



## เป่ายิ้งฉุบ (rockpaperscissors)

มีคน  $3N + 1$  คน หมายเลขตั้งแต่ 0 ถึง  $3N$  แต่ละคนเวลาเป่ายิ้งฉุบเกมจะออกเหมือนเดิมเสมอ ยกเว้นคนที่โกง กฎเหมือนเป่ายิ้งฉุบทั่วไป คือชนะกรรไกร กรรไกรชนะกระดาษ และ กระดาษชนะค้อน ถ้าออกเหมือนกันจะนับว่าเสมอ

ใน  $3N + 1$  คน มี  $N$  คนออกกระดาษ  $N$  คนออกกรรไกร  $N$  คนออกค้อน เพียงแต่จะมีหนึ่งคนโกง (ชนะคนอื่นทุกคน) คุณผู้ชอบความยุติธรรมต้องการหาว่าคนหมายเลขใดเป็นคนโกง

ในแต่ละวัน คุณสามารถถามโดยเลือกคนมาจำนวนหนึ่ง (มากกว่า 2 คนขึ้นไป) มาจัดการแข่งขันเป่ายิ้งฉุบแบบพบกันหมด แล้วคุณจะทราบจำนวนครั้งที่ชนะของคนที่คุณถามมากที่สุด

ยกตัวอย่างเช่น

- ถ้าจัดการแข่งขันโดยที่ มีคนออกค้อน 1 คน กระดาษ 3 คน และกรรไกร 1 คน คนที่ชนะมากที่สุดคือคนที่ออกกรรไกรซึ่งชนะ 3 ครั้ง
- ถ้าจัดการแข่งขันโดยที่ มีคนออกค้อน 1 คน กระดาษ 3 คน และมีคนโกง 1 คน คนที่ชนะมากที่สุดคือคนที่โกงซึ่งชนะ 4 ครั้ง

สำหรับปัญหาย่อย ที่ 1 ถึง 3 เนื่องจากคุณต้องการรู้คนโกงให้เร็วที่สุด คุณต้องการจัดการแข่งขันให้น้อยครั้งที่สุดเท่าที่จะเป็นไปได้ คุณจะได้คะแนนก็ต่อเมื่อคุณจัดการแข่งขันไม่เกิน 10 ครั้ง

สำหรับปัญหาย่อย ที่ 4 เนื่องจากงบประมาณที่ต้องใช้ในแต่ละการแข่งขันไม่เท่ากัน ให้  $S$  แทนจำนวนผู้เข้าแข่งขันในแต่ละครั้ง งบประมาณที่คุณต้องใช้ในแต่ละครั้งคือ  $S^2$  เพื่อให้ประหยัดงบประมาณ คุณต้องการให้แบ่งงบประมาณรวมสำหรับการแข่งขันที่คุณจัดน้อยที่สุดเท่าที่จะเป็นไปได้ คะแนนของคุณจะขึ้นอยู่กับงบประมาณรวมที่คุณใช้ (ดูรายละเอียดในปัญหาย่อย)

รับประกันว่าคนแรกจะไม่ใช่คนโกงเสมอ

ในการรันโปรแกรมหนึ่งครั้งอาจมีการเรียกฟังก์ชัน `find_cheater1` หรือ `find_cheater2` ไม่เกิน 10 ครั้ง และในหนึ่งครั้งในการรันโปรแกรมจะไม่เรียก `find_cheater1` และ `find_cheater2` พร้อมกัน

หมายเหตุ: เกรดเดอร์สำหรับข้อนี้เป็นแบบปรับตัวได้

## รายละเอียดการเขียนโปรแกรม

คุณจะต้องเขียนฟังก์ชันดังต่อไปนี้:

```
int find_cheater1(int N)
```

- ฟังก์ชันนี้จะถูกเรียกเรียกอย่างมาก 10 ครั้ง สำหรับปัญหาย่อยที่ 1 ถึง 3 ให้คืนค่าหมายเลขของคนโกง

- การเรียกแต่ละครั้งให้พิจารณาเป็นปัญหาที่ไม่ขึ้นต่อกัน

```
int find_cheater2(int N)
```

- ฟังก์ชันนี้จะถูกเรียกอย่างมาก 10 ครั้ง สำหรับปัญหาย่อยที่ 4 ให้คืนค่าหมายเลขของคนที่โกง
- การเรียกแต่ละครั้งให้พิจารณาเป็นปัญหาที่ไม่ขึ้นต่อกัน

ฟังก์ชัน `find_cheater1` กับ `find_cheater2` สามารถเรียกฟังก์ชันดังต่อไปนี้ได้

```
int tournament(vector<int> v)
```

- `vector<int> v` : แทนหมายเลขของคนที่ต้องการเอามาแข่งกัน ภายในเวกเตอร์ห้ามมีจำนวนซ้ำกัน
- ฟังก์ชันจะคืนค่าจำนวนครั้งที่ชนะของคนที่ชนะมากที่สุด
- หากคุณใช้งบประมาณรวมเกิน 200 000 หรือเรียกฟังก์ชันนี้ 20 000 ครั้ง ฟังก์ชันจะให้ค่าคืนเป็น  $-1$
- หากใน `vector` มีเลขซ้ำกัน หรือมีเลขที่อยู่นอกเหนือจากเลข 0 ถึง  $3N$  ฟังก์ชันจะให้ค่าคืนเป็น  $-1$  และรายงานเป็น "Output isn't correct" ใน CMS

## ขอบเขต

- $1 \leq N \leq 333$

## ปัญหาย่อย

1. (7 คะแนน)  $1 \leq N \leq 3$
2. (17 คะแนน) รับประกันว่าคนแรกออกค้อน คนที่สองออกกรรไกร และคนที่สามออกกระดาก
3. (26 คะแนน) ไม่มีเงื่อนไขเพิ่มเติม
4. (50 คะแนน) ไม่มีเงื่อนไขเพิ่มเติม

ในปัญหาย่อยที่ 4 กำหนดให้  $M$  แทนงบประมาณรวมที่ต้องใช้ในการแข่งขัน ให้  $k$  คือค่ามากที่สุดของ  $M$  ทุกกรณี ทดสอบในปัญหาย่อยนี้ คะแนนของคุณจะเป็นตามตารางต่อไปนี้:

เงื่อนไข	คะแนน
$200\,000 < k$	0 (รายงานเป็น "Output isn't correct" ใน CMS)
$20\,000 < k \leq 200\,000$	$16 - \frac{k}{20000}$
$8475 < k \leq 20\,000$	$50 - \sqrt{\frac{k-8475}{9.408}}$
$k \leq 8475$	50

## ตัวอย่าง

สำหรับในปัญหาย่อยที่ 1 ถึง 3 พิจารณาตัวอย่างที่  $N=2$  โดยที่แต่ละคนออก

[rock, rock, paper, cheater, scissors, scissors, paper] ตามลำดับ

```
find_cheater1(2)
```

ฟังก์ชันดังกล่าวอาจเรียกฟังก์ชัน tournament ตามลำดับต่อไปนี้

ฟังก์ชัน	ค่าที่ได้
$\text{tournament}([0, 1, 2])$	2
$\text{tournament}([1, 2, 3, 4])$	3
$\text{tournament}([4, 5, 6])$	1
$\text{tournament}([0, 1, 3, 4, 5])$	4

สมมติว่าได้ข้อมูลเพียงพอต่อการหาหมายเลขของคนโกง ฟังก์ชัน find\_cheater1 ควรจะคืนค่า 3

## เกรตเตอร์ตัวอย่าง

กำหนดให้  $A$  เป็นอาร์เรย์ของจำนวนเต็ม  $N$  โดยที่  $A[i]$  แทนการออกข้อคนที่  $i$

- ถ้า  $A[i] = 0$  คนที่  $i$  เป็นคนโกง
- ถ้า  $A[i] = 1$  คนที่  $i$  ออกค้อน
- ถ้า  $A[i] = 2$  คนที่  $i$  ออกกระดาด
- ถ้า  $A[i] = 3$  คนที่  $i$  ออกกระบี่

กำหนดให้  $Q$  แทนจำนวนครั้งที่ถามคำถาม

กำหนดให้  $T$  แทนปัญหาย่อยที่เราต้องการทดสอบ

- $T = 0$  ที่ 1 ถึง 3
- $T = 1$  ที่ 4

เกรตเตอร์ตัวอย่างอ่านข้อมูลดังต่อไปนี้

- บรรทัดที่ 1:  $Q$

สำหรับแต่ละคำถามย่อยที่  $i$

- บรรทัดที่  $2i$ :  $N$   $T$
- บรรทัดที่  $2i + 1$ :  $A[0]A[1]...A[3N - 1]$

สำหรับ  $T = 0$  เกรตเตอร์ตัวอย่างจะส่งออกค่าที่คืนมาจากฟังก์ชัน find\_cheater1

สำหรับ  $T = 1$  เกรตเตอร์ตัวอย่างจะส่งออกค่าที่คืนมาจากฟังก์ชัน find\_cheater2

## ข้อจำกัด

- Time limit: 2 seconds
- Memory limit: 512 MB