15. 3Sum

Medium 20889 1944 Add to List Share

Given an integer array nums, return all the triplets `[nums[i], nums[j], nums[k]]` such that `i != j, i != k,` and `j != k,` and `nums[i] + nums[j] + nums[k] == 0`.

Notice that the solution set must not contain duplicate triplets.

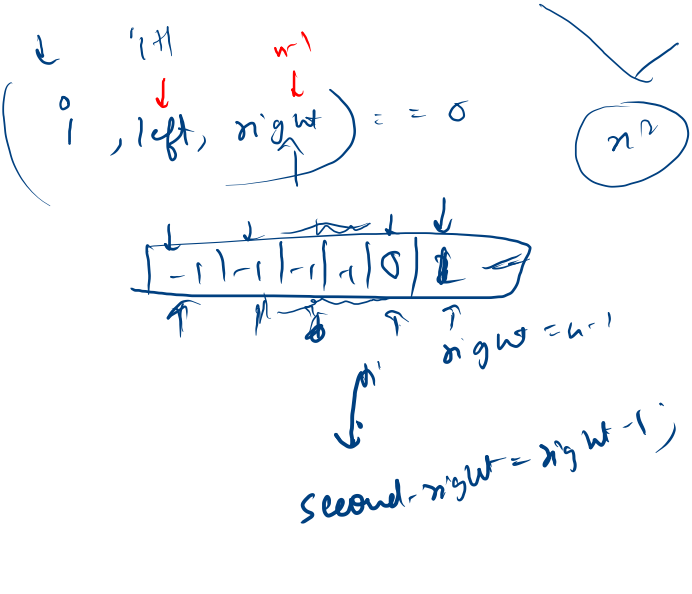
Example 1:

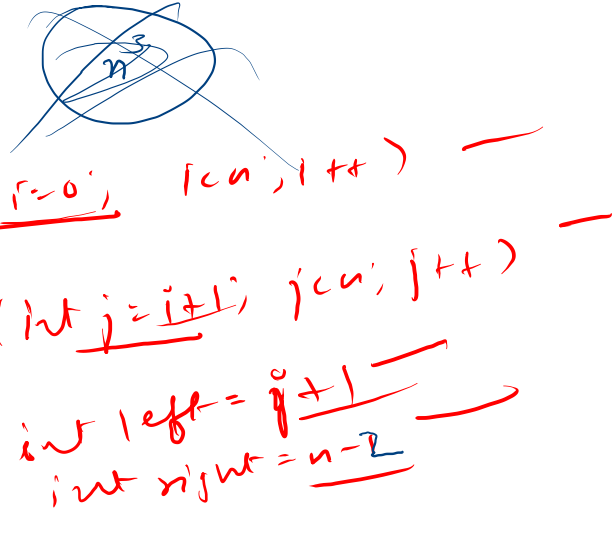
```
Input: nums = [-1,0,1,2,-1,-4]
Output: [[-1,-1,2],[-1,0,1]]
Explanation:
nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0.
nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0.
nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0.
The distinct triplets are [-1,0,1] and [-1,-1,2].
Notice that the order of the output and the order of the triplets does not matter.
```

2 elements

```
public List<List<Integer>> threeSum(int[] nums) {
    int n=nums.length;

    Arrays.sort(nums);

    for(int i=0;i<n;i++){
        int left=i+1;
        int right=n-1;

        findPair(nums,i,left,right);

        while(i+1<n && nums[i]==nums[i+1]){
            i++;
        }
    }

    return null;
}
```

```
public void findPair(int [] nums,int i,int left,int right){
    while(left<right){
        int sum=nums[i]+nums[left]+nums[right];

        if(sum==0){
            System.out.println(nums[i]+" "+nums[left]+" "+nums[right]);
            while(left+1<right && nums[left]==nums[left+1]){
                left++;
            }

            while(right-1>left && nums[right]==nums[right-1]){
                right--;
            }

            left++;
            right--;
        }else if(sum<0){
            left++;
        }else {
            right--;
        }
    }
}
```



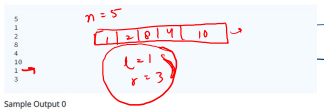


Print Prefix Sum between L and R

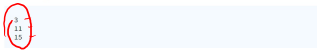
Problem | Submissions | Leaderboard | Discussions

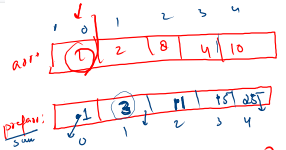
Take an integer input l and r such that (r<=array.length) Given an array Make a prefix sum array from this. The print the sum of the elements inside the array starting from the l-index till the r-index(l and r both inclusive)

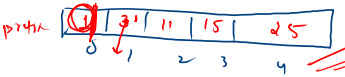
Sample Input 0



Sample Output 0







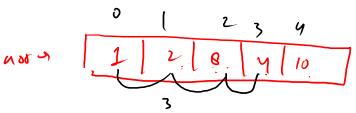
i=0 → j=0 c=0 → 1

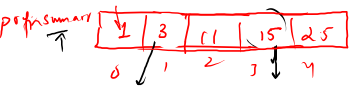
i=1 → j=0 c=1 → sum = 1+2 = 3
j=1 c=1
j=2 c=1

prefix[i] = prefix[i-1] + arr[i];
= 1 + 2 = 3

i=2 ⇒ 3+8=11
i=3 ⇒ 11+4=15
i=4 ⇒ 15+10=25

l=1, r=3





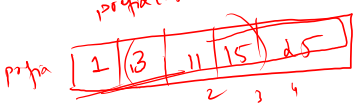
prefix max

```
for(int i=0; i<n; i++)
{
    for(int j=0; j<=i; j++)
    {
        int sum = arr[j];
    }
}
```

O(n²)

i=0; j=0 c=0 ⇒ sum=1
j=1 c=0 ✗
i=1; j=0 c=1 ⇒ sum=1
j=1 c=1 ⇒ sum=1+2=3
j=2 c=1
i=0 j=1
i=1 j=2
i=2 j=3
n²

prefix[0] = arr[0]



i=1 2 3 4

prefix[i] = prefix[i-1] + arr[i]
= 1 + 2 = 3
= 3 + 8 = 11

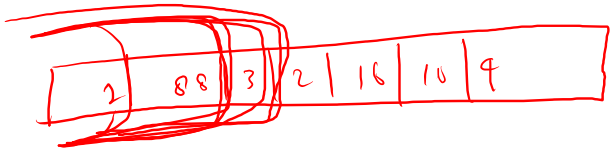
⇒ 11+4=15
⇒ 15+10=25

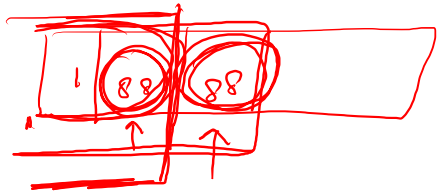
O(n)

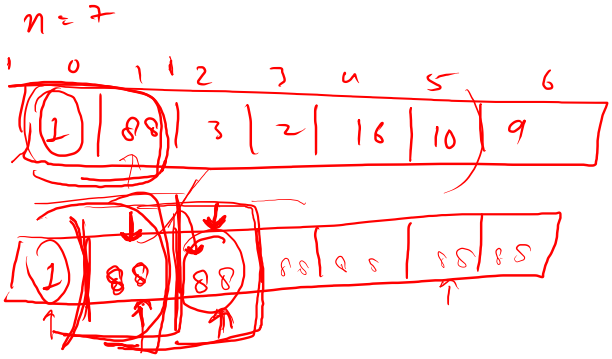
(l,r) → (1,3)

3
11
15

```
7
1
88
3
2
16
10
9
```







prefix

prefix[i] = Math.max(prefix[i-1], a[i])