

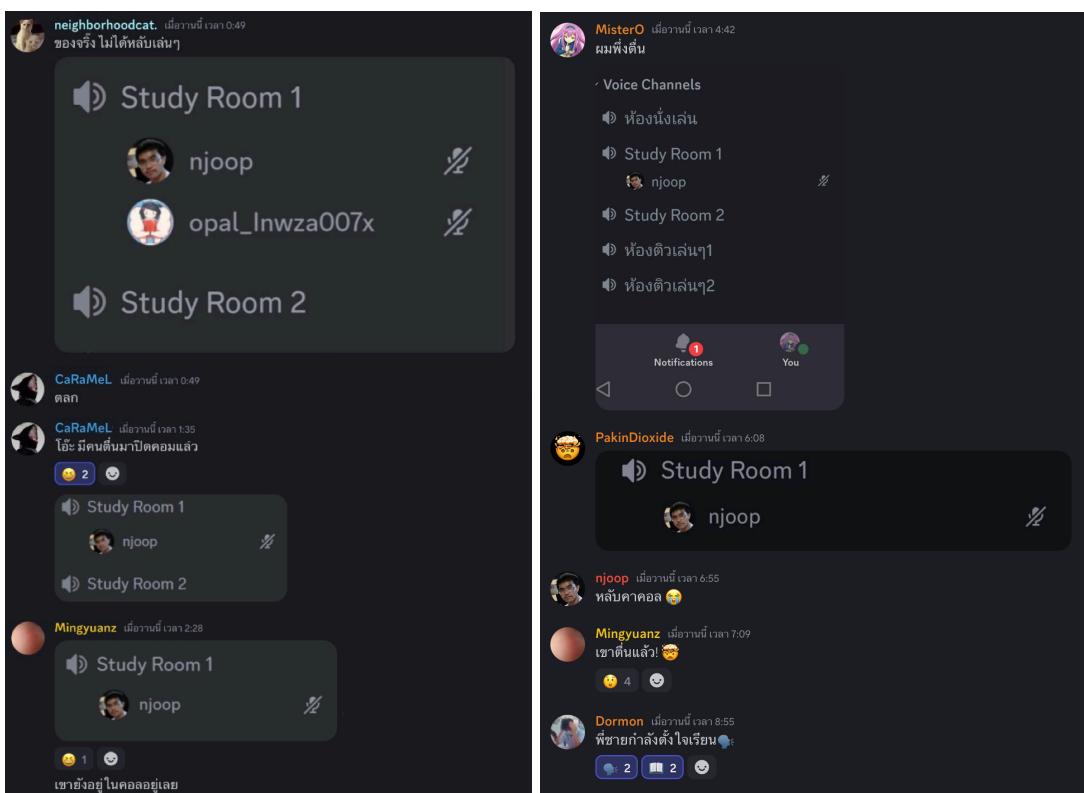


sleepntjoop

1 second, 256 megabytes

By [njoop, Mingyuanz](#)

เมื่อ njoop ได้ multitask วิจัยเกี่ยวกับ Sorting Algorithm พร้อมกับประชุมกับลูกน้องอย่างจริงจังต่อนั้น njoop ดันบังเอิญบรรลุธรรม เข้าถึงการใช้ Sleep Sort Algorithm อย่างถ่องแท้ ทำให้เข้า [หลักค่า discord](#) ถึงขนาดที่ neighborhoodcat., CaRaMeL, Mingyuanz, MisterO, Pakindioxide, และ Dormon ผู้ที่เดินผ่านไปผ่านมาแคล้ว นั้น หมายอกล้อเขากันอย่างแมมันส์ เขาหลับสนิทเกินไป จนลืมตำราสำคัญไว้ในที่ประชุม ซึ่ง Mingyuanz ก็เห็นเข้า แล้วนึกขึ้นได้ว่าเป็นตำราที่ขาดจากห้อง DevJ ศิษย์เอกของเข้า ทำให้เขาจ่วยโอกาสขโนย ตำรา "Scroll of the Infinite Shuffle" มาได้



รูปที่ 1: njoop ดันบังเอิญบรรลุธรรม เข้าถึงการใช้ Sleep sort อย่างถ่องแท้ ใน [discord server](#) แห่งหนึ่ง

เมื่อ njoop ได้เห็น กีร์สิกโกรธเป็นอย่างมาก เพราะในตำราที่มีคำว่า Bogo Sort, Quantum Bogo sort และ Bogobogo Sort ที่ใช้ Time Complexity เพียงแค่ $O(1)$ เท่านั้น และ njoop ต้องการจะศึกษามันในภายหลัง แต่ DevJ ดันขโนยตำราไปเป็นของเขาเองซะแล้ว njoop เลยต้องการทำลายตำราที่ทิ้งเสีย

สถานที่ที่ DevJ ยืนอยู่นั้น เป็นทางเดินยาวๆ ที่ประกอบไปด้วยบล็อก N บล็อก เรียงติดกัน แต่ละบล็อกจะ มีหมายเลข ของตัวเองที่แตกต่างกัน ตำราเหล่านั้นมีน้ำหนักมาก จะต้องวางอยู่กับที่บนบล็อกใดบล็อกหนึ่งตลอดเวลา แต่ DevJ ก็สามารถย้ายตำราเพื่อไปวางอีกบล็อกหนึ่งได้เสมอ เช่นกัน



สิ่งที่ DevJ สามารถทำได้คือ สร้างบล็อกทางเดินใหม่ มาต่อป้ายด้านหน้าหรือป้ายด้านหลังของทางเดิน ส่วน njoop จะสามารถทำลายบล็อกป้ายด้านหน้าหรือด้านหลังของทางเดินที่มีอยู่แล้ว

DevJ ได้ใช้ขาคิสังเกตขั้นสูง ที่ได้ฝึกมาจาก Mingyuanz ประมาณการยแท่งศาสตร์ Competitive Programming (CP) หนึ่งในผู้บรรลุการใช้ Bogo Sort, Miracle Sort, Sleep Sort และอื่นๆ จึงมองเห็นอนาคตได้มากถึง $1.5 \cdot 10^6$ วินาที ว่า njoop จะทำลายบล็อกป้ายฝั่งไหน ในลำดับใดบ้าง จึงได้เริ่มคิดแผนของตนเองแล้วเมื่อกันว่าจะต่อบล็อกที่ป้ายด้านไหนบ้าง แทรกระหว่างลำดับการทำลายบล็อกของ njoop ทำให้โดยสรุปแล้ว ทั้ง DevJ และ njoop จะมีลำดับการทำลายหรือสร้างบล็อกที่กำหนดไว้อย่างชัดเจนแล้ว รวมทั้งหมด K ครั้ง แต่ละครั้งใช้เวลา 1 วินาที โดยในแต่ละวินาที จะเกิดเหตุการณ์ขึ้นตามลำดับดังนี้

วินาทีที่ 0 เป็นวินาทีเริ่มแรกที่ DevJ ใช้เตรียมตัว ซึ่ง DevJ จะต้องย้ายตัวไปไว้บล็อกโดยบล็อกหนึ่งของทางเดินที่มีอยู่ในตอนแรก (นับเป็นการย้ายตัว 1 ครั้ง)

วินาทีที่ 1 ถึง K ในแต่ละวินาทีจะมี 2 ขั้นตอนตามลำดับดังนี้

1. DevJ สามารถย้ายตัว จากบล็อกเดิมไปวางไว้อีกบล็อกหนึ่งบนทางเดินที่มีอยู่ โดย DevJ สามารถเลือกได้ว่า จะทำหรือไม่ทำก็ได้
2. njoop ทำลายบล็อก หรือ DevJ สร้างบล็อก ที่ป้ายข้างไดข้างหนึ่งของทางเดิน โดยมี operation เหล่านี้จะ มี 4 รูปแบบได้แก่
 - 1 x - DevJ สร้างบล็อกที่มีหมายเลข x ต่อป้ายด้านหน้าของทางเดิน
 - 2 x - DevJ สร้างบล็อกที่มีหมายเลข x ต่อป้ายด้านหลังของทางเดิน
 - 3 - njoop ทำลายบล็อกป้ายด้านหน้าของทางเดิน
 - 4 - njoop ทำลายบล็อกป้ายด้านหลังของทางเดิน

โดยหมายเขของแต่ละบล็อกที่ DevJ สร้างขึ้นมาแต่ละครั้งนั้น จะไม่ซ้ำกับหมายเลขของบล็อกใดๆบนทางเดินที่มีอยู่ ณ วินาทีที่สร้างเลย

และเนื่องจาก njoop ยังไม่ได้บรรลุเข้าถึง algorithm สำคัญในตัวรันน์เลย njoop จึง Random Shuffle ลำดับการทำลายบล็อกของเขาร่วมกับ DevJ จะสร้างบล็อกใหม่ ทำให้การันตีได้ว่า ไม่มีวินาทีไหนเลย ที่ njoop จะสามารถทำลายทางเดินทั้งหมดได้ (DevJ จะมีบล็อกบนทางเดินให้วางตัวอยู่เสมอ)

เมื่อ njoop ทำลายบล็อกที่มีตัวร่วงอยู่ ตัวรากถูกทำลายไปด้วย ทำให้บังคับ DevJ จำเป็นต้องย้ายตัวเพื่อไม่ให้ถูกทำลายในเวลาต่อมา แต่ DevJ ดันลืมวางแผนไปสนใจเลยว่า เขายังต้องย้ายตัวตอนวินาทีไหน ไปไว้บล็อกหมายเลขใดบ้าง ซึ่งตัวรันน์มีหน้าที่มากและการสร้างบล็อกก็ใช้พลังงานสูงมาก เช่นกัน DevJ จึงต้องการย้ายตัวรากเป็นจำนวนครั้งที่น้อยที่สุดเท่าที่จะทำได้

โจทย์ เนื่องจาก DevJ ต้องໂฟกสักกับการสร้างบล็อกอยู่ จึงได้มารอให้คุณเขียนโปรแกรมให้เขาน้อย ว่าจะต้องย้ายตัวรันน์ที่สุดกี่ครั้ง (นับครั้งเริ่มต้นด้วย) และแต่ละครั้งต้องย้ายไปที่บล็อกหมายเลขใดบนทางเดิน ณ วินาทีที่เท่าไหร่บ้าง โดยการรับค่าของโปรแกรมจะเป็นดังนี้



ข้อมูลนำเข้า

บรรทัดแรก ประกอบด้วยจำนวนเต็มสองจำนวนได้แก่ $N \ K$:

โดยจำนวนเต็ม N คือ จำนวนของบล็อกนทางเดินที่มีอยู่ในตอนแรก ($1 \leq N \leq 100,000$)

โดยจำนวนเต็ม K คือ จำนวนครั้งของการทำลายหรือสร้างบล็อกที่ถูกกำหนดไว้แล้ว ($1 \leq K \leq 1,500,000$)

บรรทัดที่ 2 ประกอบด้วยจำนวนเต็ม N จำนวน, a_1, a_2, \dots, a_n

โดยจำนวนเต็ม a_i ($1 \leq i \leq N$) คือ หมายเลขของบล็อกที่ i นับจากฝั่งด้านหน้าของทางเดินเริ่มต้น ($1 \leq a_i \leq 10^9$)

บรรทัดที่ 3 ถึง $K + 2$ บรรทัดที่ $i + 2$ ($1 \leq i \leq K$) ประกอบด้วยจำนวนเต็ม 1 ถึง 2 จำนวน

ระบุถึง operation ในวินาทีที่ i ซึ่งมีรูปแบบดังนี้

1 x - DevJ สร้างบล็อกที่มีหมายเลข x ต่อปลายด้านหน้าของทางเดิน ($1 \leq x \leq 10^9$)

2 x - DevJ สร้างบล็อกที่มีหมายเลข x ต่อปลายด้านหลังของทางเดิน ($1 \leq x \leq 10^9$)

3 - njoop ทำลายบล็อกปลายด้านหน้าของทางเดิน

4 - njoop ทำลายบล็อกปลายด้านหลังของทางเดิน

ข้อมูลส่งออก

บรรทัดแรก จำนวนเต็ม M แทน จำนวนครั้งที่น้อยที่สุดที่ DevJ ต้องย้ายตำรา เพื่อไม่ให้ถูก njoop ทำลาย

บรรทัดที่ 2 ถึง $M + 1$ บรรทัดที่ $i + 1$ ($1 \leq i \leq M$)

ประกอบไปด้วยจำนวนเต็ม 2 จำนวน x_i และ t_i ค่านี้ด้วยช่องว่างหนึ่งช่อง

x_i แทน หมายเลขของบล็อกที่ DevJ จะย้ายตำราไปวาง ในการย้ายตำราครั้งที่ i

t_i แทน เวลา(วินาที) ที่ DevJ จะย้ายตำราครั้งที่ i

**หากข้อมูลนำเข้า มีข้อมูลส่งออกที่ถูกต้องได้หลายรูปแบบ จะสามารถเลือกตอบรูปแบบที่ถูกต้องรูปแบบใดก็ได้



ตัวอย่างข้อมูลนำเข้าและข้อมูลส่งออก

ตัวอย่างข้อมูลนำเข้า	ตัวอย่างข้อมูลส่งออก
2 5	1
4 2	4 0
4	
2 1	
1 7	
3	
4	
3 11	3
1 2 3	2 0
1 4	4 3
2 5	8 9
4	
4	
4	
4	
1 8	
1 9	
4	
3	
2 7	

คำอธิบาย

ตัวอย่างที่ 1 ให้หมายเลขที่เป็นสีฟ้า สื่อถึงบล็อกที่ DevJ นำต่อไปว่าไว้ ดังนี้

- 4 2 (DevJ ได้ย้ายต่อมาไว้ที่บล็อกหมายเลข 4 ณ วินาทีที่ 0)
- 4 (njoop ทำลายบล็อกข้างหลัง)
- 4 1 (DevJ สร้างบล็อกที่มีน้ำหนัก 1 ต่อด้านหลัง)
- 7 4 1 (DevJ สร้างบล็อกที่มีน้ำหนัก 7 ต่อด้านหน้า)
- 4 1 (njoop ทำลายบล็อกข้างหน้า)
- 4 (njoop ทำลายบล็อกข้างหลัง)

เมื่อ DevJ ย้ายต่อมาไว้ที่บล็อกหมายเลข 4 ในวินาทีที่ 0 จะทำให้ไม่ต้องย้ายต่อมาในเวลาต่อมาเลย



ตัวอย่างที่ 2 ให้หมายเลขที่เป็นสีฟ้า สื่อถึงบล็อกที่ DevJ นำทำรำไปว่างไว้ ดังนี้

- 1 **2** 3 (DevJ ได้ย้ายตำราไว้ที่บล็อกหมายเลข 2 ณ วินาทีที่ 0)
- 4 1 **2** 3
- 4 1 **2** 3 5
- 4** 1 2 3 (การย้ายตำรา ครั้งที่ 2 ณ วินาทีที่ 3)
- 4** 1 2
- 4** 1
- 4**
- 8 **4**
- 9 8 **4**
- 9 **8** (การย้ายตำรา ครั้งที่ 3 ณ วินาทีที่ 9)
- 8**
- 8** 7

สามารถแสดงได้ว่า 3 เป็นจำนวนครั้งในการย้ายตำราที่นโยบายที่สุดที่เป็นไปได้ ดังนั้น จึงแสดงตามตัวอย่าง

การให้คะแนน

คะแนนเต็ม 300 คะแนน มี 4 กลุ่มชุดทดสอบ

10 คะแนน: มีแค่ operation 2 และ 4 เท่านั้น

40 คะแนน: สามารถมี operation 1 และ 3 ได้เพียง 1 ครั้งเท่านั้น

75 คะแนน: ($1 \leq N, K \leq 5000$)

175 คะแนน: ไม่มีเงื่อนไขเพิ่มเติม

**จะได้คะแนนในแต่ละกลุ่มชุดทดสอบ ก็ต่อเมื่อโปรแกรมให้ผลลัพธ์ถูกต้องในชุดทดสอบอย่างทั้งหมด

คำแนะนำ

หากใช้ภาษา C++ แนะนำให้เพิ่มคำสั่ง `cin.tie(nullptr)->sync_with_stdio(false);`
และให้ใช้ '\n' แทน endl เช่น `cout << "Hello World" << '\n';`

หากใช้ภาษา C/C++ แนะนำให้ใช้คอมไพล์เตอร์ GNU G++17 7.3.0 ในการ Submit Code