561 Array Partition I

2018年4月15日 17:36

Given an array of **2n** integers, your task is to group these integers into **n** pairs of integer, say (a_1, b_1), (a_2, b_2), ..., (a_n, b_n) which makes sum of $\min(a_i, b_i)$ for all i from 1 to n as large as possible.

Example 1:

Input: [1,4,3,2]

Output: 4

Explanation: n is 2, and the maximum sum of pairs is 4 = min(1, 2) + min(3, 4).

Note:

1. **n** is a positive integer, which is in the range of [1, 10000].
2. All the integers in the array will be in the range of [-10000, 10000].

来自 <https://leetcode.com/problems/array-partition-i/description/>

给定长度为 **2n** 的数组, 你的任务是将这些数分成 **n** 对, 例如 (a_1, b_1), (a_2, b_2), ..., (a_n, b_n) ， 使得从1到 n 的 $\min(a_i, b_i)$ 总和最大。

示例 1:

输入: [1,4,3,2]

输出: 4

解释: n 等于 2, 最大总和为 4 = min(1, 2) + min(3, 4).

提示:

1. **n** 是正整数,范围在 [1, 10000].
2. 数组中的元素范围在 [-10000, 10000].

Solution for Python3:

```python
class Solution1:
    def arrayPairSum(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        nums.sort()
        sum = 0
        for i in range(0, len(nums), 2):
            sum += nums[i]
        return sum
```

```
12
13  class Solution2:
14      def arrayPairSum(self, nums):
15          """
16          :type nums: List[int]
17          :rtype: int
18          """
19          nums.sort()
20          return sum(nums[x] for x in range(0,
21  len(nums), 2))
22
23  class Solution3:
24      def arrayPairSum(self, nums):
25          """
26          :type nums: List[int]
27          :rtype: int
28          """
            return sum(sorted(nums)[::2])
```

Solution for C++:

```
 1  class Solution1 {
 2  public:
 3      int arrayPairSum(vector<int>& nums) {
 4          sort(nums.begin(), nums.end());
 5          int sum = 0;
 6          for (int i = 0; i < nums.size(); i += 2)
 7              sum += nums[i];
 8          return sum;
 9      }
10  };
11
12  class Solution2 {
13  public:
14      int arrayPairSum(vector<int>& nums) {
15          vector<int> hastable(20001, 0);
16          for (int num : nums)
17              hastable[num + 10000]++;
18          int flag = 1, res = 0;
19          for (int i = 0; i < 20001;) {
```

```
            if (hastable[i] > 0) {
                if (flag) {
                    flag = 0;
                    res += i - 10000;
                    --hastable[i];
                } else {
                    flag = 1;
                    --hastable[i];
                }
            } else {
                ++i;
            }
        }
        return res;
    }
};
```