quick sort best case time complexity is O(n logn)
Selection sort best case time complexity is O(n^2)
Insertion sort best case time complexity is O(n)
Bubble sort best case time complexity is O(n)

Answer31: C) Insertion – 0(n), Deletion – 0(n), Maximum – 0(1), Minimum – 0(1)
Explanation: In a sorted array, if we want to insert or delete then we have to traverse whole array and check where is the suitable position, so it will take O(n).
If array is sorted then end position will tell the maximum or minimum, so finding maximum or minimum will take O(1).

Answer32: a) Greedy method
Explanation: The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array. 1) The subarray which is already sorted. 2) Remaining subarray which is unsorted. In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray. Clearly, it is a greedy approach to sort the array. Option (A) is correct.

Answer33: c) (n log2n), O (n log2n)
Explanation: The best case, average case and worst case complexities for Merge sort algorithm are O(nlog2n). So, option (D) is correct.

Answer34: c) Long Sort

Answer35: a) radix sort

Answer36: C) Merge sort