**SEARCHING AND SORTING**

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

(A) Recurrence is T(n) = T(n-2) + O(n) and time complexity is O(n^2)

(B) Recurrence is T(n) = T(n-1) + O(n) and time complexity is O(n^2)

(C) Recurrence is T(n) = 2T(n/2) + O(n) and time complexity is O(nLogn)

(D) Recurrence is T(n) = T(n/10) + T(9n/10) + O(n) and time complexity is O(nLogn)

Answer: (B)

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)

filter\_none

edit

play\_arrow

brightness\_4

void exchange(int \*a, int \*b)

{

int temp;

temp = \*a;

\*a = \*b;

\*b = temp;

}

int partition(int arr[], int si, int ei)

{

int x = arr[ei];

int i = (si - 1);

int j;

for (j = si; j <= ei - 1; j++)

{

if(arr[j] <= x)

{

i++;

exchange(&arr[i], &arr[j]);

}

}

exchange (&arr[i + 1], &arr[ei]);

return (i + 1);

}

/\* Implementation of Quick Sort

arr[] --> Array to be sorted

si --> Starting index

ei --> Ending index

\*/

void quickSort(int arr[], int si, int ei)

{

int pi; /\* Partitioning index \*/

if(si < ei)

{

pi = partition(arr, si, ei);

quickSort(arr, si, pi - 1);

quickSort(arr, pi + 1, ei);

}

}

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

(A) O(n^2 Logn)

(B) O(n^2)

(C) O(n Logn Logn)

(D) O(nLogn)

Answer: (D)

Explanation: If we use median as a pivot element, then the recurrence for all cases

becomes

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

The above recurrence can be solved using Master Method. It falls in case 2 of master method.

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

(A) Insertion Sort

(B) Merge Sort

(C) Quick Sort

(D) Bubble Sort

Answer: (C)

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

(A) Quick Sort

(B) Heap Sort

(C) Merge Sort

(D) Insertion Sort

Answer: (D)

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 lienear time in their typical implementation.

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

(A) Insertion Sort with time complexity O(kn)

(B) Heap Sort with time complexity O(nLogk)

(C) Quick Sort with time complexity O(kLogk)

(D) Merge Sort with time complexity O(kLogk)

Answer: (B)

**6. Consider a situation where swap operation is very costly. Which of the following sorting algorithms should be preferred so that the number of swap operations are minimized in general?**

(A) Heap Sort

(B) Selection Sort

(C) Insertion Sort

(D) Merge Sort

Answer: (B)

Explanation:

Selection sort makes O(n) swaps which is minimum among all sorting algorithms mentioned above.

**7. Which of the following is not true about comparison based sorting algorithms?**

(A) The minimum possible time complexity of a comparison based sorting algorithm is O(nLogn) for a random input array

(B) Any comparison based sorting algorithm can be made stable by using position as a criteria when two elements are compared

(C) Counting Sort is not a comparison based sorting algortihm

(D) Heap Sort is not a comparison based sorting algorithm.

Answer: (D)

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

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

**Which statement is correct?**

(A) The pivot could be either the 7 or the 9.

(B) The pivot could be the 7, but it is not the 9

(C) The pivot is not the 7, but it could be the 9

(D) Neither the 7 nor the 9 is the pivot.

Answer: (A)

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

**9. Suppose we are sorting an array of eight integers using heapsort, and we have just finished some heapify (either maxheapify or minheapify) operations. The array now looks like this:**

**16 14 15 10 12 27 28**

**How many heapify operations have been performed on root of heap?**

(A) 1

(B) 2

(C) 3 or 4

(D) 5 or 6

Answer: (B)

Explanation: In Heapsort, we first build a heap, then we do following operations till the heap size becomes 1.

a) Swap the root with last element

b) Call heapify for root

c) reduce the heap size by 1.

In this question, it is given that heapify has been called few times and we see that last two elements in given array are the 2 maximum elements in array. So situation is clear, it is maxheapify whic has been called 2 times.

**10. What is the best time complexity of bubble sort?**

(A) N^2

(B) NlogN

(C) N

(D) N(logN)^2

Answer: (C)

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.

Consider the following code snippet:

int main()

{

int arr[] = {10, 20, 30, 40, 50}, i, j, isSwapped;

int n = sizeof(arr) / sizeof(\*arr);

isSwapped = 1;

for(i = 0; i < n - 1 && isSwapped; ++i)

{

isSwapped = 0;

for(j = 0; j < n - i - 1; ++j)

if (arr[j] > arr[j + 1])

{

swap(&arr[j], &arr[j + 1]);

isSwapped = 1;

}

}

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

printf("%d ", arr[i]);

return 0;

}

Please observe that in the above code, the outer loop runs only once.

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

a) Insertion sort

b) Selection sort

c) Bubble sort

d) Merge sort

ANSWER: B

**12. Which of the following is a stable sorting algorithm?**

a) Merge sort

b) Typical in-place quick sort

c) Heap sort

d) Selection sort

ANSWER: A

**13. Which of the following is not an in-place sorting algorithm?**

a) Selection sort

b) Heap sort

c) Quick sort

d) Merge sort

ANSWER: D

**14. Running merge sort on an array of size n which is already sorted is**

a) O(n)

b) O(nlogn)

c) O(n2)

d) None

ANSWER: B

**15. The time complexity of a quick sort algorithm which makes use of median, found by an O(n) algorithm, as pivot element is**

a) O(n2)

b) O(nlogn)

c) O(nloglogn)

d) O(n)

ANSWER: B

**16. Which of the following is not a noncomparison sort?**

a) Counting sort

b) Bucket sort

c) Radix sort

d) Shell sort

ANSWER: D

**17. The time complexity of heap sort in worst case is**

a) O(logn)

b) O(n)

c) O(nlogn)

d) O(n2)

ANSWER: C

**18. If the given input array is sorted or nearly sorted, which of the following algorithm gives the best performance?**

a) Insertion sort

b) Selection sort

c) Quick sort

d) Merge sort

ANSWER: A

**19. Which of the following algorithm pays the least attention to the ordering of the elements in the input list?**

a) Insertion sort

b) Selection sort

c) Quick sort

d) None

ANSWER: B

**20. Consider the situation in which assignment operation is very costly. Which of the following sorting algorithm should be performed so that the number of assignment operations is minimized in general?**

a) Insertion sort

b) Selection sort

c) Heap sort

d) None

ANSWER: B

**21. Time complexity of bubble sort in best case is**

a) θ (n)

b) θ (nlogn)

c) θ (n2)

d) θ (n(logn) 2)

ANSWER: A

**22. Given a number of elements in the range [0….n3]. which of the following sorting algorithms can sort them in O(n) time?**

a) Counting sort

b) Bucket sort

c) Radix sort

d) Quick sort

ANSWER: C

**23. Which of the following algorithms has lowest worst case time complexity?**

a) Insertion sort

b) Selection sort

c) Quick sort

d) Heap sort

ANSWER: D

**24. Which of the following sorting algorithms is/are stable**

a) Counting sort

b) Bucket sort

c) Radix sort

d) All of the above

ANSWER: D

**25. Counting sort performs …………. Numbers of comparisons between input elements.**

a) 0

b) n

c) nlogn

d) n2

ANSWER: A

**26. The running time of radix sort on an array of n integers in the range [0……..n5 -1] when using base 10 representation is**

a) θ (n)

b) θ (nlogn)

c) θ (n2)

d) none

ANSWER: B

**27. The running time of radix sort on an array of n integers in the range [0……..n5 -1] when using base n representation is**

a) θ (n)

b) θ (nlogn)

c) θ (n2)

d) None

ANSWER: A

**28. Which of the following sorting algorithm is in-place**

a) Counting sort

b) Radix sort

c) Bucket sort

d) None

ANSWER: B

**29. The radix sort does not work correctly if each individual digit is sorted using**

a) Insertion sort

b) Counting sort

c) Selection sort

d) Bubble sort

ANSWER: C

**30. Which of the following sorting algorithm has the running time that is least dependant on the initial ordering of the input?**

a) Insertion sort

b) Quick sort

c) Merge sort

d) Selection sort

ANSWER: D

**31. Time complexity to sort elements of binary search tree is**

a) O(n)

b) O(nlogn)

c) O(n2)

d) O(n2logn)

ANSWER: A

**32. The lower bound on the number of comparisons performed by comparison-based sorting algorithm is**

a) Ω (1)

b) Ω (n)

c) Ω (nlogn)

d) Ω (n2)

ANSWER: C

**33. Which of the following algorithm(s) can be used to sort n integers in range [1…..n3] in O(n) time?**

a) Heap sort

b) Quick sort

c) Merge sort

d) Radix sort

ANSWER: D

**34. Which of the following algorithm design technique is used in the quick sort algorithm?**

a) Dynamic programming

b) Backtracking

c) Divide-and-conquer

d) Greedy method

ANSWER: C

**35. Merge sort uses**

a) Divide-and-conquer

b) Backtracking

c) Heuristic approach

d) Greedy approach

ANSWER: A

**36. For merging two sorted lists of size m and n into sorted list of size m+n, we require comparisons of**

a) O(m)

b) O(n)

c) O(m+n)

d) O(logm + logn)

ANSWER: C

**37. A sorting technique is called stable if it**

a) Takes O(nlogn) times

b) Maintains the relative order of occurrence of non-distinct elements

c) Uses divide-and-conquer paradigm

d) Takes O(n) space

ANSWER: B

**38. In a heap with n elements with the smallest element at the root, the seventh smallest element can be found in time**

a) θ (nlogn)

b) θ (n)

c) θ (logn)

d) θ (1)

ANSWER: A

**39. What would be the worst case time complexity of the insertion sort algorithm, if the inputs are restricted to permutation of 1…..n with at most n inversion?**

a) θ (n2)

b) θ (nlogn)

c) θ (n1.5)

d) θ (n)

ANSWER: D

**40. In a binary max heap containing n numbers, the smallest element can be found in time**

a) θ (n)

b) θ (logn)

c) θ (loglogn)

d) θ (1)

ANSWER: A

**41. 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 item is not there at all

Answer :D

**42. f the number of records to be sorted is small, then …… sorting can be efficient**.

A. Merge

B. Heap

C. Selection

D. Bubble

Answer:C

**43. The complexity of sorting algorithm measures the …… as a function of the number n of items to be sorter.**

A. average time

B. running time

C. average-case complexity

D. case-complexity

Answer:B

**44. Which of the following is not a limitation of binary search algorithm?**

A. must use a sorted array

B. requirement of sorted array is expensive when a lot of insertion and deletions are needed

C. there must be a mechanism to access middle element directly

D. binary search algorithm is not efficient when the data elements more than 1500.

Answer:D

**45. The Average case occurs in linear search algorithm ……….**

A. when item is somewhere in the middle of the array

B. when item is not the array at all

C. when item is the last element in the array

D. Item is the last element in the array or item is not there at all

Answer:A

**46. Binary search algorithm cannot be applied to …**

A. sorted linked list

B. sorted binary trees

C. sorted linear array

D. pointer array

Answer:A

**47. Complexity of linear search algorithm is ………**

A. O(n)

B. O(logn)

C. O(n2)

D. O(n logn)

Answer:A

**48. Sorting algorithm can be characterized as ……**

A. Simple algorithm which require the order of n2 comparisons to sort n items.

B. Sophisticated algorithms that require the O(nlog2n) comparisons to sort items.

C. Both of the above

D. None of the above

Answer:C

**49. State True or False for internal sorting algorithms.**

**i) Internal sorting are applied when the entire collection if data to be sorted is small enough that the sorting can take place within main memory.**

**ii) The time required to read or write is considered to be significant in evaluating the performance of internal sorting.**

A. i-True, ii-True

B. i-True, ii-False

C. i-False, ii-True

D. i-False, ii-False

Answer:B

**50. The complexity of merge sort algorithm is ……**

A. O(n)

B. O(logn)

C. O(n2)

D. O(n logn)

Answer:D