

## สถานี (stations)

เครือข่ายแกนกลางของสิงคโปร์ (Singapore's Internet Backbone: SIB) ประกอบด้วยสถานีจำนวน  $n$  สถานี กำกับด้วยดัชนีตั้งแต่ 0 ถึง  $n - 1$  และมีการเชื่อมต่อแบบสองทิศทางจำนวน  $n - 1$  การเชื่อมต่อ กำกับด้วยหมายเลขตั้งแต่ 0 ถึง  $n - 2$  การเชื่อมต่อแต่ละอันนั้นจะเชื่อมต่อสองสถานีที่แตกต่างกัน สถานีสองสถานีที่ติดต่อกันด้วยการเชื่อมต่อเพียงอันเดียวเชื่อมจะถูกเรียกว่าเพื่อนบ้าน

เส้นทางจากสถานี  $x$  ไปยังสถานี  $y$  เป็นลำดับของสถานี  $a_0, a_1, \dots, a_p$  (ซึ่งไม่ซ้ำกัน) และมี  $a_0 = x$ ,  $a_p = y$  และ ทุก ๆ สองสถานีที่อยู่ในลำดับต่อกันในเส้นทางต้องเป็นเพื่อนบ้านกัน จะมีเส้นทางหนึ่งเส้นทางพอดีจากสถานี  $x$  ใด ๆ ไปยังสถานี  $y$  ใด ๆ

สถานี  $x$  ใด ๆ สามารถสร้างแพ็คเกจ (ก้อนข้อมูล) และส่งแพ็คเกจไปยังสถานี  $y$  ใด ๆ ก็ได้ โดยสถานีนั้นจะถูกเรียกว่า เป้าหมายของแพ็คเกจ มีวิธีการส่งต่อแพ็คเกจนี้เป็นทอด ๆ จาก  $x$  ไปยัง  $y$  ตามเส้นทางได้อยู่แบบเดียว ซึ่งเป็นดังนี้ ให้พิจารณาสถานี  $z$  ซึ่งกำลังถือแพ็คเกจนี้ ที่มีเป้าหมายเป็นสถานี  $y$  ( $z \neq y$ ) สถานี  $z$  จะต้อง

1. รัน ฟังก์ชันส่งต่อ ซึ่งกำหนดเพื่อนบ้านของ  $z$  ที่อยู่ในเส้นทางเดียวที่เป็นไปได้จาก  $z$  ไปยัง  $y$  และ
2. ส่งแพ็คเกจนี้ไปยังเพื่อนบ้านดังกล่าว

อย่างไรก็ตามสถานีต่าง ๆ นั้นมีหน่วยความจำจำกัดและไม่สามารถเก็บรายการของเส้นเชื่อมทั้งหมดใน SIB เพื่อใช้ในฟังก์ชันส่งต่อได้

งานของคุณคือเขียนรูปแบบการส่งต่อสำหรับ SIB ซึ่งประกอบด้วยสองฟังก์ชันต่อไปนี้

- ฟังก์ชันแรก รับข้อมูลนำเข้าได้แก่ ค่า  $n$ , รายการของเส้นเชื่อมใน SIB และ จำนวนเต็ม  $k \geq n - 1$  แล้วทำการกำหนดป้ายชื่อเป็นเลขจำนวนเต็มที่ไม่ซ้ำกันและมีค่าตั้งแต่ 0 ถึง  $k$  รวมหัวท้าย
- ฟังก์ชันที่สอง เป็นฟังก์ชันส่งต่อ ซึ่งจะถูกติดตั้งไว้ในทุก ๆ สถานีหลังจากได้รับการกำหนดป้ายชื่อ ฟังก์ชันนี้มีข้อมูลนำเข้าต่อไปนี้เท่านั้น
  - $s$ , ป้ายชื่อ ของสถานีที่ถือแพ็คเกจอยู่
  - $t$ , ป้ายชื่อ ของสถานีเป้าหมาย ( $t \neq s$ )
  - $c$ , รายการของป้ายชื่อของเพื่อนบ้านของ  $s$

ฟังก์ชันนี้ต้องคืนค่าป้ายชื่อของเพื่อนบ้านของ  $s$  ที่แพ็คเกจนี้จะต้องถูกส่งไป

ในปัญหาย่อยหนึ่ง คะแนนของคุณจะขึ้นอยู่กับค่าสูงสุดของป้ายชื่อที่ถูกกำหนดให้แต่ละสถานี (โดยทั่วไปคือยิ่งน้อยยิ่งดี)

## รายละเอียดการเขียนโปรแกรม

คุณต้องเขียนฟังก์ชันต่อไปนี้

```
int[] label(int n, int k, int[] u, int[] v)
```

- $n$ : จำนวนสถานีใน SIB
- $k$ : ค่าป้ายชื่อสูงสุดที่ใช้ได้
- $u$  และ  $v$ : อาร์เรย์ขนาด  $n - 1$  ซึ่งอธิบายการเชื่อมต่อ สำหรับค่า  $i$  (โดยที่  $0 \leq i \leq n - 2$ ) การเชื่อมต่อ  $i$  จะเชื่อมสถานีที่มีดัชนี  $u[i]$  กับ  $v[i]$
- ฟังก์ชันนี้ควรจะคืนอาร์เรย์  $L$  ขนาด  $n$  สำหรับค่า  $i$  (โดยที่  $0 \leq i \leq n - 1$ )  $L[i]$  คือป้ายชื่อที่กำหนดให้กับสถานีที่มีดัชนี  $i$  สมาชิกทุกตัวในอาร์เรย์  $L$  จะต้องไม่ซ้ำกันและมีค่าตั้งแต่ 0 ถึง  $k$  รวมหัวท้าย

```
int find_next_station(int s, int t, int[] c)
```

- $s$ : ป้ายชื่อของสถานีที่ถือแพ็คเกจอยู่
- $t$ : ป้ายชื่อของสถานีเป้าหมาย
- $c$ : อาร์เรย์ที่ระบุนายการของป้ายชื่อของเพื่อนบ้านทั้งหมดของ  $s$  อาร์เรย์  $c$  นี้เรียกจากน้อยไปมาก
- ฟังก์ชันนี้ต้องคืนค่าป้ายชื่อของเพื่อนบ้านของ  $s$  ที่จะต้องส่งแพ็คเกจนี้ไป

ข้อมูลทดสอบแต่ละชุดอาจจะเกี่ยวข้องกับสถานการณ์หนึ่งหรือมากกว่าสถานการณ์ที่ไม่เกี่ยวข้องกัน (กล่าวคือ เป็น SIB ที่แตกต่างกัน) สำหรับกรณีทดสอบที่มี สถานการณ์  $r$  สถานการณ์ โปรแกรมที่เรียกฟังก์ชันข้างบนนี้จะถูกเรียกสองครั้ง ดังนี้

ในการรันโปรแกรมครั้งแรก

- ฟังก์ชัน `label` จะถูกเรียก  $r$  ครั้ง
- ค่าป้ายชื่อที่ส่งคืนมาจะถูกเก็บไว้ในระบบเกรดเดอร์ และ
- ฟังก์ชัน `find_next_station` จะไม่ถูกเรียก

ในการรันโปรแกรมครั้งที่สอง

- ฟังก์ชัน `find_next_station` อาจจะถูกเรียกได้หลายครั้ง ในแต่ละครั้ง สถานการณ์ ใดๆ จะถูกเลือกขึ้นมา และป้ายชื่อที่ได้รับมาจากการเรียกฟังก์ชัน `label` สำหรับสถานการณ์นั้นจะถูกนำมาใช้เป็นข้อมูลนำเข้าของฟังก์ชัน `find_next_station`
- ฟังก์ชัน `label` จะไม่ถูกเรียก

กล่าวคือ ข้อมูลใด ๆ ที่เก็บไว้ในตัวแปรแบบ `static` หรือ `global` ในการรันครั้งแรกของโปรแกรม จะไม่สามารถใช้ได้ในฟังก์ชัน `find_next_station`

## ตัวอย่าง

ให้พิจารณาการเรียกต่อไปนี้

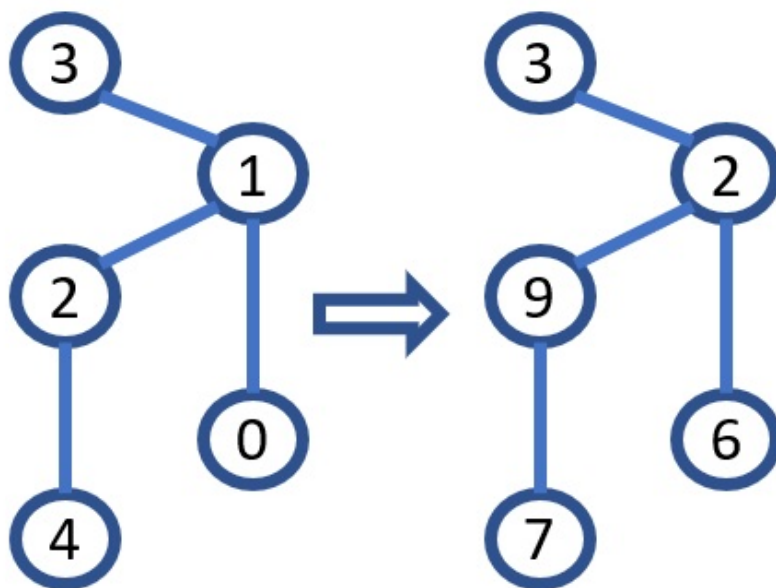
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

ซึ่งหมายความว่า มีสถานีทั้งหมด 5 สถานี และ 4 การเชื่อมต่อซึ่งเชื่อมคู่สถานีที่มีดัชนีเป็น (0, 1), (1, 2), (1, 3) และ (2, 4) ป้ายชื่อสามารถเป็นจำนวนเต็มตั้งแต่ 0 ถึง  $k = 10$

หากต้องการใช้ป้ายชื่อดังต่อไปนี้

Index	Label
0	6
1	2
2	9
3	3
4	7

ฟังก์ชัน `label` จะต้องคืนค่า [6, 2, 9, 3, 7] ตัวเลขในรูปต่อไปนี้แสดงถึงดัชนี (รูปด้านซ้าย) และ ป้ายชื่อที่กำหนดให้ (รูปด้านขวา)



ให้พิจารณาการเรียกฟังก์ชันต่อไปนี้ โดยถือว่าป้ายชื่อถูกกำหนดดังที่อธิบายในรูปด้านบน

```
find_next_station(9, 6, [2, 7])
```

ซึ่งหมายถึง สถานีที่ถือแพ็คเก็ตอยู่มีป้ายชื่อคือ 9 และป้ายชื่อของสถานีเป้าหมายคือ 6 เนื่องจากป้ายชื่อของสถานีตามเส้นทางไปยังสถานีเป้าหมายคือ [9, 2, 6] ดังนั้นการเรียกฟังก์ชันนี้ควรจะคืนค่า 2 ซึ่งเป็นป้ายชื่อของสถานีที่แพ็คเก็ตนี้ควรจะต้องถูกส่งไป (ซึ่งเป็นสถานีที่มีดัชนีเป็น 1)

ให้พิจารณาการเรียกอีกแบบที่เป็นไปได้

```
find_next_station(2, 3, [3, 6, 9])
```

ฟังก์ชันนี้ควรจะคืนค่า 3 เนื่องจากสถานีเป้าหมายซึ่งมีป้ายชื่อเป็น 3 นั้นเป็นเพื่อนบ้านของสถานีที่มีป้ายชื่อเป็น 2 ดังนั้นจึงควรจะได้รับแพ็คเกจนี้ทันที

## ข้อจำกัด

- $1 \leq r \leq 10$

สำหรับการเรียกฟังก์ชัน `label`:

- $2 \leq n \leq 1000$
- $k \geq n - 1$
- $0 \leq u[i], v[i] \leq n - 1$  (for all  $0 \leq i \leq n - 2$ )

สำหรับการเรียกฟังก์ชัน `find_next_station` ข้อมูลนำเข้าจะมาจากการเรียกฟังก์ชัน `label` ใด ๆ ก่อนหน้าก็ได้ จากการพิจารณาป้ายชื่อที่ได้มา

- $s$  และ  $t$  คือป้ายชื่อของสองสถานีที่แตกต่างกัน
- $c$  เป็นลำดับของป้ายชื่อของเพื่อนบ้านของสถานีที่มีป้ายชื่อ  $s$  เรียงจากน้อยไปมาก

สำหรับแต่ละข้อมูลทดสอบ ความยาวของอาร์เรย์  $c$  ที่ส่งให้ฟังก์ชัน `find_next_station` จะไม่เกิน 100 000 สำหรับทุก ๆ สถานการณ์รวมกัน

## ปัญหาย่อย

1. (5 คะแนน)  $k = 1000$ , ไม่มีสถานีใดมีเพื่อนบ้านมากกว่า 2 สถานี
2. (8 คะแนน)  $k = 1000$ , การเชื่อมต่อ  $i$  เชื่อมสถานี  $i + 1$  กับ  $\lfloor \frac{i}{2} \rfloor$
3. (16 คะแนน)  $k = 1\,000\,000$  มีสถานีไม่เกิน 1 สถานีที่มีเพื่อนบ้านมากกว่า 2
4. (10 คะแนน)  $n \leq 8$ ,  $k = 10^9$
5. (61 คะแนน)  $k = 10^9$

ในปัญหาย่อยที่ 5 คุณสามารถได้คะแนนเพียงบางส่วน (partial score) ให้  $m$  ค่าป้ายชื่อมากที่สุดที่ได้คืนมาจาก `label` ในสถานการณ์ทั้งหมด คะแนนของคุณจะถูกคำนวณตามตารางต่อไปนี้

Maximum label	Score
$m \geq 10^9$	0
$2000 \leq m < 10^9$	$50 \cdot \log_{5 \cdot 10^5}(\frac{10^9}{m})$
$1000 < m < 2000$	50
$m \leq 1000$	61

## เกรตเตอร์ตัวอย่าง

เกรตเตอร์ตัวอย่างอ่านข้อมูลนำเข้าในรูปแบบต่อไปนี้

- บรรทัด 1:  $r$

ตามด้วยข้อมูล  $r$  ชุด แต่ละชุดอธิบายถึงสถานการณ์หนึ่งสถานการณ์ในรูปแบบต่อไปนี้

- บรรทัด 1:  $n \ k$
- บรรทัด  $2 + i$  ( $0 \leq i \leq n - 2$ ):  $u[i] \ v[i]$
- บรรทัด  $1 + n$ :  $q$ : จำนวนครั้งในการเรียก `find_next_station`
- บรรทัด  $2 + n + j$  ( $0 \leq j \leq q - 1$ ):  $z[j] \ y[j] \ w[j]$ : **ดัชนี** ของสถานีสำหรับการเรียก `find_next_station`  $j$  สถานี  $z[j]$  คือแพ็คเกจอยู่ สถานี  $y[j]$  คือสถานีเป้าหมาย และสถานี  $w[j]$  คือสถานีที่แพ็คเกจนั้นควรจะถูกส่งไป

เกรตเตอร์ตัวอย่างจะพิมพ์ผลลัพธ์ในรูปแบบต่อไปนี้

- บรรทัด 1:  $m$

ตามด้วยข้อมูล  $r$  ชุด ตามลำดับของสถานการณ์ตามข้อมูลนำเข้า โดยแต่ละชุดจะมีรูปแบบดังต่อไปนี้

- บรรทัด  $1 + j$  ( $0 \leq j \leq q - 1$ ): **ดัชนี** ของสถานีที่ป้ายชื่อของสถานียังกล่าวได้รับคืนมาจากการเรียก `find_next_station` ครั้งที่  $j$  ในสถานการณ์นั้น

ให้สังเกตว่าในการรันเกรตเตอร์ตัวอย่างแต่ละครั้งนั้นเรียกทั้ง `label` และ `find_next_station`