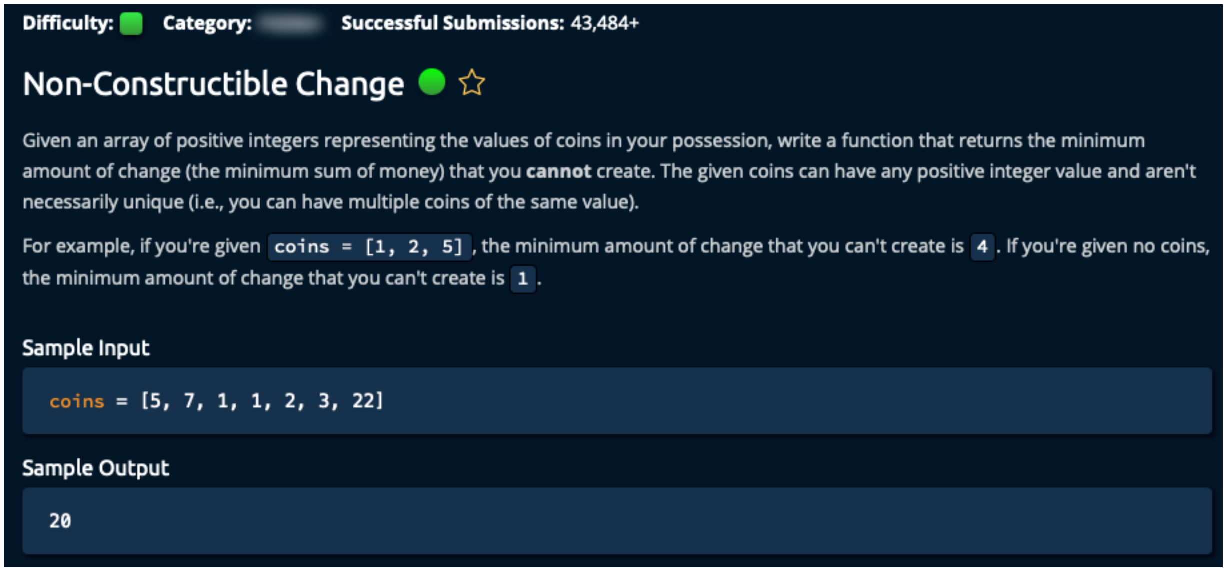
Code Challenge #18 Non-Constructible Change



Difficulty: 🟩 **Category:** [redacted] **Successful Submissions:** 43,484+

Non-Constructible Change

Given an array of positive integers representing the values of coins in your possession, write a function that returns the minimum amount of change (the minimum sum of money) that you **cannot** create. The given coins can have any positive integer value and aren't necessarily unique (i.e., you can have multiple coins of the same value).

For example, if you're given `coins = [1, 2, 5]`, the minimum amount of change that you can't create is `4`. If you're given no coins, the minimum amount of change that you can't create is `1`.

Sample Input

```
coins = [5, 7, 1, 1, 2, 3, 22]
```

Sample Output

```
20
```

```
function nonConstructibleChange(coins) {
coins.sort((a, b) => a - b);

 let currentChangeCreated = 0;
 for (const coin of coins) {
         if (coin > currentChangeCreated +1) return currentChangeCreated + 1;

         currentChangeCreated += coin;
 }

         return currentChangeCreated + 1;
 }

```

Explanation

This problem asks you to find the minimum amount of change you can create given a set number of coins. The coins don't have to be unique and are positive values only. The approach to this problem is to sort the coins using .sort((a – b) => a – b). We then create a let variable called currentChangeCreated that is equal to zero. We then use a for loop to iterate through the coins and check if the coin is greater than the currentChangeCreated amount + 1. If it is we return currentChangeCreated + 1. If not, we add the sum of the coins to currentChangeCreated. The final part of the method will return currentChangeCreated + 1 if we go through the entire array without triggering the previous return statement.