

2023-05-06 - Handout – Stacks 2

Q1. Build an Array With Stack Operations

Link: <https://leetcode.com/problems/build-an-array-with-stack-operations/description/>

You are given an integer array `target` and an integer `n`.

You have an empty stack with the two following operations:

- `"Push"`: pushes an integer to the top of the stack.
- `"Pop"`: removes the integer on the top of the stack.

You also have a stream of the integers in the range `[1, n]`.

Use the two stack operations to make the numbers in the stack (from the bottom to the top) equal to `target`. You should follow the following rules:

- If the stream of the integers is not empty, pick the next integer from the stream and push it to the top of the stack.
- If the stack is not empty, pop the integer at the top of the stack.
- If, at any moment, the elements in the stack (from the bottom to the top) are equal to `target`, do not read new integers from the stream and do not do more operations on the stack.

Return *the stack operations needed to build* `target` following the mentioned rules. If there are multiple valid answers, return **any of them**.

Constraints: $1 \leq \text{target}[i] \leq n$, `target` is strictly increasing

Input: `target = [1,3], n = 3`

Output: `["Push","Push","Pop","Push"]`

Input: `target = [1,2,3], n = 3`

Output: `["Push","Push","Push"]`

Q2. Daily Temperature

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

Given an array of integers `temperatures` represents the daily temperatures, return *an array* `answer` such that `answer[i]` is the number of days you have to wait after the i^{th} day to get a warmer temperature. If there is no future day for which this is possible, keep `answer[i] == 0` instead.

Input: `temperatures = [73,74,75,71,69,72,76,73]`

Output: `[1,1,4,2,1,1,0,0]`

Input: `temperatures = [30,40,50,60]`

Output: `[1,1,1,0]`

Q3. Basic Calculator

Link: <https://leetcode.com/problems/basic-calculator/>

Given a string `s` representing a valid expression, implement a basic calculator to evaluate it, and return *the result of the evaluation*.

Note: You are **not** allowed to use any built-in function which evaluates strings as mathematical expressions, such as `eval()`.

Input: `s = "2-1 + 2"`

Output: 3

Input: `s = "(1+(4+5+2)-3)+(6+8)"`

Output: 23

Constraints:

- `1 <= s.length <= 3 * 105`
- `s` consists of digits, `'+'`, `'-'`, `'('`, `')'`, and `' '`.
- `s` represents a valid expression.
- `'+'` is **not** used as a unary operation (i.e., `"+1"` and `"+(2 + 3)"` is invalid).
- `'-'` could be used as a unary operation (i.e., `"-1"` and `"-(2 + 3)"` is valid).
- There will be no two consecutive operators in the input.
- Every number and running calculation will fit in a signed 32-bit integer.

Q4. Trapping Rain Water

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

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

Constraints:

- `n == height.length`
- `1 <= n <= 2 * 104`
- `0 <= height[i] <= 105`