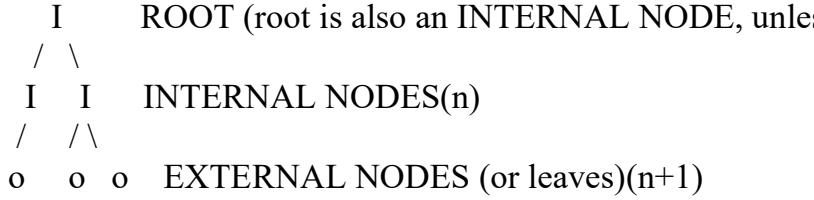
MCQ

Tree

1. The no of external nodes in a full binary tree with n internal nodes is?

A. n
B. n+1
C. 2n
D. 2n + 1
Ans : B

Explanation: An internal node (also known as an inner node, inode for short, or branch node) is any node of a tree that has child nodes. Similarly, an external node (also known as an outer node, leaf node, or terminal node) is any node that does not have child nodes. The no of external nodes in a full binary tree with n internal nodes is n+1.



2. Which of the following is a true about Binary Trees?

A. Every binary tree is either complete or full.
B. Every complete binary tree is also a full binary tree.
C. Every full binary tree is also a complete binary tree.
D. No binary tree is both complete and full.
E. None of the above
Ans : E

Explanation: A full binary tree (sometimes proper binary tree or 2-tree or strictly binary tree) is a tree in which every node other than the leaves has two children. A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible. A) is incorrect.
3. A Binary Tree can have

A. Can have 2 children
B. Can have 1 children
C. Can have 0 children
D. All of the above

Ans : D

Explanation: A Binary Tree can have 0, 1, 2 children.

4.Which of the following is not an advantage of trees?

A. Hierarchical structure
B. Faster search
C. Router algorithms
D. Undo/Redo operations in a notepad

View Answer
Ans : D

Explanation: This is an application of stack.

5.The difference between the external path length and the internal path length of a binary tree with n internal nodes is?

A. 1
B. n
C. n + 1
D. 2n

View Answer
Ans : D
Explanation: The sum over all external nodes of the lengths of the paths from the root of an extended binary tree to each node. The internal and external path lengths are related by E=I+2n or E-I=2n.

6.Height of Height of a binary tree is

A. MAX(Height of left Subtree, Height of right subtree)+1
B. MAX(Height of left Subtree, Height of right subtree)
C. MAX(Height of left Subtree, Height of right subtree)-1
D. None of the above

View Answer
Ans : A

Explanation: Height of Height of a binary tree is MAX(Height of left Subtree, Height of right subtree)+1.

7. Which of the following options is an application of splay trees ?
A. cache Implementation
B. networks
C. send values
D. receive values
View Answer
Ans : A

Explanation: Splay trees can be used for faster access to recently accessed items and hence used for cache implementations.

8. Suppose a complete binary tree has height h>0. The minimum no of leaf nodes possible in term of h is?

A. 2h -1
B. 2h -1 + 1
C. 2h -1
D. 2h +1
View Answer
Ans : C

Explanation: Suppose a complete binary tree has height h>0. The minimum no of leaf nodes possible in term of h is 2h -1.

9.What is the specialty about the inorder traversal of a binary search tree?
a)It traverses in a non increasing order
b) It traverses in an increasing order
c) It traverses in a random fashion
d)It traverses based on priority of the node
Answer:b b
Explanation: As a binary search tree consists of elements lesser than the node to the left and the ones greater than the node to the right, an inorder traversal will give the elements in an increasing order.

10.Which of the following is false about a binary search tree?

a)The left child is always lesser than its parent.
b)The right child is always greater than its parent.
c)The left and right sub-trees should also be binary search trees.
d)In order sequence gives decreasing order of elements.

Answer:d
Explanation: In order sequence of binary search trees will always give ascending order of elements. Remaining all are true regarding binary search trees.

11. What are the worst case and average case complexities of a binary search tree?
a)O(n), O(n)
b)O(logn), O(logn)
c)O(logn), O(n)
d)O(n),O(logn)

Answer:d
Explanation: Worst case arises when the tree is skewed(either to the left or right) in which case you have to process all the nodes of the tree giving O(n) complexity, otherwise O(logn) as you process only the left half or the right half of the tree.

Heap

12. A _____ is a special Tree-based data structure in which the tree is a complete binary tree.?

A. Graph
B. Heap
C. List
D. Stack

Ans : B

Explanation: A Heap is a special Tree-based data structure in which the tree is a complete binary tree.

13. How many type of heap are there?

A. 2
B. 3
C. 4
D. 5

Ans : A

Explanation: There are 2 types of heap : max-heap and min-heap.

14.In which heap the root node must be greatest among the keys present at all of it's children?

A. min-heap
B. max-heap
C. Both A and B
D. None of the above

Ans : B

Explanation : Max-Heap: In a Max-Heap the key present at the root node must be greatest among the keys present at all of it's children

15.What is the complexity of adding an element to the heap?

A. O(log n)
B. O(log h)
C. O(h)
D. Both A and C

Ans : D

Explanation : The total possible operation in re locating the new location to a new element will be equal to height of the heap.

16. Heap can be used as _____

A. Priority queue
B. Stack
C. A decreasing order array
D. Normal Array

Ans : A

Explanation : The property of heap that the value of root must be either greater or less than both of its children makes it work like a priority queue.

17. An array consists of n elements. We want to create a heap using the elements. The time complexity of building a heap will be in order of

A. O(n*n*logn)
B. O(n*logn)
C. O(n*n)
D. O(n *logn *logn)

Ans : B

Explanation: The total time taken will be N times the complexity of adding a single element to the heap. And adding a single element takes logN time, so That is equal to N*logN.

18. Which one of the following array elements represents a binary min heap?

A. 12 10 8 25 14 17
B. 8 10 12 25 14 17
C. 25 17 14 12 10 8
D. 14 17 25 10 12 8

Ans : B

Explanation : A tree is min heap when data at every node in the tree is smaller than or equal to it's children' s data. So, only 8 10 12 25 14 17 generates required tree.

19. Given an array of element 5, 7, 9, 1, 3, 10, 8, 4. Which of the following is the correct sequences of elements after inserting all the elements in a min-heap?

A. 1,3,4,5,7,8,9,10
B. 1,4,3,9,8,5,7,10
C. 1,3,4,5,8,7,9,10
D. 1,3,7,4,8,5,9,10

Ans : A

Explanation : Building a min-heap the result will a sorted array so the 1, 3, 4, 5, 7, 8, 9, 10 is correct. If we change the implementation strategy 1, 4, 3, 8, 9, 5, 7, 10 is also correct. (First filling the right child rather than left child first).

20. What is the amortized cost per operation of a skew heap?

A. O(N)
B. O(N log N)
C. O(N2)
D. O(log N)

Ans : D

Explanation : The amortized cost per operation of a skew heap is O(log N) since the worst case analysis of skew heap is O(N) and splay tree is O(M log N).

Sorting

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

A. Insertion sort
B. Selection sort
C. Bubble sort

Ans : B

Explanation:Selection sort is not a stable sorting algorithm.Itworks by finding the minimum element and then inserting it in its correct position by swapping with the element which is in the position of this minimum element. This is what makes it unstable.

23. What is an external sorting algorithm?

A. Algorithm that uses tape or disk during the sort
B. Algorithm that uses main memory during the sort
C. Algorithm that involves swapping
D. Algorithm that are considered ‘in place’

Ans : A

Explanation: As the name suggests, external sorting algorithm uses external memory like tape or disk.

24. If the number of records to be sorted is small, then …… sorting can be efficient.

A. Merge
B. Heap
C. Selection
D. Bubble

Ans : C

Explanation:Selection sorting can be efficient. The constants in the time function in selection sort is small. When the input size is small ,other algorithms prove to take greater time .but when input is large enough nlogn algorithms are efficient.

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

A. Selection sort
B. Heap sort
C. Quick sort
D. Merge sort

Ans : D

Explanation: Merge sort is not an in-place sorting algorithm. In-place means it does not occupy extra memory for merge operation as in the standard case.

26. What is the advantage of bubble sort over other sorting techniques?

A. It is faster
B. Consumes less memory
C. Detects whether the input is already sorted
D. All of the mentioned

Ans : C

Explanation: Bubble sort is one of the simplest sorting techniques and perhaps the only advantage it has over other techniques is that it can detect whether the input is already sorted.

27.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

View Answer

Ans : B

Explanation:The complexity of sorting algorithm measures the running time as a function of the number n of items to be sorter.

28. 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.

Ans : 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.

29. In the following scenarios, when will you use selection sort?

A. The input is already sorted
B. A large file has to be sorted
C. Large values need to be sorted with small keys
D. Small values need to be sorted with large keys

Ans : C

Explanation: Selection is based on keys, hence a file with large values and small keys can be efficiently sorted with selection sort.

30. What is the advantage of selection sort over other sorting techniques?

a) It requires no additional storage space
b) It is scalable
c) It works best for inputs which are already sorted
d) It is faster than any other sorting technique

View Answer

Ans : A

Explanation: Since selection sort is an in-place sorting algorithm, it does not require additional storage.

31. What is the average case complexity of selection sort?

A. O(nlogn)
B. O(logn)
C. O(n)
D. O(n2)

View Answer

Ans : D
Explanation: In the average case, even if the input is partially sorted, selection sort behaves as if the entire array is not sorted. Selection sort is insensitive to input.
32. What is the disadvantage of selection sort?

A. It requires auxiliary memory
B. It is not scalable
C. It can be used for small keys
D. None of the mentioned

View Answer
Ans : B

Explanation: As the input size increases, the performance of selection sort decreases.

33. The given array is arr = {3,4,5,2,1}. The number of iterations in bubble sort and selection sort respectively are,

A. 5 and 4
B. 4 and 5
C. 2 and 4
D. 2 and 5

View Answer
Ans : A

Explanation: Since the input array is not sorted, bubble sort takes n iterations and selection sort takes (n-1) iterations.

34. The given array is arr = {1,2,3,4,5}. (bubble sort is implemented with a flag variable)The number of iterations in selection sort and bubble sort respectively are,

A. 5 and 4
B. 1 and 4
C. 0 and 4
D. 4 and 1

Ans : D

Explanation: Selection sort is insensitive to input, hence (n-1) iterations. Whereas bubble sort iterates only once to set the flag to 0 as the input is already sorted.

35.What is the best case complexity of selection sort?

A. O(nlogn)
B. O(logn)
C. O(n)
D. O(n2)

Ans : D

Explanation: The best, average and worst case complexities of selection sort is O(n2). (n-1) + (n-2) + (n-3) + …. + 1 = (n(n-1))/2 ~ (n2)/2.

36.What is the worst case complexity of bubble sort?

A. O(nlogn)
B. O(logn)
C. O(n)
D. O(n2)

Ans : D

Explanation: Bubble sort works by starting from the first element and swapping the elements if required in each iteration.

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

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

Answer:c

Explanation: Merge sort uses divide and conquer in order to sort a given array. This is because it divides the array into two halves and applies merge sort algorithm to each half individually after which the two sorted halves are merged together.

38. What is the average case time complexity of merge sort?

a)O(nlogn)
b)$O(n^2)$
c)$O(n^2 \log n)$
d)$O(n\log n^2)$

Answer:a

Explanation: The recurrence relation for merge sort is given by $T(n) = 2T(n/2) + n$. It is found to be equal to O(n log n) using the master theorem.

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

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

d)O(nlogn)

Answer:c
Explanation: An additional space of O(n) is required in order to merge two sorted arrays. Thus merge sort is not an in place sorting algorithm.

40. What is the worst case time complexity of merge sort?
a)O(nlogn)
b)$O(n^2)$
c)$O(n^2 \log n)$
d) $O(n \log n^2)$
Answer:a
Explanation: The time complexity of merge sort is not affected by worst case as its algorithm has to implement the same number of steps in any case. So its time complexity remains to be O(n log n).

41. Which of the following method is used for sorting in merge sort?
a)merging
b)partitioning
c)selection
d)exchanging

Answer:a
Explanation: Merge sort algorithm divides the array into two halves and applies merge sort algorithm to each half individually after which the two sorted halves are merged together. Thus its method of sorting is called merging.

42. What will be the best case time complexity of merge sort?
a) O(n log n)
b) $O(n^2)$
c) $O(n^2 \log n)$
d) $O(n \log n^2)$
Answer:a
Explanation: The time complexity of merge sort is not affected in any case as its algorithm has to implement the same number of steps. So its time complexity remains to be O(n log n) even in the best case.

43. 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
Answer:d
Explanation: In-place, top down and bottom up merge sort are different variants of merge

sort. Whereas linear merge sort is not a possible variant as it is a comparison based sort and the minimum time complexity of any comparison based sort is O(n log n).

44. 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
Answer:b
Explanation: Merge sort is not an adaptive sorting algorithm. This is because it takes O(n log n) time complexity irrespective of any case.

45. Which of the following is not in place sorting algorithm by default?
a) merge sort
b) quick sort
c) heap sort
d) insertion sort
Answer:a
Explanation: Quick sort, heap sort, and insertion sort are in-place sorting algorithms, whereas an additional space of O(n) is required in order to merge two sorted arrays. Even though we have a variation of merge sort (to do in-place sorting), it is not the default option. So, among the given choices, merge sort is the most appropriate answer.

46. Which of the following is not a stable sorting algorithm?
a) Quick sort
b) Cocktail sort
c) Bubble sort
d) Merge sort
Answer:a
Explanation: Out of the given options quick sort is the only algorithm which is not stable. Merge sort is a stable sorting algorithm.

47. 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
Answer:d
Explanation: Insertion sort takes linear time to sort a partially sorted array. Though merge and quick sort takes O(n*logn) complexity to sort, merge sort is stable. Hence, Merge sort takes less time to sort partially sorted array.

.

48. Which of the following sorting algorithm makes use of merge sort?
a) tim sort
b) intro sort
c) bogo sort
d) quick sort
Answer:a
Explanation: Tim sort is a hybrid sorting algorithm as it uses more than one sorting algorithm internally. It makes use of merge sort and insertion sort.

49. 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);



}
}
```

Answer:b
Explanation: Merge sort first sorts the two halves of the array individually. Then it merges the two sorted halves in order to obtain sorted array.

50. Which of the following sorting algorithm does not use recursion?
a) quick sort
b) merge sort
c) heap sort
d) bottom up merge sort
Answer:d
Explanation: Bottom up merge sort uses the iterative method in order to implement sorting. It begins by merging a pair of adjacent array of size 1 each and then merge arrays of size 2 each in the next step and so on.

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

52. 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)
Answer:c
Explanation: The worst case performance of a quick sort algorithm is mathematically found to be $O(N^2)$.

53. Which of the following methods is the most effective for picking the pivot element?
a) first element
b) last element
c) median-of-three partitioning
d) random element
Answer:c
Explanation: Median-of-three partitioning is the best method for choosing an appropriate pivot element. Picking a first, last or random element as a pivot is not much effective.

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

55. Which is the safest method to choose a pivot element?
a) choosing a random element as pivot
b) choosing the first element as pivot
c) choosing the last element as pivot
d) median-of-three partitioning method

Answer:a
Explanation: This is the safest method to choose the pivot element since it is very unlikely that a random pivot would consistently provide a poor partition.

56. What is the average running time of a quick sort algorithm?
a) $O(N^2)$
b) O(N)
c) O(N log N)
d) O(log N)
Answer:c
Explanation: The best case and average case analysis of a quick sort algorithm are mathematically found to be O(N log N).

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

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

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

60. Which among the following is the best cut-off range to perform insertion sort within a quick sort?
a) N=0-5

b) N=5-20
c) N=20-30
d) N>30
Answer:b
Explanation: A good cut-off range is anywhere between N=5 and N=20 to avoid nasty degenerate cases.

61. How many comparisons will be made to sort the array arr = {1, 5, 3, 8, 2} using MSD radix sort?
a) 5
b) 7
c) 9
d) 0
Answer:d
Explanation: As MSD radix sort is an example of non comparison sort so it is able to sort an array without making any comparison. So the answer should be 0.

62. What is the full form of MSD in MSD radix sort?
a) most significant digit
b) many significant digit
c) more significant digit
d) must significant digit
Answer:a
Explanation: MSD stands for Most Significant Digit. It is named so because in this algorithm the processing begins from the most significant digit.

63. Which of the following combines qualities of MSD radix sort and LSD radix sort?
a) in-place MSD radix sort
b) stable MSD radix sot
c) 3 way radix quick sort
d) forward radix sort
Answer:d
Explanation: Forward radix sort combines the qualities of MSD and LSD radix sort. The sorting is done by separating the strings into groups.

64. Which of the following is the most suitable definition of radix sort?
a) It is a non comparison based integer sort
b) It is a comparison based integer sort
c) It is a non comparison based non integer sort
d) It is a comparison based non integer sort
Answer:a
Explanation: Radix sort is a non-comparison based integer sort. It sorts the given data by grouping keys which share the same significant position value.

65. Which of the following is an alternate name of MSD radix sort?

a) bottom up radix sort
b) top down radix sort
c) forward radix sort
d) backward radix sort
Answer:b
Explanation: Top down radix sort is an alternate name of MSD radix sort. It is because in this algorithm the processing starts from the most significant digit and end at least significant digit.

66. Which of the following is not true about MSD radix sort?
a) its processing starts from the most significant digit
b) it is not a stable sort
c) it is an in place sorting algorithm
d) it is non comparison based sort
Answer:c
Explanation: MSD radix sort takes non constant time for sorting the input data. So it is not an in place sorting algorithm.

68. What is the average time complexity of MSD radix sort (w= bits required to store each key)?
a) O(n + w)
b) O(n.w)
c) $O(n^2)$
d) O(n log n)
Answer:b
Explanation: Time complexity of radix sort is O(n.w). It performs better than quick sort when we have log n bits for every digit.

70.Which of the following statement is not a stable sorting algorithm?
a) LSD radix sort
b) MSD radix sort
c) Counting sort
d) Pigeonhole sort
Answer:b
Explanation:MSD radix sort is not a stable sort. It is because the elements with identical values do not appear in the same order in the output array as they were in the input array.

71. Which of the following is not true about radix sort?
a) Radix sort performs better than quick sort when we have log n bits for every digit
b) Radix sort has better cache performance than quick sort
c) Radix sort has higher values of constant factor in asymptotic notation
d) Radix sort takes more space than quick sort

Answer:b
Explanation: Quick sort has a better cache performance than radix sort. Radix sort also takes more space as compared to quick sort.

72. What is the advantage of radix sort over quick sort?
a) radix sort performs better than quick sort when we have log n bits for every digit
b) radix sort has lesser space complexity
c) radix sort is not a comparison based sorting technique
d) radix sort has better cache performance than quick sort
Answer:a
Explanation: Radix sort performs better than quick sort when we have log n bits for every digit. But radix sort takes more space as compared to quick sort.

Graph
73. 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. None of the mentioned
Ans : A

Explanation: In a walk if the vertices are distinct it is called a path, whereas if the edges are distinct it is called a trail.

74. 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
Answer:c
Explanation: The Breadth First Search Algorithm searches the nodes on the basis of level. It takes a node (level 0), explores it's neighbors (level 1) and so on.

75. 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)
Answer:a
Explanation: The Breadth First Search explores every node once and every edge once (in worst case), so it's time complexity is O(V + E).

76. The Data structure used in standard implementation of Breadth First Search is?
a) Stack
b) Queue
c) Linked List
d) Tree
Answer:b
Explanation: The Breadth First Search explores every node once and put that node in queue and then it takes out nodes from the queue and explores it's neighbors.

77. The Breadth First Search traversal of a graph will result into?
a) Linked List
b) Tree
c) Graph with back edges
d) Arrays
Answer:b
Explanation: The Breadth First Search will make a graph which don't have back edges (a tree) which is known as Breadth First Tree.

78. 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
Answer:d
Explanation: Breadth First Search can be applied to Bipartite a graph, to find the shortest path between two nodes, in GPS Navigation. In Path finding, Depth First Search is used.

79. 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

Answer:b
Explanation: When Every node will have one successor then the Breadth First Search is unique. In all other cases, when it will have more than one successor, it can choose any of them in arbitrary order.

80. In BFS, how many times a node is visited?
a) Once
b) Twice
c) Equivalent to number of indegree of the node
d) Thrice

Answer:c
Explanation: In Breadth First Search, we have to see whether the node is visited or not by it's ancestor. If it is visited, we won't let it enter it in the queue.
81. 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
Answer:a
Explanation: In Depth First Search, we explore all the nodes aggressively to one path and then backtrack to the node. Hence, it is equivalent to the pre-order traversal of a Binary Tree.

82. 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)
Answer:a
Explanation: The Depth First Search explores every node once and every edge once (in worst case), so it's time complexity is O(V + E).

83. The Data structure used in standard implementation of Breadth First Search is?
a) Stack
b) Queue
c) Linked List
d) Tree
Answer:a
Explanation: The Depth First Search is implemented using recursion. So, stack can be used as data structure to implement depth first search.

84. The Depth First Search traversal of a graph will result into?
a) Linked List
b) Tree
c) Graph with back edges
d) Array
Answer:b
Explanation: The Depth First Search will make a graph which don't have back edges (a tree) which is known as Depth First Tree.

85. 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
Answer:a
Explanation: This is the definition of the Depth First Search. Exploring a node, then aggressively finding nodes till it is not able to find any node.

86. 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 Network
Answer:d
Explanation: Depth First Search is used in the Generation of topological sorting, Strongly Connected Components of a directed graph and to detect cycles in the graph. Breadth First Search is used in peer to peer networks to find all neighborhood nodes.

87. 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
Answer:b
Explanation: When Every node will have one successor then the Depth First Search is unique. In all other cases, when it will have more than one successor, it can choose any of them in arbitrary order.

88. 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
Answer:a
Explanation: In the stack, at a time, there can be nodes which can differ in many levels. So, it can be the maximum distance between two nodes in the graph.

89. Given G is a bipartite graph and the bipartitions of this graphs are U and V respectively. What is the relation between them?
a) Number of vertices in U = Number of vertices in V
b) Sum of degrees of vertices in U = Sum of degrees of vertices in V
c) Number of vertices in U > Number of vertices in V
d) Nothing can be said

Answer:b
Explanation: We can prove this by induction. By adding one edge, the degree of vertices in U is equal to 1 as well as in V. Let us assume that this is true for n-1 edges and add one more edge. Since the given edge adds exactly once to both U and V we can tell that this statement is true for all n vertices.

90. There are four students in a class namely A, B, C and D. A tells that a triangle is a bipartite graph. B tells pentagon is a bipartite graph. C tells square is a bipartite graph. D tells heptagon is a bipartite graph. Who among the following is correct?
a)A
b)B
c)C
d)D
Answer:c
Explanation: We can prove it in this following way. Let '1' be a vertex in bipartite set X and let '2' be a vertex in the bipartite set Y. Therefore the bipartite set X contains all odd numbers and the bipartite set Y contains all even numbers. Now let us consider a graph of odd cycle (a triangle). There exists an edge from '1' to '2', '2' to '3' and '3' to '1'. The latter case ('3' to '1') makes an edge to exist in a bipartite set X itself. Therefore telling us that graphs with odd cycles are not bipartite.

91. A complete bipartite graph is a one in which each vertex in set X has an edge with set Y. Let n be the total number of vertices. For maximum number of edges, the total number of vertices that should be present on set X is?
a) n
b) n/2
c) n/4
d) data insufficient
Answer:b
Explanation: We can prove this by calculus. Let x be the total number of vertices on set X. Therefore set Y will have n-x. We have to maximize x*(n-x). This is true when x=n/2.

92.When is a graph said to be bipartite?
a) If it can be divided into two independent sets A and B such that each edge connects a vertex from to A to B
b) If the graph is connected and it has odd number of vertices
c) If the graph is disconnected
d) If the graph has at least n/2 vertices whose degree is greater than n/2
Answer:a
Explanation: A graph is said to be bipartite if it can be divided into two independent sets A and B such that each edge connects a vertex from A to B.
93. Are trees bipartite?
a) Yes
b) No

c) Yes if it has even number of vertices
d) No if it has odd number of vertices
Answer:a
Explanation: Condition needed is that there should not be an odd cycle. But in a tree there are no cycles at all. Hence it is bipartite.

94. A graph has 20 vertices. The maximum number of edges it can have is? (Given it is bipartite)
a) 100
b) 140
c) 80
d) 20
Answer:a
Explanation: Let the given bipartition X have x vertices, then Y will have 20-x vertices. We need to maximize x*(20-x). This will be maxed when x=10.

95. Can there exist a graph which is both eulerian and is bipartite?
a) Yes
b) No
c) Yes if it has even number of edges
d) Nothing can be said
Answer:a
Explanation: If a graph is such that there exists a path which visits every edge atleast once, then it is said to be Eulerian. Taking an example of a square, the given question evaluates to yes.

96. A graph is found to be 2 colorable. What can be said about that graph?
a) The given graph is eulerian
b) The given graph is bipartite
c) The given graph is hamiltonian
d) The given graph is planar
Answer:b
Explanation: A graph is said to be colorable if two vertices connected by an edge are never of the same color. 2 colorable mean that this can be achieved with just 2 colors.

Short Questions:

1) When is a binary search best applied?

A binary search is an algorithm that is best applied to search a list when the elements are already in order or sorted. The list is searched starting in the middle, such that if that middle value is not the target search key, it will check to see if it will continue the search on the lower half of the list or the higher half. The split and search will then continue in the same manner.

2) What is merge sort?

Merge sort, is a divide-and-conquer approach for sorting the data. In a sequence of data, adjacent ones are merged and sorted to create bigger sorted lists. These sorted lists are then merged again to form an even bigger sorted list, which continues until you have one single sorted list.

3)What is a linear search?

A linear search refers to the way a target key is being searched in a sequential data structure. In this method, each element in the list is checked and compared against the target key. The process is repeated until found or if the end of the file has been reached.

4) What is the advantage of the heap over a stack?

The heap is more flexible than the stack. That's because memory space for the heap can be dynamically allocated and de-allocated as needed. However, the memory of the heap can at times be slower when compared to that stack.

5) How do you insert a new item in a binary search tree?

Assuming that the data to be inserted is a unique value (that is, not an existing entry in the tree), check first if the tree is empty. If it's empty, just insert the new item in the root node. If it's not empty, refer to the new item's key. If it's smaller than the root's key, insert it into the root's left subtree, otherwise, insert it into the root's right subtree.

6) How does a selection sort work for an array?

The selection sort is a fairly intuitive sorting algorithm, though not necessarily efficient. In this process, the smallest element is first located and switched with the element at subscript zero, thereby placing the smallest element in the first position.

The smallest element remaining in the subarray is then located next to subscripts 1 through n-1 and switched with the element at subscript 1, thereby placing the second smallest element in the second position. The steps are repeated in the same manner till the last element.

7)Which sorting algorithm is considered the fastest?

There are many types of sorting algorithms: quick sort, bubble sort, balloon sort, radix sort, merge sort, etc. Not one can be considered the fastest because each algorithm is

designed for a particular data structure and data set. It would depend on the data set that you would want to sort.

8)Give a basic algorithm for searching a binary search tree.

if the tree is empty, then the target is not in the tree, end search
if the tree is not empty, the target is in the tree
check if the target is in the root item
if a target is not in the root item, check if a target is smaller than the root's value
if a target is smaller than the root's value, search the left subtree
else, search the right subtree

9)What is a bubble sort and how do you perform it?

A bubble sort is one sorting technique that can be applied to data structures such as an array. It works by comparing adjacent elements and exchanges their values if they are out of order. This method lets the smaller values "bubble" to the top of the list, while the larger value sinks to the bottom.

10)How does selection sort work?

Selection sort works by picking the smallest number from the list and placing it at the front. This process is repeated for the second position towards the end of the list. It is the simplest sort algorithm.

11)What is a graph?

A graph is one type of data structure that contains a set of ordered pairs. These ordered pairs are also referred to as edges or arcs and are used to connect nodes where data can be stored and retrieved.

12)What is an AVL tree?

An AVL tree is a type of binary search tree that is always in a state of partially balanced. The balance is measured as a difference between the heights of the subtrees from the root. This self-balancing tree was known to be the first data structure to be designed as such.

13)Which data structures are used for BFS and DFS of a graph?
Queue is used for BFS. Stack is used for DFS.DFS can also be implemented using recursion.

14)Define complete a binary tree?
A binary tree in which every non leaf node has exactly two children not necessarily on the same level.It is also called as strictly binary tree.

15)List out few of the application of tree data-structure?

The manipulation of arithmetic expression. Symbol table construction. Syntax analysis.

16)What is a spanning Tree?

A spanning tree is a tree associated with a network. All the nodes of the graph appear on the tree once. A minimum spanning tree is a spanning tree organized so that the total edge weight between nodes is minimized.

17)How does BFS Algorithm Work?

Each vertex or node in the graph is known. ...
In case the vertex V is not accessed then add the vertex V into the BFS Queue.
Start the BFS search, and after completion, Mark vertex V as visited.
The BFS queue is still not empty, hence remove the vertex V of the graph from the queue.

18)How do you use depth first search?

Rule 1 − Visit the adjacent unvisited vertex. Mark it as visited. Display it. Push it in a stack.

Rule 2 − If no adjacent vertex is found, pop up a vertex from the stack. (It will pop up all the vertices from the stack, which do not have adjacent vertices.)

Rule 3 − Repeat Rule 1 and Rule 2 until the stack is empty.

19)Where is radix sort used?

In computer science, radix sort is a non-comparative sorting algorithm. ...
Radix sort can be applied to data that can be sorted lexicographically, be they integers, words, punch cards, playing cards, or the mail.
Radix sort dates back as far as 1887 to the work of Herman Hollerith on tabulating machines.

20)Why it is called radix sort?

The algorithm is named radix sort as it specifies the radix r to be used which changes how the sort is performed. The radix, or base, of the number system is the number of digits that represent a single position in the number; a radix of 2 is binary (0-1), 10 is decimal (0-9), 16 is hexadecimal (0-F) and so on.

21)How quick sort works with example?

A large array is partitioned into two arrays one of which holds values smaller than the specified value, say pivot, based on which the partition is made and another array holds values greater than the pivot value. Quicksort partitions an array and then calls itself recursively twice to sort the two resulting subarrays.

22)Where is merge sort used?

Mergesort is used when we want a guaranteed running time of O (n log n) O(n \log n) O(nlogn), regardless of the state of the input. Mergesort is a stable sort with a space complexity of O (n) O(n) O(n).

23)Is merge sort better than quick?

Merge sort is more efficient and works faster than quick sort in case of larger array size or datasets. Quick sort is more efficient and works faster than merge sort in case of smaller array size or datasets. Sorting method : The quick sort is internal sorting method where the data is sorted in main memory.

24)Why is merge sort so fast?

Merge sort splits the list into two, calls itself recursively to sort both lists, and then merges the two lists into one. ... That means, for sufficiently large input, merge sort will be faster than bubble sort, no matter how much more efficient the bubble sort implementation is.

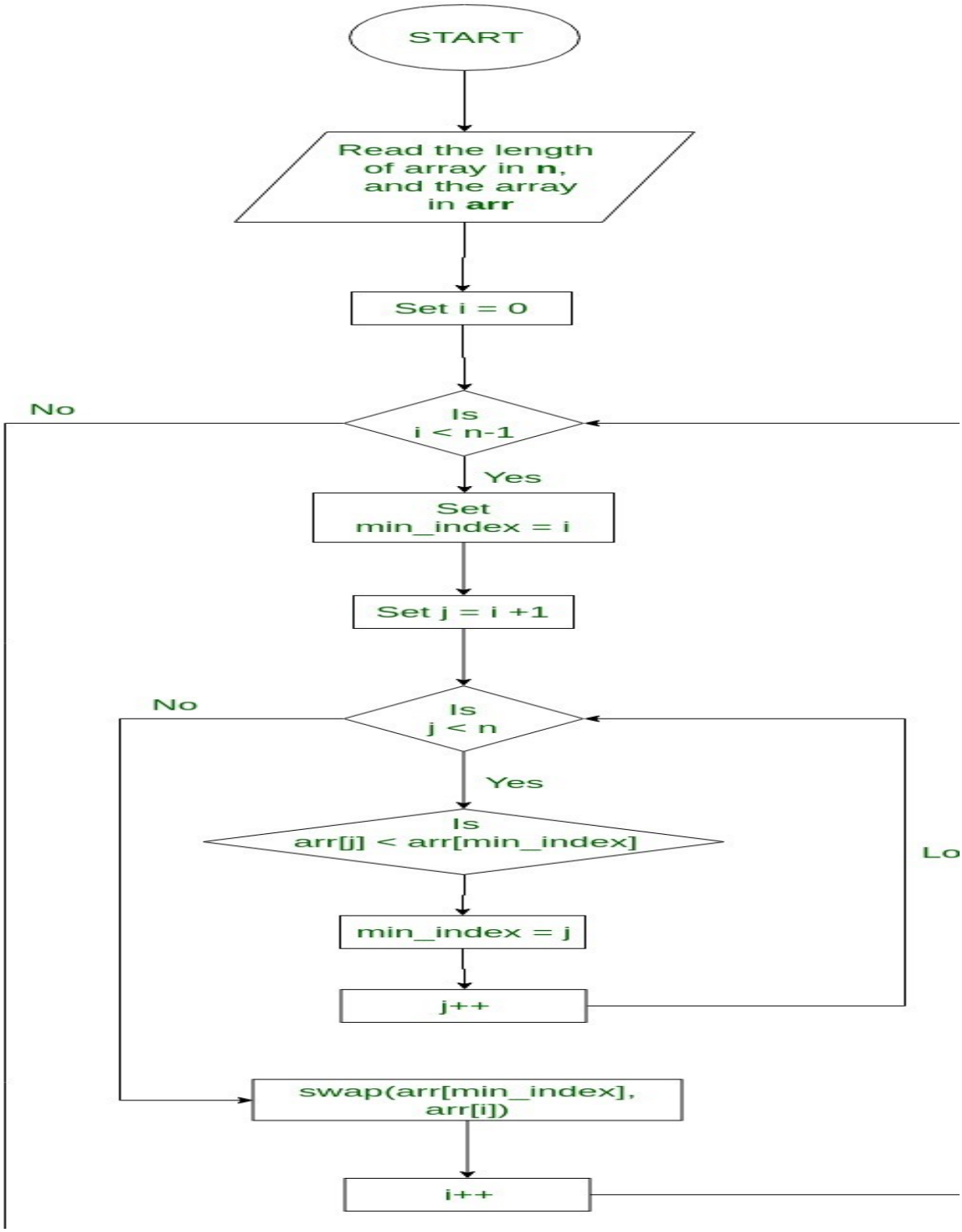
25)How do you form a max heap?

To build a max heap, you:
Assign it a value. Compare the value of the child node with the parent node. Swap nodes if the value of the parent is less than that of either child (to the left or right). Repeat until the largest element is at the root parent nodes.

26)What is min and max heap in data structure?

● A min-heap is a binary tree such that. - the data contained in each node is less than (or equal to) the data in that node's children. - the binary tree is complete. ● A max-heap is a binary tree such that. - the data contained in each node is greater than (or equal to) the data in that node's children.

selection sort (code, flowchart,complexity)



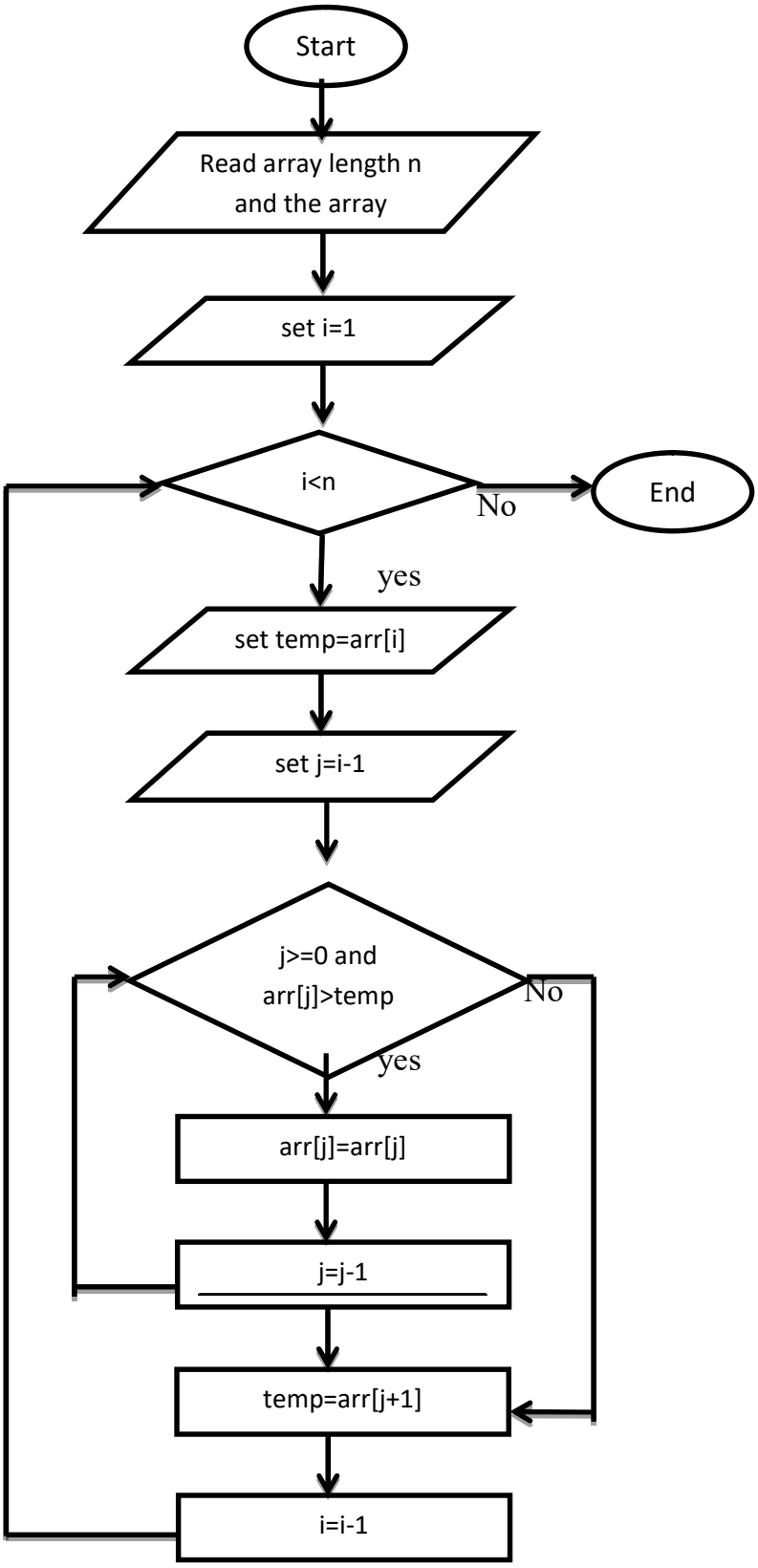
code:

```
#include <stdio.h>

void swap(int *xp, int *yp)
{
   int temp = *xp;
   *xp = *yp;
   *yp = temp;
}

void selectionSort(int arr[], int n)
{
   int i, j, min_idx;

   // One by one move boundary of unsorted subarray
   for (i = 0; i < n-1; i++)
   {
      // Find the minimum element in unsorted array
      min_idx = i;
      for (j = i+1; j < n; j++)
        if (arr[j] < arr[min_idx])
          min_idx = j;

      // Swap the found minimum element with the first element
      swap(&arr[min_idx], &arr[i]);
   }
}

/* Function to print an array */
void printArray(int arr[], int size)
{
   int i;
   for (i=0; i < size; i++)
      printf("%d ", arr[i]);
   printf("\n");
}

// Driver program to test above functions
int main()
{
   int arr[] = {64, 25, 12, 22, 11};
```

```
    int n = sizeof(arr)/sizeof(arr[0]);
    selectionSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```

Time Complexity: $O(n^2)$ as there are two nested loops.

Insertion Sort:

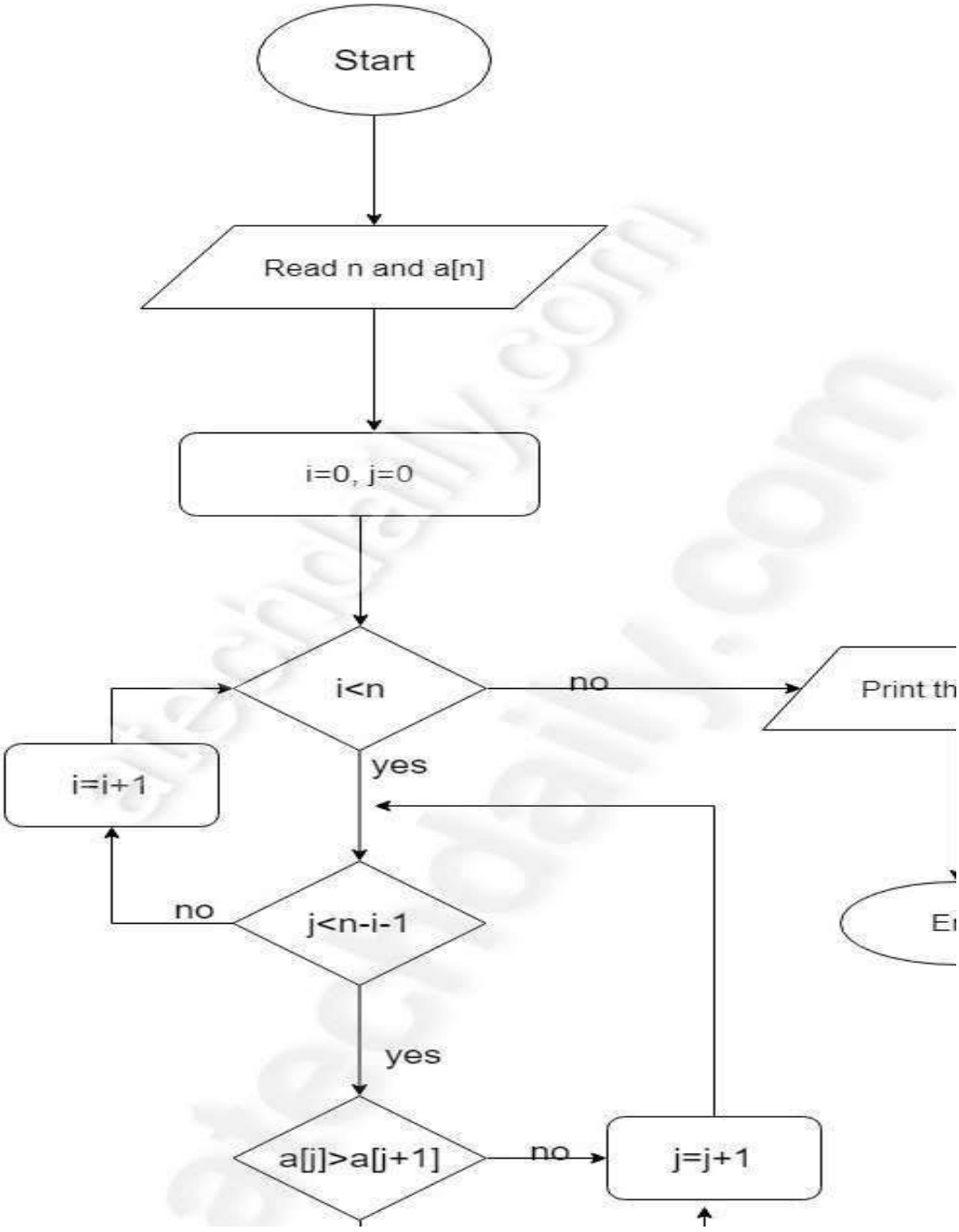


code:

```
#include <math.h>
#include <stdio.h>

/* Function to sort an array using insertion sort*/
void insertionSort(int arr[], int n)
{
   int i, key, j;
   for (i = 1; i < n; i++) {
      key = arr[i];
      j = i - 1;

      /* Move elements of arr[0..i-1], that are
        greater than key, to one position ahead
        of their current position */
      while (j >= 0 && arr[j] > key) {
         arr[j + 1] = arr[j];
         j = j - 1;
      }
      arr[j + 1] = key;
   }
}

// A utility function to print an array of size n
void printArray(int arr[], int n)
{
   int i;
   for (i = 0; i < n; i++)
      printf("%d ", arr[i]);
   printf("\n");
}

/* Driver program to test insertion sort */
int main()
{
   int arr[] = { 12, 11, 13, 5, 6 };
   int n = sizeof(arr) / sizeof(arr[0]);

   insertionSort(arr, n);
   printArray(arr, n);
```

```
    return 0;
}
```
Time Complexity: O(n^2)

Bubble Sort:



Code:

```
#include <stdio.h>

void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)

       // Last i elements are already in place
       for (j = 0; j < n-i-1; j++)
           if (arr[j] > arr[j+1])
              swap(&arr[j], &arr[j+1]);
}

/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
int main()
{
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr)/sizeof(arr[0]);
    bubbleSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
```

}
Worst and Average Case Time Complexity: O(n*n). Worst case occurs when array is reverse sorted.
Best Case Time Complexity: O(n). Best case occurs when array is already sorted.

Merge Sort:

code:

```
#include <stdio.h>
#include <stdlib.h>

// Merges two subarrays of arr[].
// First subarray is arr[l..m]
// Second subarray is arr[m+1..r]
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    /* create temp arrays */
    int L[n1], R[n2];

    /* Copy data to temp arrays L[] and R[] */
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    /* Merge the temp arrays back into arr[l..r]*/
    i = 0; // Initial index of first subarray
    j = 0; // Initial index of second subarray
    k = l; // Initial index of merged subarray
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
```

```
            j++;
        }
        k++;
    }

    /* Copy the remaining elements of L[], if there
    are any */
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
    are any */
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

/* l is for left index and r is right index of the
sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l + (r - l) / 2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}

/* UTILITY FUNCTIONS */
/* Function to print an array */
void printArray(int A[], int size)
{
```

```
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", A[i]);
    printf("\n");
}

/* Driver code */
int main()
{
    int arr[] = { 12, 11, 13, 5, 6, 7 };
    int arr_size = sizeof(arr) / sizeof(arr[0]);

    printf("Given array is \n");
    printArray(arr, arr_size);

    mergeSort(arr, 0, arr_size - 1);

    printf("\nSorted array is \n");
    printArray(arr, arr_size);
    return 0;
}
```

Time complexity of Merge Sort is θ(nLogn) in all 3 cases (worst, average and best) as merge sort always divides the array into two halves and takes linear time to merge two halves.

Quick sort:

code:

```
#include<stdio.h>
void quicksort(int number[25],int first,int last){
   int i, j, pivot, temp;

   if(first<last){
      pivot=first;
      i=first;
      j=last;

      while(i<j){
```

```
      while(number[i]<=number[pivot]&&i<last)
        i++;
      while(number[j]>number[pivot])
        j--;
      if(i<j){
        temp=number[i];
        number[i]=number[j];
        number[j]=temp;
      }
    }

    temp=number[pivot];
    number[pivot]=number[j];
    number[j]=temp;
    quicksort(number,first,j-1);
    quicksort(number,j+1,last);

  }
}

int main(){
  int i, count, number[25];

  printf("How many elements are u going to enter?: ");
  scanf("%d",&count);

  printf("Enter %d elements: ", count);
  for(i=0;i<count;i++)
    scanf("%d",&number[i]);

  quicksort(number,0,count-1);

  printf("Order of Sorted elements: ");
  for(i=0;i<count;i++)
    printf(" %d",number[i]);

  return 0;
}
```

Quicksort Complexity

Time Complexity	
Best	O(n*log n)
Worst	$O(n^2)$
Average	O(n*log n)
Space Complexity	O(log n)