Detecting Molecules

Luka je zaposlen u kompaniji KPMG i sposoban je za detekciju molekula alkohola i nikotina. Svaki molekul ima pozitivnu celobrojnu težinu. Luka ima *opseg detekcije* $[l, u]$, gde su l i u pozitivni celi brojevi. Luka može detektovati skup molekula ako i samo ako taj skup sadrži podskup molekula čija je ukupna težina u opsegu detekcije Luke.

Formalno, posmatrajmo n molekula sa težinama $w_0, \ldots, w_{n-1}$. Luka će uspešno obaviti detekciju ako postoji skup različitih indekasa $I = \{i_1, \ldots, i_m\}$ za koji važi $l \leq w_{i_1} + \ldots + w_{i_m} \leq u$.

Zbog ograničenih sposobnosti Luke, garantuje se da je razlika između l i u veća ili jednaka od razlike u težini između najtežeg i najlakšeg molekula. Formalno, važi $u - l \geq w_{max} - w_{min}$, gde je $w_{max} = \max(w_0, \ldots, w_{n-1})$ i $w_{min} = \min(w_0, \ldots, w_{n-1})$.

Da Luka ne bi bio izbačen sa olimpijade, vaš zadatak je da napišete program koji će ili pronaći bilo koji podskup molekula čija je ukupna težina u opsegu detekcije Luke ili utvrditi da takav podskup ne postoji.

Detalji implementacije

Potrebno je da implementirate jednu funkciju (metod):

- `int[] solve(int l, int u, int[] w)`
  - `l` and `u`: krajnje tačke opsega detekcije Luke,
  - `w`: težine molekula.
  - ako traženi podskup postoji, funkcija treba da vrati niz indekasa molekula koji formiraju takav podskup. Ako postoji više rešenja, vratiti bilo koje od njih.
  - ako traženi podskup ne postoji, funkcija treba da vrati prazan niz.

Za jezik C potpis funkcije je malo drugačiji:

- `int solve(int l, int u, int[] w, int n, int[] result)`
  - `n`: broj elementa u nizu `w` (tj. broj molekula),
  - ostali parametri su isti kao u gornjoj funkciji.
  - umesto da vrati niz od m indekasa (kao gore), funkcija treba da upiše indekase u prvih m pozicija niza `result` i da vrati m.
  - ako traženi podskup ne postoji, funkcija ne treba da upisuje ništa u niz `result` i treba da vrati 0.

Vaš program može da ispiše indekse u nizu koji se vraća (ili u nizu `result` u C-u) u bilo kom poretku. Koristite date templejt-fajlove za bolji uvid u detalje implementacije za vaš programski jezik.

Primeri

Primer 1

`solve(15, 17, [6, 8, 8, 7])`
U ovom primeru data su četiri molekula sa težinama 6, 8, 8 i 7. Luka može detektovati molekule čija je ukupna težina između 15 i 17, uključivo. Primetimo da važi $17 - 15 \geq 8 - 6$. Ukupna težina molekula 1 i 3 je $w_1 + w_3 = 8 + 7 = 15$, pa funkcija može vratiti `[1, 3]`. Neka od drugih tačnih rešenja su `[1, 2]` ($w_1 + w_2 = 8 + 8 = 16$) i `[2, 3]` ($w_2 + w_3 = 8 + 7 = 15$).

Primer 2

`solve(14, 15, [5, 5, 6, 6])`
U ovom primeru data su četiri molekula sa težinama 5, 5, 6 i 6, i tražimo podskup čija je ukupna težina između 14 i 15, uključivo. Primetimo da opet važi $15 - 14 \geq 6 - 5$. Ne postoji podskup molekula čija je ukupna težina između 14 i 15 pa funkcija treba da vrati prazan niz.

Primer 3

`solve(10, 20, [15, 17, 16, 18])`
U ovom primeru data su četiri molekula sa težinama 15, 17, 16 i 18, i tražimo podskup čija je ukupna težina između 10 i 20, uključivo. Kao i u prethodnim primerima, važi $20 - 10 \geq 18 - 15$. Bilo koji podskup koji sadrži tačno jedan elemenat zadovolajva uslove, pa su tačna rješenja: `[0]`, `[1]`, `[2]` i `[3]`.

Podzadaci

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

Opis priloženog grejdera

Priloženi grejder čita ulaz u sledećem formatu:

- linija 1: celi brojevi n, l, u.
- linija 2: n celih brojeva: $w_0, \ldots, w_{n-1}$.