TOPPERWORLD
LEARN & GROW

LEARN

DSA

through these

Most Asked Interview Question

www.TOPPERWORLD.in

Q.1 What is binary tree data structure? What are the applications for binary trees?

Ans.

A binary tree is a data structure that is used to organize data in a way that allows for efficient retrieval and manipulation.

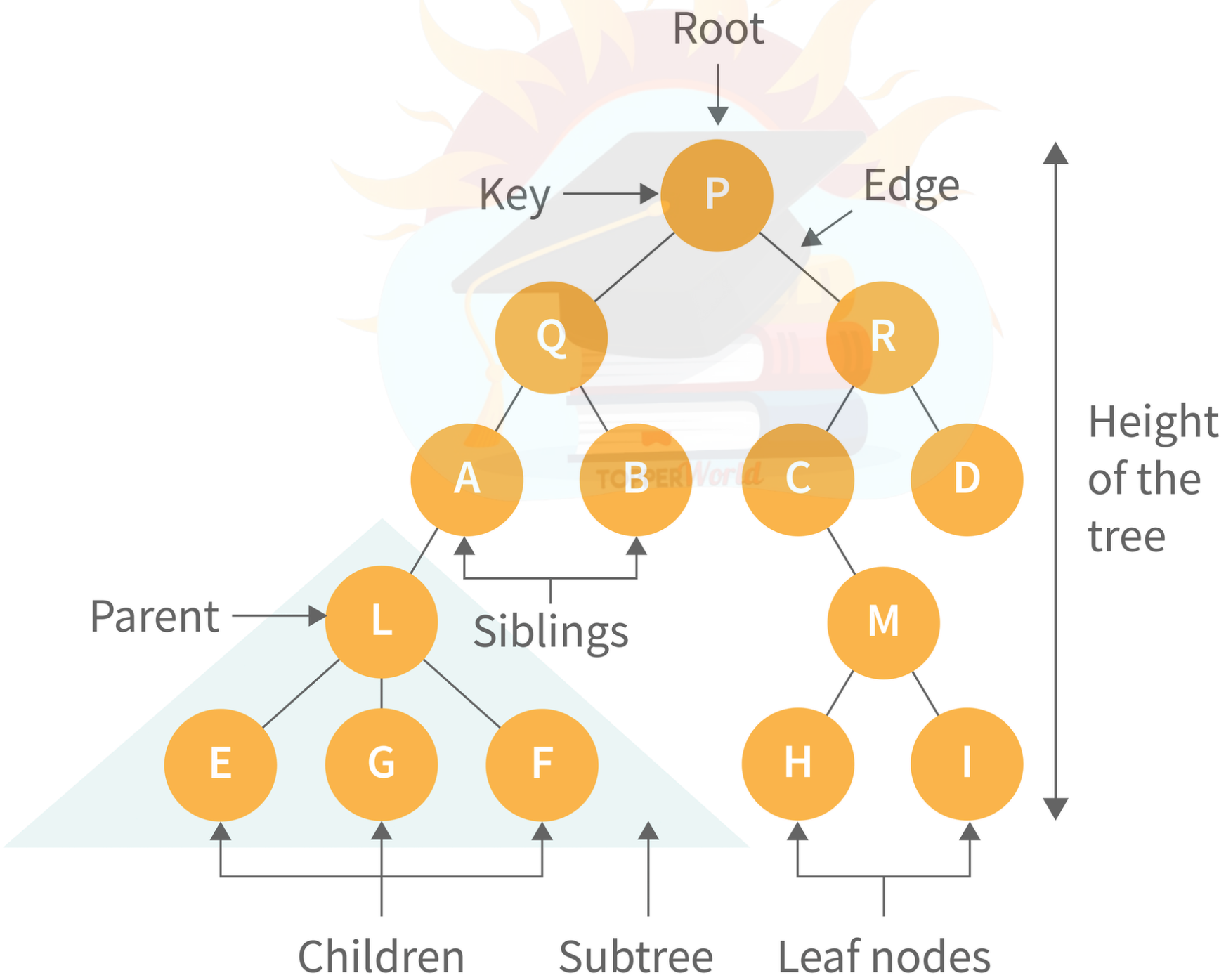
It is a data structure that uses two nodes, called leaves and nodes, to represent the data.

The leaves represent the data and the nodes represent the relationships between the leaves. Each node has two children, called siblings, and each child has one parent.

The parent is the node that is closest to the root of the tree. When a node is deleted from the tree, it is deleted from both its child and its parent.

Here are some applications for binary tree data structure:

- It's widely used in computer networks for storing routing table information.
- Decision Trees.
- Expression Evaluation.
- Database indices.



Q. 2 What are tree traversals?

Ans.

Tree traversal is the process of visiting all the nodes of a tree. Since the root (head) is the first node and all nodes are connected via edges (or links) we always start with that node. There are three ways which we use to traverse a tree -

1. Inorder Traversal:

Algorithm:

- Step 1. Traverse the left subtree, i.e., call Inorder(root.left)
- Step 2. Visit the root.
- Step 3. Traverse the right subtree, i.e., call Inorder(root.right)

2. Preorder Traversal:

Algorithm:

- Step 1. Visit the root.
- Step 2. Traverse the left subtree, i.e., call Preorder(root.left)
- Step 3. Traverse the right subtree, i.e., call Preorder(root.right)

3. Postorder Traversal:

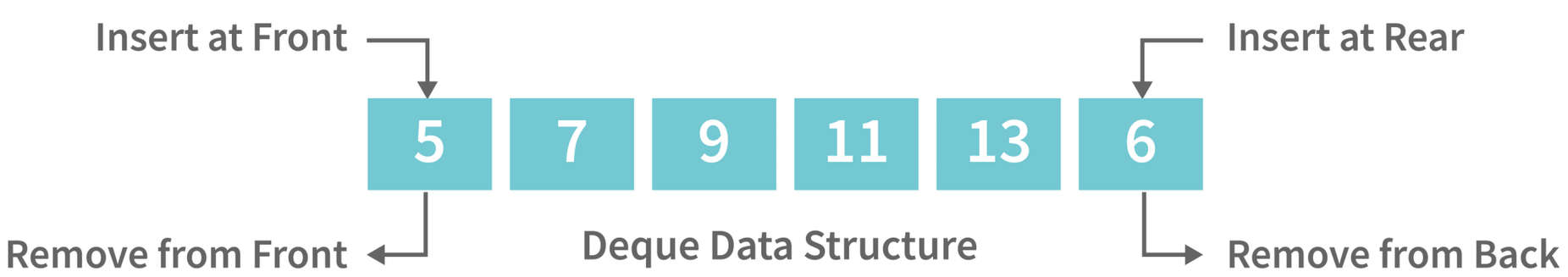
Algorithm:

- Step 1. Traverse the left subtree, i.e., call Postorder(root.left)
- Step 2. Traverse the right subtree, i.e., call Postorder(root.right)
- Step 3. Visit the root.

Q. 3 What is a deque data structure and its types?

A deque can be thought of as an array of items, but with one important difference: Instead of pushing and popping items off the end to make room, deques are designed to allow items to be inserted at either end.

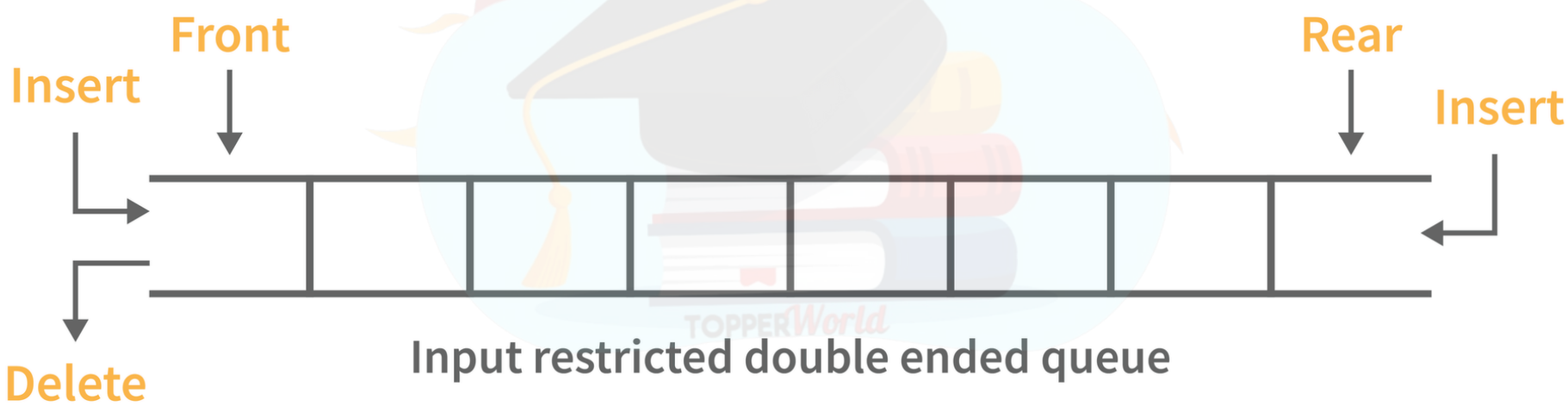
This property makes deques well-suited for performing tasks such as keeping track of inventory, scheduling tasks, or handling large amounts of data.

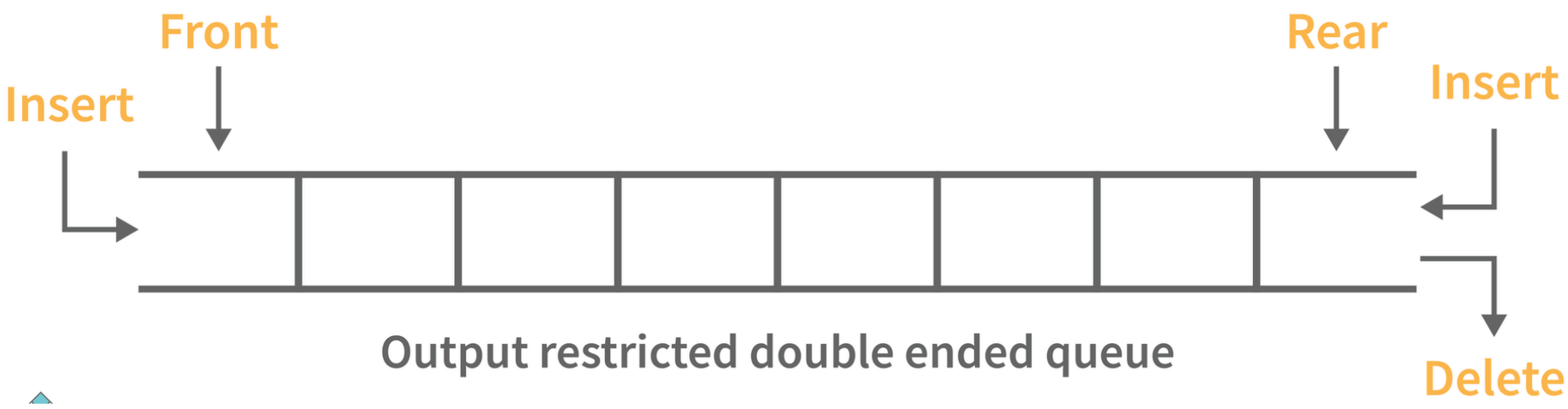


There are two types of deque:

Input Restricted Deque: Insertion operations are performed at only one end while deletion is performed at both ends in the input restricted queue.

Output Restricted Deque: Deletion operations are performed at only one end while insertion is performed at both ends in the output restricted queue.





Q. 4 What is topological sorting in a graph?

Ans.

- Topological sorting is a linear ordering of vertices such that for every directed edge ij, vertex i comes before j in the ordering.
- Topological sorting is only possible for Directed Acyclic Graph (DAG).

Applications:

- jobs scheduling from the given dependencies among jobs.
- ordering of formula cell evaluation in spreadsheets
- ordering of compilation tasks to be performed in make files,
- data serialization

Q. 5

What is the requirement for an object to be used as key or value in HashMap?

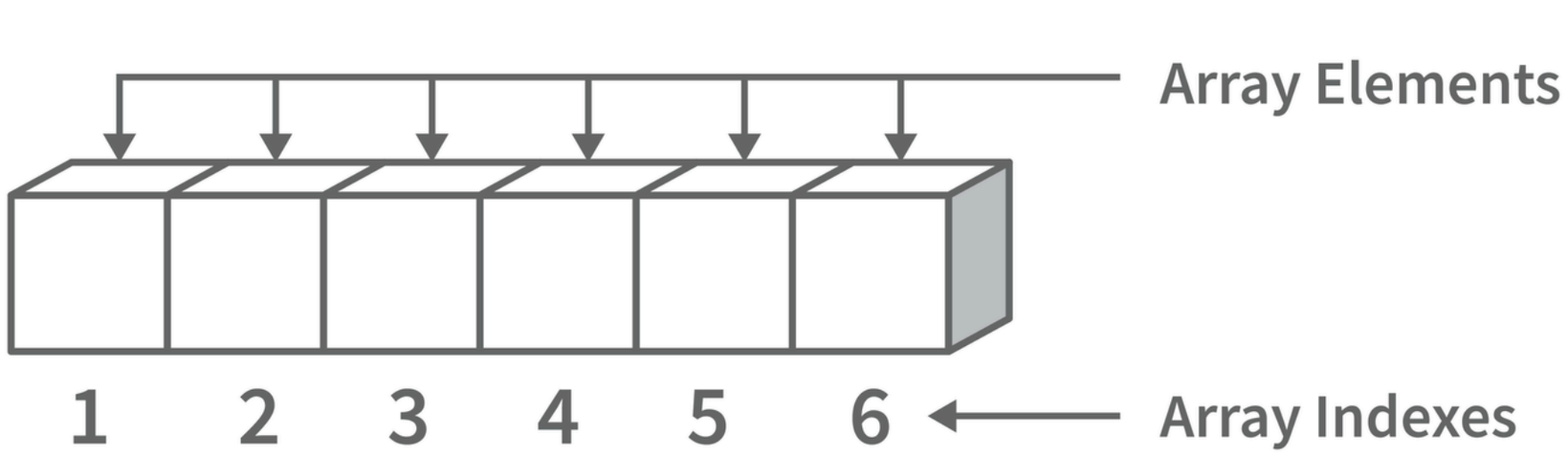
Ans.

- The key or value object that gets used in the hashmap must implement equals() and hashcode() method.
- The hash code is used when inserting the key object into the map and the equals method is used when trying to retrieve a value from the map.

Q. 6 What is array data structure?

Ans.

An array data structure is a data structure that is used to store data in a way that is efficient and easy to access. It is similar to a list in that it stores data in a sequence. However, an array data structure differs from a list in that it can hold much more data than a list can. An array data structure is created by combining several arrays together. Each array is then given a unique identifier, and each array's data is stored in the order.



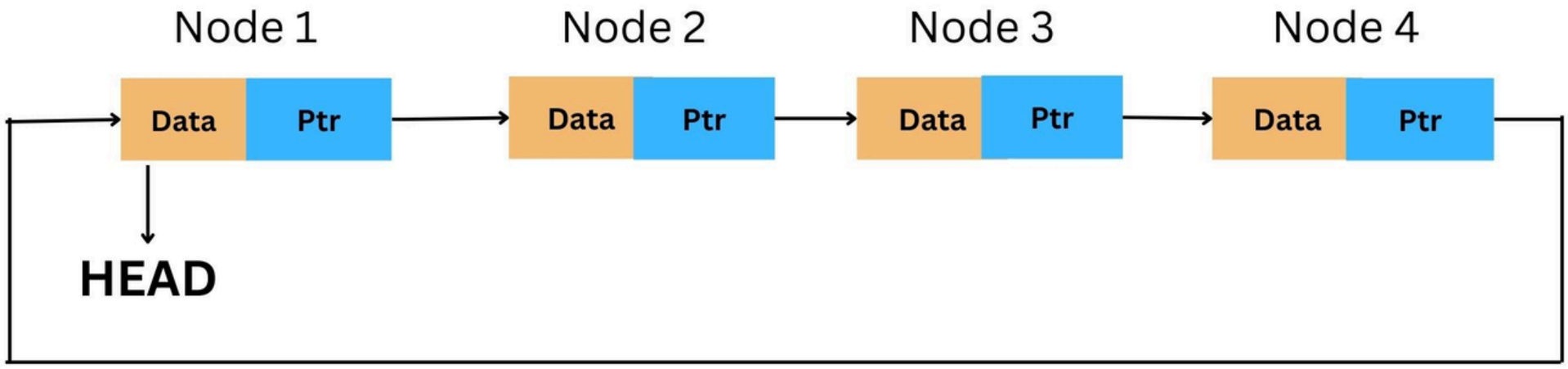
One-dimensional Array with six elements

Q. 7 What is a linked list data structure?

Ans.

A linked list can be thought of as a series of linked nodes (or items) that are connected by links (or paths). Each link represents an entry into the linked list, and each entry points to the next node in the sequence. The order in which nodes are added to the list is determined by the order in which they are created.

Circular Linked List in C++



Explain the difference between file structure and storage structure?

Ans.

File Structure: Representation of data into secondary or auxiliary memory say any device such as a hard disk or pen drives that stores data which remains intact until manually deleted is known as a file structure representation.

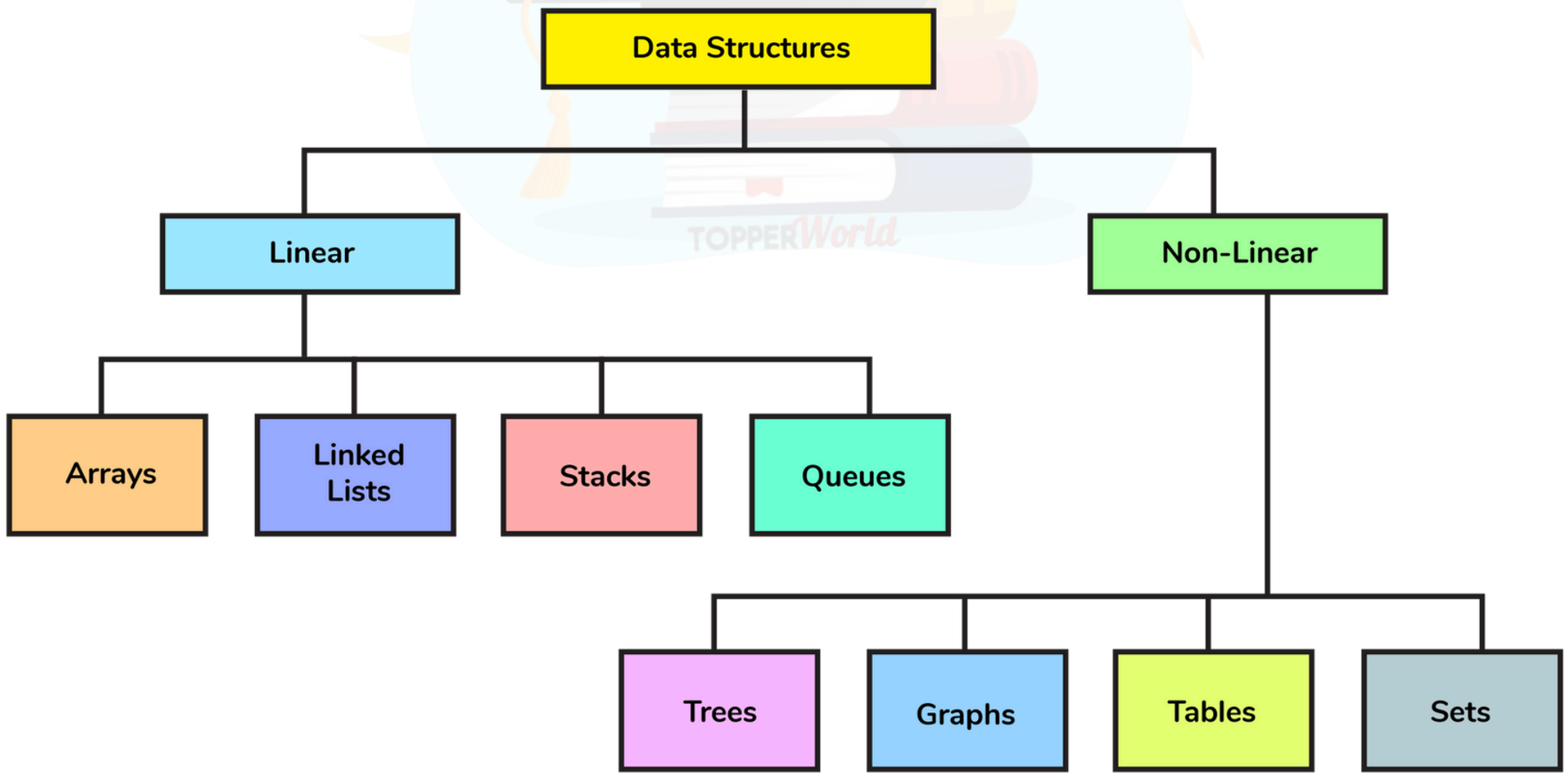
Storage Structure: In this type, data is stored in the main memory i.e RAM, and is deleted once the function that uses this data gets completely executed.

The difference is that the storage structure has data stored in the memory of the computer system, whereas the file structure has the data stored in the auxiliary memory.

Q.9 Describe the types of Data Structures?

Ans.

Linear Data Structure: A data structure that includes data elements arranged sequentially or linearly, where each element is connected to its previous and next nearest elements, is referred to as a linear data structure. Arrays and linked lists are two examples of linear data structures.



Non-Linear Data Structure: Non-linear data structures are data structures in which data elements are not arranged linearly or sequentially. We cannot walk through all elements in one pass in a non-linear data structure, as in a linear data structure. Trees and graphs are two examples of non-linear data structures.

Q. 10 What is a stack data structure? What are the applications of stack?

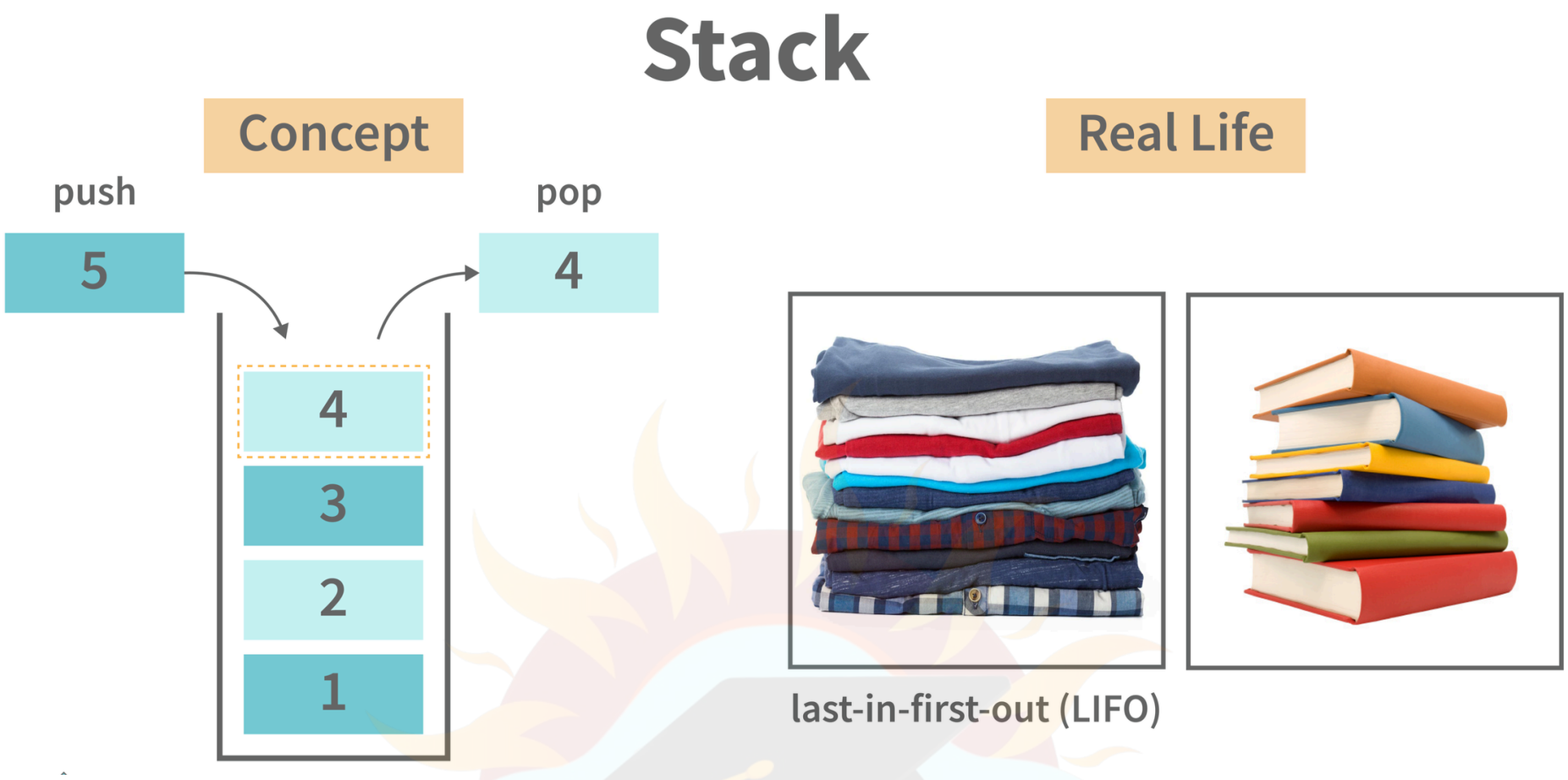
Ans.

A stack is a data structure that is used to represent the state of an application at a particular point in time. The stack consists of a series of items that are added to the top of the stack and then removed from the top.

It is a linear data structure that follows a particular order in which operations are performed. LIFO (Last In First Out) or FILO (First In Last Out) are two possible orders. A stack consists of a sequence of items.

The element that's added last will come out first, a real-life example might be a stack of clothes on top of each other.

When we remove the cloth that was previously on top, we can say that the cloth that was added last comes out first.



Following are some applications for stack data structure:

- It acts as temporary storage during recursive operations
- Redo and Undo operations in doc editors
- Reversing a string
- Parenthesis matching
- Postfix to Infix Expressions
- Function calls order

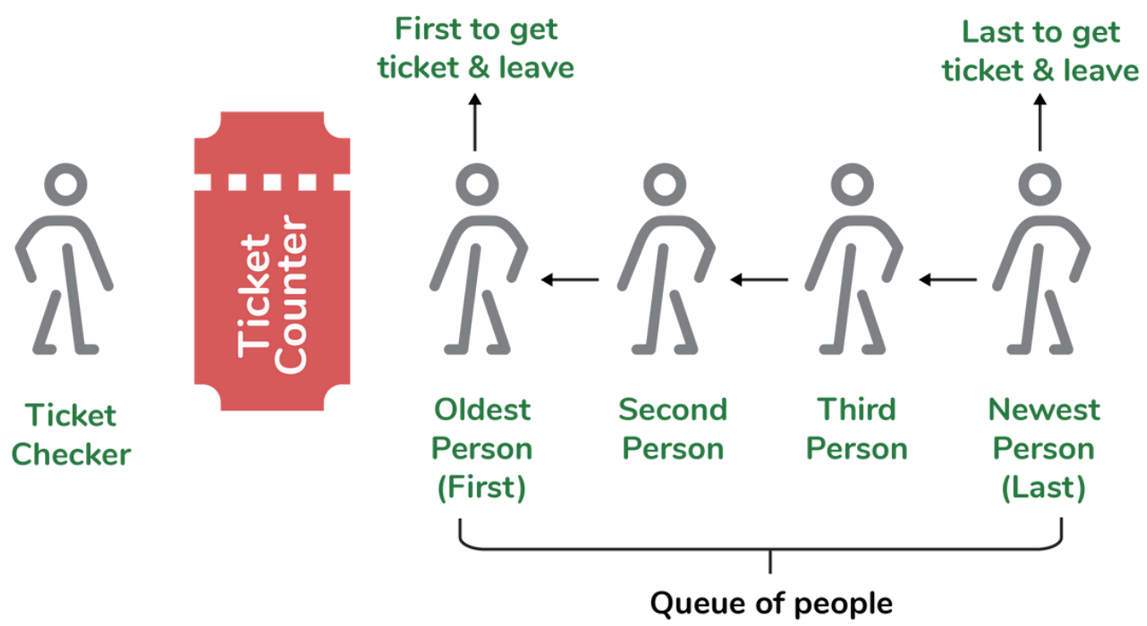
Q. 11 What is a queue data structure? What are the applications of queue?

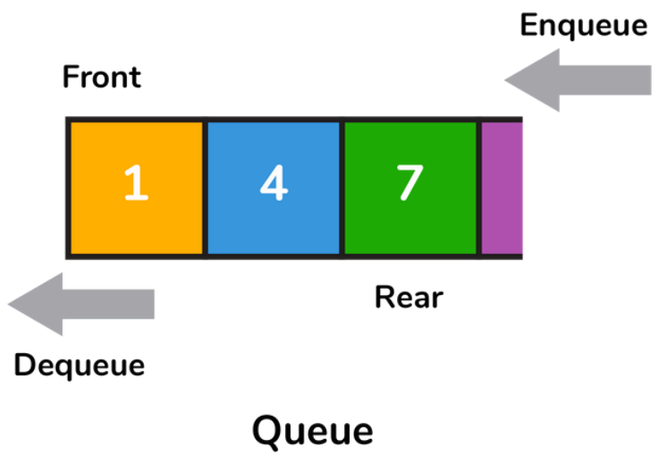
Ans.

A queue is a linear data structure that allows users to store items in a list in a systematic manner. The items are added to the queue at the rear end until they are full, at which point they are removed from the queue from the front.

Queues are commonly used in situations where the users want to hold items for a long period of time, such as during a checkout process.

A good example of a queue is any queue of customers for a resource where the first consumer is served first.





Following are some applications of queue data structure:

- Breadth-first search algorithm in graphs
- Operating system: job scheduling operations, Disk scheduling, CPU scheduling etc.
- Call management in call centres.

Difference between Array and Linked List.

Ans.

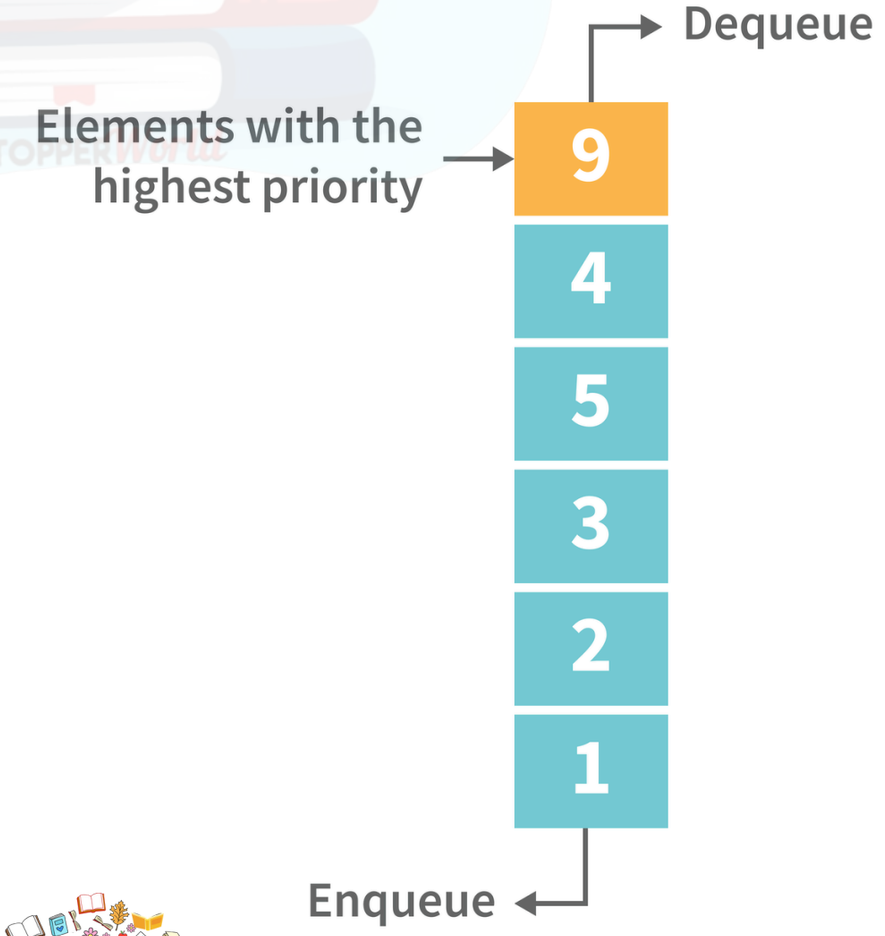
Arrays	Linked Lists
An array is a collection of data elements of the same type.	A linked list is a collection of entities known as nodes. The node is divided into two sections: data and address.
It keeps the data elements in a single memory.	It stores elements at random, or anywhere in the memory.
The memory size of an array is fixed and cannot be changed during runtime.	The memory size of a linked list is allocated during runtime.
An array's elements are not dependent on one another.	Linked List elements are dependent on one another.

It is easier and faster to access an element in an array.	In the linked list, it takes time to access an element.
Memory utilization is ineffective in the case of an array.	Memory utilization is effective in the case of linked lists.
Operations like insertion and deletion take longer time in an array.	Operations like insertion and deletion are faster in the linked list.

Q. 13 What is a priority queue?

Ans.

Priority Queue is an abstract data type that is similar to a queue in that each element is assigned a priority value. The order in which elements in a priority queue are served is determined by their priority (i.e., the order in which they are removed). If the elements have the same priority, they are served in the order they appear in the queue.



Q. 14

What is the difference between the Breadth First Search (BFS) and Depth First Search (DFS)?

Ans.

Breadth First Search (BFS)	Depth First Search (DFS)
It stands for “Breadth First Search”	It stands for “Depth First Search”
BFS (Breadth First Search) finds the shortest path using the Queue data structure.	DFS (Depth First Search) finds the shortest path using the Stack data structure.
We walk through all nodes on the same level before passing to the next level in BFS.	DFS begins at the root node and proceeds as far as possible through the nodes until we reach the node with no unvisited nearby nodes.

When compared to DFS, BFS is slower.	When compared to BFS, DFS is faster.
BFS performs better when the target is close to the source.	DFS performs better when the target is far from the source.
BFS necessitates more memory.	DFS necessitates less memory.
Nodes that have been traversed multiple times are removed from the queue.	When there are no more nodes to visit, the visited nodes are added to the stack and then removed.
Backtracking is not an option in BFS.	The DFS algorithm is a recursive algorithm that employs the concept of backtracking.
It is based on the FIFO principle (First In First Out).	It is based on the LIFO principle (Last In First Out).

Q. 15 What is AVL tree data structure, its operations, and its rotations?

Ans.

AVL trees are height balancing binary search trees named after their inventors Adelson, Velski, and Landis.

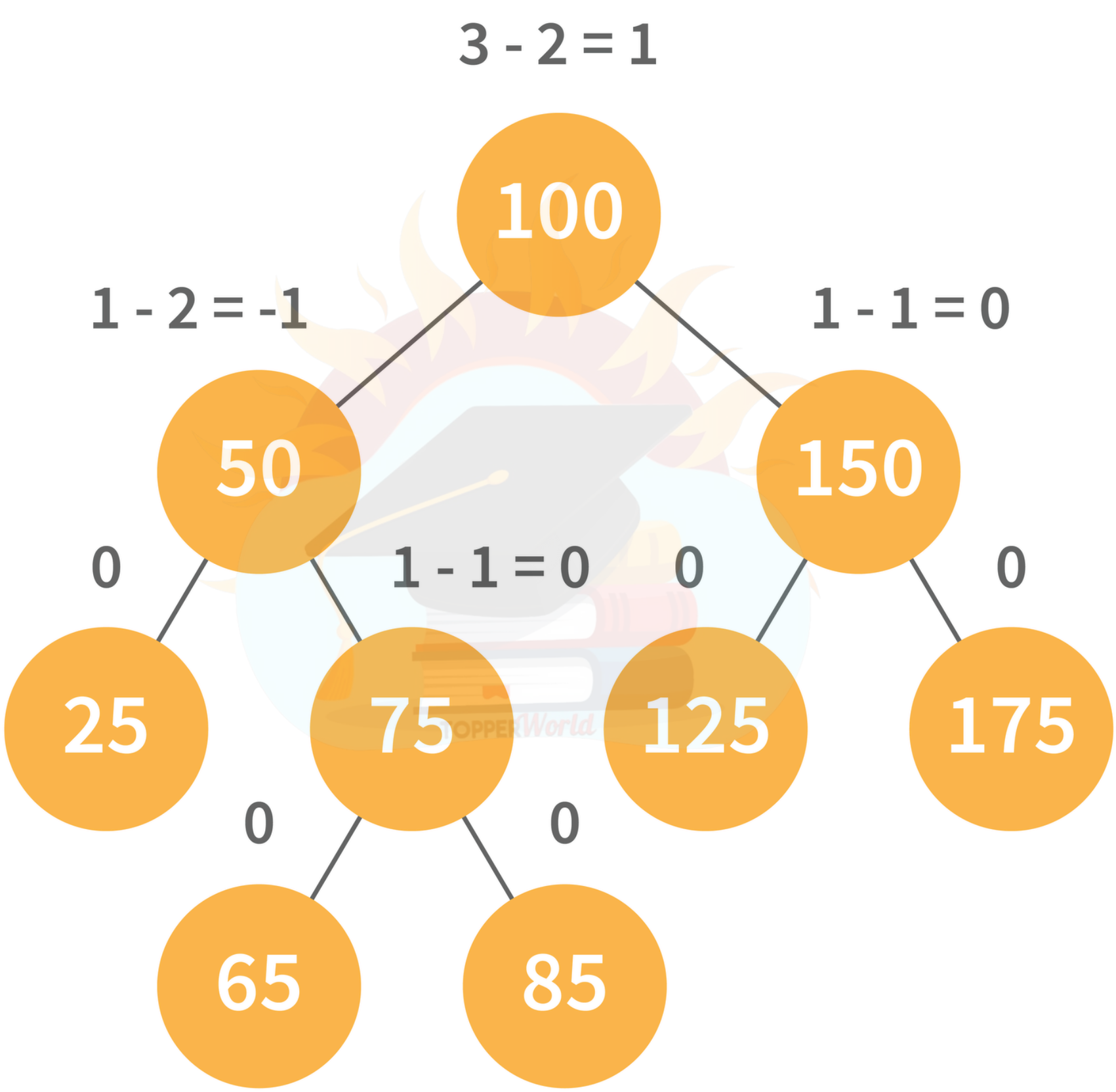
The AVL tree compares the heights of the left and right subtrees and ensures that the difference is less than one. This distinction is known as the Balance Factor.

BalanceFactor = height(left-subtree) - height(right-subtree)

We can perform the following two operations on AVL tree:

Insertion: Insertion in an AVL tree is done in the same way that it is done in a binary search tree. However, it may cause a violation in the AVL tree property, requiring the tree to be balanced. Rotations can be used to balance the tree.

Deletion: Deletion can also be performed in the same manner as in a binary search tree. Because deletion can disrupt the tree's balance, various types of rotations are used to rebalance it.



An AVL tree can balance itself by performing the four rotations listed below:

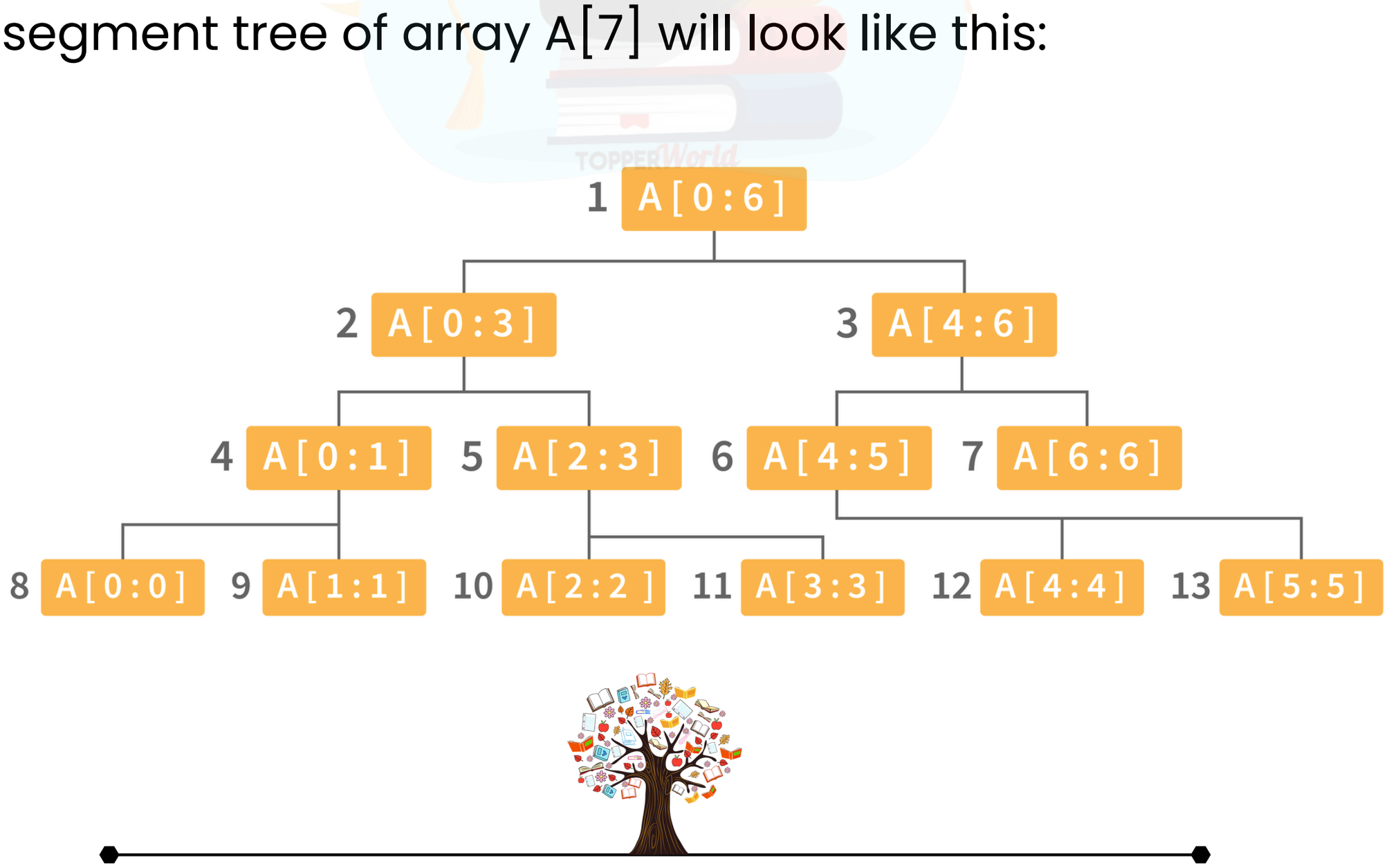
- **Left rotation:** When a node is inserted into the right subtree of the right subtree and the tree becomes unbalanced, we perform a single left rotation.
- **Right rotation:** If a node is inserted in the left subtree of the left subtree, the AVL tree may become unbalanced. The tree then requires right rotation.
- **Left-Right rotation:** The RR rotation is performed first on the subtree, followed by the LL rotation on the entire tree.
- **Right-Left rotation:** The LL rotation is performed first on the subtree, followed by the RR rotation on the entire tree.

Q. 16 Define Segment Tree data structure

Ans.

A segment Tree is a binary tree that is used to store intervals or segments. The Segment Tree is made up of nodes that represent intervals. Segment Tree is used when there are multiple range queries on an array and changes to array elements.

The segment tree of array A[7] will look like this:



Q. 17 Define Trie data structure and its applications

Ans.

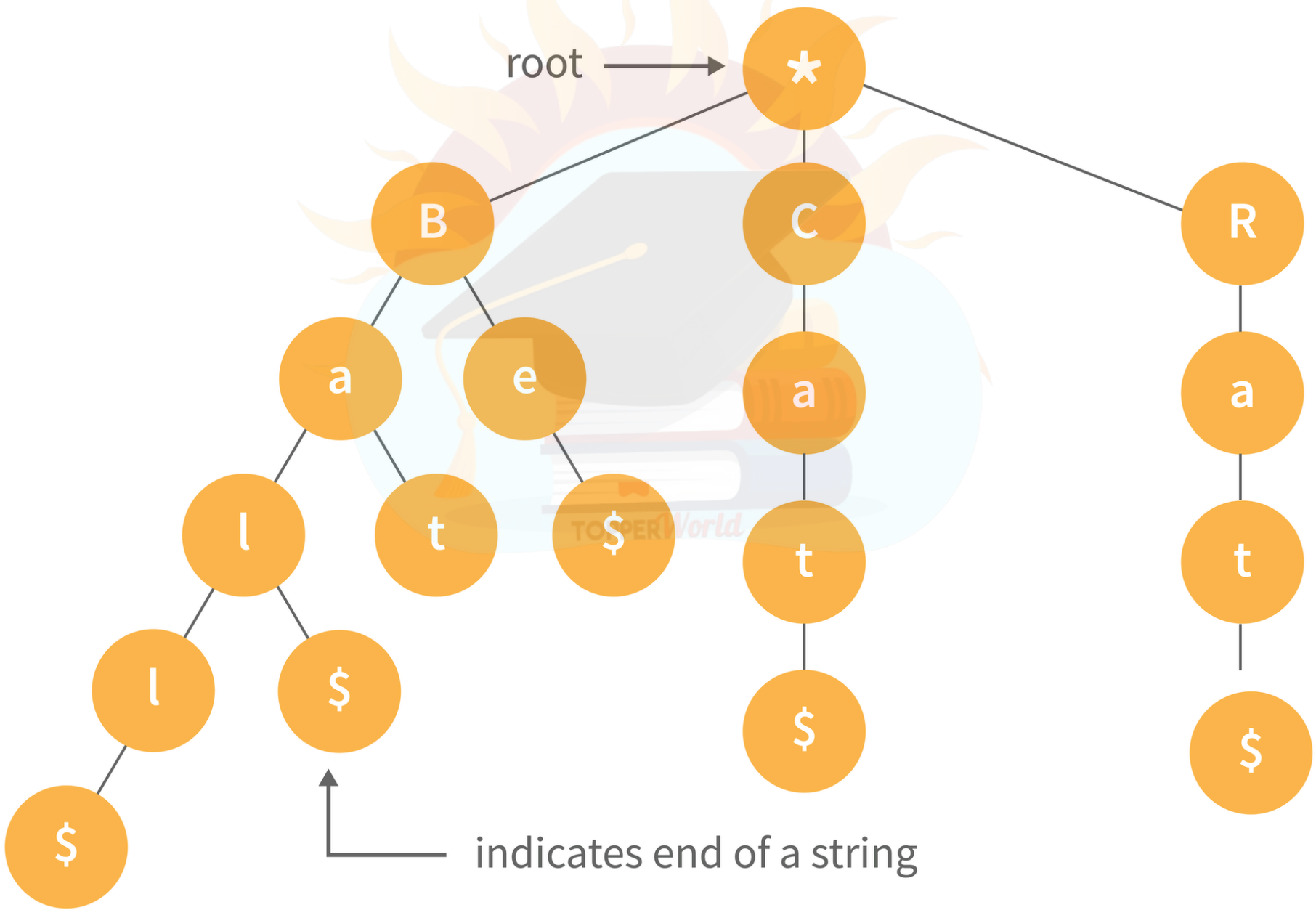
The word **"Trie"** is an abbreviation for "retrieval." Trie is a data structure that stores a set of strings as a sorted tree. Each node has the same number of pointers as the number of alphabet characters. It can look up a word in the dictionary by using its prefix. Assuming that all strings are formed from the letters 'a' to 'z' in the English alphabet, each trie node can have a maximum of 26 points.

Trie is also referred to as the digital tree or the prefix tree. The key to which a node is connected is determined by its position in the Trie. Trie allows us to insert and find strings in O(L) time, where L is the length of a single word. This is clearly faster than BST. Because of how it is implemented, this is also faster than Hashing.

Another benefit of Trie is that we can easily print all words in alphabetical order, which is not easy with hashing. Trie can also perform prefix search (or auto-complete) efficiently.

Consider the following list of strings to construct Trie

Cat, Bat, Ball, Rat, Cap & Be



The main disadvantage of tries is that they require a large amount of memory to store the strings. We have an excessive number of node pointers for each node

Following are some real-time applications for Trie data structure:

- Auto-Complete and Search for Search Engines
- Genome Analysis
- Data Analytics
- Browser History
- Spell Checker

Q. 18 Define Red-Black Tree.

Ans.

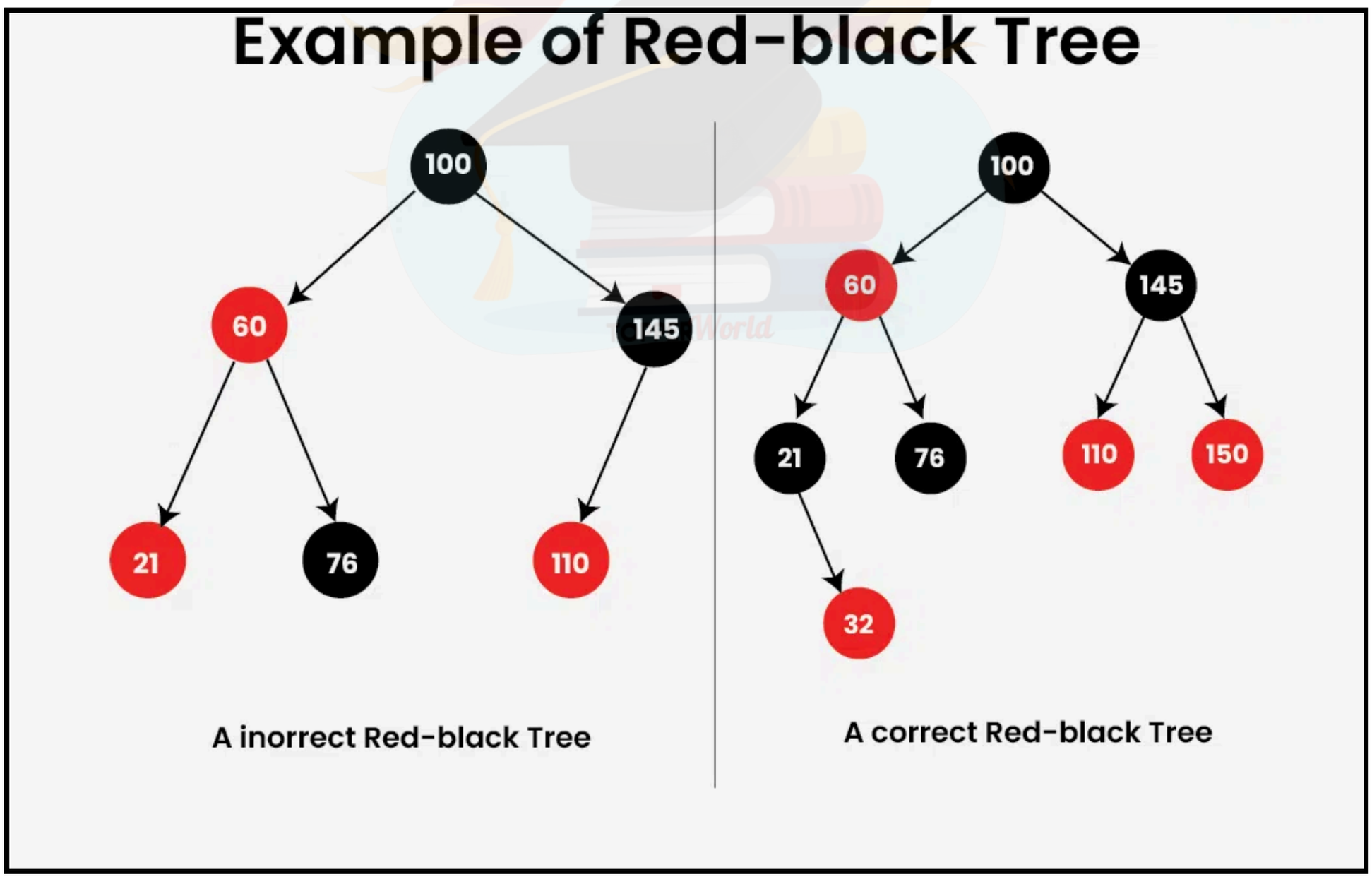
Red Black Trees are a type of self-balancing binary search tree. Rudolf Bayer invented it in 1972 and dubbed it "symmetric binary B-trees."

A red-black tree is a Binary tree in which each node has a colour attribute, either red or black. By comparing the node colours on any simple path from the root to a leaf, red-black trees ensure that no path is more than twice as long as any other, ensuring that the tree is generally balanced.

Red-black trees are similar to binary trees in that they both store their data in two's complementary binary formats.

However, red-black trees have one important advantage over binary trees: they are faster to access. Because red-black trees are so fast to access, they are often used to store large amounts of data.

Red-black trees can be used to store any type of data that can be represented as a set of values.



Q. 19 Which data structures are used for implementing LRU cache?

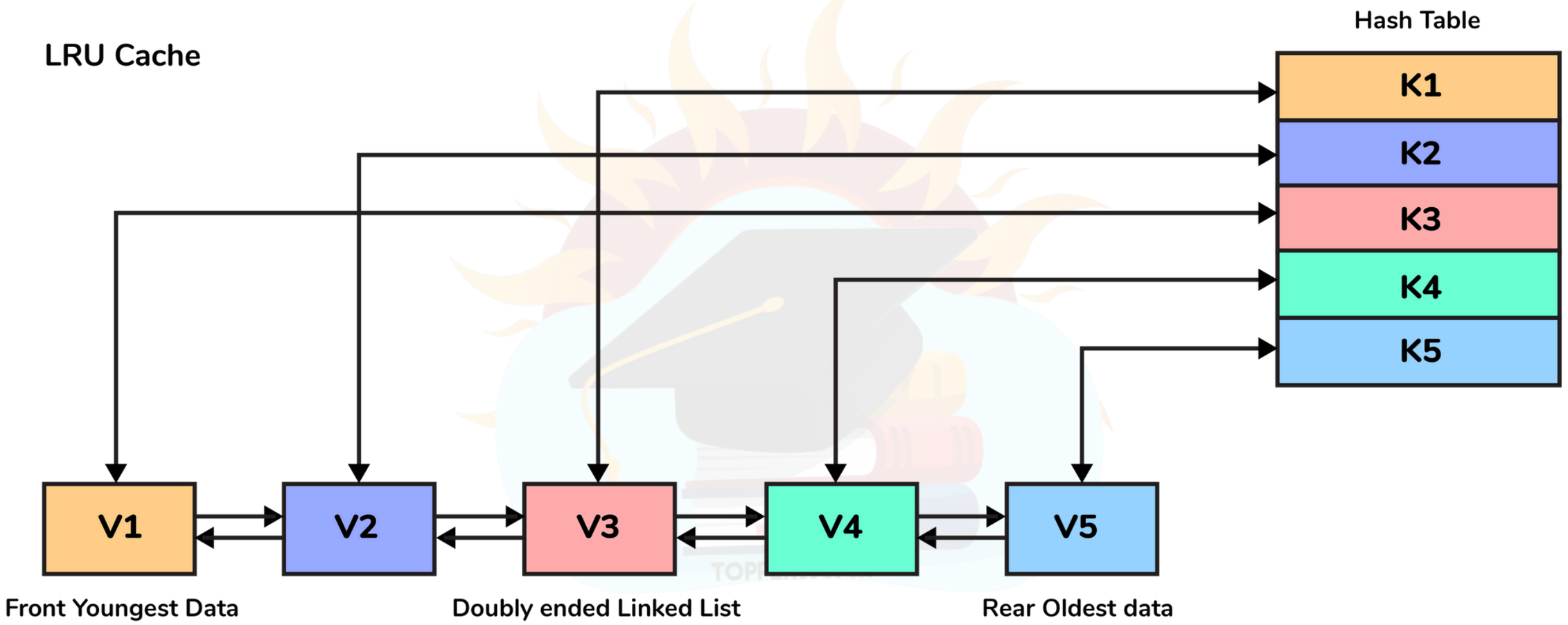
Ans.

LRU cache or Least Recently Used cache allows quick identification of an element that hasn't been put to use for the longest time by organizing items in order of use.

In order to achieve this, two data structures are used:

- **Queue –** This is implemented using a doubly-linked list. The maximum size of the queue is determined by the cache size, i.e by the total number of available frames. The least recently used pages will be near the front end of the queue whereas the most recently used pages will be towards the rear end of the queue.

- **Hashmap –** Hashmap stores the page number as the key along with the address of the corresponding queue node as the value.



Q. 20 Write Java code to count number of nodes in a binary tree

Ans.

```
int countNodes(Node root)
{
   int count =  1;          //Root itself should be counted
   if (root ==null)
      return 0;
   else
   {
      count += countNodes(root.left);
      count += countNodes(root.right);
      return count;
   }
}
```

"UNLOCK YOUR POTENTIAL"

With- TOPPERWORLD

Explore More

www. topperworld.in

DSA TUTORIAL

C TUTORIAL

C++ TUTORIAL

JAVA TUTORIAL

PYTHON TUTORIAL

www. TOPPERWORLD.in

E-mail

topperworld.in@gmail.com

Follow Us On