

Week 11- Day 1 : Coding Challenge

(Maximum marks -15)

Q-1) Implement Queue using Stacks

<https://leetcode.com/problems/implement-queue-using-stacks/>

(7.5 marks)

(Easy)

Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the functions of a normal queue (push, peek, pop, and empty).

Implement the MyQueue class:

- `void push(int x)` Pushes element x to the back of the queue.
- `int pop()` Removes the element from the front of the queue and returns it.
- `int peek()` Returns the element at the front of the queue.
- `boolean empty()` Returns true if the queue is empty, false otherwise.

Notes:

- You must use only standard operations of a stack, which means only push to top, peek/pop from top, size, and is empty operations are valid.
- Depending on your language, the stack may not be supported natively. You may simulate a stack using a list or deque (double-ended queue) as long as you use only a stack's standard operations.

Follow-up: Can you implement the queue such that each operation is amortized $O(1)$ time complexity? In other words, performing n operations will take overall $O(n)$ time even if one of those operations may take longer.

Example 1:

Input

```
["MyQueue", "push", "push", "peek", "pop", "empty"]
```

```
[[], [1], [2], [], [], []]
```

Output

```
[null, null, null, 1, 1, false]
```

Explanation

```
MyQueue myQueue = new MyQueue();
```

```
myQueue.push(1); // queue is: [1]
```

```
myQueue.push(2); // queue is: [1, 2] (leftmost is front of the queue)
```

```
myQueue.peek(); // return 1
```

```
myQueue.pop(); // return 1, queue is [2]
```

```
myQueue.empty(); // return false
```

Q-2)Trapping Rain Water

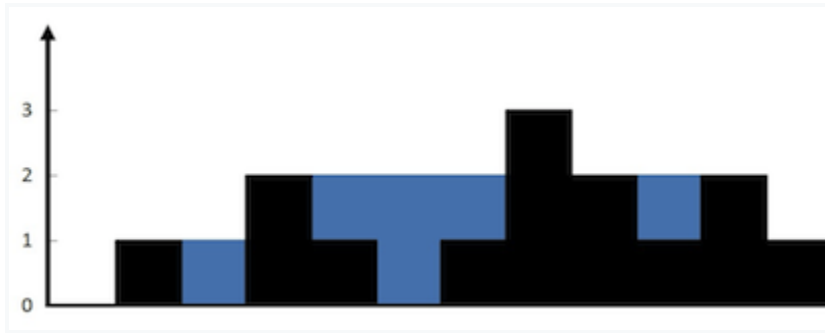
(7.5 marks)

<https://leetcode.com/problems/trapping-rain-water/>

(Hard)

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

Example 1:



Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]

Output: 6

Explanation: The above elevation map (black section) is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are being trapped.

Example 2:

Input: height = [4,2,0,3,2,5]

Output: 9

(Hint: solve using stacks to precompute L and R list, of max element on right and max element on left, respectively, for each element. Use formula $\min(\text{max_height_Right}, \text{max_height_Left}) - \text{current_height}$ to get water level. Use logic of just greatest element on the right question discussed on friday night class)

Marks distribution:

Question 1 and 2 carry 7.5 marks each.