150. Evaluate Reverse Polish Notation

Medium 3253 645 Add to List Share

Evaluate the value of an arithmetic expression in Reverse Polish Notation.

Valid operators are `+`, `-`, `*`, and `/`. Each operand may be an integer or another expression.

Note that division between two integers should truncate toward zero.

It is guaranteed that the given RPN expression is always valid. That means the expression would always evaluate to a result, and there will not be any division by zero operation.

Example 1:

```
Input: tokens = ["2","1","+","3","*"]
Output: 9
Explanation: ((2 + 1) * 3) = 9
```

Example 2:

```
Input: tokens = ["4","13","5","/","+"]
Output: 6
Explanation: (4 + (13 / 5)) = 6
```

Example 3:

```
Input: tokens =
["10","6","9","3","+","-11","*","/","*","17","+","5","+"]
Output: 22
Explanation: ((10 * (6 / ((9 + 3) * -11))) + 17) + 5
= ((10 * (6 / (12 * -11))) + 17) + 5
= ((10 * (6 / -132)) + 17) + 5
= ((10 * 0) + 17) + 5
= (0 + 17) + 5
= 17 + 5
= 22
```

Constraints:

- `1 <= tokens.length <= 10^4`
- `tokens[i]` is either an operator: `"+"`, `"-"`, `"*"`, or `"/"`, or an integer in the range `[-200, 200]`.

TimeComplexity : O(n)

Space Complexity : O(n)

For Explanation Walk through Algonotes.

1047. Remove All Adjacent Duplicates In String

Easy 3467 162 Add to List Share

You are given a string `s` consisting of lowercase English letters. A **duplicate removal** consists of choosing two **adjacent** and **equal** letters and removing them.

We repeatedly make **duplicate removals** on `s` until we no longer can.

Return *the final string after all such duplicate removals have been made*. It can be proven that the answer is **unique**.

Example 1:

```
Input: s = "abbaca"
Output: "ca"
Explanation:
For example, in "abbaca" we could remove "bb" since the letters are
adjacent and equal, and this is the only possible move.  The result
of this move is that the string is "aaca", of which only "aa" is
possible, so the final string is "ca".
```

Example 2:

```
Input: s = "azxxzy"
Output: "ay"
```

Constraints:

- `1 <= s.length <= 10^5`
- `s` consists of lowercase English letters.

TimeComplexity : O(n)

Space Complexity : O(n)

For Explanation Walk through Algonotes.

496. Next Greater Element I

Easy 3122 204 Add to List Share

The **next greater element** of some element `x` in an array is the **first greater** element that is **to the right** of `x` in the same array.

You are given two **distinct 0-indexed** integer arrays `nums1` and `nums2`, where `nums1` is a subset of `nums2`.

For each `0 <= i < nums1.length`, find the index `j` such that `nums1[i] == nums2[j]` and determine the **next greater element** of `nums2[j]` in `nums2`. If there is no next greater element, then the answer for this query is `-1`.

Return *an array* `ans` *of length* `nums1.length` *such that* `ans[i]` *is the* ***next greater element*** *as described above.*

Example 1:

```
Input: nums1 = [4,1,2], nums2 = [1,3,4,2]
Output: [-1,3,-1]
Explanation: The next greater element for each value of nums1 is as
follows:
- 4 is underlined in nums2 = [1,3,4,2]. There is no next greater
element, so the answer is -1.
- 1 is underlined in nums2 = [1,3,4,2]. The next greater element is
3.
- 2 is underlined in nums2 = [1,3,4,2]. There is no next greater
element, so the answer is -1.
```

Example 2:

```
Input: nums1 = [2,4], nums2 = [1,2,3,4]
Output: [3,-1]
Explanation: The next greater element for each value of nums1 is as
follows:
- 2 is underlined in nums2 = [1,2,3,4]. The next greater element is
3.
- 4 is underlined in nums2 = [1,2,3,4]. There is no next greater
element, so the answer is -1.
```

Constraints:

- `1 <= nums1.length <= nums2.length <= 1000`
- `0 <= nums1[i], nums2[i] <= 10^4`
- All integers in `nums1` and `nums2` are **unique**.
- All the integers of `nums1` also appear in `nums2`.

Follow up: Could you find an `O(nums1.length + nums2.length)` solution?

TimeComplexity : O(n)

Space Complexity : O(n)

For Explanation Walk through Algonotes.