Detectando Moléculas

Petr está trabajando para una compañía que construyó una máquina para detectar moléculas. Cada molécula tiene un peso que es un entero positivo. La máquina tiene un *rango de detección* $[l, u]$, donde l y u son enteros positivos. La máquina puede detectar un conjunto de moléculas si y sólo si ese conjunto contiene un subconjunto de moléculas tales que su peso total pertenezca al rango de detección de la máquina.

Formalmente, considera n moléculas con pesos enteros positivos $w_0, \ldots, w_{n-1}$. La detección es exitosa si hay un conjunto de índices distintos $I = \{i_1, \ldots , i_m\}$ tales que $l \le w_{i_1} + \ldots + w_{i_m} \le u$.

Debido a cómo funciona internamente la máquina, se puede asegurar que la diferencia entre l y u es mayor o igual a la diferencia entre el peso de la molécula más pesada y más ligera. Formalmente, $u - l \ge w_{max} - w_{min}$, donde $w_{max}=\max(w_0, \ldots , w_{n-1})$ y $w_{min}=\min(w_0, \ldots , w_{n-1})$.

Tu tarea es escribir un programa que encuentre cualquier subconjunto de moléculas con peso total dentro del rango de detección, o que determine que ningún subconjunto existe con esa propiedad.

Detalles de implementación

Debes implementar una función (método):

- `int[] solve(int l, int u, int[] w)`
  - `l` y `u`: los extremos del rango de detección,
  - `w`: los pesos de las moléculas.
  - si el subconjunto buscado existe, la función debe regresar un arreglo con los índices de las moléculas que pertenecen al subconjunto. Si hay más de una respuesta correcta, regresa cualquiera.
  - si el subconjunto buscado no existe, la función debe regresar un arreglo vacío.

Para el lenguaje C la declaración de la función es ligeramente diferente:

- `int solve(int l, int u, int[] w, int n, int[] result)`
  - `n`: el número de elementos en `w` (es decir, el número de moléculas),
  - el resto de los parámetros son iguales que arriba.
  - en vez de regresar un arreglo con m índices (como en los demás lenguajes), la función debe escribir los índices en las primeras m celdas del arreglo `result` y regresar m.

- si el subconjunto requerido no existe, la función no debe escribir nada en `result` y regresar \(0\).

Tu programa puede escribir los índices del arreglo de la respuesta (o al arreglo `result` en C) en cualquier orden.

Por favor utiliza las plantillas de tu lenguaje de programación para ver los detalles de implementación.

Ejemplos

Ejemplo 1

`solve(15, 17, [6, 8, 8, 7])`
En este ejemplo tenemos cuatro moléculas con pesos 6, 8, 8 y 7. La máquina puede detectar subconjuntos de moléculas con peso total entre 15 y 17, inclusivo. Nota que \(17-15 \ge 8-6\). El peso total de las moléculas 1 y 3 es \(w_1 + w_3 = 8 + 7 = 15\), así que la función puede regresar `[1, 3]`. Otras posibles respuestas correctas son `[1, 2]` (\(w_1 + w_2 = 8 + 8 = 16\)) y `[2, 3]` (\(w_2 + w_3 = 8 + 7 = 15\)).

Ejemplo 2

`solve(14, 15, [5, 5, 6, 6])`
En este ejemplo tenemos cuatro moléculas con pesos 5, 5, 6 y 6, y estamos buscando un subconjunto con peso total entre 14 y 15, inclusivo. De nuevo, nota que \(15-14 \ge 6-5\). No hay ningún subconjunto de moléculas con peso total entre \(14\) y \(15\), así que la función debe regresar el arreglo vacío.

Ejemplo 3

`solve(10, 20, [15, 17, 16, 18])`
En este ejemplo tenemos cuatro moléculas con pesos 15, 17, 16 y 18, y estamos buscando un subconjunto con peso total entre 10 y 20, inclusivo. De nuevo, nota que \(20-10 \ge 18-15\). Cualquier subconjunto de exactamente un elemento tiene peso total entre 10 y 20, así que las respuestas correctas posibles son `[0]`, `[1]`, `[2]` y `[3]`.

Subtareas

1. (9 puntos): \(1 \le n \le 100\), \(1 \le w_i \le 100\), \(1 \le u, l \le 1000\), todos los \(w_i\) son iguales.
2. (10 puntos): \(1 \le n \le 100\), \(1 \le w_i, u, l \le 1000\), y \(max(w_0, \ldots, w_{n-1}) - \min(w_0, \ldots, w_{n-1}) \le 1\).
3. (12 puntos): \(1 \le n \le 100\) y \(1 \le w_i,u,l \le 1000\).
4. (15 puntos): \(1 \le n \le 10\,000\) y \(1 \le w_i,u,l \le 10\,000\).
5. (23 puntos): \(1 \le n \le 10\,000\) y \(1 \le w_i,u,l \le 500\,000\).
6. (31 puntos): \(1 \le n \le 200\,000\) y \(1 \le w_i,u,l < 2^{31}\).

Evaluador de ejemplo

El evaluador de ejemplo lee la entrada en el siguiente formato:

- línea 1: los enteros \(n\), \(l\), \(u\).

- línea 2: n enteros: $w_0, \ldots , w_{n-1}$.