เรียงลำดับเทปปริศนา

นักโบราณคดีได้ค้นพบวัตถุโบราณสองชิ้น แต่ละชิ้นมีลักษณะเป็นเทปซึ่งบันทึกตัวเลขเทปละ N ตัวเรียงกันเป็นแนวเส้นตรง โดยตัวเลขของเทปอันแรกคือ $X[0], X[1], \ldots, X[N-1]$ และตัวเลขของเทปอันที่สองคือ $X[N], X[N+1], \ldots, X[2N-1]$ รับประกันว่าตัวเลขทั้ง $2N$ ตัวแตกต่างกันทุกคู่

เนื่องจากตัวเลขบนเทปดังกล่าวถูกบันทึกด้วยกรรมวิธีที่ซับซ้อนของชนเผ่าโบราณ คุณจึง**ไม่**สามารถอ่านตัวเลขบนเทปได้โดยตรง สิ่งที่คุณทราบคืออันดับของตัวเลขภายในแต่ละเทป นั่นคือคุณจะได้รับ

- อาร์เรย์ A ซึ่งเป็นการเรียงสับเปลี่ยนของ $0, 1, 2, \ldots, N-1$ ที่ทำให้

$$X[A[0]] < X[A[1]] < X[A[2]] < \cdots < X[A[N-1]]$$

- อาร์เรย์ B ซึ่งเป็นการเรียงสับเปลี่ยนของ $N, N+1, N+2 \ldots, 2N-1$ ที่ทำให้

$$X[B[0]] < X[B[1]] < X[B[2]] < \cdots < X[B[N-1]]$$

แต่ว่าคุณไม่ทราบข้อมูลใดๆ เลยเกี่ยวกับอันดับของตัวเลขระหว่างสองเทปนี้ เพื่อที่จะเรียงลำดับของตัวเลขในทั้งสองเทป นักโบราณคดีได้สร้างอุปกรณ์ชิ้นหนึ่ง ซึ่งมีหลักการทำงานดังนี้

อุปกรณ์ชิ้นนี้ประกอบไปด้วยหัวอ่าน 2 หัว หัวแรกชี้ไปตำแหน่งที่ P ($0 \le P < N$) ของเทปอันที่หนึ่ง และหัวที่สองชี้ไปที่ตำแหน่ง Q ($N \le Q < 2N$) ของเทปอันที่สอง ณ เวลาใดๆ ก็ตาม อุปกรณ์สามารถส่งกลับได้ว่า $X[P]$ มีค่าน้อยกว่า $X[Q]$ หรือไม่ แต่เนื่องจากการจะขยับตำแหน่งที่อ่านต้องทำการเลื่อนเทปโบราณซึ่งมีน้ำหนักมาก การจะเปลี่ยนตำแหน่งของหัวอ่านแรกจาก P_1 ไปเป็น P_2 จะต้องใช้พลังงาน $|P_1 - P_2|$ หน่วย และทำนองเดียวกัน การจะเปลี่ยนตำแหน่งของหัวอ่านที่สองจาก Q_1 ไปเป็น Q_2 จะต้องใช้พลังงาน $|Q_1 - Q_2|$ หน่วย

คุณต้องการเรียงลำดับเลขจากทั้งสองเทป นั่นคือคุณต้องการหาอาร์เรย์ C ซึ่งเป็นการเรียงสับเปลี่ยนของ $0, 1, 2, \ldots, 2N-1$ ที่ทำให้

$$X[C[0]] < X[C[1]] < X[C[2]] < \cdots < X[C[2N-1]]$$

รายละเอียดการเขียนโปรแกรม

คุณจะต้องเขียนฟังก์ชันดังต่อไปนี้

```
vector<int> sort_tapes(int N, vector<int> A, vector<int> B)
```

- ฟังก์ชันนี้จะถูกเรียกเพียงครั้งเดียว
- ค่า N แทนความยาวของเทป
- อาร์เรย์ A ความยาว N แทนลำดับของตัวเลขในเทปอันที่ 1

- อาร์เรย์ B ความยาว N แทนลำดับของตัวเลขในเทปอันที่ 2
- ฟังก์ชันนี้จะคืนค่าเป็นอาเรย์ C ซึ่งแทนลำดับของตัวเลขจากทั้งสองเทป

โดยภายในฟังก์ชัน `sort_tapes` สามารถเรียกฟังก์ชันต่อไปนี้ได้

```
bool compare(int P, int Q)
```

- ฟังก์ชันนี้จะทำให้หัวอ่านของอุปกรณ์ขยับมาที่ตำแหน่ง P และ Q และทำการเปรียบเทียบค่า $X[P]$ และ $X[Q]$ (เมื่อ $0 \le P < N$ และ $N \le Q < 2N$)
- ฟังก์ชันนี้จะคืนค่า `true` ถ้า $X[P] < X[Q]$ และคืนค่า `false` ถ้า $X[P] > X[Q]$

ขอบเขต

- $2 \le N \le 100\,000$
- $A[0], A[1], \ldots, A[N-1]$ เป็นการเรียงสับเปลี่ยนของ $0, 1, \ldots, N-1$
- $B[0], B[1], \ldots, B[N-1]$ เป็นการเรียงสับเปลี่ยนของ $N, N+1, \ldots, 2N-1$

ปัญหาย่อย

1. (10 คะแนน) $N \le 300$
2. (40 คะแนน) $N \le 30\,000$
3. (50 คะแนน) ไม่มีเงื่อนไขเพิ่มเติม

การให้คะแนน

ในการคิดคะแนน ถ้าผู้เข้าสอบเรียกฟังก์ชัน `compare` ทั้งหมด k ครั้ง ได้แก่ `compare`(P_1, Q_1), `compare`(P_2, Q_2), ..., `compare`(P_k, Q_k) แล้วค่าพลังงานรวมจะคิดจาก

$$\begin{aligned} E = &\big(|P_1 - P_2| + |P_2 - P_3| + \cdots + |P_{k-1} - P_k|\big) \\ &+ \big(|Q_1 - Q_2| + |Q_2 - Q_3| + \cdots + |Q_{k-1} - Q_k|\big) \end{aligned}$$

สำหรับแต่ละปัญหาย่อย ในปัญหาย่อยที่ i จะมีค่าเป้าหมาย T_i และคะแนนเต็ม S_i ดังนี้

1. $T_1 = 45\,000, S_1 = 10$
2. $T_2 = 65\,000\,000, S_2 = 40$
3. $T_3 = 450\,000\,000, S_3 = 50$

ในแต่ละข้อมูลทดสอบของปัญหาย่อยที่ i หากผู้เข้าแข่งขันทำค่าพลังงานรวม E ได้ไม่เกินเป้าหมาย T_i แล้วจะได้คะแนนเต็ม S_i

แต่หากทำค่าพลังงานรวมเกิน T_i ผู้เข้าแข่งขันจะได้คะแนนเท่ากับ

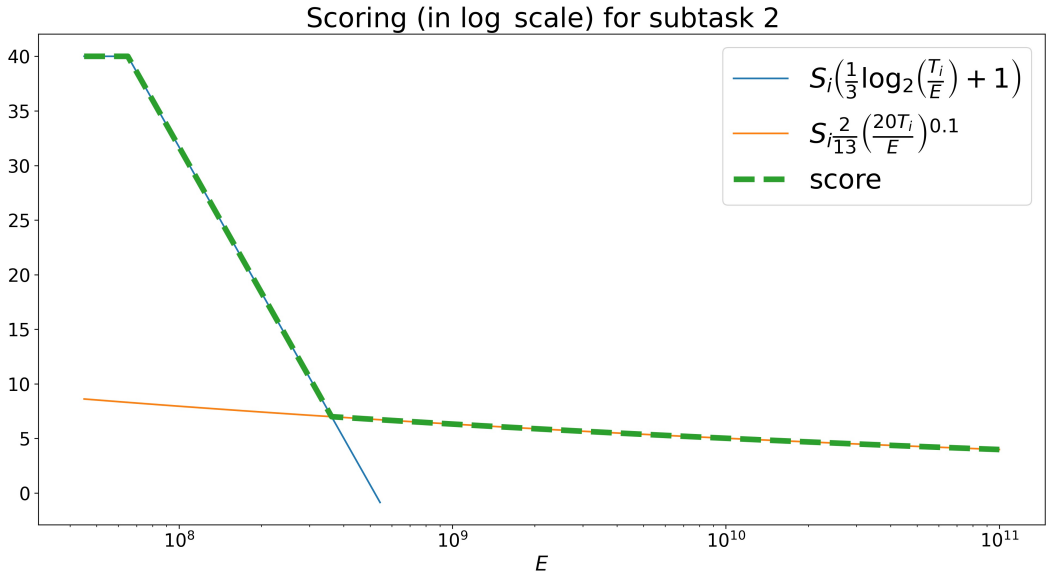
$$S_i \max\left(\frac{1}{3}\log_2\left(\frac{T_i}{E}\right)+1\,,\ \frac{2}{13}\left(\frac{20T_i}{E}\right)^{0.1}\right)$$

และคะแนนของแต่ละปัญหาย่อย คือค่าต่ำสุดของคะแนนของแต่ละข้อมูลทดสอบในปัญหาย่อยนั้น ๆ

หมายเหตุ

- หากคำตอบถูกต้องแต่ใช้ค่าพลังงานรวม E มากกว่าเป้าหมาย T_i ตัวตรวจจะแสดงผล `Output is partially correct`
- หาก $E \le T_i$ ตัวตรวจจะแสดงผล `Output is correct`
- ส่วนกรณีที่ตัวตรวจแสดงผล `Output isn't correct` จะเกิดขึ้นเมื่อคำตอบผิด นั่นคือค่า C ที่คืนมานั้นไม่ถูกต้อง

พิจารณาแผนภาพประกอบการให้คะแนน สำหรับปัญหาย่อย 2 ดังต่อไปนี้



(สำหรับปัญหาย่อยอื่น แผนภาพจะมีลักษณะคล้ายกัน เปลี่ยนเพียงค่า T_i และ S_i เท่านั้น เนื่องจากมีลักษณะคล้ายกันมากจึงไม่นำมาแนบ ณ ที่นี้)

ตัวอย่าง

พิจารณาการเรียกฟังก์ชัน

```
sort_tapes(3, [0,2,1], [5,3,4])
```

นั่นคือ คุณทราบว่า $X[0] < X[2] < X[1]$ และ $X[5] < X[3] < X[4]$ เพื่อหาข้อมูลเพิ่มเติม คุณอาจจะเรียกฟังก์ชัน `compare` ในลักษณะต่อไปนี้

ฟังก์ชัน	ค่าที่ return	ข้อมูลที่ได้รับ
`compare`$(0, 5)$	`true`	$X[0] < X[5]$
`compare`$(1, 4)$	`false`	$X[1] > X[4]$
`compare`$(2, 3)$	`true`	$X[2] < X[3]$
`compare`$(2, 5)$	`false`	$X[2] > X[5]$
`compare`$(1, 3)$	`false`	$X[1] < X[3]$

ข้อมูลข้างต้นนี้เพียงพอที่จะสรุปได้ว่า $X[0] < X[5] < X[2] < X[3] < X[4] < X[1]$ ฟังก์ชันนี้จึงควรส่งออกอาร์เรย์ $[0, 5, 2, 1, 3, 4]$ ในตัวอย่างนี้ ผู้เข้าแข่งขันใช้พลังงานรวมเท่ากับ

$$\begin{aligned}&\big(|0-1| + |1-2| + |2-2| + |2-1|\big) \\ &+ \big(|5-4| + |4-3| + |3-5| + |5-3|\big) = 9\end{aligned}$$

เกรดเดอร์ตัวอย่าง

เกรดเดอร์ตัวอย่างจะรับข้อมูลในรูปแบบดังต่อไปนี้

- บรรทัดที่ 1: N
- บรรทัดที่ 2: $X[0]$ $X[1]$ $X[2]$ $\ldots$ $X[2N-1]$

โดยที่ $1 \le X[i] \le 10^9$ ทุก $i = 0, 1, 2, \ldots, 2N-1$ และ $X[0], X[1], X[2], \ldots, X[2N-1]$ แตกต่างกันทุกคู่

เกรดเดอร์ตัวอย่างจะส่งข้อมูลคืนในรูปแบบดังต่อไปนี้

- บรรทัดที่ 1: E (ค่าพลังงานรวมที่ใช้)
- บรรทัดที่ 2: $C[0]$ $C[1]$ $C[2]$ $\ldots$ $C[2N-1]$

ข้อจำกัด

- Time limit: 5 seconds
- Memory limit: 512 MB