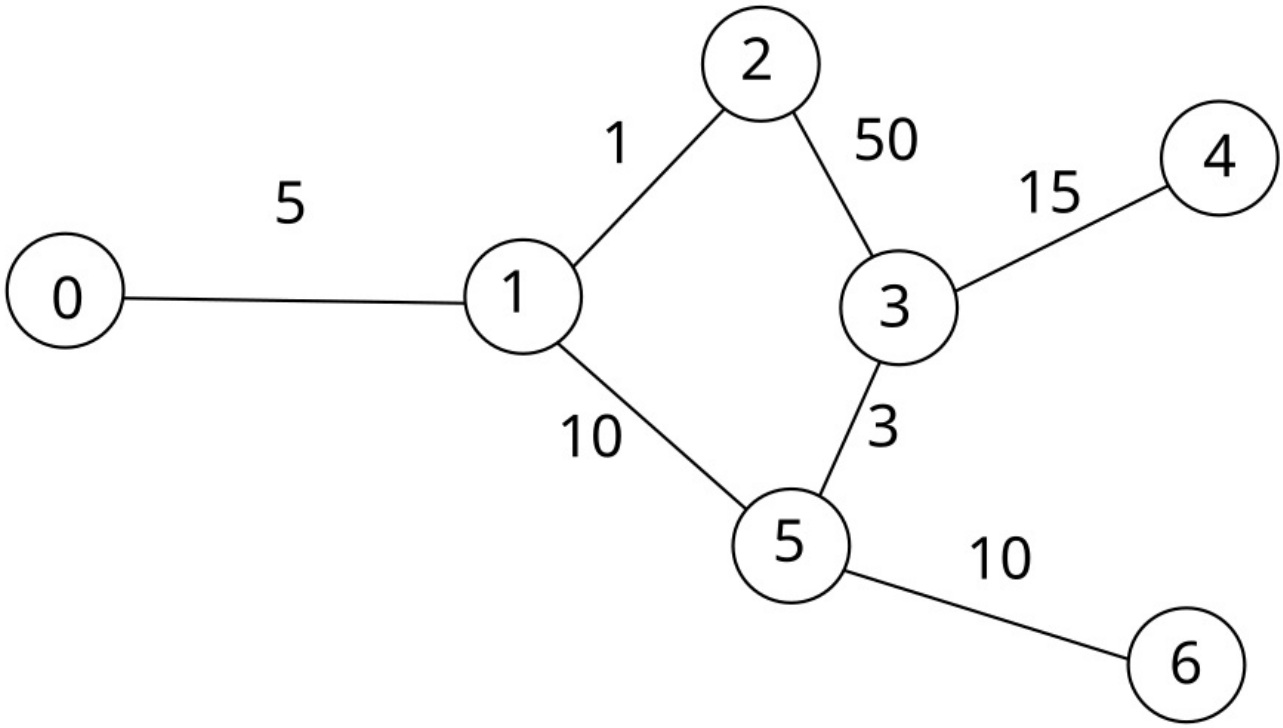
พบกันอีกครั้ง

หลังจากที่แพะได้พบ (ต่อย) กับเทพเจ้ามะนาวไปวันก่อน รอบนี้ ทั้งสองอยากพบกันเพื่อจิบกาแฟอีกครั้ง แต่ไม่ว่ายังไงก็จะต้องพบกันให้ได้ จึงเปลี่ยนเงื่อนไขการเดินจากเดิม

ในข้อนี้ ในแต่ละวินาที แพะ และ เทพเจ้ามะนาว สามารถ **เลือกที่จะเดินไปยังจุดยอดที่ติดกันได้** หรือ**เลือกที่จะหยุดอยู่กับที่ก็ได้** นอกจากนี้ หากทั้งสองคน **พบกันระหว่างกำลังเดินอยู่บนเส้นเชื่อม** จะเดินกลับไปยังจุดยอดที่ใกล้ที่สุดโดยอัตโนมัติและจิบกาแฟที่จุดยอดนั้น แต่ปัญหาคือ ด้วยการแทรกแซงทางธุรกิจจากผู้ยิ่งใหญ่ในตำนาน จึงทำให้แพะและเทพเจ้ามะนาว **ถูกส่งไปยังหนองคาย**

แน่นอนว่าแพะและเทพเจ้ามะนาว ไม่รู้จักพื้นที่หนองคาย แต่ในขณะที่กำลังถูกส่งตัวไปนั้น แพะกับเทพเจ้ามะนาวยังโชคดีที่มีสัญญาณอินเทอร์เน็ต พวกเขาจึงขอความช่วยเหลือจากคุณ (ซึ่งเป็นคนสนิทกับบุคคลปริศนาแห่งหนองคาย) โดยถามหากลยุทธ์ในการเดินมาพบกันโดยไม่เสียเวลาจนเกินไป

พิจารณาตัวอย่างด้านล่าง



สมมติว่าแพะอยู่ที่จุดยอด 0 และเทพเจ้ามะนาวอยู่ที่จุดยอด 4 สมมติว่าด้วยวิธีการที่ไม่มีใครทราบ ทั้งคู่สามารถออกแบบการเดินไปหากันได้ และถ้าแพะเดินตามเส้นทางดังนี้: $0 \to 1 \to 5 \to 3$ ส่วนเทพเจ้ามะนาวเดินตามเส้นทาง $4 \to 3$ จากนั้นรอที่จุดยอด 3 ทั้งคู่จะพบกันที่เวลา 18 (สังเกตว่าทั้งคู่ไม่สามารถนัดแนะกันก่อนได้ว่าจะเดินเส้นใด)

พิจารณาอีกกรณีหนึ่ง สมมติว่าแพะอยู่ที่จุดยอด 0 และเทพเจ้ามะนาวอยู่ที่จุดยอด 2 ถ้าแพะเดินตามเส้นทางดังนี้: $0 \to 1$ และเทพเจ้ามะนาวเดินตามเส้นทาง $2 \to 1 \to 0$ ทั้งคู่จะพบกันบนเส้นเชื่อมระหว่างจุดยอด 0 และ 1 แต่

ไม่สามารถจิบกาแฟได้ ถ้าทั้งคู่เดินไปจิบกาแฟที่จุดยอด 1 จะไปถึงที่เวลา 5 (ตามเวลาเดินของแพะ) ถ้าไปจิบที่จุดยอด 0 จะไปถึงที่เวลา 6 ตามเวลาของเทพเจ้ามะนาว ดังนั้นทั้งคู่จะเลือกไปจิบกาแฟที่จุดยอด 1 ทำให้จิบกาแฟได้ที่เวลา 5

ในการคิดคะแนน จากตัวอย่างข้างต้น สังเกตว่าถ้าทั้งสองคนเริ่มต้นที่จุดยอด 0 และ 4 ไม่มีวิธีการเดินใดที่จะพาให้ทั้งคู่มาเจอกันได้เร็วกว่าเวลา 18 ทำให้วิธีการเดินแบบใดก็ตามที่ให้คำตอบไม่แย่ไปกว่า 18 ไม่ว่าทั้งสองคนจะเริ่มจากจุดเริ่มต้นใด ๆ จะได้คะแนนเต็ม ถ้าวิธีการใช้เวลามากกว่านั้น จะได้คะแนนลดหลั่นกันไป (ดูได้ในส่วนอธิบายการให้คะแนน)

ในโจทย์ข้อนี้ ในการตรวจ โค้ดส่วนของแพะและเทพเจ้ามะนาวจะถูกเรียกให้ทำงานแยกกัน เพื่อรับประกันว่าอัลกอริทึมที่ออกแบบทั้งสองส่วนจะไม่สามารถสื่อสารกันได้

รายละเอียดการเขียนโปรแกรม

คุณจะต้องเขียนฟังก์ชันทั้งสิ้น 4 ฟังก์ชัน ส่วนของแพะและมะนาว ใน `goat.cpp` และ `lemon.cpp` ตามลำดับ ซึ่งในแต่ละส่วนจะถูกเรียกใช้ทำงานแยกกันที่เกรดเดอร์ในเครื่องตรวจ คุณสามารถประกาศตัวแปรประเภท global ได้ แต่ขอให้ทำภายใน namespace ของแต่ละส่วน (`goat` หรือ `lemon`) ตามลำดับ เพื่อป้องกันปัญหาการคอมไพล์หากมีของชื่อซ้ำกัน

ในการตรวจจริง **ทั้งสองไฟล์ไม่สามารถใช้ตัวแปรร่วมกันได้** (และตัวตรวจจะทำการเรียกแยกเป็นสอง process) **สำหรับการทำงานในเกรดเดอร์ตัวอย่างจะแตกต่างไป จะอธิบายแยกในส่วนเกรดเดอร์ตัวอย่าง---กรุณาอ่านรายละเอียดด้วย**

ส่วนของแพะใน `goat.cpp`

คุณจะต้องเขียนฟังก์ชันดังต่อไปนี้

```
void goat_init(int N, int M, vector<int> U, vector<int> V, vector<int
> W, int u)
```

- ฟังก์ชันนี้จะถูกเรียกเพียงครั้งเดียว ก่อนมีการเรียกอย่างอื่น โดยจะระบุกราฟอธิบายพื้นที่หนองคาย
- คุณสามารถใช้ตัวแปรประเภท global หรืออื่นๆ ในการเก็บข้อมูลเอาไว้เพื่อใช้คราวหลังได้ โดยจะต้องเก็บภายใน namespace `goat`
- ตัวแปร `N` และ `M` แทนค่า N และ M
- ตัวแปร `U`, `V` และ `W` เป็นเวกเตอร์ขนาด M โดยแทนเส้นเชื่อมแต่ละเส้น กล่าวคือ ในกราฟจะมีเส้นเชื่อมระหว่างจุดยอด `U[i]` กับ `V[i]` ด้วยน้ำหนัก `W[i]` สำหรับ `i` ตั้งแต่ 0 ถึง $M-1$
- ตัวแปร `u` แทนตำแหน่งของแพะในเวลาเริ่มต้น

```
int goat_move(int u)
```

- ฟังก์ชันนี้จะถูกเรียกในแต่ละครั้งที่แพะจะทำการเดิน โดยจะถูกเรียกไปเรื่อย ๆ จนกว่าแพะจะพบกับเทพเจ้ามะนาว โดยคุณสามารถใช้ตัวแปรประเภท global ที่ได้เก็บไว้ได้ภายใน namespace `goat`

- ตัวแปร u แทนตำแหน่งของแพะในปัจจุบัน (ซึ่งถ้าไม่มีความผิดพลาดเกิดขึ้นจะเท่ากับจุดยอดที่คืนไปในครั้งก่อน)
- ฟังก์ชันนี้จะต้องคืนค่าจุดยอดถัดไปที่จะเดินไป โดยจุดยอดถัดไปนี้จะต้องอยู่ติดกับจุดยอดปัจจุบัน หรือไม่ก็ต้องเป็นจุดยอดปัจจุบัน

ส่วนของมะนาวใน `lemon.cpp`

คุณจะต้องเขียนฟังก์ชันดังต่อไปนี้

```
void lemon_init(int N, int M, vector<int> U, vector<int> V, vector<in
t> W, int u)
```

- ฟังก์ชันนี้จะถูกเรียกเพียงครั้งเดียว ก่อนมีการเรียกอย่างอื่น โดยจะระบุกราฟอธิบายพื้นที่หนองคาย
- คุณสามารถใช้ตัวแปรประเภท global หรืออื่นๆ ในการเก็บข้อมูลเอาไว้เพื่อใช้คราวหลังได้ โดยจะต้องเก็บภายใน namespace `lemon`
- ตัวแปร `N` และ `M` แทนค่า N และ M
- ตัวแปร `U`, `V` และ `W` เป็นเวกเตอร์ขนาด M โดยแทนเส้นเชื่อมแต่ละเส้น กล่าวคือ ในกราฟจะมีเส้นเชื่อมระหว่างจุดยอด `U[i]` กับ `V[i]` ด้วยน้ำหนัก `W[i]` สำหรับ `i` ตั้งแต่ 0 ถึง $M-1$
- ตัวแปร u แทนตำแหน่งของเทพเจ้ามะนาวในเวลาเริ่มต้น

```
int lemon_move(int u)
```

- ฟังก์ชันนี้จะถูกเรียกในแต่ละครั้งที่เทพเจ้ามะนาวจะทำการเดิน โดยจะถูกเรียกไปเรื่อย ๆ จนกว่าจะเทพเจ้ามะนาวพบกับแพะ โดยคุณสามารถใช้ตัวแปรประเภท global ที่ได้เก็บไว้ได้ภายใน namespace `lemon`
- ตัวแปร u แทนตำแหน่งของเทพเจ้ามะนาวในปัจจุบัน (ซึ่งถ้าไม่มีความผิดพลาดเกิดขึ้นจะเท่ากับจุดยอดที่คืนไปในครั้งก่อน)
- ฟังก์ชันนี้จะต้องคืนค่าจุดยอดถัดไปที่จะเดินไป โดยจุดยอดถัดไปนี้จะต้องอยู่ติดกับจุดยอดปัจจุบัน หรือไม่ก็ต้องเป็นจุดยอดปัจจุบัน

วิธีการเรียกที่เกรดเดอร์ที่ CMS จะเรียกใช้งาน

โดยเริ่มต้น เกรดเดอร์จะทำการเรียก `goat_init` ที่ process ของแพะและ `lemon_init` ที่ process ของมะนาวต่อมาเราจะนับเวลา $t_{\text{goat}} = t_{\text{lemon}} = 0$

ถัดมา เราจะทำการกระทำต่อไปนี้ซ้ำไปเรื่อย ๆ จนกว่าจะเข้าเงื่อนไข **[แพะพบกับเทพเจ้ามะนาว]** คือ เราจะเรียกบุคคลที่มี t ต่ำกว่าอีกคน (ถ้าเท่ากันให้ถือว่าแพะเริ่มก่อน) แล้วเรียกฟังก์ชัน `move` ของบุคคลนั้น หากจุดเดิมคือ u และ `move` คืนค่า u' แล้ว ให้ $w_{u,u'}$ แทนน้ำหนักของเส้นเชื่อมระหว่าง u กับ u' แล้วเราจะเพิ่มค่า t ของคนนั้นไป $w_{u,u'}$ พร้อมกับย้ายตำแหน่งบุคคลนั้น จาก u ไปยัง u'

ในกรณีที่ค่าที่คืนจาก `move` คือจุดยอดเดิม (u) เพื่อประสิทธิภาพในการคำนวณเราจะถือว่าบุคคลนั้นจะอยู่ที่จุดยอด u ไปจนถึงเวลาที่อีกคนเดินถึงเป้าหมาย หรือเวลาเพิ่มขึ้น 1 วินาทีในกรณีที่อีกคนยังไม่ได้เดิน

แล้ววนกลับไปทำซ้ำ คือเลือกผู้เล่นปัจจุบัน p แล้วเรียกฟังก์ชัน p_move

ในการกระทำข้างต้น เราจะกล่าวว่า สอดคล้องกับเงื่อนไข **[แพะพบกับเทพเจ้ามะนาว]** เมื่ออย่างใดอย่างหนึ่งต่อไปนี้เป็นจริง

- ในขณะที่บุคคลปัจจุบัน p กำลังจะเริ่มเคลื่อนจาก u ไป u' ซึ่งจะออกเดินทางในวินาทีที่ t_p และถึงในวินาทีที่ $t_p + w_{u,u'}$ นั้น บุคคลอีกคนหนึ่งอยู่ที่ตำแหน่ง u' (และกำลังรออยู่)
- ในขณะที่บุคคลปัจจุบัน p กำลังจะเริ่มเคลื่อนจาก u ไป u' ซึ่งจะไปถึงในวินาทีที่ $t_p + w_{u,u'}$ นั้น บุคคลอีกคนกำลังเคลื่อนจาก u' มายัง u ซึ่งจะไปถึงวินาทีที่ $t_q + w_{u',u}$ อยู่เช่นกัน

สำหรับกรณีแรก จะกล่าวว่า $t_p + w_{u,u'}$ คือ **เวลาที่แพะพบกับเทพเจ้ามะนาว** และในกรณีที่สอง ทั้งสองคนจะพบกันกลางทางและจะเดินไปยังจุดหมายที่ใกล้ที่สุดของหนึ่งในสองคนนั้น คือจะถือว่า**เวลาที่แพะพบกับเทพเจ้ามะนาว** คือเวลา $\min\{t_p + w_{u,u'}, t_q + w_{u',u}\}$

หากระหว่างการทำงานของเกรดเดอร์ พบว่าเข้าสู่เงื่อนไข **[แพะพบกับเทพเจ้ามะนาว]** แล้วจะคิดคะแนนด้วย **เวลาที่แพะพบกับเทพเจ้ามะนาว**

ขอบเขต

- $2 \le N \le 2\,000$
- $N - 1 \le M \le 5\,000$
- กราฟของหนองคายเป็นกราฟไม่ระบุทิศทาง ระบุน้ำหนัก และเป็นกราฟที่เชื่อมต่อกัน
- U[i] และ V[i] เป็นจำนวนเต็มระหว่าง 0 ถึง $N - 1$ และ W[i] เป็นจำนวนเต็มระหว่าง 1 ถึง $1\,000\,000$ สำหรับทุก i ตั้งแต่ 0 ถึง $M - 1$
- ไม่มีเส้นเชื่อมสองเส้นที่เชื่อมระหว่างคู่จุดยอดเดียวกัน กล่าวคือ ไม่มีคู่ i $\neq$ j ที่ U[i] $=$ U[j] และ V[i] $=$ V[j] และ ไม่มีคู่ i $\neq$ j ที่ U[i] $=$ V[j] และ U[j] $=$ V[i]
- ไม่มีเส้นเชื่อมที่เชื่อมจุดยอดไปยังตัวเอง กล่าวคือ U[i] $\neq$ V[i] สำหรับทุก i ตั้งแต่ 0 ถึง $M - 1$

ปัญหาย่อย

สำหรับทุกปัญหาย่อยในข้อนี้ จะมีการให้คะแนนแบบบางส่วน โดยพิจารณาในส่วนถัดจากนี้

1. (15 คะแนน) รับประกันว่า $M = N - 1$, U[i]$=$i, V[i]$=$i+1 และ W[i]$= 1$ สำหรับทุก i ตั้งแต่ 0 ถึง $N - 2$
2. (15 คะแนน) $M = N - 1$
3. (20 คะแนน) $N \le 200$
4. (15 คะแนน) W[i]$= 1$ สำหรับทุก i ตั้งแต่ 0 ถึง $M - 1$
5. (35 คะแนน) ไม่มีเงื่อนไขเพิ่มเติม

การให้คะแนน

เรารับประกันว่ามีวิธีที่จะทำให้แพะพบกับเทพเจ้ามะนาวได้ในเวลา $1\,000\,000\,000\,000$ วินาที ถ้าคุณใช้เวลามากกว่านั้นจะถือว่าใช้เวลามากเกินไป (นั่นคือใช้เวลานานกว่า $30\,000$ ปี ซึ่งเกินอายุขัยของแพะ) และจะไม่ได้คะแนน

เราจะพิจารณากลยุทธ์ทั้งหมดที่เป็นไปได้ที่จะทำให้แพะได้เจอกับเทพเจ้ามะนาว ให้ $\mathcal{A}$ แทนกลยุทธ์ (ที่ดีที่สุด) ของกรรมการที่กรรมการสามารถหาได้ เราจะพิจารณากรณีตำแหน่งเริ่มต้นของแพะและเทพเจ้ามะนาวทั้งหมดทั้ง N^2 แบบ สมมติแพะเริ่มที่จุดยอด u และเทพเจ้ามะนาวเริ่มที่จุดยอด v จะได้ว่า **ค่าเสียเวลา** ของกลยุทธ์ $\mathcal{A}$ บนจุดเริ่มต้น (u, v) มีค่าเท่ากับ **เวลาที่แพะพบกับเทพเจ้ามะนาว** จากกลยุทธ์ $\mathcal{A}$ เขียนแทนด้วย $T_{\mathcal{A}}(u, v)$

นิยาม **ค่าเสียเวลา** ของกลยุทธ์ $\mathcal{A}$ เขียนแทนด้วย $T_{\mathcal{A}}$ นิยามโดย $T_{\mathcal{A}} := \max_{u,v \in V} T_{\mathcal{A}}(u, v)$

ให้ $\mathcal{B}$ แทนกลยุทธ์ของคุณ นิยาม **ค่าเสียเวลา** ของกลยุทธ์ของคุณ $T_{\mathcal{B}} := \max_{(u,v) \in S} T_{\mathcal{B}}(u, v)$ โดย S แทนเซตของจุดเริ่มต้นในข้อมูลทดสอบข้อมูลนั้น (อาจไม่ครบทั้งหมดทั้ง N^2 แบบ)

ต่อมา ให้ $r = \frac{T_{\mathcal{A}}}{T_{\mathcal{B}}}$ แล้วคะแนนที่คุณจะได้รับ จะเป็นไปตามตารางดังนี้

เงื่อนไข	อัตราส่วนคะแนน
$r \geq 1$	1.0
$0.75 \leq r < 1$	$2r - 1$
$0.5 \leq r < 0.75$	0.3
$0 \leq r < 0.5$	0.15

ตัวอย่าง

สมมติว่าแผนที่เป็นดังรูปด้านบน และแพะเริ่มที่จุดยอด 0 และเทพเจ้ามะนาวอยู่ที่จุดยอด 4

เกรดเดอร์จะเรียกฟังก์ชัน `goat_init` และ `lemon_init` (ที่ทำงานอยู่คนละ process กันใน cms) ดังนี้

```
goat_init(7, 7, [0,1,5,5,3,2,3],
                [1,2,1,6,5,3,4],
                [5,1,10,10,3,50,15], 0)
```

และ

```
lemon_init(7, 7, [0,1,5,5,3,2,3],
                 [1,2,1,6,5,3,4],
                 [5,1,10,10,3,50,15], 4)
```

ถ้าแผนการเดินของแพะคือไปตามเส้นทาง $0 \to 1 \to 5 \to 3$ ส่วนเทพเจ้ามะนาวไปตามเส้นทาง $4 \to 3$ ผลการทำงานจะเป็นดังนี้

เกรดเดอร์จะเรียก

```
goat_move(0)
```

ซึ่งจะคืนค่า 1 สังเกตว่าแพะจะเดินทางถึงจุดยอด 1 ที่เวลา 5 ต่อไปเกรดเดอร์จะเรียก

```
lemon_move(4)
```

ซึ่งจะคืนค่า 3 เทพเจ้ามะนาวจะเดินทางถึงจุดยอด 3 ที่เวลา 15 จากนั้นเกรดเดอร์จะเรียก

```
goat_move(1)
```

ซึ่งจะเป็นการถามจุดยอดต่อไปหลังจากที่ถึงจุดยอด 1 แล้วที่เวลา 5 ฟังก์ชันจะคืนค่า 5 จากนั้นสังเกตว่าทั้งแพะและเทพเจ้ามะนาวพร้อมจะเดินทางต่อที่เวลา 15 เท่ากัน เกรดเดอร์จะเรียก

```
goat_move(5)
```

ซึ่งจะคืนค่า 3 และเรียก

```
lemon_move(3)
```

ซึ่งจะคืนค่า 3 แต่ในกรณีนี้ สังเกตว่าแพะจะเดินถึงเป้าหมายที่เวลา 18 เราจะสมมติว่าเทพเจ้ามะนาวจะรออยู่ที่จุดยอด 3 ถึงเวลา 18 ด้วยเช่นกัน ทำให้เจอกันที่จุดยอดดังกล่าวในเวลา 18

เนื่องจาก 18 เป็นเวลาที่ดีที่สุดแล้วถ้าแพะและเทพเจ้ามะนาวเริ่มจากที่ใดก็ได้ วิธีการเดินแบบนี้จะได้คะแนนเต็ม

พิจารณาอีกกรณีหนึ่งที่แพะอยู่ที่จุดยอด 0 และเทพเจ้ามะนาวอยู่ที่จุด 2 เกรดเดอร์จะเรียกฟังก์ชัน `goat_init` และ `lemon_init` (ที่ทำงานอยู่คนละ process กันใน cms) ดังนี้

```
goat_init(7, 7, [0,1,5,5,3,2,3],
                [1,2,1,6,5,3,4],
                [5,1,10,10,3,50,15], 0)
```

และ

```
lemon_init(7, 7, [0,1,5,5,3,2,3],
                 [1,2,1,6,5,3,4],
                 [5,1,10,10,3,50,15], 2)
```

สมมติว่าทั้งสองคนวางแผนตามที่ระบุในคำอธิบายโจทย์ข้างต้น

หลังจากเรียก init เกรดเดอร์จะเรียก

```
goat_move(0)
```

ซึ่งจะคืนค่า 1 จากนั้นเกรดเดอร์จะเรียก

```
lemon_move(2)
```

ซึ่งคืนค่า 1 เทพเจ้ามะนาวจะถึงจุดยอด 1 ในเวลา 1 และเกรดเดอร์จะเรียก

```
lemon_move(1)
```

อีกครั้ง ซึ่งคืนค่า 0 ทำให้ทั้งสองคนพบกันบนเส้นเชื่อม และตัดสินใจมาจิบกาแฟที่จุดยอด 1 ที่เวลา 5

สังเกตว่าเวลาที่ดีที่สุดที่ทำได้ในกรณีที่ทั้งสองคนเริ่มเดินจากจุดที่แตกต่างกันใด ๆ คือ 18 และคุณสามารถทำเวลาได้ไม่แย่กว่า 18 ทำให้คำตอบนี้ได้คะแนนเต็ม

เกรดเดอร์ตัวอย่าง (กรุณาอ่านรายละเอียดการทำงาน)

เกรดเดอร์ตัวอย่างจะอ่านข้อมูลนำเข้าในรูปแบบต่อไปนี้:

- Line 1: N M
- Line $2+i$: $U[i]$ $V[i]$ $W[i]$
- Line $2+M$: s_1 s_2

โดยที่ s_1 และ s_2 จะเป็นจุดยอดเริ่มต้นของแพะและมะนาวตามลำดับ

เกรดเดอร์ตัวอย่างจะพิมพ์เวลาที่ทั้งสองคนเจอกัน

ในส่วนของเกรดเดอร์ตัวอย่างนั้น จะทำงานเป็น process เดียว โดยเกรดเดอร์จะเรียก `goat_init`, `lemon_init` แล้วเรียก `goat_move` กับ `lemon_move` ตามเงื่อนไขที่ระบุในโจทย์

อย่างไรก็ตาม เนื่องจากการทำงานเป็นแบบ process เดียว ตัวแปร global ในส่วน `goat` และ `lemon` สามารถปะปนกันได้ ซึ่งถ้าเขียนอย่างระมัดระวังไม่ไปเรียกใช้ตัวแปรข้ามส่วนจะไม่ส่งผลให้การทำงานในเครื่องผู้แข่งขันแตกต่างจากในเครื่องจริง (ยกเว้นเวลาที่บนเกรดเดอร์บน CMS อาจจะทำงานนานกว่า)

จึงเป็นเหตุผลว่า สำหรับการส่งไปในระบบ CMS เราจะขอให้ผู้เข้าแข่งขัน **เก็บตัวแปรประเภท global รวมถึงการประกาศฟังก์ชันแยก เอาไว้ใน namespace ของแต่ละส่วนแยกกัน** เพื่อป้องกันไม่ให้ตัวแปรตีกัน โดยหากตัวแปรเกิดตีกันขึ้นมาแล้วมีปัญหาในการคอมไพล์บน CMS จะถือว่าเป็นความรับผิดชอบของผู้เข้าแข่งขันที่จะต้องแก้ไขเอาเอง

การป้องกันการเข้าถึงนี้ มีไว้ช่วยให้การรันเกรดเดอร์ในเครื่องมีพฤติกรรมเหมือนกับที่ CMS เท่านั้น (ป้องกันความผิดพลาดของคุณ) แต่ในการตรวจจริงโค้ดสองส่วนจะทำงานแยกกัน ซึ่งจะรับประกันว่าจะไม่สามารถสื่อสารกันได้อยู่แล้ว

ข้อจำกัด

- Time limit: 5 seconds
- Memory limit: 512 MB