ตรวจจับโมเลกุล

ปีเตอร์ทำงานให้บริษัทที่ผลิตเครื่องตรวจจับโมเลกุล แต่ละโมเลกุลมีน้ำหนักเป็นจำนวนเต็มบวก โดยเครื่องดังกล่าวมี *ระยะตรวจจับ* $[l, u]$, โดยที่ l และ u เป็นจำนวนเต็มบวก เครื่องสามารถตรวจจับเซ็ตของโมเลกุลได้ก็ต่อเมื่อเซ็ตนี้มีสับเซ็ตที่มีผลรวมน้ำหนักโมเลกุลอยู่ภายในระยะตรวจจับของเครื่อง

กล่าวคือ: เมื่อพิจารณาโมเลกุล n โมเลกุล ซึ่งมีน้ำหนัก $w_0, \ldots, w_{n-1}$ การตรวจจับจะประสบผลสำเร็จถ้ามีเซ็ตของตัวเลขดัชนี $I = \{i_1, \ldots, i_m\}$ โดยที่ $l \le w_{i_1} + \ldots + w_{i_m} \le u$

เนื่องด้วยลักษณะเฉพาะของเครื่อง เราสามารถรับประกันได้ว่าส่วนต่างระหว่าง l และ u จะต้องมากกว่าหรือเท่ากับส่วนต่างน้ำหนักระหว่างโมเลกุลที่หนักที่สุดกับโมเลกุลที่เบาที่สุด กล่าวคือ: $u - l \ge w_{max} - w_{min}$, โดยที่ $w_{max} = \max(w_0, \ldots, w_{n-1})$ และ $w_{min} = \min(w_0, \ldots, w_{n-1})$

ภารกิจของคุณคือ จงเขียนโปรแกรมเพื่อค้นหาสับเซ็ตของโมเลกุลซึ่งมีผลรวมน้ำหนักอยู่ภายในระยะตรวจจับ หรือตัดสินว่าสับเซ็ตดังกล่าวไม่มีอยู่จริง

รายละเอียดการเขียนโปรแกรม

จงเขียนฟังก์ชัน:

- `int[] solve(int l, int u, int[] w)`
  - `l` และ `u`: ตัวเลขที่ระบุระยะตรวจจับ
  - `w`: น้ำหนักของโมเลกุล
  - หากสับเซ็ตที่ต้องการมีอยู่จริง ฟังก์ชันควรคืนค่าอาเรย์ของดัชนีโมเลกุลซึ่งรวมตัวกันเป็นสับเซ็ตดังกล่าว หากมีคำตอบที่ถูกต้องหลายคำตอบให้คืนค่าคำตอบใดก็ได้
  - หากสับเซ็ตที่ต้องการไม่มีอยู่จริง ฟังก์ชันควรคืนค่าอาเรย์ว่างเปล่า

สำหรับภาษา C หัวฟังก์ชันมีความแตกต่างเล็กน้อย:

- `int solve(int l, int u, int[] w, int n, int[] result)`
  - `n`: จำนวนสมาชิกใน `w` (นั่นคือ: จำนวนโมเลกุล)
  - พารามิเตอร์อื่น เหมือนกับที่กล่าวไว้ด้านบน
  - แทนที่จะต้องคืนค่าเป็นอาเรย์ของดัชนี m ตัว (แบบด้านบน), ฟังก์ชันนี้ควรเขียนค่าดัชนีลงไปยัง m ช่องแรกของอาเรย์ `result` แล้วค่อยคืนค่า m
  - หากสับเซ็ตที่ต้องการไม่มีอยู่จริง ฟังก์ชันนี้ไม่ควรเขียนอะไรลงในอาเรย์ `result` และคืนค่า 0

โปรแกรมของคุณจะเขียนดัชนีในลำดับใดก็ได้ลงในอาเรย์ที่คืนค่า (หรือลงในอาเรย์ `result` ในภาษา C)

สำหรับรายละเอียดการเขียนโปรแกรมในภาษาของคุณ โปรดดูไฟล์ต้นแบบที่ได้เตรียมไว้ให้

ตัวอย่าง

ตัวอย่างที่ 1

`solve(15, 17, [6, 8, 8, 7])`
ในตัวอย่างนี้เรามีโมเลกุลสี่โมเลกุลที่มีน้ำหนัก 6, 8, 8 และ 7 เครื่องสามารถตรวจจับสับเซตของโมเลกุลที่มีน้ำหนักรวมตั้งแต่ 15 ถึง 17 (รวมหัวท้าย) สังเกตว่า $17 - 15 \geq 8 - 6$ น้ำหนักรวมของโมเลกุลที่ 1 และ 3 คือ $w_1 + w_3 = 8 + 7 = 15$ ดังนั้นฟังก์ชันสามารถคืนค่า `[1, 3]` เป็นคำตอบ คำตอบอื่นที่ถูกต้องเช่นกันได้แก่ `[1, 2]` ($w_1 + w_2 = 8 + 8 = 16$) และ `[2, 3]` ($w_2 + w_3 = 8 + 7 = 15$)

ตัวอย่างที่ 2

`solve(14, 15, [5, 5, 6, 6])`
ในตัวอย่างนี้เรามีโมเลกุลสี่โมเลกุลที่มีน้ำหนัก 5, 5, 6 และ 6 และเราต้องการหาสับเซตที่มีน้ำหนักรวมตั้งแต่ 14 ถึง 15 (รวมหัวท้าย) โปรดสังเกตอีกครั้งว่า $15 - 14 \geq 6 - 5$ แต่ไม่มีสับเซตใดที่มีผลรวมน้ำหนักโมเลกุลตั้งแต่ 14 ถึง 15 เลย ฟังก์ชันนี้จึงควรคืนค่าอาเรย์ว่างเปล่า

ตัวอย่างที่ 3

`solve(10, 20, [15, 17, 16, 18])`
ในตัวอย่างนี้เรามีโมเลกุลสี่โมเลกุลที่มีน้ำหนัก 15, 17, 16 และ 18 และเราต้องการหาสับเซตที่มีน้ำหนักรวมตั้งแต่ 10 ถึง 20 (รวมหัวท้าย) โปรดสังเกตอีกครั้งว่า $20 - 10 \geq 18 - 15$ สับเซ็ตใดก็ตามที่มีสมาชิกเพียงหนึ่งตัวย่อมมีผลรวมน้ำหนักตั้งแต่ 10 ถึง 20 ดังนั้นคำตอบที่ถูกต้องได้แก่: `[0]`, `[1]`, `[2]` และ `[3]`

ปัญหาย่อย

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

เกรดเดอร์ตัวอย่าง

เกรดเดอร์ตัวอย่างอ่านข้อมูลนำเข้าในรูปแบบต่อไปนี้:

- บรรทัดที่ 1: จำนวนเต็ม n, l, u
- บรรทัดที่ 2: จำนวนเต็ม n จำนวน: $w_0, \ldots, w_{n-1}$