Olimpíadas Nacionais de Informática 2002

www.api.pt

3 de Maio de 2002

Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa

Problema 2 – Trocadeuro

1 Problema

Com a entrada em circulação do euro, muitas máquinas de pagamento automático que dão trocos tiveram de ser reprogramadas. As máquinas de "primeira geração" usam um esquema determinista muito simples para calcular os trocos, que consiste em usar o menor número de moedas que for possível. Por exemplo, se o troco for €3.50, dão uma moeda de €2, uma de €1 e uma de €0,50.

No entanto, isso é muito monótono e, para mais, verifica-se que assim as moedas grandes tendem a ser gastas mais depressa, pelo que ao fim de algum tempo a máquina fica com muitas moedas pequenas e poucas grandes, o que é inconveniente.

Por isso, os fabricantes estão a pensar introduzir as máquinas de segunda geração, que usam um programa mais "inteligente". O objectivo é não dar o troco sempre da mesma maneira, variar um pouco, para quebrar a monotonia, e assim distribuir melhor as moedas.

A ideia é a seguinte. Suponhamos que temos de trocar €3,50. Já vimos que no mínimo são necessárias três moedas. Mas suponhamos que a máquina está programada com uma *tolerância* de 2. Então, por hipótese, está autorizada a entregar o troco em 3 moedas, em 4 moedas ou em 5 moedas. Isso dá lugar a muitas outras possibilidades, por exemplo: <€1; €1; €1; €0,50>, <€2; €1; €0,20; €0,20; €0,10>, <€1, €1, €0,50, €0,50, €0,50>.

Pois bem, o problema é, dada uma quantia em euros e uma tolerância *t*, calcular todas as maneiras de trocar essa quantia em moedas, de maneira que o número de moedas não exceda em mais de *t* o mínimo possível.

2 Tarefa

A sua tarefa é escrever um programa para resolver o problema indicado.

3 Dados

Os dados para o problema vêm num ficheiro de texto, de nome "TROCOS.IN", com duas linhas. A primeira linha tem dois números não negativos, que representam a quantia a trocar: nesta linha o primeiro dos dois números é o número inteiro de euros e o segundo é o número inteiro de cêntimos. O número de cêntimos é inferior a 100. A segunda linha contém um número não negativo que representa a tolerância. Por exemplo

```
3 50
2
```

4 Resultado

O resultado do programa é um ficheiro de texto, de nome "TROCOS.OUT", com uma ou mais linhas. Em cada linha haverá 8 números não negativos separados entre eles por um único espaço, números esses que representam uma maneira de fazer o troco da quantia dada, dentro da tolerância. Em cada linha, o primeiro número representa o número de moedas de €2, o segundo o número de moedas de €1 e assim sucessivamente para as moedas de €0,50, €0,20, €0,10, €0,05, €0,02 e €0,01. Haverá tantas linhas quantas as maneiras de fazer o troco, uma para cada uma delas.

As linhas devem estar ordenadas por ordem crescente do número de moedas e nos casos de empate, por ordem lexicográfica inversa. (Veja abaixo o significado de ordem lexicográfica e de ordem lexicográfica inversa).

No caso do ficheiro de dados indicado, o resultado deve ser o seguinte.

```
1 1 1 0 0 0 0 0
1 0 3 0 0 0 0 0
0 3 1 0 0 0 0 0
1 1 0 2 1 0 0 0
0 2 3 0 0 0 0 0
```

5 Limites

A quantia máxima a trocar é €10 e a tolerância máxima é 8.

6 Pontuação parcial

O seu programa terá pontuação parcial de 30% se apenas calcular a maneira de fazer o troco usando o número mínimo de moedas. Terá pontuação parcial de 50% se calcular metade ou mais das maneiras de fazer o troco dentro da tolerância, mas não todas. Terá pontuação parcial de 70% se calcular todas as maneiras mas o resultado não for apresentado pela ordem especificada.

7 Note bem

Respeite rigorosamente os nomes do ficheiros e o formato do ficheiro de resultado. O júri copiará o seu programa para uma pasta de teste no computador de avaliação e tentará correr o programa com ficheiros de dados com o nome indicado, na pasta de teste. O júri espera encontrar depois o ficheiro de resultado também na pasta de teste.

Os ficheiros de teste usados pelo júri são válidos, garantidamente, pelo que o seu programa escusa de ser preocupar em validar os dados.

8 Ordem lexicográfica

Considere duas sequências de números, s_1 e s_2, diferentes, ambas de comprimento n: $s_1 = <x_1, x_2, ..., x_n>$ e $s_2 = <y_1, y_2, ..., y_n>$. A sequência s_1 vem antes de s_2 na ordem lexicográfica (e vem depois de s_2 na ordem lexicográfica inversa) se existir um $i <= n$ tal que para todo o $j < i$ se tem $x_j = y_j$ e, além disso, $x_i < y_i$.

Por exemplo: a sequência <3, 5, 7> vem antes da sequência <4, 5, 7> e esta vem antes de <4, 6, 7>.

No problema usamos a ordem lexicográfica inversa. É por isso que no ficheiro dos resultados a sequência <1 0 3 0 0 0 0 0> aparece antes da sequência <0 3 1 0 0 0 0 0>, por exemplo.