Description | Editorial | Solutions (3.9K) | Submissions

1351. Count Negative Numbers in a Sorted Matrix

Hint

Easy | 3.9K | 109

Companies

Given a `m x n` matrix `grid` which is sorted in non-increasing order both row-wise and column-wise, return *the number of* ***negative*** *numbers in* `grid`.

Example 1:

```
Input: grid = [[4,3,2,-1],[3,2,1,-1],[1,1,-1,-2],[-1,-1,-2,-3]]
Output: 8
Explanation: There are 8 negatives number in the matrix.
```

Example 2:

```
Input: grid = [[3,2],[1,0]]
Output: 0
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 100`
- `-100 <= grid[i][j] <= 100`

Follow up: Could you find an `O(n + m)` solution?

Java | Auto

```java
class Solution {
    public int countNegatives(int[][] grid) {
        int result = 0;

        for(int[] a : grid) {
            result += findNegative(a);
        }
        return result;
    }
    public int findNegative(int[] a){
        if(a[0] < 0) return a.length;

        if(a[a.length - 1] >= 0) return 0;
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

grid =

[[4,3,2,-1],[3,2,1,-1],[1,1,-1,-2],[-1,-1,-2,-3]]

Output

8

Expected

8

Console | Run | Submit

Description | Editorial | Solutions (12K) | Submissions

19. Remove Nth Node From End of List

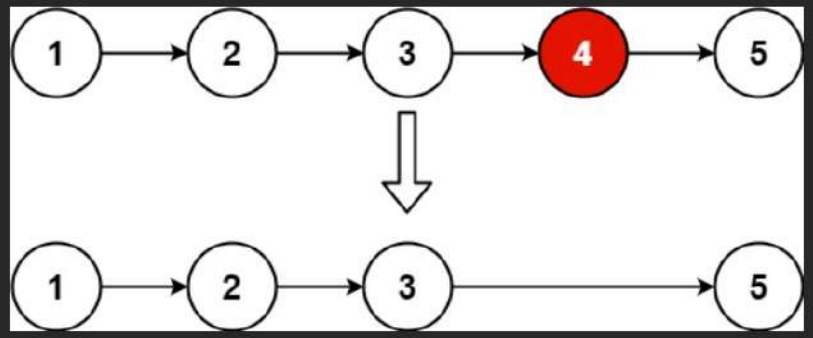
Hint

Medium | 15.9K | 664

Companies

Given the `head` of a linked list, remove the `n^th` node from the end of the list and return its head.

Example 1:




```
Input: head = [1,2,3,4,5], n = 2
Output: [1,2,3,5]
```

Example 2:

```
Input: head = [1], n = 1
Output: []
```

Example 3:

```
Input: head = [1,2], n = 1
Output: [1]
```

Java | Auto

```
10  */
11 class Solution {
12     public ListNode removeNthFromEnd(ListNode head, int n) {
13         ListNode slow = head, fast = head;
14         for(int i=1; i<=n; i++){
15             fast = fast.next;
16         }
17
18         if(fast == null){
19             return head.next;
20         }
21
22         while(fast != null && fast.next != null){
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

head =
[1,2,3,4,5]

n =
2

Output

[1,2,3,5]

Console | Run | Submit

Description | Editorial | Solutions (3.4K) | Submissions

18. 4Sum

Medium | 9.4K | 1.1K

Companies

Given an array `nums` of `n` integers, return *an array of all the **unique** quadruplets* `[nums[a], nums[b], nums[c], nums[d]]` such that:

- `0 <= a, b, c, d < n`
- `a`, `b`, `c`, and `d` are **distinct**.
- `nums[a] + nums[b] + nums[c] + nums[d] == target`

You may return the answer in **any order**.

Example 1:

```
Input: nums = [1,0,-1,0,-2,2], target = 0
Output: [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]
```

Example 2:

```
Input: nums = [2,2,2,2,2], target = 8
Output: [[2,2,2,2]]
```

Constraints:

- `1 <= nums.length <= 200`
- `-10^9 <= nums[i] <= 10^9`
- `-10^9 <= target <= 10^9`

Java | Auto

```java
class Solution {
    public List<List<Integer>> fourSum(int[] nums, int target) {
        List<List<Integer>> quadruplets = new ArrayList<>();
        int n = nums.length;
        // Sorting the array
        Arrays.sort(nums);
        for (int i = 0; i < n - 3; i++) {
            // Skip duplicates
            if (i > 0 && nums[i] == nums[i - 1]) {
                continue;
            }
            for (int j = i + 1; j < n - 2; j++) {
                // Skip duplicates
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

nums =
[1,0,-1,0,-2,2]

target =
0

Output

[[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]

Console | Run | Submit