

## Array Partition I

Given an array of  $2n$  integers, your task is to group these integers into  $n$  pairs of integer, say  $(a_1, b_1), (a_2, b_2), \dots, (a_n, b_n)$  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.

### 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  $[-10000, 10000]$ .

## Solution 1

The algorithm is first sort the input array and then the sum of 1st, 3rd, 5th..., is the answer.

```
public class Solution {  
    public int arrayPairSum(int[] nums) {  
        Arrays.sort(nums);  
        int result = 0;  
        for (int i = 0; i < nums.length; i += 2) {  
            result += nums[i];  
        }  
        return result;  
    }  
}
```

Let me try to prove the algorithm...

1. Assume in each pair  $i$ ,  $b_i \geq a_i$ .
2. Denote  $S_m = \min(a_1, b_1) + \min(a_2, b_2) + \dots + \min(a_n, b_n)$ . The biggest  $S_m$  is the answer of this problem. Given 1,  $S_m = a_1 + a_2 + \dots + a_n$ .
3. Denote  $S_a = a_1 + b_1 + a_2 + b_2 + \dots + a_n + b_n$ .  $S_a$  is constant for a given input.
4. Denote  $d_i = |a_i - b_i|$ . Given 1,  $d_i = b_i - a_i$ . Denote  $S_d = d_1 + d_2 + \dots + d_n$ .
5. So  $S_a = a_1 + a_1 + d_1 + a_2 + a_2 + d_2 + \dots + a_n + a_n + d_n = 2S_m + S_d \Rightarrow S_m = (S_a - S_d) / 2$ . To get the max  $S_m$ , given  $S_a$  is constant, we need to make  $S_d$  as small as possible.
6. So this problem becomes finding pairs in an array that makes sum of  $d_i$  (distance between  $a_i$  and  $b_i$ ) as small as possible. Apparently, sum of these distances of adjacent elements is the smallest. If that's not intuitive enough, see attached picture. Case 1 has the smallest  $S_d$ .



written by [shawngao](#) original link [here](#)

## Solution 2

So if we are given [1.2.3.4], is this a possibility?

(1,4) and (2,3)

In that case, won't the correct answer be 5. Our goal is to get the largest possible sum from the above 4 integers. So, how can 4 be the answer?

Thanks

written by [djordan](#) original link [here](#)

## Solution 3

```
class Solution(object):  
  
    def arrayPairSum(self, nums):  
        """  
        :type nums: List[int]  
        :rtype: int  
        """  
        return sum(sorted(nums)[:2])
```

written by [ecdubs](#) original link [here](#)

From [Leetcode](#).