41. How do you find the height of a node in a tree?

The height of a node is equal to the number of edges in the longest path to the leaf from that node. Here, the depth of a leaf node is zero.

I) List the types of trees.

a) General Tree

A generic tree with no constrained hierarchy. Here, nodes can have an endless number of offspring and the rest of the trees are subsets of the tree.

b) Binary Tree

It is the most commonly-known tree. Here, each parent has a minimum of two offspring, which are called the left and right youngsters. When a binary tree has certain features and limitations, other trees, such as RBT, AVL, and binary search trees, are used.

c) Binary Search Tree

Also known as BST, it is an extension of a binary tree with various optional constraints. The left child value of a node should be less than or equal to the parent value, and the correct child value should be higher than or equal to the parent value.

e) Red and Black Tree

It is an auto-balancing tree with red or black nodes. Hence, the name Red-Black Tree. It keeps the forest balanced. The tree might not be balanced, but the searching process takes O(log n) time.

f) N-ary Tree

N represents the maximum number of children. In this tree, every node’s children are either 0 or N.

42. How does bubble sort work?

A bubble sort is a commonly-used sorting method appalled to arrays where adjacent elements are compared to each other. Also, values are exchanged as per the order followed in the arrangement. It is named bubble sort as it involves exchanging elements like bubbles floating in the water and larger entities sinking to the bottom.

43. What is a Jagged array?

A jagged array is used to store rows of data elements of varying sizes and dimensions. It enhances efficiency while working with multidimensional arrays.

44. How do you find a loop in a linked list?

There are multiple ways to detect a loop in a linked list. One can use the following methods:

- Hashing
- Floyd’s cycle-finding algorithm
- Visited nodes method

45. What are the advantages of a binary search over a linear search?

- A binary search is comparatively more efficient than a linear search as it involves fewer comparisons.
- Linear search allows you to eliminate only one element per comparison each time you fail to find the value you want. However, a binary search allows you to eliminate half the data set with each comparison.
- The binary search runs in O(log n) time while the linear search runs in O(n) time. So, the more elements in the search array, the faster the binary search is.

46. What is a heap data structure?

Heap is a unique tree-based non-linear data structure with a complete binary tree. It means that all the levels are filled except possibly the last one, which consists of all the elements, and all the nodes are left-justified.

Broadly, a heap is of two types:

- **Max-Heap:** The value of the root node must be greatest from all other keys of its children node. The same is recursively true for all sub-trees in that binary tree.

- **Min-Heap:** The value of the key at the root node must be the smallest among all keys of the children node. The same stands recursively true for all other sub-trees in that binary tree.

47. Define Red-Black Tree.

A red-black tree is a binary tree invented in 1972 by Rudolf Bayer and was dubbed as “symmetric binary B-tree”. It is a self-balancing binary tree that stores data in two’s complementary binary formats.

Each node is either red or black, and by comparing these colours on a path from root to leaf, the tree ensures that the path is not more than twice as long as others, making it balanced. It can be used to store any kind of data represented as a set of values.

Although it is similar to a binary tree, there is one difference. The read-black tree is faster to access, making it suitable for storing a large amount of data.

48. What is a Trie data structure?

Trie stands for ‘Retrieval’ and is also known as the prefix tree or digital tree. It stores a set of strings in the form of a sorted tree. Let’s say strings range from letters ‘a to z’ of the English alphabet, so each trie node can’t have more than 26 points.

Also, every individual node has the same number of pointers as alphabet characters. This data structure can check the work in the dictionary by its prefix. Moreover, it allows you to print all words alphabetically, which is an advantage over hashing. It ensures efficient prefix search or auto-complete.

The key to which each node is connected is based on its position in the Trie data structure. You can add or find strings in O(L) time, where L = Length of a single word. This is a faster way than BST and hashing, considering the way it is implemented. You don’t need to compute a hash function or manage collisions like in open addressing or separate chaining.