J223 - Ice Cream

David Wai {wjx}

2022-01-29

香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Background

Problem idea by David Wai

Preparation by Brian Lee, Daniel Yeh

Problem Restatement

Given S_1 1-dollar coins, S_2 2-dollar coins, and S_5 5-dollar coins

The price of each scoop is X_A and X_B at shop A and shop B respectively

You can choose to go to which shop first and the set of coins to pay

Let N = number of shops you need to get change from

Your task is to minimize N and output the construction

Sample Input	Sample output
0 5 2 10 6	Possible 0 B 0 0 2 0 3 0
0 5 2 10 7	Possible 1 B 0 3 1 0 1 1
0 5 2 10 11	Impossible

Statistics

Task	Attempts	Max	Mean	Std Dev
J223 - Ice Cream	75	100	11.813	16.831

Subtasks					
3: 67	17: 9	17: 14	7: 34	26: 1	30: 1

First solved by **dbsboscowang** at **2:59:35**

SUBTASKS

For all cases: $0 \leq S_1, S_2, S_5, X_A, X_B \leq 10^8$

	Points	Constraints
1	3	$S_2 = S_5 = 0$
2	17	$S_2 = 0$
3	17	$S_5 = 0$
4	7	$S_1 = S_2 = S_5 = 10^8$
5	26	$0 \leq S_1, S_2, S_5, X_A, X_B \leq 500$
6	30	No additional constraints

Subtask 1

Subtask 1 (3%): $S_2 = S_5 = 0$

- Only 1-dollar coins
- $N = 0$
- Check if $S_1 \geq X_A + X_B$
  - If yes, give X_A coins to shop A and X_B coins to shop B
  - Otherwise, output “Impossible”

Subtask 4

Subtask 4 (7%): $S_1 = S_2 = S_5 = 10^8$

- The answer is always possible since $0 \le X_A, X_B \le 10^8$
- $N = 0$
- There are many ways to do this subtask
- One of the easiest ways is to reserve 1 1-dollar coin for the shops with odd-number price, then use 2-dollar coins to pay the remaining price
- $S_2 * 2 \ge X_A + X_B$, 2-dollar coins are always enough

Subtask 3

Subtask 3 (17%): $S_5 = 0$

Only 1-dollar coins and 2-dollar coins

We can divide the problem into the following 3 cases:

1. Both X_A and X_B are even numbers, same as subtask 1 ($N = 0$)
2. One of X_A and X_B is an odd number
   - 2.1. Assume that X_A is odd
   - 2.2. If $S_1 = 0$, then we need to get change from shop A ($N = 1$)
   - 2.3. Otherwise, give 1 1-dollar coin to shop A, go to case 1 ($N = 0$)
3. Both X_A and X_B are odd numbers
   - 3.1. If $S_1 = 0$, give ($X_A + 1$) 2-dollar coins to shop A and receive 1 1-dollar coin, go to 2.3 ($N = 1$)
   - 3.2. If $S_1 = 1$, give the 1-dollar coin to shop A, go to 2.2 ($N = 1$)
   - 3.3. Otherwise, **reserve** 1 1-dollar coin for each shop, go to case 1 ($N = 0$)

Subtask 2

Subtask 2 (17%): $S_2 = 0$

- Only 1-dollar coins and 5-dollar coins
- It is always better if we use 1 5-dollar coin instead of 5 1-dollar coins
- Exhaust go to which shop first
  - For the first visited shop
    - Use as many 5-dollar coins as possible, then try to use 1-dollar coins to fill the remaining price
    - If 1-dollar coins are not enough, instead of using 1-dollar coins, just simply add 1 5-dollar coin and get change from the shop (*N* increases by 1)
  - Use a similar greedy approach for the second visited shop
    - There may be some 2-dollar coins when visiting the second shop
    - Use 5-dollar coins first, followed by 2-dollar coins, at last 1-dollar coins

Greedy Solution

- The subtask 2 solution can be extended to a more general greedy solution
- Exhaust go to which shop first, then greedy give coins in the $5 \rightarrow 2 \rightarrow 1$ order

In fact, this greedy solution can pass subtasks 1-4, which can get 44 points

Subtask 5

Subtask 5 (26%): $0 \le S_1, S_2, S_5, X_A, X_B \le 500$

- Exhaust go to which shop first
- Exhaust how many 1-dollar coins, 2-dollar coins and 5-dollar coins to use in the first shop
- Use the greedy approach to pay to the second shop

Time complexity: $O(S_1 S_2 S_3)$

Full Solution

Subtask 6 (30%): No additional constraints

- The greedy approach has some problem
- It is because 5-dollar coins cannot totally replace 2-dollar coins
  - For example, we should use 3 2-dollar coins instead of 1 5-dollar coin and 1 2-dollar coin so we don't need to get change
  - Special handle this case in the greedy approach
- Also, we need to ensure that we have enough 1-dollar or 5-dollar coins to pay for the second shop if the price of the second shop is an odd number
  - Try to reserve some 5-dollar coins, followed by 1-dollar coins to change the price of each shop to an even number

Full Solution 2

Subtask 6 (30%): No additional constraints

- The greedy approach has some problem
- It is because 5-dollar coins cannot totally replace 2-dollar coins
- Also, we need to ensure that we have enough 1-dollar or 5-dollar coins to pay for the second shop if the price of the second shop is an odd number
- You may also combine the subtask 5 solution with the greedy approach
  - Exhaust to reserve how many 1-dollar coins, 2-dollar coins and 5-dollar coins for the first shop (bounded by a small number, like 500 or even 10)
  - Do the greedy approach

Tips On Implementation

- The time complexity of the full solution is *O(1)*
- You may use some C++ STL features to simplify your code although it needs more time

```cpp
const vector coins{5, 2, 1};

vector<int> pay(int x, vector<int>& s) {
    vector used{0, 0, 0};
    for (int i = 0; i < 3; i++) {
        int num = min(s[i], x / coins[i]);
        s[i] -= num;
        used[i] += num;
        x -= num * coins[i];
    }
    return used;
}
```