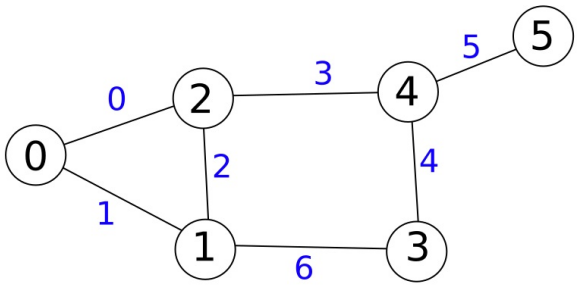
บ้านเธอบ้านฉัน (twohomes)

ที่เมืองแห่งหนึ่งมีบ้าน N หลัง เรียกเป็นบ้าน 0 ถึงบ้าน $N-1$ บ้านเหล่านี้เชื่อมกันด้วยถนนสองทิศทาง M เส้น ถนนเส้นที่ i สำหรับ $0 \leq i < M$ เชื่อมระหว่างบ้านหลังที่ $R[i][0]$ กับ $R[i][1]$ รับประกันว่าระหว่างบ้านสองหลังใด ๆ สามารถไปหากันได้ผ่านทางถนนเหล่านี้ และไม่มีบ้านหลังใดเชื่อมกับถนนมากกว่า 5 เส้น

Alice แม่ครัวชื่อดังอยู่บ้านที่ s ส่วน Bob อยู่บ้านที่ t โดยที่ $s \neq t$ เราไม่ทราบ s และ t แต่เราทราบว่าทุกวัน Bob จะเดินทางจากบ้านไปซื้อข้าวกลางวันที่บ้านของ Alice (ถ้าสามารถทำได้) เราสามารถเลือกเซตของถนน C และสั่งปิดทุกถนนในเซตนั้นแล้วสังเกตว่า Bob ได้ออกไปซื้ออาหารกลางวันหรือไม่เพื่อตรวจสอบว่ามีเส้นทางในเมืองจากบ้าน t ไป s ที่ไม่ผ่านถนนเหล่านั้นหรือไม่ได้

คุณสามารถถามได้ไม่เกิน Q ครั้ง

พิจารณาตัวอย่างต่อไปนี้ที่ $N = 6$ และ $M = 7$ หมายเลขถนนแสดงเป็นเลขสีน้ำเงิน



สมมติว่า $s = 1$ และ $t = 5$ ตารางด้านล่างแสดงว่าถ้าสั่งปิดเซตของถนนใด แล้ว Bob จะเดินทางไปหา Alice เพื่อซื้ออาหารกลางวันได้หรือไม่

สับเซตของถนน	Bob สามารถไปบ้าน Alice ได้หรือไม่
$\{0, 2, 3\}$	`yes`
$\{0, 1\}$	`yes`
$\{2, 3, 4, 6\}$	`no`
$\{5\}$	`no`
$\{0, 1, 2, 3\}$	`yes`

เมื่อคุณสั่งปิดถนนจนสามารถทราบได้ว่าบ้านของ Alice และ Bob เป็นบ้านใดแล้ว คุณจะต้องตอบว่า s และ t คืออะไร (สามารถตอบสลับกันได้)

รายละเอียดการเขียนโปรแกรม

คุณจะต้องเขียนฟังก์ชันต่อไปนี้

```
pair<int,int> find_homes(int N, int M, vector<vector<int>> R)
```

- ฟังก์ชันนี้จะถูกเรียกหนึ่งครั้ง
- ฟังก์ชันจะต้องคืนค่า `pair` h ที่ $\{h.first, h.second\} = \{s, t\}$
- พารามิเตอร์ R ระบุข้อมูลของถนนในเมืองจำนวน M เส้น กล่าวคือ สำหรับ $0 \leq i < M$ ถนนเส้นที่ i เชื่อมระหว่างบ้าน $R[i][0]$ กับบ้าน $R[i][1]$

ฟังก์ชัน `find_homes` จะสามารถเรียกฟังก์ชันด้านล่างเพื่อตรวจสอบได้ไม่เกิน Q ครั้ง

```
bool is_reachable(vector<int> C)
```

- ฟังก์ชันนี้จะถูกเรียกจาก `find_homes` ได้ไม่เกิน Q ครั้ง
- จะต้องส่งรายการของถนนที่ต้องการปิดในอาร์เรย์ C
- ฟังก์ชันจะคืนค่า `true` ก็ต่อเมื่อ Bob สามารถเดินทางไปหา Alice ได้ โดยไม่ใช้ถนนใด ๆ ในรายการ C เลย

เงื่อนไข

- $2 \leq N \leq 1\,000$
- $2 \leq M \leq 2\,500$
- $0 \leq R[i][0] \leq N-1, 0 \leq R[i][1] \leq N-1$
- $Q \in \{70, 1\,000\}$

Subtasks

1. (7 points) $Q = 1\,000, N \leq 300, M \leq 750$
2. (10 points) $Q = 70, M = N-1$ และถนนเชื่อมต่อเป็นเส้น นั่นคือมีบ้านสองหลังที่เชื่อมกับถนนหนึ่งเส้น ส่วนบ้านที่เหลือล้วนแต่เชื่อมกับถนนสองเส้น
3. (8 points) $Q = 70, M = N-1$ และ $N = 2^k - 1$ สำหรับบางจำนวนเต็ม k และถนนเชื่อมต่อเป็น complete binary tree
4. (5 points) $Q = 70, M = N-1$ และ $N = 4^k - 1$ สำหรับบางจำนวนเต็ม k และถนนเชื่อมต่อเป็น complete 4-nary tree
5. (25 points) $Q = 70, M = N-1, s = 0$, และ s ติดกับถนนเส้นเดียว
6. (30 points) $Q = 70, M = N-1$
7. (15 points) $Q = 70$

Examples

จากตัวอย่างข้างต้น เกรดเดอร์จะเรียกฟังก์ชัน `find_homes` ดังนี้

```
find_homes(6,7,
           [[0,2], [0,1], [2,1], [4,2], [3,4], [4,5], [1,3]])
```

ฟังก์ชันดังกล่าวจะสามารถเรียกฟังก์ชัน `is_reachable` ตามตัวอย่างด้านล่าง

```
is_reachable([0,2,3])
```

จะคืนค่า `true` และถ้าเรียก

```
is_reachable([0,1])
```

จะคืนค่า `true` และถ้าเรียก

```
is_reachable([2,3,4,6])
```

จะคืนค่า `false` และถ้าเรียก

```
is_reachable([5])
```

จะคืนค่า `false` และถ้าเรียก

```
is_reachable([0,1,2,3])
```

จะคืนค่า `true`

ฟังก์ชัน `find_homes` ที่ทำงานถูกต้องจะต้องคืน `pair` ที่ประกอบไปด้วย 1 และ 5 (จะเป็น `[1,5]` หรือ `[5,1]` ก็ได้)

เกรดเดอร์ตัวอย่าง

เกรดเดอร์ตัวอย่างจะอ่านข้อมูลนำเข้าดังนี้

- บรรทัดที่ 1: N M Q s t
- บรรทัดที่ $2+i$ ถึง $2+M-1$: $R[i][0]$ $R[i][1]$

ถ้าถามเกิน Q ครั้ง เกรดเดอร์ตัวอย่างจะพิมพ์ข้อความแสดงความผิดพลาด ถ้าไม่เช่นนั้น เกรดเดอร์จะตรวจค่าที่คืนมาจากฟังก์ชัน `find_homes` และพิมพ์ว่าคำตอบถูกหรือไม่

Limits

- Time limit: 1 seconds
- Memory limit: 512 MB