https://www.interviewbit.com/data-structure-mcq/

1.

How is an array initialized in C language?

1. int a[3] = {1, 2, 3};
2. int a = {1, 2, 3};
3. int a[] = new int[3]
4. int a(3) = [1, 2, 3];

Ans (A)

An array is initialized in C as per the syntax in Option (A).

2.

Which of the following is a linear data structure?

1. Array
2. AVL Trees
3. Binary Trees
4. Graphs

Ans (A)

3.

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

1. \*a + 2
2. \*(a + 2)
3. \*(\*a + 2)
4. &(a + 2)

Ans (B)

4.

Which of the following is not the type of queue?

1. Priority queue
2. Single-ended queue
3. Circular queue
4. Ordinary queue

Ans (B)

Answer - B) Single Ended Queue is not a type of queue.

5.

From following which is not the operation of data structure?

1. Operations that manipulate data in some way
2. Operations that perform a computation
3. Operations that check for syntax errors
4. Operations that monitor an object for the occurrence of a controlling event

Check Answer

Answer - A) Operations that manipulate data in some way

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;

}

1. 5
2. 15
3. 9
4. 6

Ans (C)

The code snippet basically calculates the sum of elements at even indices of the array, so we get 1 + 3 + 5 = 9 as the result.

7.

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;

}

1. 2
2. 15
3. Syntax Error
4. 3

Ans (D)

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.

8.

What is the disadvantage of array data structure?

1. The amount of memory to be allocated should be known beforehand.
2. Elements of an array can be accessed in constant time.
3. Elements are stored in contiguous memory blocks.
4. Multiple other data structures can be implemented using arrays.

Ans (A)

The disadvantage of arrays is that the amount of memory to be allocated should be known beforehand.

9.

How are String represented in memory in C?

1. An array of characters.
2. The object of some class.
3. Same as other primitive data types.
4. LinkedList of characters.

Ans (A)

Strings are represented in C as array 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();

}

}

1. 3 2 1
2. 1 2 3
3. 3
4. 1

Ans (A)

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

Ans (B)

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++?

1. push\_back()
2. append()
3. push()
4. insert()

Ans (A)

push\_back() function is used for the purpose.

13.

Which one of the following is an application of queue data structure

1. When a resource is shared among multiple consumers.
2. When data is transferred asynchronously
3. Load Balancing
4. All of the above

Ans (D)

Answer - D) All the above statements are the applications of queue data structure.

14.

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

1. Overflow
2. Underflow
3. Syntax Error
4. Garbage Value

Check Answer

pop() on an empty queue causes Underflow.

15.

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;

}

O(n)

O(n^2)

O(1)

O(log n)

Check Answer

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).

16.

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

Stack

Arrays

LinkedList

All of the Above

Check Answer

Stack, Arrays, and LinkedList can be used to implement Queues so all the options are correct.

17.

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

Stack

Arrays

LinkedList

Queues

Check Answer

Stacks find their use in recursion.

18.

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

Stack

Deque

Queue

Strings

Check Answer

Deque is a data structure allowing insertion and deletion from both ends.

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;

}

1 2 3 4 5

5 4 3 2 1

1 3 5 2 4

5 3 1 2 4

Check Answer

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

Check 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

Check 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

Check Answer

Merge Sort is a Divide and Conquer algorithm.

23.

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

Bubble Sort

Merge Sort

Counting Sort

Heap Sort

Check 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.

Check Answer

All the above options are applicable.

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.

Check Answer

The 0/1 Knapsack is not an NP-Hard 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

Check Answer

All the above algorithms are used for string and pattern matching.

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))

Check Answer

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).

Check Answer

Both (A) and (B) are used to handle queries on trees.

29.

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;

}

1 2 3 4 5

5 4 3 2 1

1 3 5 2 4

2 4 1 3 5

Check Answer

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.

30.

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.

Check Answer

Segment trees would be an optimal choice for such operations.

31.

What is the time complexity of the binary search algorithm?

O(n)

O(1)

O(log2n)

O(n^2)

Check Answer

Time complexity of binary search algorithm is O(log2n).

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.

Check 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;

}

6

7

0

1

Check Answer

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.

Check Answer

Red-Black Trees are used to implement Maps in C++.

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;

}

5

4

6

3

Check Answer

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.

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;

}

3

4

0

2

Check Answer

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)

Check Answer

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)

Check Answer

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)

Check Answer

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

20

5

1

Check Answer

The code calculates the sum of values of the nodes of the given LinkedList, which is 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

Check Answer

All the above operations can be performed with LinkedList.

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

Check Answer

The address of the next node and value of the current node are needed to be stored in a LinkedList’s node.

44.

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

2

0

1

n

Check Answer

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)

O(n \* logn)

O(1)

O(logn)

Check Answer

In the worst case, we might need to visit all the nodes in the BST.

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

Check Answer

Postorder traversal follows the order of left->right->root in a binary tree.

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)

Check Answer

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.

Check Answer

All the above options are applicable for an AVL Tree.

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;

}

Calculates the number of edges in an undirected graph.

Calculates the number of nodes in a given graph.

Calculates the sum of degrees of all nodes in a given graph.

None of the above.

Check Answer

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

Check 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

Check Answer

Such a node in a tree is called a centroid of the tree.

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);

}

}

}

Stores depths of all the nodes in a given tree, with respect to some root node.

Counts the number of nodes in a given tree.

Finds the diameter of a tree.

Checks if all the nodes are reachable in a given tree.

Check Answer

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.

Check 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)

Check Answer

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

Check Answer

Answer - B) Stacks are used for t