6062. Design an ATM Machine

My Submissions (/contest/biweekly-contest-76/problems/design-an-atm-machine/submissions/) Back to Contest (/contest/biweekly-contest-76/)

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

There is an ATM machine that stores banknotes of `5` denominations: `20`, `50`, `100`, `200`, and `500` dollars. Initially the ATM is empty. The user can use the machine to deposit or withdraw any amount of money.

When withdrawing, the machine prioritizes using banknotes of **larger** values.

- For example, if you want to withdraw `$300` and there are `2` `$50` banknotes, `1` `$100` banknote, and `1` `$200` banknote, then the machine will use the `$100` and `$200` banknotes.
- However, if you try to withdraw `$600` and there are `3` `$200` banknotes and `1` `$500` banknote, then the withdraw request will be rejected because the machine will first try to use the `$500` banknote and then be unable to use banknotes to complete the remaining `$100`. Note that the machine is **not** allowed to use the `$200` banknotes instead of the `$500` banknote.

Implement the ATM class:

- `ATM()` Initializes the ATM object.
- `void deposit(int[] banknotesCount)` Deposits new banknotes in the order `$20`, `$50`, `$100`, `$200`, and `$500`.
- `int[] withdraw(int amount)` Returns an array of length `5` of the number of banknotes that will be handed to the user in the order `$20`, `$50`, `$100`, `$200`, and `$500`, and update the number of banknotes in the ATM after withdrawing. Returns `[-1]` if it is not possible (do **not** withdraw any banknotes in this case).

Example 1:

```
Input
["ATM", "deposit", "withdraw", "deposit", "withdraw", "withdraw"]
[[], [[0,0,1,2,1]], [600], [[0,1,0,1,1]], [600], [550]]
Output
[null, null, [0,0,1,0,1], null, [-1], [0,1,0,0,1]]

Explanation
ATM atm = new ATM();
atm.deposit([0,0,1,2,1]); // Deposits 1 $100 banknote, 2 $200 banknotes,
                          // and 1 $500 banknote.
atm.withdraw(600);        // Returns [0,0,1,0,1]. The machine uses 1 $100 banknote
                          // and 1 $500 banknote. The banknotes left over in the
                          // machine are [0,0,0,2,0].
atm.deposit([0,1,0,1,1]); // Deposits 1 $50, $200, and $500 banknote.
                          // The banknotes in the machine are now [0,1,0,3,1].
atm.withdraw(600);        // Returns [-1]. The machine will try to use a $500 banknote
                          // and then be unable to complete the remaining $100,
                          // so the withdraw request will be rejected.
                          // Since the request is rejected, the number of banknotes

                          // in the machine is not modified.
atm.withdraw(550);        // Returns [0,1,0,0,1]. The machine uses 1 $50 banknote
                          // and 1 $500 banknote.
```

Constraints:

- `banknotesCount.length == 5`
- `0 <= banknotesCount[i] <=` 10^9
- `1 <= amount <=` 10^9
- At most `5000` calls **in total** will be made to `withdraw` and `deposit`.
- At least **one** call will be made to each function `withdraw` and `deposit`.

JavaScript

```
const v = [20, 50, 100, 200, 500];
function ATM() {
    let d = [0, 0, 0, 0, 0];
    return { deposit, withdraw }
    function deposit(banknotesCount) {
        for (let i = 0; i < 5; i++) {
            d[i] += banknotesCount[i];
        }
    }
    function withdraw(amount) {
        let pre = amount, res = Array(5).fill(0);
        for (let i = 4; i >= 0; i--) {
            let used = Math.min(d[i], amount / v[i] >> 0);
            amount -= used * v[i];
        }
        if (amount != 0) return [-1];
        amount = pre;
        for (let i = 4; i >= 0; i--) {
            let used = Math.min(d[i], amount / v[i] >> 0);
            amount -= used * v[i];
            res[i] = used;
            d[i] -= used;
        }
        return res;
    }
}
```

□ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/681961912/) More Details (/submissions/detail/681961912/)

Share your acceptance!

1

Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Online Interview (/interview/) | Students (/student) | Terms (/terms) | Privacy Policy (/privacy)

United States (/region)