Rainforest Jumps

ในดินแดนป่าดิบชื้นแห่งสุมาตรา มีต้นไม้ N ต้นเรียงเป็นแถว โดยมีหมายเลขเรียงกันตั้งแต่ 0 ถึง $N-1$ จากซ้ายไปขวา ต้นไม้เหล่านี้มีความสูง **แตกต่างกันทั้งหมด** โดยต้นไม้ที่ i จะมีความสูง $H[i]$

ปัก เด็งเคล็กกำลังฝึกให้อุรังอุตังกระโดดไปมาระหว่างต้นไม้ ในการกระโดดหนึ่งครั้ง อุรังอุตังสามารถกระโดดจากยอดต้นไม้ต้นหนึ่ง ไปยังยอดของต้นไม้ที่อยู่ด้านซ้ายหรือขวาที่อยู่ใกล้ที่สุดที่มีความสูงมากกว่าต้นที่เธออยู่ ถ้าจะกล่าวอย่างเป็นทางการก็คือ ถ้าอุรังอุตังอยู่ที่ต้นไม้ x เธอจะสามารถกระโดดไปต้นไม้ y ก็ต่อเมื่อ (if and only if) เงื่อนไขข้อใดข้อหนึ่งต่อไปนี้เป็นจริง

- y เป็นจำนวนเต็มที่ไม่เป็นลบที่มากที่สุดแต่น้อยกว่า x ที่ $H[y] > H[x]$; หรือ
- y เป็นจำนวนเต็มที่ไม่เป็นลบที่น้อยที่สุดแต่มากกว่า x ที่ $H[y] > H[x]$

ปัก เด็งเคล็กมีแผนการกระโดดอยู่ Q แผน แต่ละแผนจะสามารถเขียนแทนได้ด้วยจำนวนเต็มสี่จำนวน A, B, C, และ D ($A \le B < C \le D$) สำหรับแต่ละแผน ปัก เด็งเคล็กอยากทราบว่าเป็นไปได้หรือไม่ที่อุรังอุตังจะเริ่มต้นกระโดดที่ต้นไม้ s บางต้น ($A \le s \le B$) และสิ้นสุดการกระโดดที่ต้นไม้ e สักต้น ($C \le e \le D$) โดยใช้ลำดับการกระโดดตามที่ระบุข้างต้น ถ้าเป็นไปได้ ปัก เด็งเคล็กอยากทราบจำนวนการกระโดดที่น้อยที่สุดที่อุรังอุตังต้องการใช้ในการกระโดดตามแผนการนั้น

Implementation Details

คุณต้องเขียนฟังก์ชันสองฟังก์ชันต่อไปนี้

```
void init(int N, int[] H)
```

- N: จำนวนต้นไม้
- H: อาร์เรย์ความยาว N โดยที่ $H[i]$ คือความสูงของต้นไม้ i
- ฟังก์ชันนี้จะถูกเรียกหนึ่งครั้งเท่านั้น ก่อนการเรียกฟังก์ชัน `minimum_jumps` ทั้งหมด

```
int minimum_jumps(int A, int B, int C, int D)
```

- A, B: ขอบเขตของต้นไม้ที่อุรังอุตังเริ่มต้นกระโดด
- C, D: ขอบเขตของต้นไม้ที่อุรังอุตังจะต้องสิ้นสุดการกระโดด
- ฟังก์ชันนี้จะต้องคืนจำนวนการกระโดดที่น้อยที่สุดที่จะทำตามแผน หรือ -1 ถ้าไม่สามารถทำได้
- ฟังก์ชันนี้จะถูกเรียก Q ครั้ง

Example

พิจารณาการเรียกฟังก์ชันดังนี้:

```
init(7, [3, 2, 1, 6, 4, 5, 7])
```

หลังจากการเตรียมค่าเริ่มต้นแล้ว พิจารณาการเรียกฟังก์ชันต่อไปนี้:

```
minimum_jumps(4, 4, 6, 6)
```

การเรียกฟังก์ชันดังกล่าวหมายความว่าอุรังอุตังจะต้องเริ่มที่ต้นไม้ 4 (ที่มีความสูง 4) และสิ้นสุดการกระโดดที่ต้นไม้ 6 (ที่มีความสูง 7) วิธีหนึ่งที่จะกระโดดให้ได้ตามแผนและใช้จำนวนการกระโดดน้อยที่สุดคือเริ่มกระโดดไปยังต้นไม้ 3 (ที่สูง 6) จากนั้นกระโดดไปที่ต้นไม้ 6 อีกวิธีคือกระโดดไปที่ต้นไม้ 5 (ที่สูง 5) แล้วกระโดดต่อไปที่ต้นไม้ 6 ดังนั้นฟังก์ชัน `minimum_jumps` จะต้องคืนค่า 2

พิจารณาการเรียกฟังก์ชันตัวอย่างด้านล่างอีกรูปแบบหนึ่ง:

```
minimum_jumps(1, 3, 5, 6)
```

การเรียกฟังก์ชันดังกล่าวหมายความว่าอุรังอุตังจะต้องเริ่มที่ต้นไม้ 1 (ที่มีความสูง 2), ต้นไม้ 2 (ที่สูง 1) หรือต้นไม้ 3 (ที่สูง 6) และสิ้นสุดการกระโดดที่ต้นไม้ 5 (ที่มีความสูง 5) หรือต้นไม้ 6 (ที่สูง 7) วิธีเดียวที่จะกระโดดให้ได้ตามแผนและใช้จำนวนการกระโดดน้อยที่สุดคือเริ่มกระโดดที่ต้นไม้ 3 จากนั้นกระโดดไปที่ต้นไม้ 6 ด้วยการกระโดดครั้งเดียว ดังนั้นฟังก์ชัน `minimum_jumps` จะต้องคืนค่า 1

พิจารณาอีกตัวอย่างหนึ่ง:

```
minimum_jumps(0, 1, 2, 2)
```

การเรียกฟังก์ชันดังกล่าวหมายความว่าอุรังอุตังจะต้องเริ่มที่ต้นไม้ 0 (ที่มีความสูง 3) หรือต้นไม้ 1 (ที่สูง 2) และสิ้นสุดการกระโดดที่ต้นไม้ 2 (ที่มีความสูง 1) เนื่องจากต้นไม้ 2 เป็นต้นไม้ที่เตี้ยที่สุด จึงไม่มีทางที่จะไปถึงได้โดยการกระโดดจากต้นไม้ต้นอื่น ๆ ที่สูงกว่ามัน ดังนั้นฟังก์ชัน `minimum_jumps` จะต้องคืนค่า -1

Constraints

- $2 \leq N \leq 200\,000$
- $1 \leq Q \leq 100\,000$
- $1 \leq H[i] \leq N$ (สำหรับทุก $0 \leq i \leq N-1$)
- $H[i] \neq H[j]$ (สำหรับทุก $0 \leq i < j \leq N-1$)
- $0 \leq A \leq B < C \leq D \leq N-1$

Subtasks

1. (4 points) $H[i] = i + 1$ (สำหรับทุก $0 \leq i \leq N-1$)
2. (8 points) $N \leq 200$, $Q \leq 200$

3. (13 points) $N \leq 2000$, $Q \leq 2000$
4. (12 points) $Q \leq 5$
5. (23 points) $A = B$, $C = D$
6. (21 points) $C = D$
7. (19 points) ไม่มีเงื่อนไขเพิ่มเติม

Sample Grader

เกรดเดอร์ตัวอย่างอ่านข้อมูลนำเข้าในรูปแบบดังนี้:

- line 1: $N\ Q$
- line 2: $H[0]\ H[1]\ \ldots\ H[N-1]$
- line $3+i$ ($0 \leq i \leq Q-1$): $A\ B\ C\ D$ สำหรับการเรียกฟังก์ชัน `minimum_jumps` ครั้งที่ i

เกรดเดอร์ตัวอย่างพิมพ์ผลลัพธ์ในรูปแบบดังนี้:

- line $1+i$ ($0 \leq i \leq Q-1$): ค่าที่คืนจากการเรียก `minimum_jumps` ครั้งที่ i