Source: Interview Bit

1. **How is an array initialized in C language?**

   int a[3] = {1, 2, 3};

   int a = {1, 2, 3};

   int a[] = new int[3]

   int a(3) = [1, 2, 3];

2.Which of the following is a linear data structure?

   Array

   AVL Trees

   Binary Trees

   Graphs

3. **How is the 2nd element in an array accessed based on pointer notation?**

   *a + 2

   ***(a + 2)** **[a[2] is equivalent to *(a + 2) in pointer notation.]**

   *(*a + 2)

   &(a + 2)

4. **Which of the following is not the type of queue?**

   Priority queue

   Single-ended queue

   Circular queue

   Ordinary queue

5. **From following which is not the operation of data structure?**

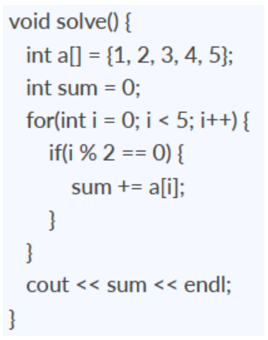
   Operations that manipulate data in some way

   Operations that perform a computation

   Operations that check for syntax errors

   Operations that monitor an object for the occurrence of a controlling event

6. **What will be the output of the following code snippet?**



```
void solve() {
  int a[] = {1, 2, 3, 4, 5};
  int sum = 0;
  for(int i = 0; i < 5; i++) {
    if(i % 2 == 0) {
      sum += a[i];
    }
  }
  cout << sum << endl;
}
```

// **9**

7. What is the disadvantage of array data structure?

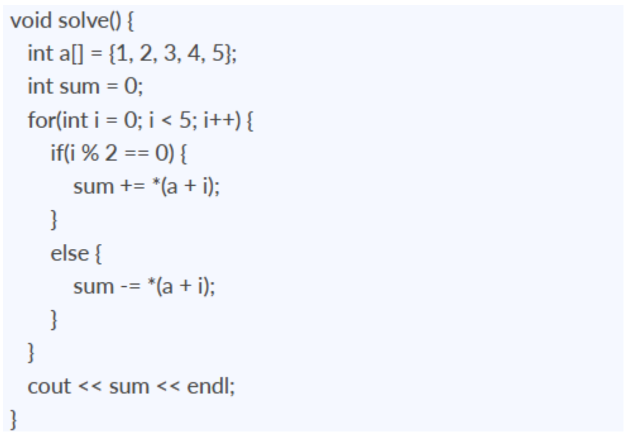
The amount of memory to be allocated should be known beforehand.

Elements of an array can be accessed in constant time.

Elements are stored in contiguous memory blocks.

Multiple other data structures can be implemented using arrays.

8. **What will the output of the following code snippet?**



```
void solve() {
  int a[] = {1, 2, 3, 4, 5};
  int sum = 0;
  for(int i = 0; i < 5; i++) {
     if(i % 2 == 0) {
        sum += *(a + i);
     }
     else {
        sum -= *(a + i);
     }
  }
  cout << sum << endl;
}
```

3

This is an example of accessing an array element through pointer notation. So *(a + i) is equivalent to a[i], and we are adding all the even indices elements, and subtracting all the elements at odd indices, which gives a result of 3.

9. How are String represented in memory in C?

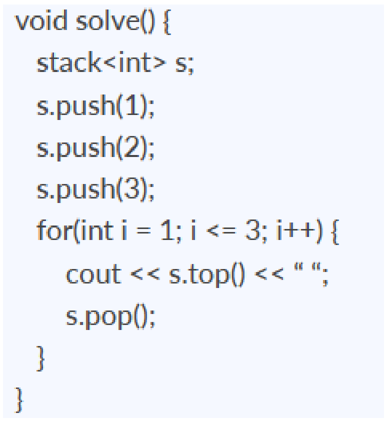
An array of characters.

The object of some class.

Same as other primitive data types.

LinkedList of characters.

10. **What is the output of the following code snippet?**



```
void solve() {
  stack<int> s;
  s.push(1);
  s.push(2);
  s.push(3);
  for(int i = 1; i <= 3; i++) {
     cout << s.top() << “ “;
     s.pop();
  }
}
```

3 2 1

Stack follows a last in first out methodology, so the elements are popped out in reverse order.

11. **Which of the following is the advantage of the array data structure?**

Elements of mixed data types can be stored.

Easier to access the elements in an array

Index of the first element starts from 1.

Elements of an array cannot be sorted.

Answer - B) Elements in an array are stored in a contiguous block of memory. So, easier to access the elements in an array

12. **What function is used to append a character at the back of a string in C++?**

push_back()

append()

push()

insert()

13. **When a pop() operation is called on an empty queue, what is the condition called?**

Overflow

Underflow

Syntax Error

Garbage Value

14. **Which one of the following is an application of queue data structure**

When a resource is shared among multiple consumers.

When data is transferred asynchronously

Load Balancing

All of the above

15. Which of the following data structures can be used to implement queues?

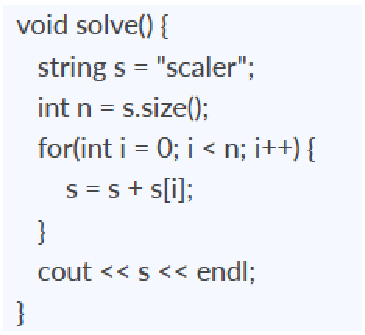
Stack

Arrays

LinkedList

All of the Above [Stack, Arrays, and LinkedList can be used to implement Queues so all the options are correct.]

16. What is the time complexity of the following code snippet in C++?



```
void solve() {
   string s = "scaler";
   int n = s.size();
   for(int i = 0; i < n; i++) {
      s = s + s[i];
   }
   cout << s << endl;
}
```

The s = s + s[i] line first makes a copy of the original string and then appends the new character in it, leading to each operation being O(n). So the total time complexity is **O(n^2).**

17. Which of the following data structures finds its use in recursion?

Stack

Arrays

LinkedList

Queues

18. Which of the following data structures allow insertion and deletion from both ends?

Stack

Deque

Queue

Strings

19. **What will be the output of the following code snippet?**

```
void solve() {
  deque<int> dq;
  for(int i = 1; i <= 5; i++) {
     if(i % 2 == 0) {
        dq.push_back(i);
     }
     else {
        dq.push_front(i);
     }
  }
  for(auto x: dq) {
     cout << x << " ";
  }
  cout << endl;
}
```

5 3 1 2 4

The push_back() function emplaces an element at the back of the deque, and the push_front() function emplaces an element at the front of the deque. In the code snippet, we are alternatively pushing at the front and back of the deque, before printing its contents.

20. **Which of the following sorting algorithms provide the best time complexity in the worst-case scenario?**

Merge Sort

Quick Sort

Bubble Sort

Selection Sort

Answer: Merge Sort will always have a time complexity of O(n * logn) which is the best in the worst case among these algorithms.

21. **What is the maximum number of swaps that can be performed in the Selection Sort algorithm?**

n - 1

n

1

n – 2

answer: n - 1 swaps are performed at max to sort any array by Selection Sort.

22. **Which of the following is a Divide and Conquer algorithm?**

Bubble Sort

Selection Sort

Heap Sort

Merge Sort

23. **What will be the best sorting algorithm, given that the array elements are small (<= 1e6)?**

Bubble Sort

Merge Sort

Counting Sort

Heap Sort

Answer: Counting sort sorts an array in O(n) time complexity, taking up an extra space complexity of O(max(a[i]))..

24. **Which of the following are applications of Topological Sort of a graph?**

Sentence Ordering.

Course Scheduling.

OS Deadlock Detection.

All of the above.

25. **Which of the following is known to be not an NP-Hard Problem?**

Vertex Cover Problem.

0/1 Knapsack Problem.

Maximal Independent Set Problem.

Travelling Salesman Problem.

26. **Which of the following algorithms are used for string and pattern matching problems?**

Z Algorithm

Rabin Karp Algorithm

KMP Algorithm

All of the above

27. **Consider we have a function, getLCA(), which returns us the Lowest Common Ancestor between 2 nodes of a tree. Using this getLCA() function, how can we calculate the distance between 2 nodes, given that distance from the root, to each node is calculated?**

dist(u) + dist(v) - 2 * dist(getLCA(u, v))

dist(u) + dist(v) + 2 * dist(getLCA(u, v))

dist(u) + dist(v)

dist(u) + dist(v) - dist(getLCA(u, v))

The distance between 2 nodes, can be broken down into the sum of distances from the root, to each of the nodes. Observe, that in each of these paths, the path from the root to the LCA comes in each of them. So, this needs to be removed from the answer, and hence we get our formula in Option (A).

28. **Which of the following algorithms are useful for processing queries on trees?**

Centroid Decomposition.

Heavy Light Decomposition.

Both (A) and (B).

Neither (A) nor (B).

29. **Consider the following code snippet:**

```
void solve(vector<int> &a) {
  int queries;
  cin >> queries;
  while(queries--) {
     int type;
     cin >> type;
     if(type == 1) {
        int index, value;
        cin >> index >> value;
        update(a, index, value);
     }
     else {
        int l, r;
        cin >> l >> r;
        cout << getXOR(a, l, r) << endl;
     }
  }
}
```

The update() function updates the element at the given index in the array to some given value. The getXOR() function returns the XOR of the elements in the array a, in the range [l, r]. Which of the following data structures can perform the above tasks optimally?

Segment Trees.

Prefix XOR Arrays.

Tries.

Stacks.

30. **What will the output of the following code snippet be?**

```
void solve() {
  vector<int> a = {1, 2, 3, 4, 5};
  sort(a.begin(), a.end(), [&](const int &x, const int &y) {
     return x % 2 < y % 2;
  });
  for(int x: a) {
     cout << x << " ";
  }
  cout << endl;
}
```

2 4 1 3 5

The above code snippet sorts the array, based on a custom comparator. The comparator sorts the array based on the rule, the elements with even parities, have higher priority than the elements with odd parities.

31. **What is the time complexity of the binary search algorithm?**

O(n)

O(1)

O(log2n)

O(n^2)

32. **Kruskal's Algorithm for finding the Minimum Spanning Tree of a graph is a kind of a?**

DP Problem.

Greedy Algorithm.

Adhoc Problem.

None of the above.

Answer: Kruskal's Algorithm works on the greedy algorithm of taking the lowest weight edges in the MST of a graph unless it forms a cycle.

33. **What will be the output of the following code snippet?**

```
void solve() {
  string s = "00000001111111";
  int l = 0, r = s.size() - 1, ans = -1;
  while(l <= r) {
     int mid = (l + r) / 2;
     if(s[mid] == '1') {
        ans = mid;
        r = mid - 1;
     }
     else {
        l = mid + 1;
     }
  }
  cout << ans << endl;
}
```

7

This code snippet shows one of the many applications of binary search. Here, we are binary searching for the index of the first occurrence of the character '1' in the given string. When we get the character '1' at the mid index, we store it as the answer and move to the left half which will have the first index of '1' if it occurs. Else we move to the right half. So, the answer will be 7 (0-based indexing).

34. **Maps in C++ are implemented using which of the following data structures?**

Red-Black Trees.

Binary Search Trees.

AVL Trees.

Hash Tables.

35. **What will be the output of the following code snippet?**

```
void solve() {
  int n = 24;
  int l = 0, r = 100, ans = n;
  while(l <= r) {
     int mid = (l + r) / 2;
     if(mid * mid <= n) {
        ans = mid;
        l = mid + 1;
     }
     else {
        r = mid - 1;
     }
  }
  cout << ans << endl;
}
```

The code snippet basically uses binary search to calculate the floor of the square root of a number. Since the square root is an increasing function, so binary search is applicable here. Here, for n = 24, **the answer is 4.**

36. **What is the time complexity of the Sieve of Eratosthenes to check if a number is prime?**

O(nlog(logn)) Precomputation, O(1) for check.

O(n) Precomputation, O(1) for the check.

O(n * logn) Precomputation, O(logn) for check.

O(n) Precomputation, O(logn) for check.

Answer: The Sieve of Eratosthenes checks if a number is prime in O(1) in the range [1, n] by using an O(nlog(logn)) precomputation and O(n) space complexity.

37. **What will be the output of the following code snippet?**

```
int search(int l, int r, int target, vector<int> &a) {
  int mid = (l + r) / 2;
  if(a[mid] == target) {
     return mid;
  }
  else if(a[mid] < target) {
     return search(mid + 1, r, target, a);
  }
  else {
     return search(0, mid - 1, target, a);
  }
}
void solve() {
  vector<int> a = {1, 2, 3, 4, 5};
  cout << search(0, 4, 4, a) << endl;
}
```

The above code is just a recursive implementation of binary search which searches for the index of value 4, which is **3**.

38. **What is the best case time complexity of the binary search algorithm?**

O(1)

O(n)

O(log2n)

O(n^2)

The best-case time complexity occurs when the target element occurs exactly at the middle of the array and the algorithm terminates in 1 move.

39. **What is the time complexity to insert an element to the front of a LinkedList(head pointer given)?**

O(n)

O(1)

O(logn)

O(n * logn)

We set the next node to the head of the list, and then return that node as the new head.

40. **What is the time complexity to insert an element to the rear of a LinkedList(head pointer given)?**

O(n)

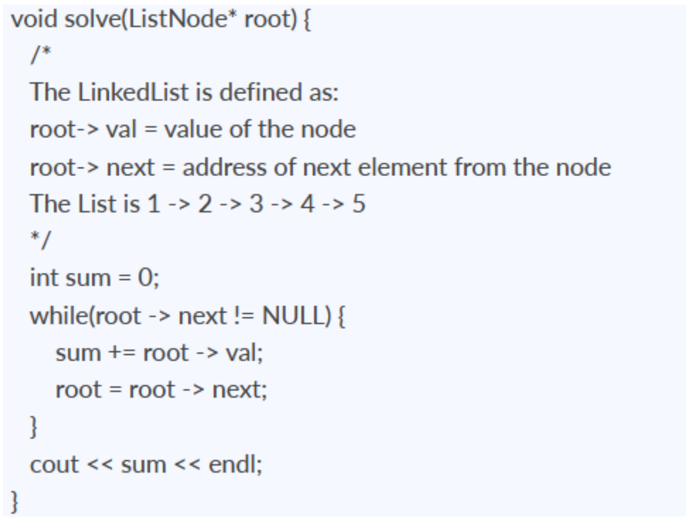
O(1)

O(logn)

O(n * logn)

We need to traverse to the end of the LinkedList and set it next to the new element. So, the traversal will take O(n) time complexity.

41. **What will be the value of “sum” after the following code snippet terminates?**



```
void solve(ListNode* root) {
  /*
  The LinkedList is defined as:
  root-> val = value of the node
  root-> next = address of next element from the node
  The List is 1 -> 2 -> 3 -> 4 -> 5
  */
  int sum = 0;
  while(root -> next != NULL) {
     sum += root -> val;
     root = root -> next;
  }
  cout << sum << endl;
}
```

10

42. **Which of the following can be done with LinkedList?**

Implementation of Stacks and Queues

Implementation of Binary Trees

Implementation of Data Structures that can simulate Dynamic Arrays

All of the above

43. What is the information, which a LinkedList's Node must store?

The address of the next node if it exists

The value of the current node

Both (A) and (B)

None of the above

44. **What is the maximum number of children a node can have in an n-ary tree?**

2

0

1

n

An n-ary tree can have at most n children.

45. Worst case time complexity to access an element in a BST can be?

O(n) (In the worst case, we might need to visit all the nodes in the BST.)

O(n * logn)

O(1)

O(logn)

46. **Which of the following represents the Postorder Traversal of a Binary Tree?**

Left -> Right -> Root

Left -> Root -> Right

Right -> Left -> Root

Right -> Root -> Left

47. **In what time complexity can we find the diameter of a binary tree optimally?**

O(V + E)

O(V)

O(E)

O(V * logE)

We can compute the diameter of a binary tree using a single DFS, which takes the time of O(V + E).

48. **Which of the following statements is true about AVL Trees?**

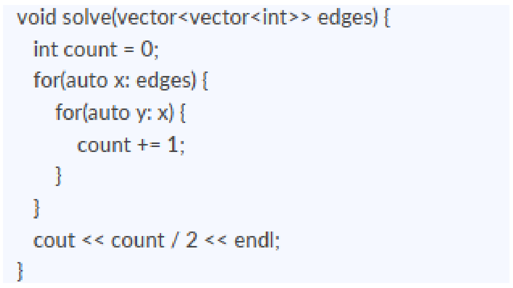
The difference between the heights of left and right nodes cannot be more than 1.

The height of an AVL Tree always remains of the order of O(logn)

AVL Trees are a type of self-balancing Binary Search Trees.

All of the above.

49. **What does the following code snippet calculate (edges represent the adjacency list representation of a graph)?**



```
void solve(vector<vector<int>> edges) {
  int count = 0;
  for(auto x: edges) {
     for(auto y: x) {
        count += 1;
     }
  }
  cout << count / 2 << endl;
}
```

The above code snippet calculates the number of edges in an undirected graph.

50. **In a graph of n nodes and n edges, how many cycles will be present?**

Exactly 1

At most 1

At most 2

Depends on the graph

Answer: A tree contains by definition n nodes and n - 1 edges, and it is an acyclic graph. When we add 1 edge to the tree, we can form exactly one cycle by adding this edge.

51. **A node in a tree, such that removing it splits the tree into forests, with size of each connected component being not greater than n / 2 is called?**

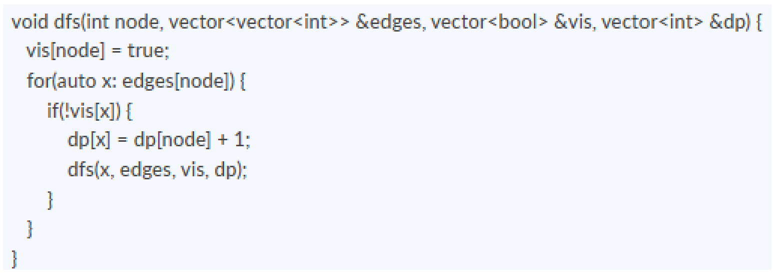
Center

Diameter

Centroid

Path

52. **What does the following code snippet do?**



```
void dfs(int node, vector<vector<int>> &edges, vector<bool> &vis, vector<int> &dp) {
  vis[node] = true;
  for(auto x: edges[node]) {
     if(!vis[x]) {
        dp[x] = dp[node] + 1;
        dfs(x, edges, vis, dp);
     }
  }
}
```

The above code snippets finds the depth of each node, in the tree, with respect to some root node, and stores them in the dp array, using a dfs traversal on the tree.

53. **Which of the following algorithms are used to find the shortest path from a source node to all other nodes in a weighted graph?**

BFS.

Djikstra's Algorithm.

Prims Algorithm.

Kruskal's Algorithm.

Answer: [The Djikstra's algorithm is used to find the shortest path from a source node to all other nodes in a weighted graph.]

54. **What is the best time complexity we can achieve to precompute all-pairs shortest paths in a weighted graph?**

O(n^3)

O(n^2)

O(n)

O(n^4)

The Floyd-Warshall Algorithm computes All Pairs Shortest Paths in a weighted graph, in **O(n^3)**.

55. **Which data structure is mainly used for implementing the recursive algorithm?**

Queue

Stack

Array

List

Algorithms:

Source Java Tutorials:

1. Which IDE is recommended for developing Java applications?

A. Eclipse

B. Notepad

C. Visual Studio

D. Sublime Text

2. What is necessary to run a Java program?

A. Java Runtime Environment (JRE)

B. Java Development Kit (JDK)

C. Java Compiler

D. All of the above

3. What does JDK stand for?

A. Java Development Kit

B. Java Default Kit

C. Java Data Kit

D. Java Document Kit

4. Which command is used to compile a Java program?

A. java

B. javac

C. compile

D. run

5. What is the primary purpose of algorithms in data structures?

A. To store data

B. To perform calculations

C. To manipulate data efficiently

D. To visualize data

Answer (C): To manipulate data efficiently
Algorithms are designed to manipulate data efficiently, allowing for effective data retrieval and processing.

6. What does the time complexity of an algorithm describe?

A. The amount of memory used

B. The speed of the algorithm

C. The number of steps to complete

D. The efficiency of coding

Answer (C): The number of steps to complete
Time complexity measures how the execution time of an algorithm increases with the size of the input.

7. Which data structure uses LIFO (Last In, First Out) principle?

A. Queue

B. Stack

C. Array

D. LinkedList

Answer (B): Stack
A stack is a data structure that follows the LIFO principle, where the last element added is the first to be removed.

8. In which scenario would you use a binary search algorithm?

A. Unsorted data

B. Sorted data

C. Large datasets only

D. Data with duplicates

Answer (B): Sorted data
Binary search is an efficient algorithm that requires the dataset to be sorted before it can be applied.

9. What is a data structure?

A. A way to organize data

B. A type of algorithm

C. A programming language

D. None of the above

Answer (A): A way to organize data
A data structure is a way to organize and store data in a computer so that it can be accessed and modified efficiently.

10. What is the main purpose of using a stack?

- **A.** To store data permanently
- **B.** To manage function calls
- **C.** To retrieve data in random order
- **D.** To sort data

Answer (B): To manage function calls
Stacks are primarily used to manage function calls and maintain the order of operations in programming.

11. Which data structure uses First In First Out (FIFO) principle?

- **A.** Array
- **B.** Stack
- **C. Queue**
- **D.** Linked List

Answer (C): Queue
A queue operates on the FIFO principle, where the first element added is the first one to be removed.

12. How do you declare an array of integers in Java?

- **A.** int[] arr;
- **B.** array int arr;
- **C.** int arr[];
- **D. Both int[] arr; and int arr[];**

Answer (D): Both int[] arr; and int arr[];
Both int[] arr; and int arr[]; are valid syntax for declaring an integer array in Java.

13. Can an array in Java store objects?

- **A. Yes**
- **B.** No
- **C.** Only primitive types
- **D.** Only strings

Answer (A): Yes
Yes, an array in Java can store objects, which can include instances of classes.

14. What is the primary advantage of linked lists over arrays?

- **A. Better memory utilization**
- **B.** Faster access times
- **C.** Easier to implement
- **D.** Fixed size

Answer (A): Better memory utilization
The primary advantage of linked lists over arrays is better memory utilization, as linked lists can grow and shrink dynamically.

15. Which method is used to insert a new node at the end of a doubly linked list?

A. insertAtBeginning()

B. insertAtEnd()

C. deleteNode()

D. displayList()

Answer (B): insertAtEnd()
The method used to insert a new node at the end of a doubly linked list is called insertAtEnd().

16. Which method is used to add an element to the stack in Java?

A. push()

B. add()

C. insert()

D. append()

Answer (A): push()
The push() method is used to add elements to the top of the stack.

17. What will happen if you try to pop an element from an empty stack?

A. Return null

B. Throw an exception

C. Return zero

D. Do nothing

Answer (B): Throw an exception
Attempting to pop from an empty stack will throw an EmptyStackException.

18. What is the primary purpose of parsing expressions in Java?

A. To execute code

B. To analyze and evaluate mathematical expressions

C. To compile Java programs

D. To manage memory

Answer (B): To analyze and evaluate mathematical expressions
Parsing expressions allows for the analysis and evaluation of mathematical expressions within Java programs.

19. Which data structure is commonly used to evaluate expressions in Java?

A. Array

B. Linked List

C. Stack

D. Queue

Answer (C): Stack
Stacks are commonly used to evaluate expressions due to their Last In First Out (LIFO) nature.

20. What does an expression tree represent?

A. A graphical representation of data structures

B. A hierarchy of operations and operands

C. The flow of control in a program

D. A database schema

Answer (B): A hierarchy of operations and operands
An expression tree represents a hierarchical structure of operations and operands in an expression.

21. In the context of parsing, what does the term 'shunting yard algorithm' refer to?

A. A method for sorting data

B. An algorithm for converting infix expressions to postfix

C. A technique for searching data

D. An algorithm for optimizing code

Answer (B): An algorithm for converting infix expressions to postfix
The shunting yard algorithm is used to convert infix expressions to postfix notation, making them easier to evaluate.

22. Which Java class is typically used to handle exceptions during expression parsing?

A. IllegalArgumentException

B. ExpressionParseException

C. NullPointerException

D. ArithmeticException

Answer (B): ExpressionParseException
A custom exception like ExpressionParseException can be created to handle parsing errors specifically.

23. Which of the following operations is NOT typically associated with a Queue?

A. Enqueue

B. Dequeue

C. Peek

D. Sort

Answer (D): Sort
Sorting is not an operation typically associated with a Queue; it primarily supports Enqueue, Dequeue, and Peek operations.

24. Which Java class is commonly used to implement a Queue?

A. ArrayList

B. LinkedList

C. HashMap

D. Stack

Answer (B): LinkedList
The LinkedList class in Java implements the Queue interface, allowing for efficient queue operations.

25. In a circular queue, what is the advantage over a linear queue?

A. Less memory usage

B. No overflow

C. Better performance

D. No underflow

Answer (A): Less memory usage
A circular queue uses memory more efficiently than a linear queue, reducing the chance of overflow when there is available space.

26. What is a priority queue?

A. A queue where elements are processed in FIFO order

B. A queue where each element has a priority

C. A data structure that stores only integers

D. A queue that does not allow duplicates

Answer (B): A queue where each element has a priority
A priority queue is a data structure where each element has a priority and elements are processed based on their priority.

27. Which class in Java is used to implement a priority queue?

A. ArrayList

B. LinkedList

C. PriorityQueue

D. HashMap

Answer (C): PriorityQueue
The PriorityQueue class in Java is used to implement a priority queue.

28. What is the time complexity of adding an element to a priority queue?

A. O(1)

B. O(log n)

C. O(n)

D. O(n log n)

Answer (B): O(log n)
Adding an element to a priority queue has a time complexity of O(log n).

29. Which method is used to remove the highest priority element from a priority queue?

A. remove()

B. poll()

C. dequeue()

D. extract()

Answer (B): poll()
The poll() method is used to remove and return the highest priority element from a priority queue.

30.Can a priority queue contain duplicate elements?

A. Yes

B. No

C. Only if they have the same priority

D. Depends on implementation

Answer (A): Yes
Yes, a priority queue can contain duplicate elements regardless of their priority.

31. What is a tree in data structures?

A. A linear structure

B. A hierarchical structure

C. A graph structure

D. A flat structure

Answer (B): A hierarchical structure
A tree is a hierarchical structure where each node has a parent and can have multiple children.

32. Which of the following is a characteristic of a binary tree?

A. Each node has at most two children

B. Each node has any number of children

C. Nodes are arranged in a linear fashion

D. None of the above

Answer (A): Each node has at most two children
In a binary tree, each node has at most two children, referred to as the left and right child.

33. What is the maximum number of nodes in a binary tree of height h?

A. h

B. 2^h - 1

C. 2h

D. h^2

Answer (B): 2^h - 1
A binary tree of height h can have a maximum of 2^h - 1 nodes.

34. What is a balanced tree?

A. A tree with an equal number of nodes on each side

B. A tree where the height difference between left and right subtrees is at most 1

C. A tree that is always complete

D. None of the above

Answer (B): A tree where the height difference between left and right subtrees is at most 1
A balanced tree is one where the height difference between the left and right subtrees for any node is at most 1.

35. What is a hash table?

A. A data structure that uses key-value pairs

B. A type of array

C. A function for sorting data

D. A method for searching in databases

Answer (A): A data structure that uses key-value pairs
A hash table is a data structure that stores data in key-value pairs for efficient retrieval.

36. What is the primary advantage of using a hash table?

A. Fast data retrieval

B. Easy implementation

C. Low memory usage

D. Ordered data storage

Answer (A): Fast data retrieval
The primary advantage of using a hash table is fast data retrieval, as it allows for average-case constant time complexity for lookups.

37. What happens when two keys hash to the same index in a hash table?

A. The second key is ignored

B. An error occurs

C. Collision handling mechanisms are used

D. The first key is overwritten

Answer (C): Collision handling mechanisms are used
When two keys hash to the same index, collision handling mechanisms, such as chaining or open addressing, are used.

38. What is the load factor in the context of hash tables?

A. The maximum number of elements allowed

B. The ratio of stored elements to the table size

C. The speed of retrieval

D. The number of collisions occurred

Answer (B): The ratio of stored elements to the table sizes
The load factor is defined as the ratio of the number of stored elements to the total capacity of the hash table.

39. What is a heap data structure?

A. A binary tree

B. A complete binary tree

C. A balanced binary tree

D. An unordered list

Answer (B): A complete binary tree
A heap is a complete binary tree that satisfies the heap property.

40. What are the two types of heaps mentioned in the tutorial?

A. Min-Heap and Max-Heap

B. Binary-Heap and Fibonacci-Heap

C. Max-Heap and AVL-Heap

D. Min-Heap and Binary-Heap

Answer (A): Min-Heap and Max-Heap
The two types of heaps are Min-Heap and Max-Heap.

41. Which operation is used to add a new element to the heap?

A. Insert

B. Push

C. Add

D. Enqueue

Answer (A): Insert
The operation used to add a new element to the heap is called Insert.

42.What is the time complexity for extracting the maximum element from a Max-Heap?

A. O(1)

B. O(log n)

C. O(n)

D. O(n log n)

Answer (B): O(log n)
The time complexity for extracting the maximum element from a Max-Heap is O(log n).

43. In a Min-Heap, which property is maintained?

A. Parent nodes are less than or equal to their children

B. Parent nodes are greater than or equal to their children

C. All nodes are sorted in ascending order

D. All nodes are sorted in descending order

Answer (A): Parent nodes are less than or equal to their children
In a Min-Heap, every parent node is less than or equal to its child nodes.

44. What is a graph in data structures?

A. A collection of nodes and edges

B. A linear data structure

C. A type of tree

D. None of the above

Answer (A): A collection of nodes and edges
A graph is defined as a collection of nodes (vertices) and edges connecting them.

45.Which of the following is NOT a type of graph?

- **A.** Directed graph
- **B.** Undirected graph
- **C.** Weighted graph
- **D. Sequential graph**

Answer (D): Sequential graph
Sequential graph is not a standard type of graph in data structures.

46. What is the primary use of graphs?

- **A.** To represent hierarchical data
- **B. To model pairwise relationships**
- **C.** To store data in sorted order
- **D.** To perform arithmetic operations

Answer (B): To model pairwise relationships
Graphs are primarily used to model pairwise relationships between entities.

47. Which algorithm can be used to traverse a graph?

- **A.** Binary Search
- **B. Depth First Search**
- **C.** Bubble Sort
- **D.** Quick Sort

Answer (B): Depth First Search
Depth First Search (DFS) is one of the algorithms used to traverse or search through a graph.

48. What does an edge in a graph represent?

- **A. A connection between two vertices**
- **B.** A node in the graph
- **C.** The weight of the graph
- **D.** None of the above

Answer (A): A connection between two vertices
An edge in a graph represents a connection between two vertices (nodes).

49. Which data structure is commonly used to implement graphs?

- **A.** Array
- **B.** Linked List
- **C.** Hash Table
- **D. Both Array and Linked List**

Answer (D): Both Array and Linked List
Graphs can be implemented using both arrays (adjacency matrix) and linked lists (adjacency list).

50. Which of the following is a comparison-based sorting algorithm?

A. Bubble Sort

B. Counting Sort

C. Radix Sort

D. Bucket Sort

Answer (A): Bubble Sort
Bubble Sort is a comparison-based sorting algorithm that repeatedly steps through the list and swaps adjacent elements if they are in the wrong order.

51. What is the time complexity of Quick Sort in the average case?

A. O(n log n)

B. O(n^2)

C. O(log n)

D. O(n)

Answer (A): O(n log n)
The average case time complexity of Quick Sort is O(n log n).

52. What is the main advantage of using Merge Sort?

A. It is faster than other algorithms

B. It is stable and works well with large datasets

C. It requires less space

D. It is easier to implement

Answer (B): It is stable and works well with large datasets
Merge Sort is stable and performs well with large datasets due to its O(n log n) time complexity.

53. What is recursion in programming?

A. A method of sorting data

B. A technique where a function calls itself

C. A way to optimize loops

D. A type of data structure

Answer (B): A technique where a function calls itself
Recursion is a programming technique where a function calls itself to solve a problem.

54. Which of the following is a base case in recursion?

A. The case that ends the recursion

B. The case that starts the recursion

C. A loop within the recursion

D. An error handling case

Answer (A): The case that ends the recursion
The base case is the condition under which the recursion will stop.

55. What will happen if a recursive function does not have a base case?

A. The program will run successfully

B. It will cause a stack overflow

C. It will return null

D. It will execute faster

Answer (B): It will cause a stack overflow
Without a base case, the recursive function will continue to call itself indefinitely, leading to a stack overflow.

Source : Scholar Hat

1.Which algorithm stops the execution when it finds the solution otherwise start the problem from the top?

1. **Backtracking**
2. Divide and conquer
3. Branch and Bound
4. Dynamic programming

Ans. Backtracking solves the problem recursively and removes the solution if it does not satisfy the problem constraints. Whenever a solution fails we trace back to the failure point, build on the next solution, and continue this process till we find the solution or all possible solutions are looked after.

2. The data structure used to check whether an expression contains a balanced parenthesis is?

1. Queue
2. **Stack**
3. Tree
4. Array

Ans. Stack works according to the LIFO principle. Open parenthesis are pushed into the stack, and closed parenthesis pop out elements until the top element of the stack is its corresponding open parenthesis. If the stack is empty, the parenthesis are balanced.

3. A one-dimensional array containing one-dimensional arrays is called

1. **Two-dimensional array**
2. Multi-casting array
3. Multi-dimensional array
4. Three-dimensional array

4. The complexity of the average case of an algorithm is

1. **more complicated to analyze than the worst-case**
2. Much simpler to analyze than the worst-case
3. Sometimes more complicated and some other times simpler than the worst-case
4. None of the above

5. The time complexity of the dequeue operation in a queue is

1. **O(1)**
2. O(n)
3. O(long)
4. O(n logn)

Ans. The dequeue operation involves removing the front element and updating the front pointer.

6. Which of the following data structures does the Tower of Hanoi algorithm use?

1. Queue
2. Linked List
3. Heap
4. **Stack**

7. A binary search algorithm cannot be applied to

1. **Sorted linked list**
2. Sorted linear array
3. Sorted binary tree
4. Pointer array

Ans. A binary search algorithm cannot be efficiently applied to a sorted linked list because it relies on random access to elements, which is a key feature of arrays but not linked lists.

8. What is an AVL tree?

1. an unbalanced and height-balanced tree
2. **a balanced and height-balanced tree**
3. a tree with at most 3 children
4. a tree with three children

Ans. AVL tree is a popular self-balancing binary search tree where the difference between the heights of left and right subtrees for any node does not exceed one. It automatically adjusts its structure to maintain the minimum possible height after any operation with the help of a balance factor for each node.

9. In a max-heap, the element with the greatest key is always in which node?

1. Leaf node
2. First node of left sub-tree
3. **root node**
4. First node of the right sub-tree

Ans. In a max-heap, all the nodes (including the root) are greater than their respective child nodes. The key of the root node is always the largest among all other nodes.

10. In a binary search tree, which traversals would print the numbers in ascending order?

1. Level-order traversal
2. Pre-order traversal
3. Post-order traversal
4. **In-order traversal**

Ans. In a binary search tree, each left subtree has values below the root and each right subtree has values above the root. An in-order traversal first visits the left child, then visits the node, and finally, the right child.

11. When do you prefer Red-black trees over AVL trees?

1. **when there are more insertions or deletions**
2. when a large search operation is required
3. when the tree must be balanced
4. when log(nodes) time complexity is needed

Ans. Red-Black Trees require fewer rotations to maintain balance. On average, a Red-Black Tree requires at most 2 rotations for insertion and 3 rotations for deletion, while AVL Trees may require more rotations.

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

1. Multi Graph
2. **Regular Graph**
3. Simple Graph
4. Complete Graph

Ans. Regular Graph is an undirected graph where every vertex has the same number of edges or neighbours.

13. What is the term used when several elements compete for the same location in the hash table?

1. Diffusion
2. Replication
3. **Collision**
4. Duplication

Ans. A hash collision refers to a situation where two different inputs produce the same hash value or hash code when processed by a hash function.

14. A full binary tree can be generated using

1. **post-order and pre-order traversal**
2. pre-order traversal
3. post-order traversal
4. in-order traversal

Ans. A full binary tree is a tree in which every node has either 0 or 2 children.

15. B+ Trees are called balanced trees because

1. **the lengths of the paths from the root to all leaf nodes are equal.**
2. the lengths of the paths from the root to all leaf nodes differ from each other by at most 1
3. the number of children of any two non-leaf sibling nodes differs by at most 1
4. the number of records in any two leaf nodes differs by at most 1

16. An algorithm design method is used when the solution to a problem can be viewed as the result of a sequence of decisions

1. **Dynamic programming**
2. Backtracking
3. Branch and bound
4. Greedy method

Ans. This algorithm uses the already found solution to avoid repetitive calculation of the same part of the problem. It divides the problem into smaller overlapping subproblems, solves them, and stores the intermediate results.

17. Which algorithm type is used in solving the 4 Queens problem?

1. Greedy
2. Dynamic
3. Branch and Bound
4. **Backtracking**

18. Given an undirected graph G with V vertices and E edges, what will be the sum of the degrees of all vertices?

1. E
2. **2E**
3. V + E
4. 2V

Ans. The degree of a vertex in a graph is the number of edges connected to that vertex.

19. The necessary condition to be checked before deletion from the queue is

1. Overflow
2. **Underflow**
3. Rear value
4. Front value

Ans. Before deletion, we need to check whether the queue is empty or not.

20. BFS is best compared to DFS in the case of

1. The graph's width is large
2. **The graph's depth is large**
3. The graph consists of many nodes
4. The graph is complex

Ans. BFS explores all nodes at the present "depth" level before moving on to nodes at the next depth level. This ensures that the first time BFS reaches a node, it has found the shortest path to that node (in terms of the number of edges).

21. One of the differences between a queue and a stack is:

1. Queues require dynamic memory, but stacks do not.
2. Stacks require dynamic memory, but queues do not.
3. **Queues use two ends of the structure; stacks use only one**.
4. Stacks use two ends of the structure, and queues use only one

Ans. The stack has only one end, the top, at which both insertion and deletion take place. A queue has two ends, rear and front, for insertion and deletion respectively.

22. Which array operations have a time complexity of O(1)?

1. Searching any element
2. **Accessing any element**
3. Inserting an element
4. deleting an element

Ans. Accessing an element in an array can be done through indexing. So, it takes very little time.

23. What is the number of edges in a graph's minimum spanning tree with N vertices and E edges?

1. E - 1
2. **N - 1**
3. N + E - 1
4. N + E - 2

Ans. The number of edges in a spanning tree is equal to the number of nodes or vertices minus one i.e. n-1.

24. What is the time complexity of the merge sort algorithm?

1. O(n)
2. O(n^2)
3. O(log n)
4. **O(n log n)**

25. A self-balancing binary search tree can be used to implement

1. **Priority queue**
2. Hash table
3. Heap sort
4. Priority queue and Heap sort

Ans. A self-balancing binary search tree can implement a priority queue by efficiently managing insertions, deletions, and minimum/maximum element retrieval in O(logn) time.

26. What would be the colour of a newly created node while inserting a new element in a Red-black tree?

1. Black, if the new node is not a root node
2. Red, if the new node is not a root node
3. Black, if the new node is a root node
4. **Both b and c**

Ans. If the newly created node is a root node, then it will be Black; otherwise, it will be Red.

27. Which one of the following algorithms does not return the optimal solution?

1. Dynamic Programming
2. **Backtracking**
3. Branch and Bound
4. Greedy Method

Ans. Backtracking solves the problem recursively and removes the solution if it does not satisfy the constraints of a problem. Whenever a solution fails, we trace back to the failure point, build on the next solution, and continue this process till we find the solution or all possible solutions are looked after.

28. In a priority queue, insertion and deletion takes place at

1. front, rear end
2. only at the rear end
3. only at the front end
4. **any position**

Ans. In a priority queue, insertion takes place at the appropriate position to maintain the heap property, and deletion takes place at the root or the corresponding node.

29. O(n) means computing time is

1. Constant
2. Quadratic
3. **Linear**
4. Cubic

Ans. O(n) means the computing time grows linearly with the input size n.

30. The total number of comparisons in a bubble sort is

1. **n(n-1)/2**
2. 2n
3. n^2
4. n^3

Ans. The **bubble sort algorithm** repeatedly compares the adjacent elements, from left to right, and swaps them if they are out-of-order.

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

1. To find the lower/upper bound in an ordered sequence
2. Union of intervals
3. Debugging
4. **To search in an unordered list**

Ans. Binary Search is a searching algorithm that searches for an element's position in a sorted array only.

32. What makes selection sorting different from other sorting techniques?

1. **It requires no additional storage space**
2. It is scalable
3. It works best for already-sorted inputs
4. It is faster than any other sorting technique

Ans. Selection sort is an in-place comparison sort algorithm. In-place sorting algorithms rearrange the elements within the array that is to be sorted, without using any additional space or memory.

33. A linearly ordered sequence of memory cells is known as

1. **node**
2. link
3. variable
4. null

34. Associative arrays can be implemented using

1. B-tree
2. A doubly linked list
3. A single linked list
4. **A self-balancing binary search tree**

Ans. Associative arrays can be implemented using self-balancing binary search trees like AVL trees or Red-Black trees.

35. What is the worst-case time complexity of binary search using recursion?

1. **O(log n)**
2. O(n)
3. O(n^2)
4. O(nlogn)

36. What is the advantage of a recursive approach over an iterative approach?

1. Consumes less memory
2. **Less code and easy to implement**
3. Consumes more memory
4. More code has to be written

37. What is the hash function used in the division method?

1. h(k) = k/m
2. **h(k) = k mod m**
3. h(k) = m/k
4. h(k) = m mod k

Ans. The hash function in Data Structures is a function that takes a key and returns an index into the hash table. After that, it returns the value stored at that index which is known as the hash value.

Data Structure - II Chapters

Here's the list of chapters on the "Data Structure - II" subject covering 100+ topics. You can practice the MCQs chapter by chapter starting from the 1st chapter or you can jump to any chapter of your choice.

1. Searching

The section contains multiple choice questions and answers on linear and binary search iteratives, linear search recursive, jump search, exponential search, uniform binary search and fibonacci search, interpolation and substring searching algorithms.

⍰ Searching	⍰ Jump Search
⍰ Linear Search Iterative	⍰ Fibonacci Search
⍰ Linear Search Recursive	⍰ Exponential Search
⍰ Binary Search Iterative	⍰ Interpolation Searching Algorithm
⍰ Uniform Binary Search	⍰ Substring Searching Algorithm

2. Sorting

The section contains questions and answers on sorting techniques like selection sort, bubble sort, merge sort, pancake sort, insertion sort, quicksort, shellsort, heapsort, introsort, timsort, binary tree sort, comb sort, cube sort, cycle sort, library sort, strand sort, cpcktail sort, gnome sort, pigeonhole sort, bogosort, bucket sort, bead sort, stooge sort, recursive bubble sort, tree sort, binary and recursive insertion sort, sleep sort, lsd and msd radix sort, inplace merge sort, bottom-up mergesort, counting sort, odd even and permutation sort.

- Sorting
- Insertion Sort – 1
- Insertion Sort – 2
- Selection Sort
- Bubble Sort
- Merge Sort
- In-place Merge Sort
- Bottom-Up Mergesort
- Quicksort – 1
- Quicksort – 2
- Quicksort – 3
- Quicksort using Random Sampling
- Quicksort using Median-of-Three Partitioning
- Shell Sort – 1
- Shell Sort – 2
- Heapsort – 1
- Heapsort – 2
- Introsort
- Timsort
- Cubesort
- Binary Tree Sort
- Cycle Sort

- Library Sort
- Strand Sort
- Cocktail Sort
- Comb Sort
- Gnome Sort
- Bogosort
- Sleep Sort
- Pigeonhole Sort
- LSD Radix Sort
- MSD Radix Sort
- Counting Sort
- Bucket Sort (Uniform Keys)
- Bead Sort
- Pancake Sort
- Odd-Even Sort
- Stooge Sort
- Permutation Sort
- Recursive Bubble Sort
- Binary Insertion Sort
- Recursive Insertion Sort
- Tree Sort

3. String Matching

The sections contains MCQs on rabin-karp and quick search algorithms.

- Rabin-Karp Algorithm
- Quick Search Algorithm

4. Number Theory

The section contains multiple choice questions and answers on euclids algorithm, strassens algorithm, permutations and combinations generation, partitions and subsets generation, inclusion and exclusion principles.

Euclid’s Algorithm	Generating Combinations
Euler’s Totient Function	Generating Partitions
Strassen’s Algorithm	Generating Subsets
Pseudorandom Number Generators	Floyd’s cycle-finding Algorithm
Generating Permutations	Inclusion-Exclusion Principle

5. Computational Geometry

The sections contains questions and answers on line point distance, cross product, closest pair problem, quickhull and chan’s algorithm.

Line Point Distance	Quickhull
Closest Pair Problem	Chan’s Algorithm
Cross Product	

6. Graph Search

The sections contains MCQs on depth first search, non recursive dfs, branch and bound, breadth first search and best first search.

Depth First Search	Best First Search
Non-recursive Depth First Search	Branch and Bound
Breadth First Search	

7. Minimum Spanning Tree

The section contains multiple choice questions and answers on minimum spanning tree, kruskal’s and prim’s algorithm.

Minimum Spanning Tree	Prim’s Algorithm
Kruskal’s Algorithm	

8. Shortest Path

The section contains questions and answers on dijkstra’s algorithm, bellman ford and floyd warshall algorithms.

Shortest Paths	Bellman-Ford Algorithm
Dijkstra’s Algorithm	Floyd-Warshall Algorithm

9. Flow Networks

The sections contain MCQs on maximum flow problem.

Maximum Flow Problem

10. Matching

The section contains multiple choice questions and answers on stable marriage problem and maximum bipartite matching.

- Matching
- Stable Marriage Problem
- Maximum Bipartite Matching

11. Minimum Cut

The sections contains questions and answers on minimum cut.

- Cut Vertices
- Minimum Cut
- Karger’s Algorithm

12. Graph Coloring

The sections contains MCQs on chromatic number, vertex and edge coloring.

- Vertex Coloring
- Chromatic Number
- Edge Coloring
- Edge Coloring – 2

13. Bipartite Graphs & Eulerian Tour

The section contains multiple choice questions and answers on complete bipartite graph, bipartite graphs and its properties.

- Bipartite Graph
- Properties of Bipartite Graphs
- Complete Bipartite Graph
- Eulerian Tour
- Fleury’s Algorithm

14. Recursion

The section contains questions and answers on recursion, factorial using recursion, fibonacci using recursion, sting reversal using recursion, matrix multiplication, gcd and lcm using recursion, decimal to binary conversions, length of a string, array, linked list using recursion, recursive selection sort, master’s theorem, searching element in array and linkedlist by using recursion.

⍰ Recursion	
⍰ Factorial using Recursion	⍰ Stack Reversal using Recursion
⍰ Fibonacci using Recursion	⍰ Recursive Selection Sort
⍰ Catalan Numbers	⍰ Largest and Smallest Number in an Array using Recursion
⍰ Sum of n Natural Numbers using Recursion	⍰ Largest and Smallest Number in a Linked List using Recursion
⍰ GCD and LCM using Recursion – 1	⍰ Search an Element in an Array using Recursion – 1
⍰ GCD and LCM using Recursion – 2	⍰ Search an Element in an Array using Recursion – 2
⍰ Sum of Digits of a Number using Recursion	⍰ Search an Element in a Linked List using Recursion
⍰ String Reversal using Recursion	⍰ Power of a Number using Recursion in Logn Time
⍰ Decimal to Binary Conversion using Recursion	⍰ Towers of Hanoi using Recursion
⍰ Length of a Linked List using Recursion	⍰ Master’s Theorem – 1
⍰ Length of a String using Recursion	⍰ Master’s Theorem – 2
⍰ Matrix Multiplication using Recursion	

15. Greedy Algorithms

The sections contains MCQs on fractional knapsack problem, activity selection problem and huffman code.

⍰ Greedy Algorithms	⍰ Activity Selection Problem
⍰ Fractional Knapsack Problem	⍰ Huffman Code

16. Backtracking

The section contains multiple choice questions and answers on backtracking, eight queens and n queens problem.

⍰ Backtracking	⍰ N Queens Problem
⍰ Eight Queens Problem	

17. Dynamic Programming

The sections contains questions and answers on dynamic programming, fibonacci using dynamic programming, coin change problem, kadane algorithm, longest increasing subsequence, rod cutting, minimum no of jumps, 0/1 knapsack problem, matrix chain multiplication, longest common subsequence, edit distance problem, wagner-fischer algorithm, balanced partition, dice throw problem and counting boolean parenthesizations.

- Dynamic Programming
- Fibonacci using Dynamic Programming
- Coin Change Problem
- Maximum Sum of Continuous Subarray – 1
- Maximum Sum of Continuous Subarray – 2
- Kadane’s Algorithm
- Longest Increasing Subsequence
- Rod Cutting
- Minimum Number of Jumps
- 0/1 Knapsack Problem
- Matrix-chain Multiplication
- Longest Common Subsequence
- Longest Palindromic Subsequence
- Edit Distance Problem
- Wagner-Fischer Algorithm
- Catalan Number using Dynamic Programming
- Assembly Line Scheduling
- Minimum Insertions to form a Palindrome
- Maximum Sum Rectangle in a 2D Matrix
- Balanced Partition
- Dice Throw Problem
- Counting Boolean Parenthesizations

18. Cryptography

The sections contains MCQs on monoalphabetic cipher, morse code, polyalphabetic and vigenere cipher, transposition and columnar transposition, pigpen cipher, atbash and gronsfeld cipher, beaufort cipher, autokey and playfair cipher, hill cipher, railfence and route cipher, trithemius cipher, polybius square, running key cipher, bifid and affine cipher.

- Monoalphabetic Cipher
- Polyalphabetic Cipher
- Transposition
- Pigpen Cipher
- Vigenère Cipher
- Atbash Cipher
- Gronsfeld Cipher
- Beaufort Cipher
- Autokey Cipher
- Playfair Cipher
- Hill Cipher
- Rail Fence Cipher
- Route Cipher
- Trithemius Cipher
- Polybius Square
- Running Key Cipher
- Bifid Cipher
- Morse Code – 1
- Morse Code – 2
- Columnar Transposition
- Affine Cipher
- Coalesced Hashing

19. Checksum, Complexity Classes & NP Complete Problems

The section contains multiple choice questions and answers on hamming code, hamiltonian path problem, subset sum and set partition problems, p, np, np-hard and np-complete complexity classes.

Hamming Code	Vertex Cover Problem
P, NP, NP-hard, NP-complete Complexity Classes	Independent Set Problem
Hamiltonian Path Problem	Dominating Set Problem
Subset Sum Problem	Set Partition Problem

20. Page Replacement Algorithms

The section contains questions and answers on optimal page replacement and first in first out algorithm.

Optimal Page Replacement Algorithm	Least Recently Used Algorithm (LRU)
Not Recently Used Algorithm (NRU)	Random Page Replacement Algorithm
First-in, First-out Algorithm (FIFO)	

21. Miscellaneous

The sections contain MCQs on topological sorting, quickselect, coordinate compression and square root decomposition.

Topological Sorting	Co-ordinate Compression
Flooding Algorithm	Square Root Decomposition
Quickselect	