Gap

มีตัวเลขจำนวนเต็มไม่ติดลบอยู่ทั้งหมด N ตัว $a_1, a_2, \ldots, a_N$ ซึ่งมีคุณสมบัติว่า $0 \le a_1 < a_2 < \cdots < a_N \le 10^{18}$. Jeehak อยากจะทราบค่าที่ *มากที่สุดที่เป็นไปได้* ของ $a_{i+1} - a_i$ โดย i มีค่าได้ตั้งแต่ 1 ถึง $N-1$. ตัวเลขอินพุตตัวเลขอินพุตจะไม่ถูกป้อนเข้าสู่โปรแกรมของ Jeehak โดยตรง แต่จะสามารถเข้าถึงได้ผ่านทางฟังก์ชันพิเศษ. ดูรายละเอียดได้ที่หัวข้อ Implementation สำหรับภาษาของคุณ

Task

ช่วย Jeehak เขียนฟังก์ชันซึ่งคืนค่าที่มากที่สุดของ $a_{i+1} - a_i$ โดยที่ i มีค่าได้ตั้งแต่ 1 ถึง $N-1$.

Implementation สำหรับ C และ C++

คุณจะต้องเขียนฟังก์ชั่น `findGap(T, N)` ซึ่งรับพารามิเตอร์ต่อไปนี้ และคืนค่าเป็นจำนวนเต็มชนิด `long long`:

- `T` — หมายเลขของภารกิจย่อย (1 หรือ 2)
- `N` — จำนวนตัวเลขที่มีให้

ฟังก์ชั่น `findGap` ของคุณสามารถเรียกใช้ฟังก์ชั่น `MinMax(s, t, &mn, &mx)` โดยที่สองพารามิเตอร์แรก `s` และ `t` เป็นจำนวนเต็มชนิด `long long` และสองพารามิเตอร์หลัง `&mn` and `&mx` เป็นพอยเตอร์ชี้ไปที่ตัวแปรจำนวนเต็มชนิด `long long`, กล่าวคือ `mn` and `mx` เป็นตัวแปรจำนวนเต็มชนิด `long long`. เมื่อ `MinMax(s, t, &mn, &mx)` ทำงานเสร็จ, ตัวแปร `mn` จะมีค่าเท่ากับ a_i ที่น้อยที่สุดที่มากกว่าหรือเท่ากับ `s` ส่วนตัวแปร `mx` จะมีค่าเท่ากับ a_j ที่มากที่สุดที่น้อยกว่าหรือเท่ากับ `t`. ในกรณีที่ไม่มีตัวเลขอินพุตใดๆ เลยระหว่าง `s` กับ `t` (โดยรวมทั้งสองตัวนั้นด้วย), ค่าของ `mn` และ `mx` จะเท่ากับ `-1`. ค่าของ `s` ไม่ควรจะมากกว่าค่าของ `t` เมื่อเรียกใช้ `MinMax`. ถ้าไม่เช่นนั้นแล้ว โปรแกรมจะหยุดทำงานและจะมี exit code ที่ไม่ใช่ศูนย์

Implementation สำหรับ Pascal

คุณจะต้องเขียนฟังก์ชั่น `findGap(T, N)` ซึ่งรับพารามิเตอร์ต่อไปนี้ และคืนค่าเป็นจำนวนเต็มชนิด `Int64`:

- `T` — หมายเลขของภารกิจย่อย (1 หรือ 2) (ชนิด `Integer`)
- `N` — จำนวนตัวเลขที่มีให้ (ชนิด `LongInt`)

ฟังก์ชั่น `findGap` ของคุณสามารถเรียกใช้โพรซีเดอร์ `MinMax(s, t, mn, mx)` โดยที่สองพารามิเตอร์แรก `s` และ `t` เป็นจำนวนเต็มชนิด `Int64` และสองพารามิเตอร์หลัง `mn` and `mx` เป็นตัวแปร **called by reference** ชนิด `Int64`, กล่าวคือ `mn` and `mx` เป็นตัวแปรจำนวนเต็มชนิด `Int64`. เมื่อ `MinMax(s, t, mn, mx)` ทำงานเสร็จ, ตัวแปร `mn` จะมีค่าเท่ากับ a_i ที่น้อยที่สุดที่มากกว่าหรือเท่ากับ `s` ส่วนตัวแปร `mx` จะมีค่าเท่ากับ a_j

ที่มากที่สุดที่น้อยกว่าหรือเท่ากับ t. ในกรณีที่ไม่มีตัวเลขอินพุตใดๆ เลยระหว่าง s กับ t (โดยรวมทั้งสองตัวนั้นด้วย), ค่าของ mn และ mx จะเท่ากับ -1. ค่าของ s ไม่ควรจะมากกว่าค่าของ t เมื่อเรียกใช้ MinMax. ถ้าไม่เช่นนั้นแล้ว โปรแกรมจะหยุดทำงาน

Implementation สำหรับทุกคน

นอกเหนือจากข้อบังคับพื้นฐาน (เวลา , หน่วยความจำ , ไร้ runtime errors, ฯลฯ) โปรแกรมของคุณจะต้องเป็นไปตามข้อกำหนดเหล่านี้ด้วยจึงจะเรียกได้ว่าสามารถแก้ test case ได้

- ฟังก์ชั่น findGap ของคุณจะต้องคืนค่าคำตอบที่ถูกต้อง,
- ราคา M ของการเรียกใช้ MinMax จะต้องไม่สูงเกินกว่าขอบเขตที่อนุญาต (ดูหัวข้อ Scoring).

Example for C, C++

พิจารณากรณีที่ $N = 4$ และ $a_1 = 2, a_2 = 3, a_3 = 6,$ และ $a_4 = 8$.

คำตอบคือ 3 ซึ่งฟังก์ชั่น findGap สามารถคำนวณและคืนค่าได้ถ้า MinMax ถูกเรียกใช้ดังต่อไปนี้:

- MinMax(1, 2, &mn, &mx) ถูกเรียกใช้ และ mn กับ mx ต่างมีค่าเท่ากับ 2.
- MinMax(3, 7, &mn, &mx) ถูกเรียกใช้ และ mn มีค่าเท่ากับ 3 ส่วน mx มีค่าเท่ากับ 6.
- MinMax(8, 9, &mn, &mx) ถูกเรียกใช้ และ mn กับ mx ต่างมีค่าเท่ากับ 8.

Example for Pascal

พิจารณากรณีที่ $N = 4$ และ $a_1 = 2, a_2 = 3, a_3 = 6,$ และ $a_4 = 8$.

คำตอบคือ 3 ซึ่งฟังก์ชั่น findGap สามารถคำนวณและคืนค่าได้ถ้า MinMax ถูกเรียกใช้ดังต่อไปนี้:

- MinMax(1, 2, mn, mx) ถูกเรียกใช้ และ mn กับ mx ต่างมีค่าเท่ากับ 2.
- MinMax(3, 7, mn, mx) ถูกเรียกใช้ และ mn มีค่าเท่ากับ 3 ส่วน mx มีค่าเท่ากับ 6.
- MinMax(8, 9, mn, mx) ถูกเรียกใช้ และ mn กับ mx ต่างมีค่าเท่ากับ 8.

Scoring

สำหรับทุกภารกิจย่อย ข้อกำหนด $2 \le N \le 100,000$ จะเป็นจริงเสมอ

ภารกิจย่อยที่ 1 (30 คะแนน): การเรียกใช้ MinMax แต่ละครั้งจะเพิ่มค่า M ขึ้นอีก 1. คุณจะได้คะแนนเต็มสำหรับภารกิจย่อยนี้ถ้า $M \le \frac{N+1}{2}$ สำหรับ test cases ทั้งหมด.

ภารกิจย่อยที่ 2 (70 คะแนน): ให้ k เป็นจำนวนตัวเลขอินพุตที่มีค่ามากกว่าหรือเท่ากับ s แต่น้อยกว่าหรือเท่ากับ t ในการเรียกใช้ MinMax ครั้งหนึ่งๆ. แต่ละครั้งที่เรียกใช้ MinMax ค่า M จะเพิ่มขึ้นอีก $k + 1$. คะแนนรวมสุดท้ายจะถูกคำนวณโดยใช้กฎดังนี้:
คะแนนรวมสุดท้ายของภารกิจย่อยจะเท่ากับคะแนนที่ต่ำที่สุดที่คุณได้รับจาก test cases

ทั้งหมด. สำหรับแต่ละ test case ถ้า $M \leq 3N$ คุณจะได้คะแนน 70;
ไม่เช่นนั้นแล้วคุณจะได้คะแนนเท่ากับ $\frac{60}{\sqrt{\frac{M}{N}+1-1}}$.

Experimentation

ตัวให้คะแนนตัวอย่างซึ่งสามารถดาวน์โหลดได้จากระบบให้คะแนนจะอ่านข้อมูลจากอินพุตมาตร
บรรทัดแรกของอินพุตประกอบด้วยจำนวนเต็มสองตัว: หมายเลขภารกิจย่อย T, และ N.
บรรทัดต่อไปประกอบด้วยจำนวนเต็ม N ตัวเรียงจากน้อยไปมาก.
ตัวให้คะแนนตัวอย่างจะเขียนข้อมูลเหล่านี้ออกสู่เอาพุตมาตรฐาน: มูลค่าที่ได้รับการคืนค่าจาก `findGap` ในบรรทัดแรก และค่าของ M ที่เหมาะสมสำหรับภารกิจย่อยที่ test case ตัวอย่างได้กำหนดไว้

ด้านล่างนี้เป็นอินพุตของตัวอย่างที่ให้ไว้ด้านบน:

```
2 4
2 3 6 8
```