Individuazione di molecole

Petr lavora per una compagnia che ha costruito una macchina per individuare molecole. Ogni molecola ha per peso un intero positivo. La macchina ha un *range di individuazione* $[l, u]$, dove l e u sono interi positivi. La macchina può individuare un insieme di molecole se e solo se esso contiene un sottoinsieme di molecole la cui somma dei pesi appartiene al range di individuazione della macchina.

Formalmente, consideriamo n molecole di peso $w_0, \ldots, w_{n-1}$. L'individuazione ha successo se c'è un insieme di indici distinti $I = \{i_1, \ldots, i_m\}$ tali che $l \le w_{i_1} + \ldots + w_{i_m} \le u$.

A causa delle specifiche tecniche della macchina, è garantito che la differenza tra l ed u è maggiore o uguale alla differenza tra il peso della molecola più leggera e quello della molecola più pesante. Formalmente, $u - l \ge w_{max} - w_{min}$, dove $w_{max} = \max(w_0, \ldots, w_{n-1})$ e $w_{min} = \min(w_0, \ldots, w_{n-1})$.

Il tuo compito è di scrivere un programma che trovi un qualsiasi sottoinsieme di molecole il cui peso totale è nel range di individuazione, o che determini che un tale insieme non esiste.

Dettagli di implementazione

Devi implementare la seguente funzione (metodo):

- `int[] solve(int l, int u, int[] w)`
  - `l` ed `u`: gli estremi del range di individuazione.
  - `w`: i pesi delle molecole.
  - se il sottoinsieme richiesto esiste, la funzione deve restituire un array di indici di molecole che formano un tale sottoinsieme. Se sono possibili più risposte, può esserne restituita una qualsiasi.
  - se il sottoinsieme richiesto non esiste, la funzione deve restituire un array vuoto.

Per il linguaggio C la signature della funzione è leggermente diversa:

- `int solve(int l, int u, int[] w, int n, int[] result)`
  - `n`: il numero di elementi in `w` (cioè il numero di molecole),
  - i parametri `l`, `u`, `w` sono come sopra.
  - invece di restituire un array di m indici (come sopra), la funzione deve scrivere gli indici nelle prime m celle dell'array `result` e poi restituire m.
  - se il sottoinsieme richiesto non esiste, la funzione non deve scrivere nulla nell'array `result` e deve restituire `0`.

Il tuo programma può scrivere gli indici nell'array restituito (o nell'array `result` in C)

in un ordine qualsiasi. Per ulteriori dettagli, vedi il template fornito per il tuo linguaggio di programmazione.

Esempi

Esempio 1

`solve(15, 17, [6, 8, 8, 7])`
In questo esempio ci sono quattro molecole con pesi 6, 8, 8, 7. La macchina può individuare sottoinsiemi di molecole con peso totale tra 15 e 17 compresi. Nota che $17 - 15 \geq 8 - 6$. Il peso totale delle molecole 1 e 3 è $w_1 + w_3 = 8 + 7 = 15$, quindi la funzione può restituire `[1, 3]`. Altre possibili risposte corrette sono `[1, 2]`, dato che $w_1 + w_2 = 8 + 8 = 16$, e `[2, 3]` dato che $w_2 + w_3 = 8 + 7 = 15$.

Example 2

`solve(14, 15, [5, 5, 6, 6])`
In questo esempio ci sono quattro molecole con pesi 5, 5, 6, 6 e cerchiamo un loro sottoinsieme con peso tra 14 e 15 compresi. Nota che $15 - 14 \geq 6 - 5$. Non ci sono sottoinsiemi di molecole con peso totale tra 14 e 15, quindi la funzione deve restituire un array vuoto.

Example 3

`solve(10, 20, [15, 17, 16, 18])`
In questo esempio ci sono quattro molecole con pesi 15, 17, 16, 18 e cerchiamo un loro sottoinsieme con peso tra 10 e 20 compresi. Nota che $20 - 10 \geq 18 - 15$. Ogni sottoinsieme con esattamente un elemento ha peso totale compreso tra 10 e 20, quindi le possibili risposte corrette sono `[0]`, `[1]`, `[2]` e `[3]`.

Subtask

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

Grader di esempio

Il grader di esempio legge l'input nel formato seguente:

- riga 1: tre interi n, l, u.
- riga 2: n interi: $w_0, \ldots, w_{n-1}$.