ตั๋วงานเทศกาล (tickets)

ริงโกอยู่ในงานเทศกาลในสิงค์โปร์ เขามีตั๋วชิงรางวัลในกระเป๋าสำหรับเล่นเกมชิงรางวัลที่แผงเกม ตั๋วแต่ละใบจะมีสีหนึ่งสีจากทั้งหมด n สีที่เป็นไปได้ และมีจำนวนเต็มที่ไม่เป็นลบระบุอยู่ จำนวนเต็มที่พิมพ์อยู่นั้นอาจจะซ้ำกันได้ เนื่องจากความผิดพลาดบางอย่าง n จะเป็น**จำนวนเต็มคู่**เสมอ

สำหรับแต่ละสี ริงโกจะมีตั๋วอยู่ทั้งสิ้น m ใบพอดี รวมแล้วเขามีตั๋วทั้งสิ้น $n \cdot m$ ใบ ตั๋วใบที่ j ของสี i จะมีจำนวนเต็ม $x[i][j]$ พิมพ์อยู่ (สำหรับ $0 \leq i \leq n-1$ และ $0 \leq j \leq m-1$)

เกมชิงรางวัลจะเล่นทั้งสิ้น k รอบ เรียกเป็นรอบที่ 0 ถึง $k-1$ เกมแต่ละรอบจะดำเนินไปดังนี้

- ริงโกจะเลือก**เซต**ที่ประกอบไปด้วยตั๋ว n ใบจากกระเป๋าของเขา โดยที่ตั๋วแต่ละใบจะมีสีแตกต่างกัน เขาจะให้เซตของตั๋วเหล่านั้นกับพิธีกรผู้จัดการเกม
- พิธีกรจะจดจำนวนเต็มที่พิมพ์อยู่บนตั๋วเหล่านั้น $a[0], a[1], \ldots, a[n-1]$ ลำดับของจำนวนเต็มเหล่านี้ไม่มีความสำคัญ
- พิธีกรจะดึงบัตรพิเศษออกมาจากกล่องปริศนาและจดจำนวนเต็ม b ที่พิมพ์อยู่บนบัตรนั้น
- พิธีกรจะคำนวนผลต่างสัมบูรณ์ระหว่าง $a[i]$ กับ b สำหรับ i จาก 0 ถึง $n-1$ ให้ S เป็นผลรวมของผลต่างเหล่านี้
- สำหรับรอบนี้ พิธีกรจะมอบรางวัลมูลค่าเท่ากับ S กับริงโก
- ตั๋วในเซตเหล่านั้นจะถูกทิ้งไป และจะไม่สามารถใช้ในรอบถัด ๆ ไปได้อีก

ตั๋วที่เหลือหลังจากการเล่น k รอบจะถูกทิ้งไป

ริงโกแอบสังเกตเห็นว่าเกมชิงรางวัลดังกล่าวนั้นถูกโกง! ภายในกล่องปริศนามีเครื่องพิมพ์บัตรอยู่ ในแต่ละรอบ พิธีกรจะหาจำนวนเต็ม b ที่ทำให้มูลค่าของรางวัลในรอบนั้นน้อยที่สุด จำนวนเต็มที่ถูกเลือกนั้นจะถูกพิมพ์ลงบนบัตรที่ดึงจากกล่องปริศนา

ด้วยข้อมูลนี้ ริงโกต้องการที่จะจัดสรรตั๋วสำหรับเล่นเกมในรอบต่าง ๆ นั่นคือ เขาต้องการเลือกเซตของตั๋วสำหรับใช้ในการเล่นแต่ละรอบเพื่อที่จะทำให้มูลค่ารางวัลรวมที่เขาได้รับนั้นมีค่ามากที่สุด

รายละเอียดการเขียนโปรแกรม

คุณจะต้องเขียนฟังก์ชันต่อไปนี้:

```
int64 find_maximum(int k, int[][] x)
```

- k: จำนวนรอบของการเล่นเกม
- x: อาร์เรย์ขนาด $n \times m$ ที่ระบุจำนวนบนตั๋วแต่ละใบ ค่าตั๋วของแต่ละสีจะถูกเรียงจากน้อยไปหามาก
- ฟังก์ชันนี้จะถูกเรียกหนึ่งครั้งเท่านั้น

- ฟังก์ชันนี้จะต้องเรียกฟังก์ชัน `allocate_tickets` (ดูรายละเอียดด้านล่าง) หนึ่งครั้ง เพื่อระบุเซตของตั๋วจำนวน k เซต สำหรับแต่ละรอบ การจัดสรรดังกล่าวจะต้องทำให้ผลรวมของรางวัลที่ได้รับนั้นมากที่สุด
- ฟังก์ชันจะต้องคืนค่าผลรวมของรางวัลที่มากที่สุด

ฟังก์ชัน `allocate_tickets` นิยามดังด้านล่าง:

```
void allocate_tickets(int[][] s)
```

- s: อาร์เรย์ขนาด $n \times m$ ค่าใน $s[i][j]$ จะต้องเป็น r ถ้าตั๋วที่ j ของสี i ถูกใช้ในรอบที่ r ในการเล่นเกม และเป็น -1 ถ้าตั๋วนั้นไม่ได้ถูกใช้
- สำหรับทุก ๆ ค่า i ที่ $0 \leq i \leq n-1$ ถ้าพิจารณาค่าในอาร์เรย์ $s[i][0], s[i][1], \ldots, s[i][m-1]$ จะต้องมีค่า $0, 1, 2, \ldots, k-1$ ปรากฏค่าละหนึ่งครั้งพอดี และช่องอื่น ๆ จะต้องมีค่า -1
- ถ้ามีวิธีการจัดสรรหลายแบบ สามารถรายงานการจัดสรรใด ๆ ก็ได้

ตัวอย่าง

ตัวอย่างที่ 1

พิจารณาการเรียกต่อไปนี้:

```
find_maximum(2, [[0, 2, 5],[1, 1, 3]])
```

หมายความว่า:

- มีการเล่นเกมทั้งสิ้น $k = 2$ รอบ
- จำนวนเต็มที่พิมพ์อยู่บนตั๋วที่มีสี 0 คือ 0, 2 และ 5 ตามลำดับ
- จำนวนเต็มที่พิมพ์อยู่บนตั๋วที่มีสี 1 คือ 1, 1 และ 3 ตามลำดับ

การจัดสรรรูปแบบหนึ่งที่ทำให้ได้ผลรวมของรางวัลมากที่สุดคือ

- ในรอบที่ 0, ริงโกเลือกตั๋วที่ 0 ของสี 0 (ที่มีจำนวนเต็ม 0 พิมพ์อยู่) และตั๋วที่ 2 ของสี 1 (ที่มีจำนวนเต็ม 3 พิมพ์อยู่) มูลค่าของรางวัลที่น้อยที่สุดที่เป็นไปได้ของรอบนี้คือ 3 ตัวอย่างเช่นพิธีกรสามารถเลือกให้ $b = 1$ ซึ่งทำให้ $|1 - 0| + |1 - 3| = 1 + 2 = 3$
- ในรอบที่ 1, ริงโกเลือกตั๋วที่ 2 ของสี 0 (ที่มีจำนวนเต็ม 5 พิมพ์อยู่) และตั๋วที่ 1 ของสี 1 (ที่มีจำนวนเต็ม 1 พิมพ์อยู่) มูลค่าของรางวัลที่น้อยที่สุดที่เป็นไปได้ของรอบนี้คือ 4 ตัวอย่างเช่นพิธีกรสามารถเลือกให้ $b = 3$ ซึ่งทำให้ $|3 - 1| + |3 - 5| = 2 + 2 = 4$
- ดังนั้นมูลค่ารางวัลรวมคือ $3 + 4 = 7$

ในการรายงานการจัดสรรนี้ ฟังก์ชัน `find_maximum` จะต้องเรียกฟังก์ชัน `allocate_tickets` ดังนี้:

- `allocate_tickets([[0, -1, 1], [-1, 1, 0]])`

สุดท้ายฟังก์ชัน `find_maximum` ควรคืนค่า 7

ตัวอย่างที่ 2

พิจารณาการเรียกต่อไปนี้:

```
find_maximum(1, [[5, 9], [1, 4], [3, 6], [2, 7]])
```

หมายความว่า:

- มีการเล่นหนึ่งรอบเท่านั้น
- จำนวนเต็มที่พิมพ์อยู่บนตั๋วที่มีสี 0 คือ 5 และ 9 ตามลำดับ
- จำนวนเต็มที่พิมพ์อยู่บนตั๋วที่มีสี 1 คือ 1 และ 4 ตามลำดับ
- จำนวนเต็มที่พิมพ์อยู่บนตั๋วที่มีสี 2 คือ 3 และ 6 ตามลำดับ
- จำนวนเต็มที่พิมพ์อยู่บนตั๋วที่มีสี 3 คือ 2 และ 7 ตามลำดับ

การจัดสรรรูปแบบหนึ่งที่ทำให้ได้ผลรวมของรางวัลมากที่สุดคือ

- ในรอบที่ 0, ริงโกเลือกตั๋วที่ 1 ของสี 0 (ที่มีจำนวนเต็ม 9 พิมพ์อยู่), ตั๋วที่ 0 ของสี 1 (ที่มีจำนวนเต็ม 1 พิมพ์อยู่), ตั๋วที่ 0 ของสี 2 (ที่มีจำนวนเต็ม 3), และตั๋วที่ 1 ของสี 3 (ที่มีจำนวนเต็ม 7 พิมพ์อยู่) มูลค่าของรางวัลที่น้อยที่สุดที่เป็นไปได้ของรอบนี้คือ 12 เมื่อพิธีกรเลือก $b = 3$ ทำให้ได้รางวัลมูลค่า $|3 - 9| + |3 - 1| + |3 - 3| + |3 - 7| = 6 + 2 + 0 + 4 = 12$

ในการรายงานการจัดสรรนี้ ฟังก์ชัน `find_maximum` จะต้องเรียกฟังก์ชัน `allocate_tickets` ดังนี้:

- `allocate_tickets([[-1, 0], [0, -1], [0, -1], [-1, 0]])`

สุดท้ายฟังก์ชัน `find_maximum` ควรคืนค่า 12

ข้อจำกัด

- $2 \le n \le 1500$ และ n เป็นจำนวนเต็มคู่
- $1 \le k \le m \le 1500$
- $0 \le x[i][j] \le 10^9$ (สำหรับทุก ๆ ค่า i และ j ที่ $0 \le i \le n-1$ และ $0 \le j \le m-1$)
- $x[i][j-1] \le x[i][j]$ (สำหรับทุก ๆ ค่า i และ j ที่ $0 \le i \le n-1$ และ $1 \le j \le m-1$)

ปัญหาย่อย

1. (11 คะแนน) $m = 1$
2. (16 คะแนน) $k = 1$
3. (14 คะแนน) $0 \le x[i][j] \le 1$ (สำหรับทุก ๆ ค่า i และ j ที่ $0 \le i \le n-1$ และ $0 \le j \le m-1$)
4. (14 คะแนน) $k = m$
5. (12 คะแนน) $n, m \le 80$
6. (23 คะแนน) $n, m \le 300$
7. (10 คะแนน) ไม่มีข้อจำกัดเพิ่มเติมอื่น ๆ

เกรดเดอร์ตัวอย่าง

เกรดเดอร์ตัวอย่างจะอ่านข้อมูลนำเข้าในรูปแบบต่อไปนี้:

- บรรทัดที่ 1: $n \;\; m \;\; k$
- บรรทัดที่ $2 + i$ ($0 \le i \le n - 1$): $x[i][0] \;\; x[i][1] \;\; \ldots \;\; x[i][m-1]$

เกรดเดอร์ตัวอย่างจะพิมพ์คำตอบของโปรแกรมของคุณในรูปแบบต่อไปนี้:

- บรรทัดที่ 1: ค่าที่คืนจาก `find_maximum`
- บรรทัดที่ $2 + i$ ($0 \le i \le n - 1$): $s[i][0] \;\; s[i][1] \;\; \ldots \;\; s[i][m-1]$