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), ..., (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.

```java
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`, `bi >= ai`.
2. Denote `Sm = min(a1, b1) + min(a2, b2) + ... + min(an, bn)`. The biggest `Sm` is the answer of this problem. Given `1`, `Sm = a1 + a2 + ... + an`.
3. Denote `Sa = a1 + b1 + a2 + b2 + ... + an + bn`. `Sa` is constant for a given input.
4. Denote `di = |ai - bi|`. Given `1`, `di = bi - ai`. Denote `Sd = d1 + d2 + ... + dn`.
5. So `Sa = a1 + a1 + d1 + a2 + a2 + d2 + ... + an + an + di = 2Sm + Sd => Sm = (Sa - Sd) / 2`. To get the max `Sm`, given `Sa` is constant, we need to make `Sd` as small as possible.
6. So this problem becomes finding pairs in an array that makes sum of `di` (distance between `ai` and `bi`) 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 `Sd`.

written by shawngao original link here

Solution 2

```
class Solution(object):

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

written by ecdubs original link here

Solution 3

```
class Solution {
public:
    int arrayPairSum(vector<int>& nums) {
        vector<int> hashtable(20001,0);
        for(size_t i=0;i<nums.size();i++)
        {
            hashtable[nums[i]+10000]++;
        }
        int ret=0;
        int flag=0;
        for(size_t i=0;i<20001;){
            if((hashtable[i]>0)&&(flag==0)){
                ret=ret+i-10000;
                flag=1;
                hashtable[i]--;
            }else if((hashtable[i]>0)&&(flag==1)){
                hashtable[i]--;
                flag=0;
            }else i++;
        }
        return ret;
    }
};
```

with the range of numbers,it is easy to using vector,and if we don't know the range of numbers,maybe using STL multiset,but using multiset is O(nlogn).

written by disillusion028 original link here