575 Distribute Candies

2018年4月15日 20:54

Given an integer array with **even** length, where different numbers in this array represent different **kinds** of candies. Each number means one candy of the corresponding kind. You need to distribute these candies **equally** in number to brother and sister. Return the maximum number of **kinds** of candies the sister could gain.

Example 1:

Input: candies = [1,1,2,2,3,3]

Output: 3

Explanation:

There are three different kinds of candies (1, 2 and 3), and two candies for each kind.
Optimal distribution: The sister has candies [1,2,3] and the brother has candies [1,2,3], too.
The sister has three different kinds of candies.

Example 2:

Input: candies = [1,1,2,3]

Output: 2

Explanation: For example, the sister has candies [2,3] and the brother has candies [1,1].
The sister has two different kinds of candies, the brother has only one kind of candies.

Note:

1. The length of the given array is in range [2, 10,000], and will be even.
2. The number in given array is in range [-100,000, 100,000].

来自 <https://leetcode.com/problems/distribute-candies/description/>

给定一个**偶数**长度的数组，其中不同的数字代表着不同种类的糖果，每一个数字代表一个糖果。你需要把这些糖果**平均**分给一个弟弟和一个妹妹。返回妹妹可以获得的最大糖果的种类数。

注意:

1. 数组的长度为[2, 10,000]，并且确定为偶数。
2. 数组中数字的大小在范围[-100,000, 100,000]内。

Solution for Python3:

```python
class Solution1:
    def distributeCandies(self, candies):
        """
        :type candies: List[int]
        :rtype: int
        """
        res = len(collections.Counter(candies))
        if res > len(candies) // 2:
           return len(candies) // 2
        return res

class Solution2:
    def distributeCandies(self, candies):
        """
        :type candies: List[int]
        :rtype: int
        """
        # s = set(candies)
        # return min(len(s), len(candies) // 2)
        return min(len(candies) // 2,
```

```
        len(set(candies)))
```

Solution for C++:

```cpp
class Solution1 {
public:
    int distributeCandies(vector<int>& candies) {
        unordered_set<int> uniqueSet(candies.begin(), candies.end());
        return min(uniqueSet.size(), candies.size()/2);
    }
};

class Solution2 {
public:
    int distributeCandies(vector<int>& candies) {
        bitset<200001> hash;
        for (int i : candies)
            hash.set(i + 100000);
        return min(hash.count(), candies.size()/2);
    }
};

class Solution3 {
public:
    int distributeCandies(vector<int>& candies) {
        bitset<200001> hash;
        int cnt = 0;
        for (int i : candies) {
            if (!hash.test(i + 100000)) {
                cnt++;
                hash.set(i + 100000);
            }
        }
        return min(cnt, int(candies.size()/2));
    }
};
```

Appendix:

C++ bitset:存储二进制数位。

1) 每个元素都能单独被访问。
2) 整数类型和布尔数组都能转化成bitset。
3) 大小在编译时就需要确定。如果你想要不确定长度的bitset，请使用（奇葩的）vector<bool>。
4) 定义：

bitset<16> foo;	0000000000000000
bitset<16> bar (0xfa2);	0000111110100010
bitset<16> baz (string("0101111001"));	0000000101111001

5) 运算：bitset的运算就像一个普通的整数一样，可以进行与(&)、或(|)、异或(^)、左移(<<)、右移(>>)等操作。

6) 函数：bitset<16> foo;

foo.size()	返回大小（位数）
foo.count()	返回1的个数
foo.any()	返回是否有1
foo.none()	返回是否没有1
Foo.set()	全都设为1
Foo.reset()	全都变为0
Foo.set(p)	第p位设为1(index从0开始)
Foo.reset(p)	第p位变为0
foo.set(p,x)	第p位设为x
foo.test(p)	返回第p位是否为1
foo.flip()	全都取反
foo.flip(p)	将第p位取反
foo.to_ulong()	返回它转换为unsigned long的结果，超出报错
foo.to_ullong()	返回它转换为unsigned long long的结果，超出报错
foo.to_string()	返回它转换为string的结果