Competitive Programming Workshop: Algoritmos voraces

Giovanny Alfonso Chávez Ceniceros

Universidad Autónoma de Chihuahua
Facultad de Ingeniería

gchavezceniceros@acm.com

16 de marzo de 2019

Contenido

1 Algoritmos voraces (greedy)
- Devolver el cambio
- Maximizando el lote

2 Referencias

¿Qué es algoritmo *greedy*?

- Se dice que un algoritmo es codicioso si toma la decisión óptima a nivel local en cada paso con la esperanza de llegar a la solución global óptima.
- En algunos casos, la codicia funciona: la solución es corta y se ejecuta de manera eficiente. Para muchos otros, sin embargo, no lo hace.
- Cuando una estrategia voraz falla al producir resultados óptimos en todas las entradas, en lugar de algoritmo suele denominarse heurística.

Devolver el cambio

Descripción

El objetivo de este problema es encontrar el número mínimo de monedas necesarias para devolver el cambio del valor de entrada (un número entero) en monedas con denominaciones 1, 5 y 10.

Entrada

Un solo entero m.

Salida

El número mínimo de monedas con denominaciones 1, 5, 10 que se devuelven m.

Devolver el cambio

Restricciones

$0 \leq m \leq 10^3$.

Casos de prueba

Entrada	Salida
2	2
28	6

$$2 = 1 + 1$$
$$28 = 10 + 10 + 5 + 1 + 1 + 1$$

Devolver el cambio

```
1  int greedyChange( int money )
2  {
3      vector<int> coins{10, 5, 1}; // el orden importa
4      int i = 0, change = 0;
5
6      while(money > 0)
7          while(i < coins.size( ))
8          {
9              if(coins[i] > money)
10                 ++i;
11             else // decision  optima  a nivel local
12             {
13                 money -= coins[i];
14                 change += 1;
15             }
16         }
17     return change;
18 }
```

Referencias


Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2009)
Introduction to algorithms


Skiena, S. S. (2003)
Programing Challenges. The Programming Contest Training Manual

Our dear Sultan is visiting a country where there are n different types of coin. He wants to collect as many different types of coin as you can. Now if he wants to withdraw X amount of money from a Bank, the Bank will give him this money using following algorithm.

```
withdraw(X){
    if( X == 0) return;
    Let Y be the highest valued coin that does not exceed X.
    Give the customer Y valued coin.
    withdraw(X-Y);
}
```

Now Sultan can withdraw any amount of money from the Bank. He should maximize the number of different coins that he can collect in a single withdrawal.

Input

First line of the input contains T the number of test cases. Each of the test cases starts with n ($1 \leq n \leq 1000$), the number of different types of coin. Next line contains n integers C_1, C_2, ..., C_n the value of each coin type. $C_1 < C_2 < C_3 < \ldots < C_n < 1000000000$. C_1 equals to 1.

Output

For each test case output one line denoting the maximum number of coins that Sultan can collect in a single withdrawal. He can withdraw infinite amount of money from the Bank.

Sample Input

```
2
6
1 2 4 8 16 32
6
1 3 6 8 15 20
```

Sample Output

```
6
4
```