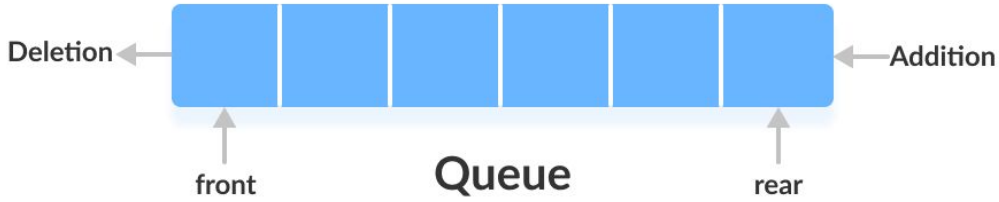
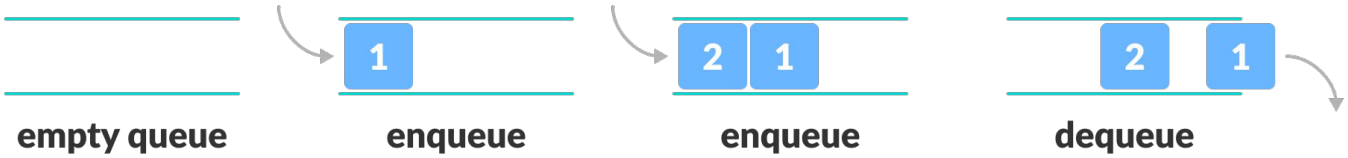
Queue Basics

Queue Data Structure



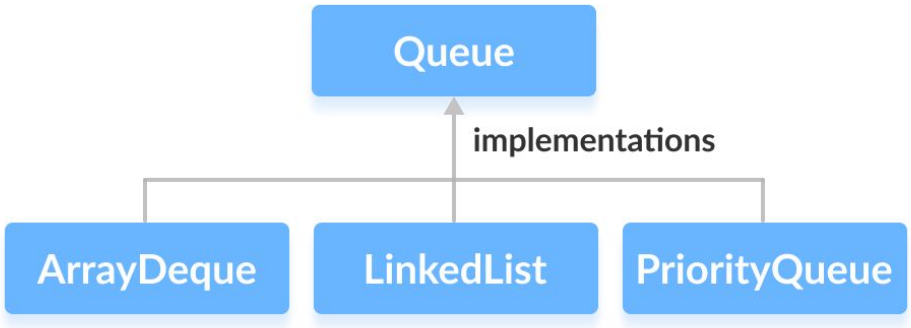


Implement Queue Using Linked List

Implement Queue Using Array

Implement Queue Using Circular Array

Queue Interface in Collection Framework



Methods of Queue

- **add()** - Inserts the specified element into the queue. If the task is successful, `add()` returns `true`, if not it throws an exception.
- **offer()** - Inserts the specified element into the queue. If the task is successful, `offer()` returns `true`, if not it returns `false`.
- **element()** - Returns the head of the queue. Throws an exception if the queue is empty.
- **peek()** - Returns the head of the queue. Returns `null` if the queue is empty.
- **remove()** - Returns and removes the head of the queue. Throws an exception if the queue is empty.
- **poll()** - Returns and removes the head of the queue. Returns `null` if the queue is empty.

Implement Queue using Two Stacks

Practice Problems

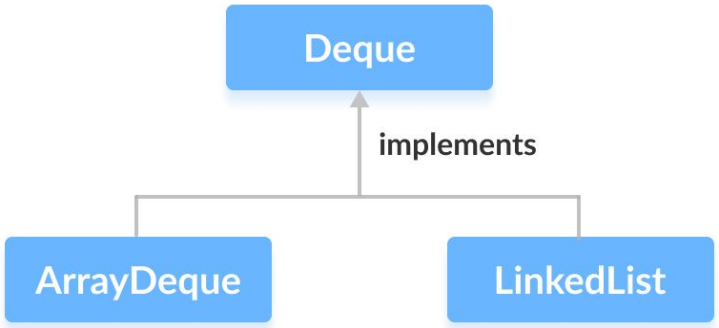
1. Implement Stack using Two Queues
2. Implement Stack using One Queue
3. Reverse a Queue
4. Reverse the first k elements of the Queue

Queue Problems - I

Reverse the first k elements of a Queue

Circular Tour Problem

Deque Data Structure



Methods of Deque

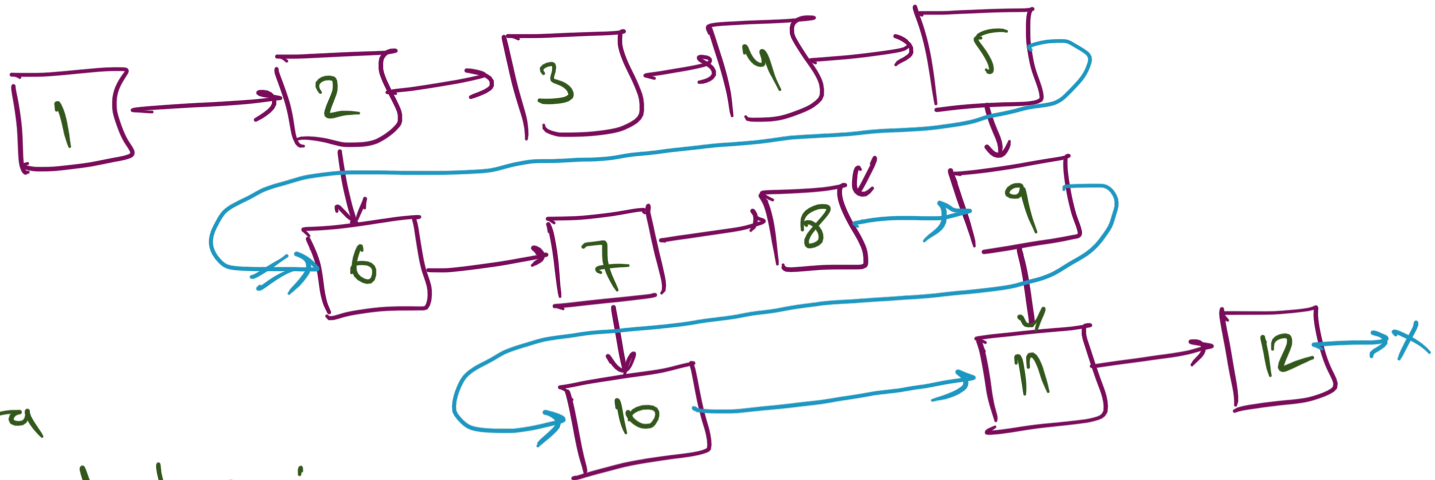
- **addFirst()** - Adds the specified element at the beginning of the deque. Throws an exception if the deque is full.
- **addLast()** - Adds the specified element at the end of the deque. Throws an exception if the deque is full.
- **offerFirst()** - Adds the specified element at the beginning of the deque. Returns `false` if the deque is full.
- **offerLast()** - Adds the specified element at the end of the deque. Returns `false` if the deque is full.
- **getFirst()** - Returns the first element of the deque. Throws an exception if the deque is empty.
- **getLast()** - Returns the last element of the deque. Throws an exception if the deque is empty.
- **peekFirst()** - Returns the first element of the deque. Returns `null` if the deque is empty.
- **peekLast()** - Returns the last element of the deque. Returns `null` if the deque is empty.
- **removeFirst()** - Returns and removes the first element of the deque. Throws an exception if the deque is empty.
- **removeLast()** - Returns and removes the last element of the deque. Throws an exception if the deque is empty.
- **pollFirst()** - Returns and removes the first element of the deque. Returns `null` if the deque is empty.
- **pollLast()** - Returns and removes the last element of the deque. Returns `null` if the deque is empty.

Sliding Window Maximum Problem

Practice Problems

1. Flatten a Multi-level Linked List
2. Read more on Usage of Queue: https://medium.com/techie-delight/queue-data-structure-practice-problems-and-interview-questions-f459bf0578db
3. Solve Problems on Stacks and Queues: https://www.interviewbit.com/courses/programming/stacks-and-queues

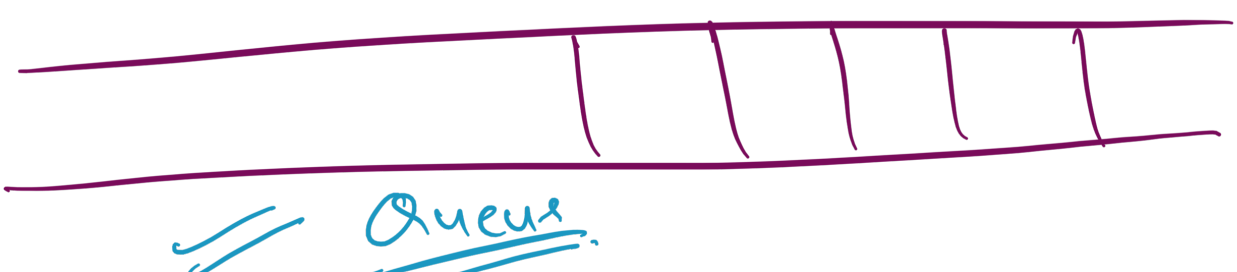
Flatten a Multilevel LL



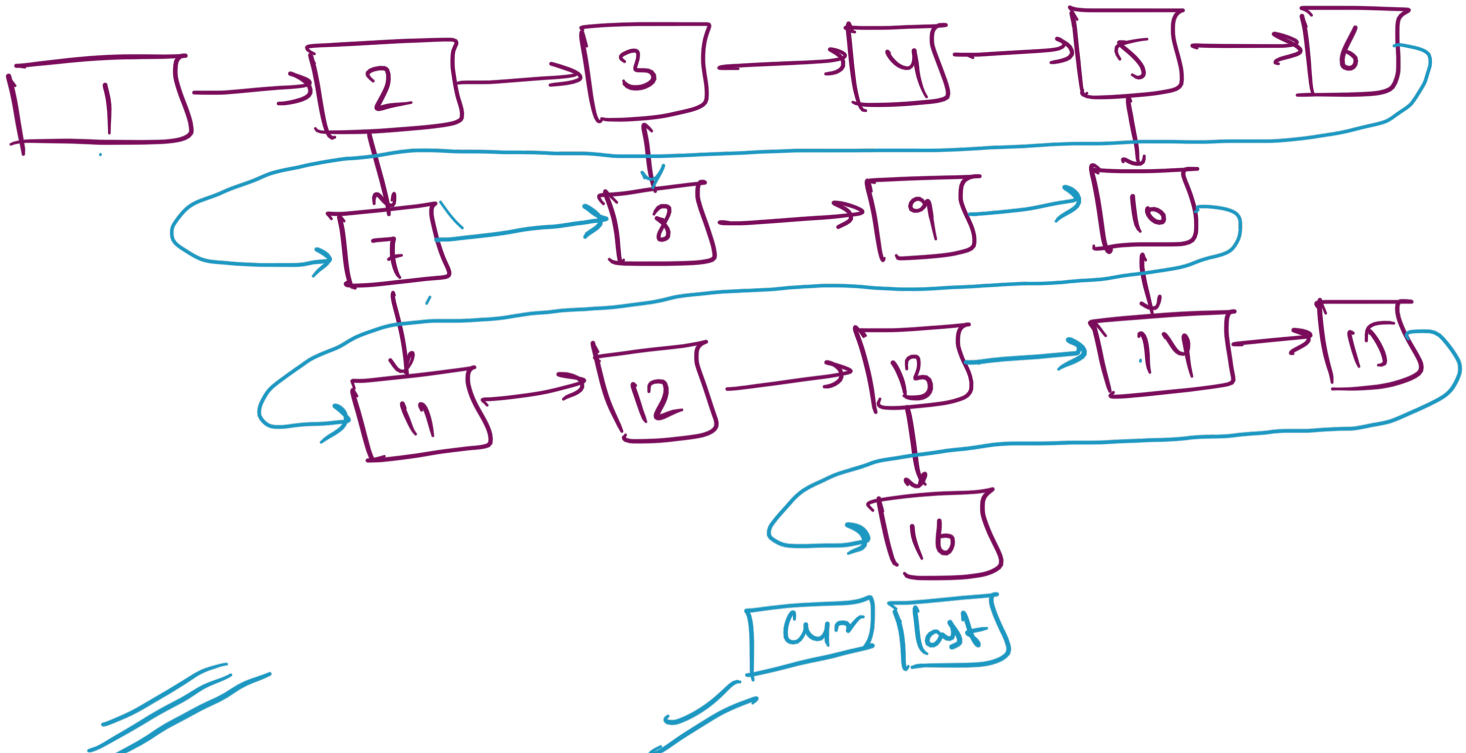

```
Node {
   int data
   Node next, down;
}
```

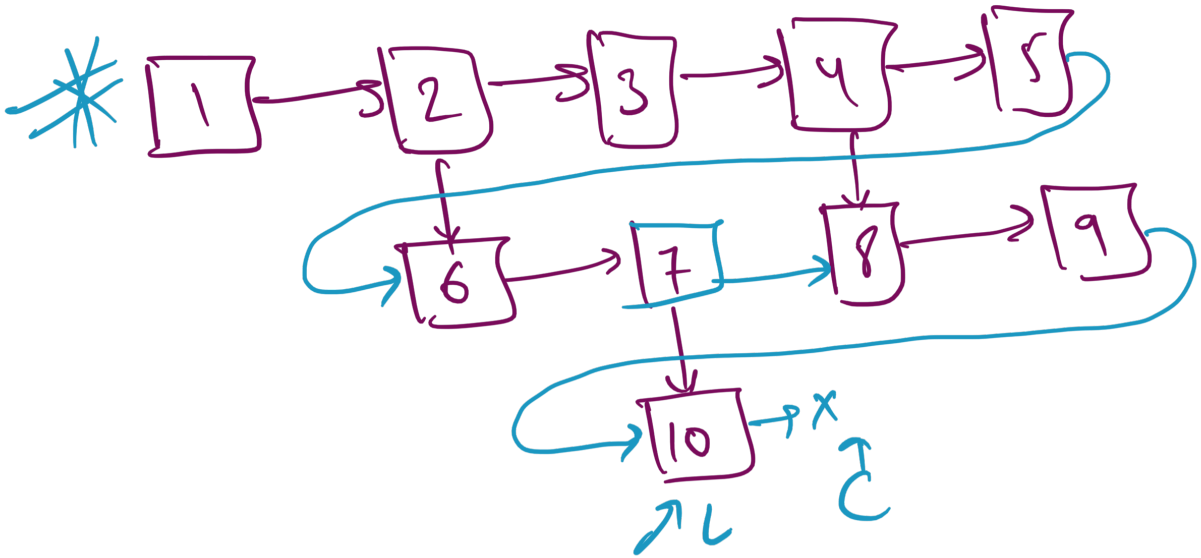
1, 2, 3 4, 5, 6, 7, 8, 9, 10, 11, 12

Queue<Node>



// Queue.


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
cur
last


1
2
3
4
5
6
7
8
9
10
x

Binary Tree Basics

Binary Tree Data Structure

Creation of a Binary Tree

Inorder Traversal in a Binary Tree

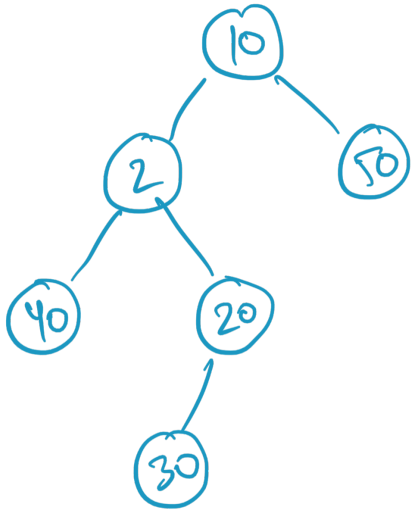
Preorder Traversal in a Binary Tree

Height of a Binary Tree

Print all the nodes present at a distance k from Root

Practice Problems

1. Postorder Binary Tree Traversal
2. Size of a Binary Tree
3. Maximum / Minimum in a Binary Tree
4. Inorder/Preorder/Postorder without using Recursion



Minimum

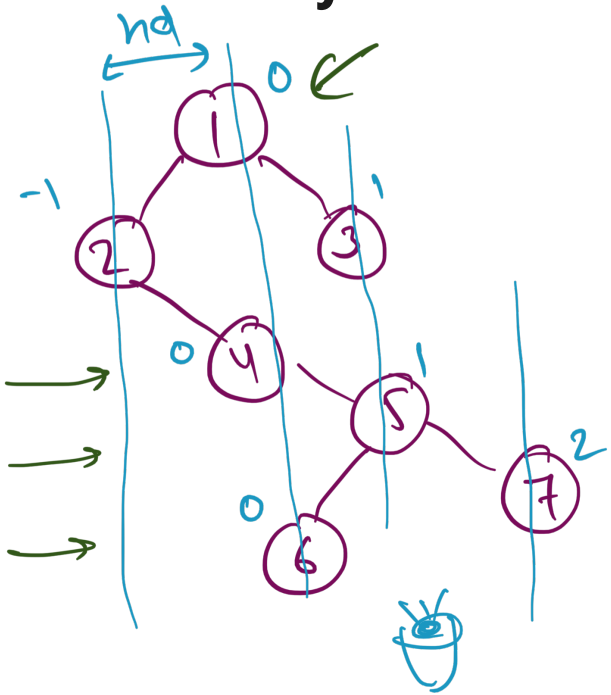
```
minTree (root) {
    if (root == null) return Max value;
    return Math.min(root.data, Math.min(
        minTree(root.left),
        minTree (root.right)
    ));
}
```

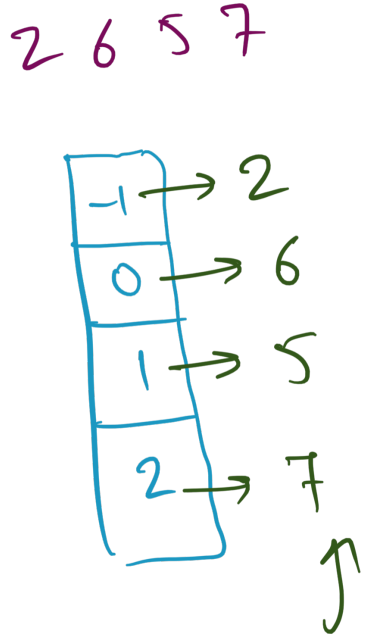
Binary Tree - I

Print Level-Order Traversal of a Binary Tree

Print Right-view of a Binary Tree

Print Top View of a Binary Tree





Practice Problems

1. Print the Zig-Zag order Traversal of a Binary Tree
2. Print the Left View of a Binary Tree
3. Print the Bottom View of a Binary Tree
4. Modified Vertical order traversal: https://leetcode.com/problems/vertical-order-traversal-of-a-binary-tree/
5. Cousins in a Binary Tree: https://leetcode.com/problems/cousins-in-binary-tree/
6. Populate the Next Pointer in Each Node: https://leetcode.com/problems/populating-next-right-pointers-in-each-node/