```
class Solution(object):
    def fairCandySwap(self, A, B):
        Sa, Sb = sum(A), sum(B)
        setB = set(B)
        for x in A:
            if x + (Sb - Sa) / 2 in setB:
                return [x, x + (Sb - Sa) / 2]
```

Quick Navigation

Average Rating: 4.45 (51 votes)

Solution

Approach 1: Solve the Equation

Intuition

If Alice swaps candy `x`, she expects some specific quantity of candy `y` back.

Algorithm

Say Alice and Bob have total candy S_A, S_B respectively.

If Alice gives candy x, and receives candy y, then Bob receives candy x and gives candy y. Then, we must have

$$S_A - x + y = S_B - y + x$$

for a fair candy swap. This implies

$$y = x + \frac{S_B - S_A}{2}$$

Our strategy is simple. For every candy x that Alice has, if Bob has candy $y = x + \frac{S_B - S_A}{2}$, we return $[x, y]$. We use a `Set` structure to check whether Bob has the desired candy y in constant time.

Java Python Copy

```
class Solution(object):
    def fairCandySwap(self, A, B):
        Sa, Sb = sum(A), sum(B)
        setB = set(B)
        for x in A:
            if x + (Sb - Sa) / 2 in setB:
                return [x, x + (Sb - Sa) / 2]
```

Complexity Analysis

- Time Complexity: $O(A.\text{length} + B.\text{length})$.
- Space Complexity: $O(B.\text{length})$, the space used by `setB`. (We can improve this to $\min(A.\text{length}, B.\text{length})$ by using an if statement.)

Report Article Issue

Testcase Run Code Result Debugger

Accepted Runtime: 32 ms

Your input
[1,1]
[2,2]

Output [1,2] Diff

Expected [1,2]

Console Use Example Testcases