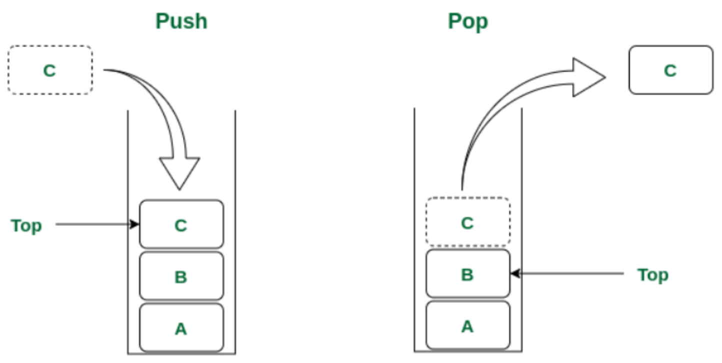
Session 3 Stacks & Queues

Dijkstra:
Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks.

A Star:
A* is a graph traversal and path search algorithm, which is used in many fields of computer science due to its completeness, optimality, and optimal efficiency.

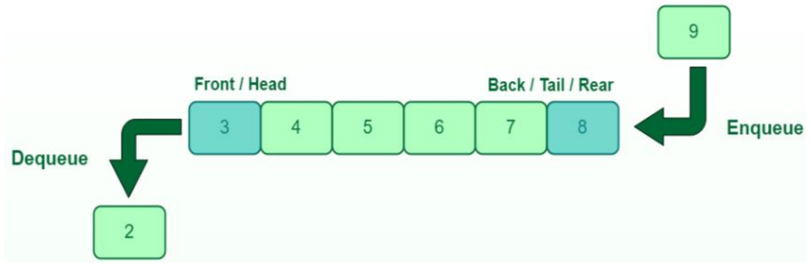
Stack:
Stack is a linear data structure that follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out).



Stack Operation:
-Pop
-Push
-Peek
-IsEmpty
-Size
They all require O(1)

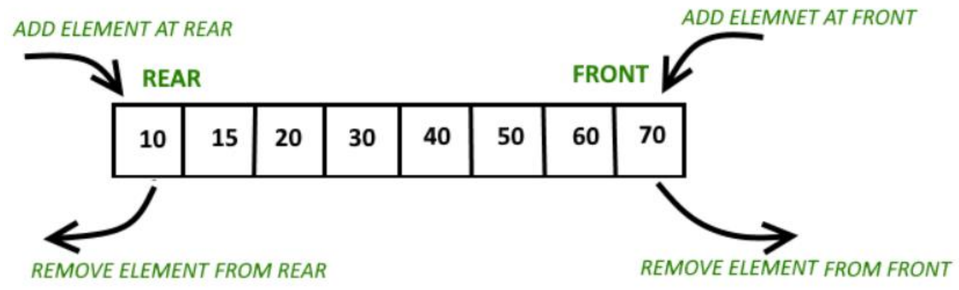
Mono Stack:
A monotonic stack is a stack whose elements are monotonically increasing or decreasing. It contains all qualities that a typical stack has and its elements are all monotonic decreasing or increasing.

Queue:
A Queue is defined as a linear data structure that is open at both ends and the operations are performed in First In First Out (FIFO) order.



Queue Operation:
-Create
-Enqueue
-Dequeue
-IsEmpty
-IsFull
-Size
They all require O(1)

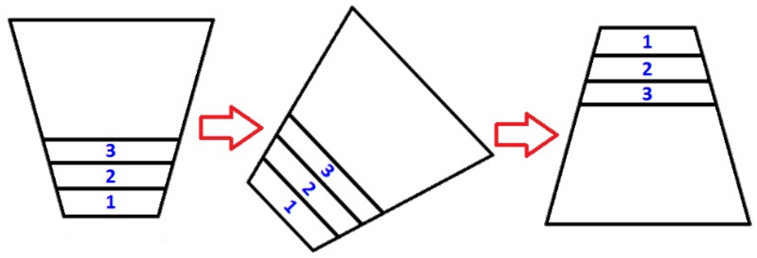
Deque:
Deque or Double Ended Queue is a generalised version of Queue data structure that allows insert and delete at both ends.



Operations on Deque:
-insertFront(): Adds an item at the front of Deque.
-insertLast(): Adds an item at the rear of Deque.
-deleteFront(): Deletes an item from the front of Deque.
-deleteLast(): Deletes an item from the rear of Deque. In addition to the above -operations, the following operations are also supported.
-getFront(): Gets the front item from the queue.
-getRear(): Gets the last item from the queue.
-isEmpty(): Checks whether Deque is empty or not.
-isFull(): Checks whether Deque is full or not.
Vector:
Vectors are the same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container. Vector elements are placed in contiguous

storage so that they can be accessed and traversed using iterators. In vectors, data is inserted at the end. Inserting at the end takes differential time, as sometimes the array may need to be extended. Removing the last element takes only constant time because no resizing happens. Inserting and erasing at the beginning or in the middle is linear in time.

<u>How can we make a queue from a stack:</u>
Use 2 stacks first one store elements are given then pop them to the second stack, this leads to formation of a queue as the sequence will flip.



<u>Find max in stack:</u>
Naive approach: O(n) we iterate each time to find the max.
Better approach: O(1) use a stack of pairs or 2 stacks one has ordered numbers once I pop the max from main I also pop it from max stack.
-Create an auxiliary stack, say ‘trackStack’ to keep the track of maximum element
-Push the first element to both mainStack and the trackStack.
-Now from the second element, push the element to the main stack. Compare the element with the top element of the track stack, if the current element is greater than the top of trackStack then push the current element to trackStack otherwise push the top element of trackStack again into it.
-If we pop an element from the main stack, then pop an element from the trackStack as well.
-Now to compute the maximum of the main stack at any point, we can simply print the top element of the Track stack.

<u>Notes:</u>
-Iterator in stack, use iterators instead of 2 stacks to go over stack elements.

<u>Problems:</u>
https://leetcode.com/problems/daily-temperatures/