Soluciones entrenos OIE

Monedas

Baq tiene n monedas, cada una con un valor m_i. A Baq no le gusta que las máquinas donde compra los billetes de metro le devuelvan cambio, por eso quiere saber si con sus n monedas puede pagar con exactitud cualquier cantidad.

¿Cuál es la mínima cantidad de dinero que Baq no puede pagar con exactitud con sus n monedas?

Input Format

La entrada consiste en un entero t seguido de t casos.

Cada caso empieza con un entero n seguido de los n valores de las monedas, m_i.

Constraints

$1 \leq t \leq 500$

$1 \leq m_i \leq 10^9$

$1 \leq n$

Para cada caso la suma de p_i no es mayor que 10^5.

- 13 Puntos: $n \leq 10$
- 26 Puntos: $n \leq 10^2$
- 30 Puntos: $n \leq 10^3$
- 31 Puntos: $n \leq 10^4$

Output format

Una línea para cada caso con la mínima cantidad de dinero que Baq no puede pagar con exactitud.

Solución

Primero ordenamos las monedas que tenemos por valor. Si la suma de las primeras m monedas s, es menor a la m-ésima moneda menos 1, entonces no podemos formar la cantidad $s+1$. Tenemos que mirar cuál es la primera vez que esto sucede. Nota: consideramos que tenemos una moneda con valor 0, en otras palabras, si no tenemos la moneda 1 entonces la solución será 1.

Pero, ¿por qué sucede esto? Es decir, si las primeras m monedas suman s y en ningún momento hemos tenido que para algún $n < m$ la suma de las n primeras monedas $+1$ sea menor a m_{n+1}, ¿por qué podemos

asegurar que todas las cantidades $\leq s$ se pueden formar? La idea es que podemos representar cualquier moneda de valor $v_0 \leq s$ de manera greedy: cogemos la primera moneda m_i de valor $\leq v$, usaremos m_i en la representación de v_0, con lo que ahora queremos representar el valor $v_1 = v_0 - m_i$. Como que $m_1 + m_2 + \cdots + m_i \geq v_0$ por hipótesis, entonces $m_1 + m_2 + \cdots + m_{i-1} \geq v_1$. Es decir, para este nuevo valor (v_1) la suma de las primeras $i-1$ monedas también vale por lo menos v_1. Podemos ir repitiendo el proceso: ahora cogeríamos m_j la primera moneda de las primeras $i-1$ que sea $\leq v_1$ y tendríamos que encontrar una representación para $v_2 = v_1 - m_j$. Al final tendremos que $1 \geq v_k$ y claramente podremos representar la cantidad v_k pues esta solo puede ser o 0 o 1 y la moneda 1 la tenemos.

Código

C++

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;
typedef long long int ll;

int main() {
    int t;
    cin >> t;

    for (int caso = 0; caso < t; ++caso){
        int n;
        cin >> n;

        vector <int> monedas(n);
        for (int i = 0; i < n; ++i) cin >> monedas[i];

        sort(monedas.begin(), monedas.end()); //ordenamos las monedas de manera
        ↪ creciente

        bool encontrado = false; //esta variable nos indica si ya hemos encontrado
        ↪ un valor que no podamos representar
        ll suma = 0;
        for (int i = 0; i < n and not encontrado; ++i){
            if (suma + 1 < monedas[i]){ //si no podemos representar suma + 1
                cout << suma + 1 << endl;
                encontrado = true;
            }
            suma += monedas[i];
        }
        if (not encontrado) cout << suma + 1 << endl;
    }
}
```

Python

```python
import sys

t = int(input())

for caso in range(t):
    n = int(input())
    monedas = list(map(int, input().strip().split(' ')))

    monedas.sort() #ordenamos las monedas de manera creciente

    encontrado = False #esta variable nos indica si ya hemos encontrado un valor que no podamos representar
    suma = 0
    for moneda in monedas:
        if (not encontrado):
            if (suma + 1 < moneda): #si no podemos representar suma + 1
                print(suma+1)
                encontrado = True

            suma += moneda

    if (not encontrado): print(suma+1)
```