Algorithm Design Laboratory with Applications

Prof. Stefano Leucci

Problem: *Souvenirs.*

Algoland has a total of D districts indexed from 1 to D, each of which has a different (and sometimes weird) coin system. More precisely, the i-th district, uses a certain number $k_i \geq 1$ of coin denominations[1] $c_1^{(i)}, \dots, c_{k_i}^{(i)}$, where $c_1^{(i)}$ is always equal to 1 and, for $j = 2, \dots, k_i$, $c_{j-1}^{(i)} < c_j^{(i)}$. You are touring all the districts and, for each district i, you want to buy a souvenir that costs n_i units in the local currency.

You want to travel light but you also want to avoid travelling back with leftover coins. Your task is twofold:

1. For each district i, find the minimum number η_i of coins needed to be able to pay for n_i without receiving any change back.
2. Solve the above problem in the special case in which, $\forall i = 1, \dots, D$, the coin denominations $c_j^{(i)}$ are consecutive powers of a (small) positive integer p_i, i.e., $c_j^{(i)} = p_i^{j-1}$ for some $p_i \geq 2$.

Input. The first line of the input contains the number D of districts. Each district is described by 2 lines. The first line of each district contains k_i and n_i. The second line of each district contains the k_i integers $c_1^{(i)}, \dots, c_{k_i}^{(i)}$ separated by a space.

Output. The output consists of D lines. The i-th line contains the minimum number of coins η_i needed to pay n_i units of currency in the i-th district.

Assumptions.

For task 1:

$1 \leq D \leq 100$; $\forall i = 1, \dots, D,\ 1 \leq n_i < 2^{19}$; $\forall i = 1, \dots, D,\ 1 \leq k_i < 2^8$; $\forall i = 1, \dots, D, \forall j = 1, \dots, k_i,\ 1 \leq c_j^{(i)} < 2^{14}$.

For task 2:

$1 \leq D \leq 100$; $\forall i = 1, \dots, D,\ 1 \leq n_i < 2^{31}$; $\forall i = 1, \dots, D,\ 1 \leq k_i < 2^4$; $\forall i = 1, \dots, D, \forall j = 1, \dots, k_i,\ 1 \leq c_j^{(i)} < 2^{31}$.

Example.

Input (for the general case):

```
2
3 6
1 3 4
5 1023
1 2 5 10 50
```

Output:

```
2
24
```

Requirements. Your general algorithm should require $O(\sum_{i=1}^{D} n_i k_i)$ time (with reasonable hidden constants). Your algorithm for the special case of denominations that are consecutive powers of p_i should require time $O(\sum_{i=1}^{D} k_i)$.

Notes. A reasonable implementation should not require more than 1 second for each input file. Inputs for task 2 have the suffix `-special-case`.

[1] Apparently, there are no banknotes in Algoland.