15. 3Sum

Given numbers, return the combination whose sum is 0, the res should not contain duplicate triplets.

```
class Solution:
    def threeSum(self, nums):
        """
        :type nums: List[int]
        :rtype: List[List[int]]
        """
        nums.sort()
        res = []
        l = len(nums)
        for i in range(l-2):
            if i == 0 or nums[i] != nums[i-1]:
                p1 = i+1
                p2 = l-1
                while p1 < p2:
                    if p1 == i+1 or nums[p1] != nums[p1-1]:
                        if nums[i] + nums[p1] + nums[p2] > 0:
                            p2 = p2 -1
                        elif nums[i] + nums[p1] + nums[p2] < 0:
                            p1 = p1 +1
                        else:
                            res.append([nums[i] , nums[p1] , nums[p2]])
                            p1 = p1+1
                    else:
                        p1=p1+1
            else:
                i=i+1
        return res
```

16. 3Sum Closest

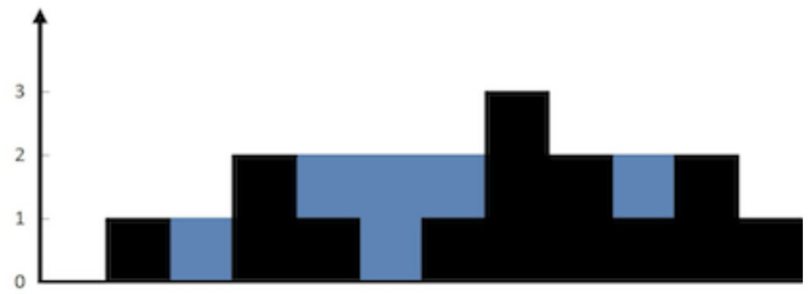
what is the absolute value in python?

answer: abs()

```
class Solution:
    def threeSumClosest(self, nums, target):
        """
        :type nums: List[int]
        :type target: int
        :rtype: int
        """
        nums.sort()

        min_distance = abs(nums[0] + nums[1] + nums[2] - target)
        res = nums[0] + nums[1] + nums[2]

        for i in range(len(nums) - 2):

            l = i + 1
            r = len(nums) - 1

            while l < r:
                sum_ = nums[i] + nums[l] + nums[r]
                if abs(sum_ - target) < min_distance:
                    min_distance = abs(sum_ - target)
                    res = sum_

                if sum_ < target:
                    l = l+1
                elif sum_ > target:
                    r = r-1
                else:
                    return target

        return res
```

42. Trapping Rain Water

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



The above elevation map 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. **Thanks Marcos** for contributing this image!

Example:

```
Input: [0,1,0,2,1,0,1,3,2,1,2,1]
Output: 6
```

use two pointers, the time complexity is O(N), space is O(1)

---

class Solution: def trap(self, height): """ :type height: List[int] :rtype: int 0,1,0,2,1,0,1,3,2,1,2,1 i j left_max = 0 right_max = 1

```
    so we firstly move i forward,
    """

    # use two pointers still
    if len(height) == 0 or len(height) == 1:
        return  0

    i = 0
    j = len(height) - 1

    right_max = height[j]
    left_max = height[i]
    area = 0

    while i < j:
        if left_max<= right_max:
            i = i+1
            if height[i] < left_max :
                area = area + left_max - height[i]
            else:
                left_max = height[i]
        else:

            j = j-1
            if height[j] < right_max:
                area = area + right_max - height[j]
            else:
                right_max = height[j]

    return area
```

48. Rotate Image

```
class Solution:
    def rotate(self, matrix):
        """
        :type matrix: List[List[int]]
        :rtype: void Do not return anything, modify matrix in-place instead.
        """
        matrix.reverse()
        for i in range(len(matrix)):
            for j in range(i):
                matrix[i][j], matrix[j][i] = matrix[j][i], matrix[i][j]
```

56. Merge Intervals

Given a collection of intervals, merge all overlapping intervals.

Example 1:

```
Input: [[1,3],[2,6],[8,10],[15,18]]
Output: [[1,6],[8,10],[15,18]]
Explanation: Since intervals [1,3] and [2,6] overlaps, merge them into [1,6].
```

Example 2:

```
Input: [[1,4],[4,5]]
Output: [[1,5]]
Explanation: Intervals [1,4] and [4,5] are considered overlapping.
```

1. how to sort by the first number in python?

sorted([('abc', 121),('abc', 231),('abc', 148), ('abc',221)], key=lambda x: x[1])

```
# Definition for an interval.
# class Interval:
#     def __init__(self, s=0, e=0):
#         self.start = s
#         self.end = e

class Solution:
    def merge(self, intervals):
        """
        :type intervals: List[Interval]
        :rtype: List[Interval]
        """
        if len(intervals) == 0 or len(intervals) == 1:
            return  intervals

        intervals = sorted(intervals, key = lambda x:x.start)
        start = intervals[0].start
        end = intervals[0].end
        res = []

        for i in range(1, len(intervals)):
            if intervals[i].start > end:
                res.append(Interval(start, end))
                start = intervals[i].start
                end = intervals[i].end
            else:
                end = max(end, intervals[i].end)

        res.append(Interval(start,end))
        return res
```

152. Maximum Product Subarray

```
class Solution:
    def maxProduct(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        [2,3,-2,4]


        """

        min_ = nums[0]
        max_ = nums[0]
        res = nums[0]

        for i in range(1,len(nums)):
            num = nums[i]
            if num < 0:
                temp = min_
                min_ = max_
                max_ = temp

            max_ = max(max_*num, num)
            min_ = min(min_*num, num)
            res = max(max_, res)
        return res
```

238. Product of Array Except Self

```
class Solution:
    def productExceptSelf(self, nums):
        """
        :type nums: List[int]
        :rtype: List[int]
        1 2 3 4
        1 1 2 6
        24  12  4  1
        """

        res = []
        cur = 1
        l = len(nums)
        for i in range(l-1):
            res.append(cur)
            cur = cur * nums[i]
        res.append(cur)

        cur = 1
        for i in range(l-1, 0, -1):
            res[i] = res[i] * cur
            cur = cur * nums[i]
        res[0] = res[0] * cur

        return res
```