OJ 166

Making Change

Prof. Edson Alves – UnB/FGA

Problema

Given an amount of money and unlimited (almost) numbers of coins (we will ignore notes for this problem) we know that an amount of money may be made up in a variety of ways. A more interesting problem arises when goods are bought and need to be paid for, with the possibility that change may need to be given. Given the finite resources of most wallets nowadays, we are constrained in the number of ways in which we can make up an amount to pay for our purchases – assuming that we can make up the amount in the first place, but that is another story.

Problema

The problem we will be concerned with will be to minimise the number of coins that change hands at such a transaction, given that the shopkeeper has an adequate supply of all coins. (The set of New Zealand coins comprises $5c, 10c, 20c, 50c, \$1$ and $\$2$.) Thus if we need to pay $55c$, and we do not hold a $50c$ coin, we could pay this as $2 \times 20c + 10c + 5c$ to make a total of 4 coins. If we tender $\$1$ we will receive $45c$ in change which also involves 4 coins, but if we tender $\$1.05(\$1 + 5c)$, we get $50c$ change and the total number of coins that changes hands is only 3.

Write a program that will read in the resources available to you and the amount of the purchase and will determine the minimum number of coins that change hands.

Entrada e saída

Input

Input will consist of a series of lines, each line defining a different situation. Each line will consist of 6 integers representing the numbers of coins available to you in the order given above, followed by a real number representing the value of the transaction, which will always be less than $\$5.00$. The file will be terminated by six zeroes $(0\ 0\ 0\ 0\ 0\ 0)$. The total value of the coins will always be sufficient to make up the amount and the amount will always be achievable, that is it will always be a multiple of $5c$.

Output

Output will consist of a series of lines, one for each situation defined in the input. Each line will consist of the minimum number of coins that change hands right justified in a field 3 characters wide.

Exemplo de entradas e saídas

Sample Input

```
2 4 2 2 1 0 0.95
2 4 2 0 1 0 0.55
0 0 0 0 0 0
```

Sample Output

```
2
3
```

Solução $O(NM)$

- Seja $m \geq C$ a quantidade paga ao caixa, onde C é o valor da conta, em centavos
- Defina $G(m, xs)$ como o número mínimo de moedas utilizadas para pagar m usando xs, onde

  $$xs = \{x_1, x_2, \ldots, x_N\}$$

  e x_i é o número de moedas disponíveis do tipo c_i
- Para o cidadão xs é dado na entrada
- Para o caixa, $xs = ys$, com $y_i = \infty$ para todo $i \in [1, N]$
- Caso não seja possível dar um troco para m usando xs, faça $G(m, xs) = \infty$

Solução $O(NM)$

- Quando o cidadão paga m ao caixa, ele recebe $m - C$ de troco
- Utilizando a notação descrita, a solução S do problema é dada por

$$S = \min_i \{G(C + 5i, xs) + G(5i, ys)\},$$

  onde $i \in [0, (M - C)/5]$ e M é o maior valor possível para a conta
- Como $G(m, xs)$ tem complexidade $O(N)$, a solução proposta tem complexidade $O(NM)$ por caso de teste

Solução $O(NM)$

```cpp
#include <bits/stdc++.h>

using namespace std;

const int oo { 1000000010 }, N = 6;
const vector<int> cs { 5, 10, 20, 50, 100, 200 }, ys(cs.size(), oo);

int greedy(int m, const vector<int>& qs)
{
    int res = 0;

    for (int i = N - 1; i >= 0; --i)
    {
        auto k = min(qs[i], m / cs[i]);
        m -= k*cs[i];
        res += k;
    }

    return m > 0 ? oo : res;
}
```

Solução $O(NM)$

```
int solve(int M, const vector<int>& xs)
{
    int total = 0, ans = oo;

    for (int i = 0; i < N; ++i)
        total += xs[i] * cs[i];

    for (int m = M; m <= total; m += 5)
        ans = min(ans, greedy(m, xs) + greedy(m - M, ys));

    return ans;
}

int main()
{
    while (true) {
        vector<int> xs(6);

        for (int i = 0; i < N; ++i)
            scanf("%d", &xs[i]);
```

Solução $O(NM)$

```
        if (accumulate(xs.begin(), xs.end(), 0) == 0)
            break;

        int d, c;
        scanf("%d.%d", &d, &c);

        auto ans = solve(100*d + c, xs);

        printf("%3d\n", ans);
    }

    return 0;
}
```