

Detectando Moléculas

Petr está trabajando para una compañía que ha construido una máquina para detectar moléculas. Cada molécula tiene un peso 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 solo si tiene un subconjunto con peso total perteneciente al rango de la máquina

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

Debido a especificaciones de la máquina, se garantiza que la diferencia de peso entre u y l es mayor o igual a la diferencia entre la molécula más pesada y la más liviana.

Formalmente, $u - l \geq w_{max} - w_{min}$, donde $w_{max} = \max(w_0, \dots, w_{n-1})$ y $w_{min} = \min(w_0, \dots, w_{n-1})$.

Su tarea es escribir un programa que encuentre un subconjunto de moléculas con un peso total en el rango de detección, o que determine que no existe tal subconjunto.

Detalles de implementación

Debe implementar una función (método):

- `int[] solve(int l, int u, int[] w)`
 - l y u : puntos extremos del rango de detección,
 - w : pesos de las moléculas.
 - si el subconjunto requerido existe, la función debe retornar un arreglo de índices de moléculas que formen un subconjunto válido. Si hay varias respuestas correctas, retorne cualquiera de ellas.
 - si el subconjunto requerido no existe, la función debe retornar un arreglo vacío.

Para el lenguaje C la declaración de la función se ve un poco distinta:

- `int solve(int l, int u, int[] w, int n, int[] result)`
 - n : número de elementos en w (es decir, número de moléculas),
 - los otros parámetros son los mismos de arriba.
 - en vez de retornar un arreglo de m índices (como arriba), la función debe escribir los índices a las primeras m posiciones del arreglo `result` y después retornar m .
 - si el subconjunto requerido no existe, la función no debería escribir nada en

el arreglo `result` y debe retornar `0`.

Su programa puede escribir los índices en el arreglo retornado (o el arreglo `result` en C) en cualquier orden.

Por favor usar los archivos ejemplo dados para ver detalles de implementación en su lenguaje de programación.

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 un peso total entre 15 y 17, inclusive. Note que $17 - 15 \geq 8 - 6$. El peso total de las moléculas 1 y 3 es

$w_1 + w_3 = 8 + 7 = 15$, así que la función puede retornar `[1, 3]`. Otras posibles respuestas correctas `[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 de ellas con un peso total entre 14 y 15, inclusive. Nuevamente, note que $15 - 14 \geq 6 - 5$. No hay un subconjunto de moléculas con un peso total entre 14 y 15. Por lo tanto la función debe devolver un 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 el peso total entre 10 y 20, inclusive. Nuevamente, note que $20 - 10 \geq 18 - 15$. Cualquier subconjunto consistente de exactamente un elemento satisface el requerimiento, por lo tanto las respuestas correctas son: `[0]`, `[1]`, `[2]` y `[3]`.

Subtareas

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

Calificador ejemplo

El calificador ejemplo lee la entrada en el siguiente formato:

- línea 1: enteros n, l, u .
- línea 2: n enteros: w_0, \dots, w_{n-1} .