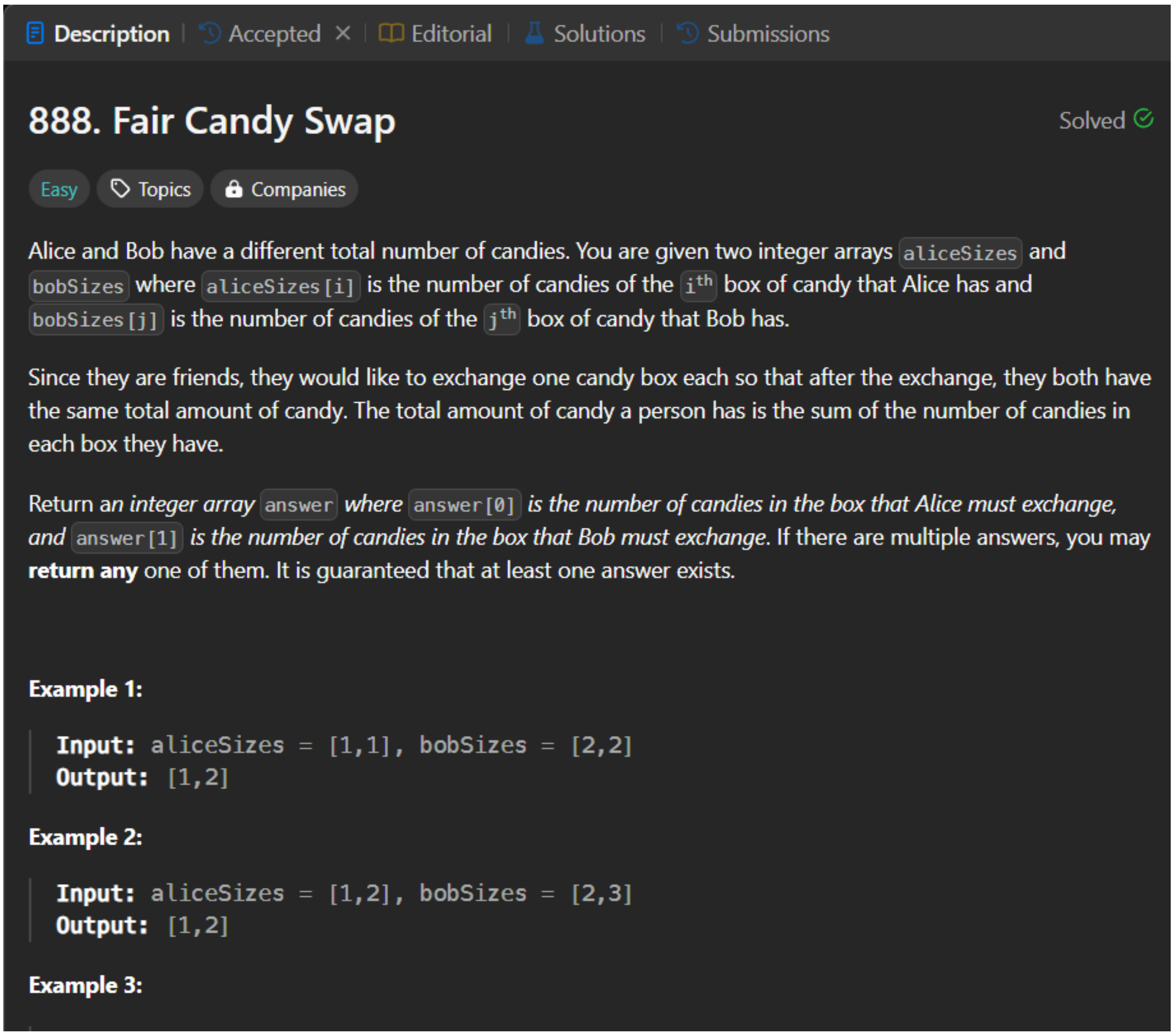


888. Fair Candy Swap

Alice and Bob have a different total number of candies. You are given two integer arrays `aliceSizes` and `bobSizes` where `aliceSizes[i]` is the number of candies of the `i`th box of candy that Alice has and `bobSizes[j]` is the number of candies of the `j`th box of candy that Bob has.

Since they are friends, they would like to exchange one candy box each so that after the exchange, they both have the same total amount of candy. The total amount of candy a person has is the sum of the number of candies in each box they have.

Return an *integer array* `answer` *where* `answer[0]` *is the number of candies in the box that Alice must exchange, and* `answer[1]` *is the number of candies in the box that Bob must exchange*. If there are multiple answers, you may **return any** one of them. It is guaranteed that at least one answer exists.

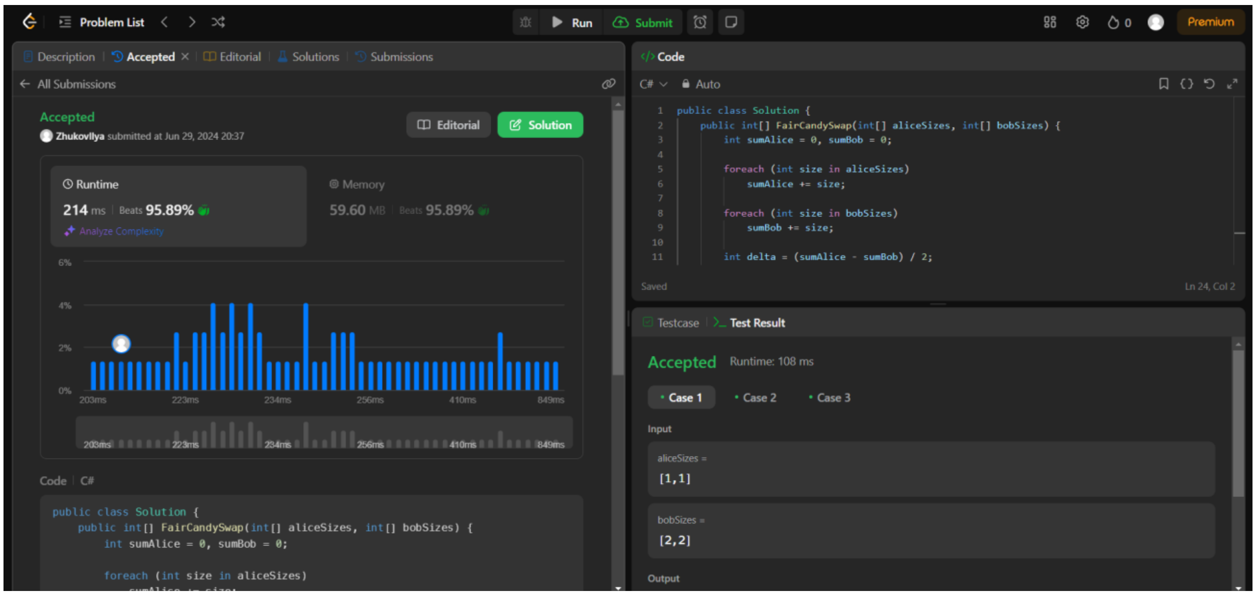
Example 1:

```
Input: aliceSizes = [1,1], bobSizes = [2,2]
Output: [1,2]
```

Example 2:

```
Input: aliceSizes = [1,2], bobSizes = [2,3]
Output: [1,2]
```

Example 3:



Accepted

Zhukovllya submitted at Jun 29, 2024 20:37

Runtime
214 ms | Beats 95.89%

Memory
59.60 MB | Beats 95.89%

Code | C#

```
public class Solution {
    public int[] FairCandySwap(int[] aliceSizes, int[] bobSizes) {
        int sumAlice = 0, sumBob = 0;

        foreach (int size in aliceSizes)
```

```
public class Solution {
    public int[] FairCandySwap(int[] aliceSizes, int[] bobSizes) {
        int sumAlice = 0, sumBob = 0;

        foreach (int size in aliceSizes)
            sumAlice += size;

        foreach (int size in bobSizes)
            sumBob += size;

        int delta = (sumAlice - sumBob) / 2;
```

Saved   Ln 24, Col 2

Accepted   Runtime: 108 ms

Input

aliceSizes =
[1,1]

bobSizes =
[2,2]

Output

Код:

```
public class Solution
{
    public int[] FairCandySwap(int[] aliceSizes, int[] bobSizes)
    {
        int sumAlice = 0, sumBob = 0;
        foreach (int size in aliceSizes)
            sumAlice += size;

        foreach (int size in bobSizes)
            sumBob += size;

        int delta = (sumAlice - sumBob) / 2;

        HashSet<int> setAlice = new HashSet<int>(aliceSizes);

        foreach (int y in bobSizes)
        {
            int x = y + delta;
            if (setAlice.Contains(x))
            {
                return new int[] { x, y };
            }
        }

        throw new ArgumentException("No valid swap found");
    }
}
```