ธนาคารทุนนิยม

ประเทศหนึ่งมีเมือง N เมือง เรียกเป็นเมือง 0 ถึง $N-1$ มีถนนแบบไปกลับสองทิศทาง $N-1$ เส้นที่เชื่อมระหว่างเมืองเหล่านี้ที่รับประกันว่าสามารถเดินทางจากเมืองใด ๆ ไปเมืองอื่น ๆ ได้เสมอ ถนนเส้นที่ i สำหรับ $0 \leq i \leq N-1$ เชื่อมระหว่างเมือง $R[i][0]$ กับเมือง $R[i][1]$ และมีค่าเดินทาง $R[i][2]$ บาท

การคิดค่าธรรมเนียมเป็นสิ่งปกติของประเทศนี้ ถ้าคุณต้องการถอนเงิน คุณก็จะต้องเดินทางไปที่ธนาคาร (เสียค่าเดินทาง) และเสียค่าธรรมเนียมค่าถอน ค่าธรรมเนียมจะคิดเป็นเปอร์เซ็นต์ของเงินที่ต้องการถอน และ มีค่าใช้ค่าธรรมเนียมการใช้บริการเป็นค่าคงที่อีกจำนวนหนึ่ง รับประกันว่าจำนวนเงินในการถอนเงินเป็นมูลค่าที่หาร 100 ลงตัวเสมอ

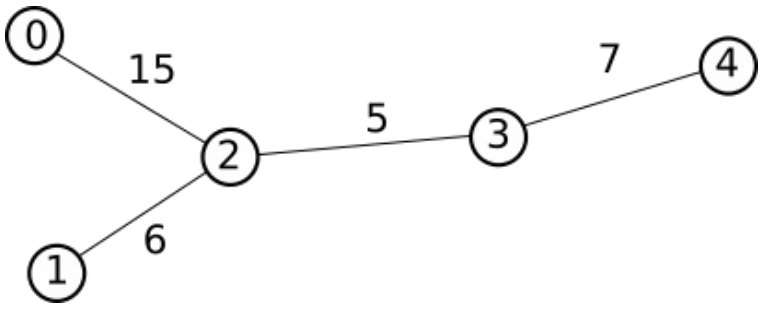
ธนาคารแห่งหนึ่งที่กำลังจะพยายามขยายสาขา เมื่อเริ่มต้นมีสาขาจำนวน K สาขา สำหรับ $0 \leq j \leq K-1$, สาขา j อยู่ที่เมือง $B[j][0]$ มีค่าธรรมเนียมในการถอนเงินคิดเป็น $B[j][1]$ % ของจำนวนเงินที่จะถอน บวกด้วยค่าใช้ธรรมเนียมการใช้บริการของธนาคารนั้นๆ $B[j][2]$ บาท ไม่ว่าจะถอนเงินแค่ไหนก็ตาม

ในแต่ละช่วงเวลา มีเหตุการณ์เกิดขึ้นได้ดังนี้

- **การเพิ่มสาขา**: ธนาคารจะเพิ่มสาขาที่เมือง P และจะมีค่าธรรมเนียมในการถอนเงิน F % และมีค่าธรรมเนียมคงที่ของธนาคารนั้น L บาท ซึ่งอาจจะสร้างซ้ำในเมืองที่มีธนาคารอยู่แล้วได้
- **การถอนเงิน**: คนร้อนเงินอยู่ที่เมือง S ต้องการเดินทางไปถอนเงินมูลค่า Y ที่ธนาคาร โดยต้องการให้มีค่าใช้จ่ายน้อยที่สุด (นั่นคือมีค่าเดินทางรวมกับค่าธรรมเนียมการถอนเงินน้อยที่สุด)

ทั้งสองเหตุการณ์จะเกิดขึ้นไม่เกิน Q ครั้ง ให้เขียนโปรแกรมตอบคำถามการถอนเงินให้ถูกต้อง

พิจารณาตัวอย่างต่อไปนี้ ที่ $N = 5$ มีถนน 4 เส้นดังแสดงในรูปด้านล่าง



สมมติว่ามีธนาคารอยู่แล้วสองที่ ($K = 2$) ที่เมือง $B[0][0] = 1$ และเมือง $B[1][0] = 4$ ธนาคารทั้งสองมีอัตราค่าธรรมเนียมคือ $B[0][1] = 5\%, B[0][2] = 3$ และ $B[1][1] = 2\%, B[1][2] = 5$

สมมติว่าคนต้องการถอนเงินอยู่ที่เมือง 2 และต้องการถอนเงิน 100 บาท เขาสามารถเลือกไปธนาคารที่เมือง 1 เสียค่าเดินทาง 6 บาท และค่าธรรมเนียม $0.05 \times 100 + 3 = 8$ บาท รวมมีค่าใช้จ่าย 14 บาท ถ้าเขาเลือกไปธนาคารที่เมือง 4 จะเสียค่าใช้จ่ายเป็นค่าเดินทาง 12 บาทและค่าธรรมเนียม

$0.02 \times 100 + 5 = 7$ บาท รวม 19 บาท ดังนั้นค่าใช้จ่ายน้อยที่สุดคือ 14 บาท โดยไปธนาคารที่เมือง 1

เวลาถัดมา สมมติว่าธนาคารเปิดสาขาเพิ่มที่เมือง $P = 0$ โดยมีอัตราค่าธรรมเนียมที่ $F = 1\%, L = 10$

หลังจากนั้นถ้ามีคนที่เมือง 2 ต้องการถอนเงิน $1\,000$ บาท เขามีทางเลือก 3 ทาง ถ้าไปที่เมือง 1 จะจ่าย $6 + 50 + 3 = 59$ บาท ถ้าไปที่เมือง 4 จะจ่าย $12 + 20 + 5 = 37$ บาท และถ้าไปเมือง 0 จะจ่าย $15 + 10 + 10 = 35$ บาท ดังนั้นค่าใช้จ่ายน้อยที่สุดคือ 35 บาท

ถัดมาถ้ามีคนที่เมือง 3 ต้องการถอนเงิน 500 บาท เช่นเคยเขามีทางเลือก 3 ทาง แต่เขาควรไปที่เมือง 4 ที่มีค่าใช้จ่าย $7 + 10 + 5 = 22$ บาท ซึ่งน้อยที่สุดในบรรดาทางเลือกทั้งสามทาง

รายละเอียดการเขียนโปรแกรม

คุณจะต้องเขียนฟังก์ชันต่อไปนี้

```
void initialize(int N, int K,
                vector<vector<int>> R,
                vector<vector<long long>> B)
```

- ฟังก์ชันนี้จะถูกเรียกครั้งเดียวเท่านั้นก่อนเริ่มการเรียกฟังก์ชันอื่นๆ
- เวกเตอร์ R จะมีขนาด $N - 1$ โดยที่สำหรับ $0 \le i \le N - 2$ ถนนเส้นที่ i จะเชื่อมระหว่างเมือง $R[i][0]$ กับ $R[i][1]$ และมีความยาว $R[i][2]$
- เวกเตอร์ B ระบุข้อมูลธนาคารตั้งต้น กล่าวคือ สำหรับ j ที่ $0 \le j \le K - 1$, ธนาคารสาขา j อยู่ที่เมือง $B[j][0]$ และมีค่าธรรมเนียม $B[j][1]$ % และค่าธรรมเนียมคงที่ $B[j][2]$ บาท

คุณจะต้องเขียนอีกสองฟังก์ชันต่อไปนี้เพื่อรองรับเหตุการณ์

```
void update_bank(int P, int F, long long L)
```

- เป็นเหตุการณ์เปิดธนาคารใหม่ที่เมือง P
- ธนาคารสาขานี้จะมีค่าธรรมเนียมในการถอนเงิน F %
- ธนาคารสาขานี้จะมีค่าธรรมเนียมคงที่ในการถอนเงิน L บาท

และฟังก์ชัน

```
long long find_best_bank(int S, int Y)
```

- เป็นการถามว่าผู้ถอนเงินจากเมือง S ที่ต้องการถอนเงิน Y หน่วย จะมีค่าใช้จ่ายน้อยที่สุดเท่าใดในการถอนเงิน ค่าใช้จ่ายในที่นี้คือค่าใช้จ่ายในการเดินทางไปยังธนาคารและค่าธรรมเนียมในการถอนเงินรวมกัน

ฟังก์ชัน `update_bank` และ `find_best_bank` จะถูกเรียกรวมกันไม่เกิน Q ครั้ง

ขอบเขต

- $1 \le N \le 100\,000$, $0 \le K \le N$
- $1 \le Q \le 100\,000$
- สำหรับ $0 \le i < N-1$, $0 \le R[i][0] \le N-1$, $0 \le R[i][1] \le N-1$, $R[i][0] \ne R[i][1]$
- สำหรับ $0 \le i < N-1$, $1 \le R[i][2] \le 10^6$
- สำหรับ $0 \le j \le K-1$, $0 \le B[j][0] \le N-1$, $0 \le B[j][1] \le 10^5$, $0 \le B[j][2] \le 10^{14}$
- $0 \le P \le N-1$, $0 \le F \le 10^5$, $0 \le L \le 10^{14}$
- $0 \le S \le N-1$, $1 \le Y \le 10^9$

ปัญหาย่อย

1. (10 คะแนน) $N, Q \le 10^3$
2. (13 คะแนน) $N \le 10^5, Q \le 10^5, B[j][1] = 0, B[j][2] = 0, F = 0, L = 0$
3. (17 คะแนน) $N \le 10^5, Q \le 10^5, B[j][1] = 0, F = 0$
4. (12 คะแนน) $N \le 10^5, Q \le 10^5$ แต่จะไม่มีการเรียกฟังก์ชัน `update_bank`
5. (17 คะแนน) $N \le 10^5, Q \le 10^5$ แต่จะไม่มีการเรียกฟังก์ชัน `update_bank` หลังจากเกิดการเรียกฟังก์ชัน `find_best_bank`
6. (31 คะแนน) ไม่มีเงื่อนไขเพิ่มเติม

ตัวอย่าง

จากตัวอย่างข้างต้น จะมีการเรียกฟังก์ชัน `initialize` ดังนี้

```
initialize(5, 2,
           [[0, 2, 15], [1, 2, 6], [2, 3, 5], [3, 4, 7]],
           [[1, 5, 3], [4, 2, 5]])
```

หลังจากนั้นจะมีคำถาม

```
find_best_bank(2, 100)
```

ฟังก์ชันที่ทำงานถูกต้องจะคืนค่า 14

ฟังก์ชัน

```
update_bank(0, 1, 10)
```

จะถูกเรียกเพื่อเพิ่มธนาคารที่เมือง 0 ด้วยอัตราค่าธรรมเนียม 1% ค่าธรรมเนียมคงที่ 10 บาท

สำหรับคำถามถัดมา

```
find_best_bank(2, 1000)
```

จะต้องตอบ 35

และคำถาม

```
find_best_bank(3, 500)
```

จะต้องตอบ 22

เกรดเดอร์ตัวอย่าง

เกรดเดอร์ตัวอย่างอ่านข้อมูลดังนี้

- บรรทัดที่ 1: N, K, Q
- บรรทัดที่ $2, \ldots, N$: $R[i][0]$, $R[i][1]$, $R[i][2]$
- บรรทัดที่ $N+1, \ldots, N+K$: $B[i][0], B[i][1], B[i][2]$
- บรรทัดที่ $N+K+1, \ldots, N+K+Q$: จะอยู่ในรูปแบบใดแบบหนึ่งดังนี้
  - $1\ P\ F\ L$: เปิดธนาคารที่เมือง P อัตราค่าธรรมเนียม $F\%$ ค่าธรรมเนียมคงที่ L บาท
  - $2\ S\ Y$: เป็นคำถาม ว่าจากเมือง S ถอนเงิน Y ค่าใช้จ่ายต่ำสุดเป็นเท่าใด

เกรดเดอร์จะพิมพ์ค่าที่คืนจากฟังก์ชัน `find_best_bank` บรรทัดละตัว

ข้อจำกัด

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