Detetando Moléculas

O Petr trabalha para uma empresa que construiu uma máquina que deteta moléculas. Cada molécula tem um peso inteiro positivo. A máquina tem um *intervalo de detecção* $[l, u]$, onde l e u são inteiros positivos. A máquina consegue detetar um conjunto de moléculas se e só se este conjunto contém um subconjunto de moléculas cujo peso total pertence ao intervalo de detecção da máquina.

Formalmente, considere n moléculas com pesos inteiros positivos $w_0, \ldots, w_{n-1}$. A detecção é bem sucedida se existe um conjunto de índices distintos $I = \{i_1, \ldots, i_m\}$ tais que $l \leq w_{i_1} + \ldots + w_{i_m} \leq u$.

Devido às especificações da máquina, é garantido que a diferença entre l e u é maior do que ou igual à diferença entre o peso da molécula mais pesada e o da molécula mais leve. Formalmente, $u - l \geq w_{max} - w_{min}$, onde $w_{max} = \max(w_0, \ldots, w_{n-1})$ e $w_{min} = \min(w_0, \ldots, w_{n-1})$.

A sua tarefa é escrever um programa que, ou encontra um subconjunto de moléculas com peso total incluído no intervalo de detecção, ou determina que não existe um subconjunto nessas condições.

Detalhes da implementação

Você deve implementar uma função (método):

- `int[] solve(int l, int u, int[] w)`
  - `l` e `u`: os extremos do intervalo de detecção,
  - `w`: os pesos das moléculas,
  - se o subconjunto pedido existe, a função deve retornar um vetor de indíces das moléculas que formam um tal subconjunto. Se existirem várias respostas corretas, devolva uma qualquer,
  - se o subconjunto pedido não existe, a função deve retornar um vetor vazio.

Para a linguagem C a assinatura da função é ligeiramente diferente:

- `int solve(int l, int u, int[] w, int n, int[] result)`
  - `n`: o número de elementos em `w` (i.e., o número de moléculas),
  - os restantes parâmetros são os mesmos que os anteriores.
  - em vez de retornar um vetor com m indíces (como acima), a função deve escrever os índices para as primeiras m posições do vetor `result` e depois retornar m.
  - se o subconjunto pretendido não existir, a função não deve escrever nada

para o vetor `result` e deve retornar 0.

O seu programa pode escrever os índices para o vetor retornado (ou para o vetor `result` em C) em qualquer ordem.

Por favor use os arquivos modelo providenciados para ver mais detalhes sobre a implementação na sua linguagem de programação.

Exemplos

Exemplo 1

`solve(15, 17, [6, 8, 8, 7])`
Neste exemplo temos quatro moléculas com pesos 6, 8, 8 e 7. A máquina consegue detectar subconjuntos de moléculas com peso total entre 15 e 17, inclusive. Note que $17 - 15 \geq 8 - 6$. O peso total das moléculas 1 e 3 é $w_1 + w_3 = 8 + 7 = 15$, por isso a função pode retornar `[1, 3]`. Outras possíveis respostas corretas são `[1, 2]` ($w_1 + w_2 = 8 + 8 = 16$) e `[2, 3]` ($w_2 + w_3 = 8 + 7 = 15$).

Exemplo 2

`solve(14, 15, [5, 5, 6, 6])`
Neste exemplo temos quatro moléculas com pesos 5, 5, 6 e 6, e estamos à procura de um subconjunto delas com peso total entre 14 e 15, inclusive. Novamente, note que $15 - 14 \geq 6 - 5$. Não há nenhum subconjunto de moléculas com peso total entre 14 e 15 por isso a função deve retornar um vetor vazio.

Exemplo 3

`solve(10, 20, [15, 17, 16, 18])`
Neste exemplo temos quatro moléculas com pesos 15, 17, 16, 18, e estamos à procura de um subconjunto delas com peso total entre 10 e 20, inclusive. Novamente, nota que $20 - 10 \geq 18 - 15$. Qualquer subconjunto que consista de exatamente um elemento tem peso total entre 10 e 20, logo as possíveis respostas corretas são: `[0]`, `[1]`, `[2]` ou `[3]`.

Subtarefas

1. (9 pontos): $1 \leq n \leq 100$, $1 \leq w_i \leq 100$, $1 \leq u, l \leq 1000$, todos os w_i são iguais.
2. (10 pontos): $1 \leq n \leq 100$, $1 \leq w_i, u, l \leq 1000$ e $\max(w_0, \ldots, w_{n-1}) - \min(w_0, \ldots, w_{n-1}) \leq 1$.
3. (12 pontos): $1 \leq n \leq 100$ e $w_i, u, l \leq 1000$.
4. (15 pontos): $1 \leq n \leq 10\,000$ e $w_i, u, l \leq 10\,000$.
5. (23 pontos): $1 \leq n \leq 10\,000$ e $w_i, u, l \leq 500\,000$
6. (31 pontos): $1 \leq n \leq 200\,000$ e $w_i, u, l < 2^{31}$.

Corretor exemplo

O corretor exemplo lê a entrada no seguinte formato:

- linha 1: inteiros n, l, u.
- linha 2: n inteiros: $w_0, \ldots, w_{n-1}$.