경찰과 도둑 (police)

KOI 마을은 N개의 집과 집들을 잇는 $N-1$개의 양방향 도로로 이루어져 있으며, 임의의 서로 다른 두 집들을 도로만을 사용하여 오갈 수 있다. 즉, KOI 마을의 도로망은 트리 구조를 이룬다.

KOI 마을의 집들에는 0부터 $N-1$까지의 서로 다른 번호가 붙어 있으며, KOI 마을의 도로들에는 0부터 $N-2$까지의 서로 다른 번호가 붙어 있다. 모든 $0 \le i \le N-2$에 대해 i번 도로는 $A[i]$번 집과 $B[i]$번 집을 연결하며 길이는 $D[i]$미터이다.

최근 KOI 마을에 도둑이 자주 들어 주민들이 어려움을 겪고 있다. 이에 KOI 마을의 한 집에 경찰을 대기시켜, 도둑이 나타났을 때를 대비하려고 한다. KOI 마을의 사람들은 도둑이 드는 상황에서 경찰이 얼마나 빠르게 도둑을 잡을 수 있을지 궁금해졌다.

여러분에게 Q개의 시나리오가 주어진다. 시나리오에는 0부터 $Q-1$까지의 서로 다른 번호가 붙어 있다. 하나의 시나리오는 다음과 같이 이루어진다.

- j번 시나리오에서 경찰은 $P[j]$번 집에서 출발하며 1초에 최대 $V1[j]$미터를 이동할 수 있다.
- j번 시나리오에서 도둑은 $T[j]$번 집에서 출발하며 1초에 최대 $V2[j]$미터를 이동할 수 있다.
- 경찰이 출발하는 집과 도둑이 출발하는 집은 다르다. 즉, $P[j] \neq T[j]$이다.
- 집의 크기는 충분히 작으므로 집은 점으로 취급한다. 도로의 너비는 충분히 좁으므로 도로는 선분으로 취급한다. 도로들은 교차하지 않는다.
- 경찰과 도둑은 각각 자신의 최대 속력 내에서 KOI 마을 안을 자유롭게 이동할 수 있다. 이동하지 않는 것도 가능하다.
- 경찰이 도둑과 같은 위치에 있다면 경찰은 도둑을 잡을 수 있다. 이때, 집뿐만이 아니라 도로의 중간에서도 도둑을 잡는 것이 가능하다.
- 시나리오 내에서 경찰과 도둑은 자신과 상대방의 속도를 알고 있으며, 어느 시점이든 상대방의 위치를 알 수 있다.
- 경찰과 도둑은 최선의 전략을 사용한다. 즉, 경찰은 도둑을 가장 빠르게 잡는 전략을, 도둑은 가장 오랫동안 도망치는 전략을 사용한다. 경찰과 도둑이 최선의 전략을 사용할 때, 반드시 유한한 시간 안에 도둑이 잡힘을 증명할 수 있다.

여러분은 각 시나리오마다 도둑이 잡히는 데에 걸리는 시간을 구해야 한다.

함수 목록 및 정의

여러분은 아래 함수를 구현해야 한다.

```
vector< array<long long, 2> > police_thief(vector<int> A, vector<int> B,
vector<int> D, vector<int> P, vector<int> V1, vector<int> T, vector<int> V2)
```

- A, B, D: 크기가 $N-1$인 정수 배열. 모든 $0 \le i \le N-2$에 대해 i번 도로는 $A[i]$번 집과 $B[i]$번 집을 연결하며 길이는 $D[i]$미터이다.
- P, $V1$, T, $V2$: 크기가 Q인 정수 배열. 모든 $0 \le j \le Q-1$에 대해 j번 시나리오에서 경찰은 $P[j]$번 집에서 출발하며 1초에 최대 $V1[j]$미터를 이동할 수 있고 도둑은 $T[j]$번 집에서 출발하며 1초에 최대 $V2[j]$미터를 이동할 수 있다.
- 이 함수는 각 원소가 크기 2의 배열인 크기가 Q인 배열 C를 반환해야 한다. 모든 $0 \le j \le Q-1$에 대해 j번 시나리오에서 도둑이 잡히는 데에 걸리는 시간(초 단위)을 분수로 나타낸 형태가 $C[j][0]/C[j][1]$이어야 한다.
- $C[j][0]/C[j][1]$은 기약분수가 아니여도 무방하다. 단, $C[j][0]$과 $C[j][1]$은 1 이상 10^{18} 이하의 정수여야 한다. 제약 조건을 만족하는 모든 입력에 대해, 정답을 항상 해당 형태의 분수로 표현할 수 있음을 증명할 수 있다.

제출하는 소스 코드의 어느 부분에서도 입출력 함수를 실행해서는 안 된다.

제약 조건

- $2 \le N \le 100\,000$
- $1 \le Q \le 100\,000$
- 모든 $0 \le i \le N-2$에 대해 $0 \le A[i], B[i] \le N-1$, $A[i] \ne B[i]$
- 모든 $0 \le i \le N-2$에 대해 $1 \le D[i] \le 1\,000\,000$
- KOI 마을은 트리 구조를 이룬다.
- 모든 $0 \le j \le Q-1$에 대해 $0 \le P[j], T[j] \le N-1$, $P[j] \ne T[j]$
- 모든 $0 \le j \le Q-1$에 대해 $1 \le V1[j], V2[j] \le 1\,000\,000$

부분문제

1. (15점)
    - $N \leq 5\,000$
    - $Q \leq 5\,000$
2. (21점)
    - $N \leq 50\,000$
    - $Q \leq 50\,000$
3. (5점)
    - 모든 $0 \leq i \leq N-2$에 대해 $A[i] = i$, $B[i] = i+1$
4. (6점)
    - 모든 $0 \leq i \leq N-2$에 대해 $A[i] = 0$, $B[i] = i+1$
5. (14점)
    - 모든 $0 \leq j \leq Q-1$에 대해 $V1[j] \leq V2[j]$
6. (9점)
    - 모든 $0 \leq j \leq Q-1$에 대해 $P[j]$번 집과 $T[j]$번 집을 잇는 단순 경로 상의 도로의 개수는 10개 이하
7. (9점)
    - 모든 $0 \leq j \leq Q-1$에 대해 $P[j] = 0$
8. (10점)
    - 모든 $0 \leq j \leq Q-1$에 대해 $T[j] = 0$
9. (11점)
    - 추가적인 제약 조건이 없다.

예제 1

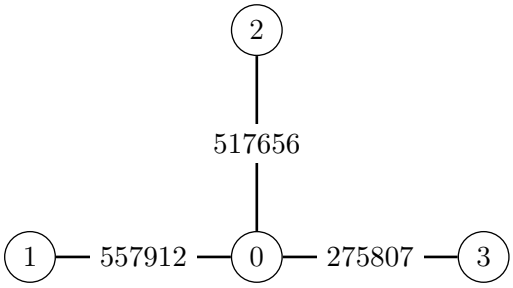
$N = 4$, $Q = 3$, $A = [0, 0, 0]$, $B = [1, 2, 3]$, $D = [557912, 517656, 275807]$,

$P = [3, 0, 0]$, $V1 = [265381, 190435, 195025]$, $T = [0, 2, 3]$, $V2 = [1000000, 12345, 67890]$인 경우를 생각해 보자.

그레이더는 다음과 같이 함수를 호출한다.

```
police_thief([0, 0, 0], [1, 2, 3], [557912, 517656, 275807], [3, 0, 0],
[265381, 190435, 195025], [0, 2, 3], [1000000, 12345, 67890])
```

아래 그림은 KOI 마을의 구조를 나타낸다.



0번 시나리오에서 도둑은 1번 집으로 향하는 경로를 따라 이동하며, 1번 집에서 잡히게 된다. 1번과 2번 시나리오에서 도둑은 출발하는 집에서 움직이지 않는다.

함수의 반환값으로 가능한 값은 $[[833719, 265381], [517656, 190435], [275807, 195025]]$가 있다. 이 외에도 $[[833719, 265381], [517656, 190435], [551614, 390050]]$ 등이 가능한 반환값이 된다.

예제 2

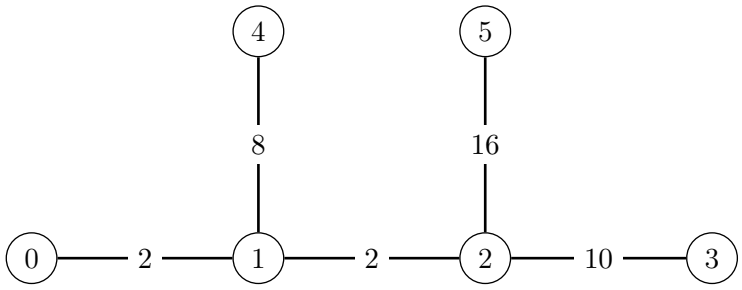
$N = 6$, $Q = 4$, $A = [0, 1, 2, 1, 2]$, $B = [1, 2, 3, 4, 5]$, $D = [2, 2, 10, 8, 16]$,

$P = [3, 3, 3, 3]$, $V1 = [4, 2, 19, 20]$, $T = [0, 0, 0, 0]$, $V2 = [3, 1, 9, 19]$인 경우를 생각해 보자.

그레이더는 다음과 같이 함수를 호출한다.

```
police_thief([0, 1, 2, 1, 2], [1, 2, 3, 4, 5], [2, 2, 10, 8, 16], [3, 3, 3,
3], [4, 2, 19, 20], [0, 0, 0, 0], [3, 1, 9, 19])
```

아래 그림은 KOI 마을의 구조를 나타낸다.



0번 시나리오에서 도둑은 5번 집으로 향하는 경로를 따라 이동하다가 2번 집과 5번 집을 연결하는 도로의 중간에서 잡히게 된다.

함수의 반환값으로 가능한 값은 $[[6, 1], [10, 1], [1, 1], [13, 10]]$가 있다.

예제 3

$N = 10$, $Q = 10$, $A = [4, 2, 9, 9, 3, 7, 1, 6, 5]$, $B = [9, 8, 0, 1, 1, 6, 2, 2, 9]$, $D = [7, 8, 4, 5, 1, 2, 5, 10, 2]$,

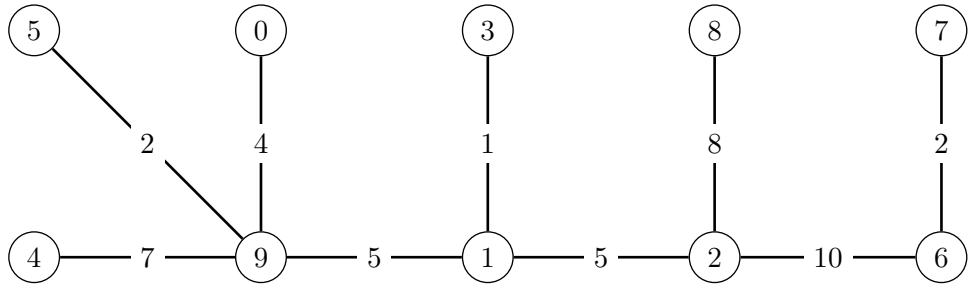
$P = [3, 0, 5, 2, 0, 5, 5, 7, 9, 8]$, $V1 = [1, 6, 6, 5, 2, 6, 5, 4, 1, 5]$,

$T = [5, 5, 9, 1, 6, 2, 0, 1, 8, 4]$, $V2 = [9, 7, 6, 7, 4, 10, 10, 8, 7, 5]$인 경우를 생각해 보자.

그레이더는 다음과 같이 함수를 호출한다.

```
police_thief([4, 2, 9, 9, 3, 7, 1, 6, 5], [9, 8, 0, 1, 1, 6, 2, 2, 9], [7, 8, 4, 5, 1, 2, 5, 10, 2], [3, 0, 5, 2, 0, 5, 5, 7, 9, 8], [1, 6, 6, 5, 2, 6, 5, 4, 1, 5], [5, 5, 9, 1, 6, 2, 0, 1, 8, 4], [9, 7, 6, 7, 4, 10, 10, 8, 7, 5])
```

아래 그림은 KOI 마을의 구조를 나타낸다.



함수의 반환값으로 가능한 값은 $[[18, 1], [13, 3], [4, 1], [17, 5], [13, 1], [4, 1], [6, 5], [29, 4], [22, 1], [5, 1]]$가 있다.

예제 4

$N = 10$, $Q = 10$, $A = [6, 8, 8, 3, 4, 9, 0, 1, 8]$, $B = [7, 5, 2, 9, 1, 7, 4, 3, 4]$, $D = [1, 1, 4, 4, 4, 7, 3, 4, 7]$,

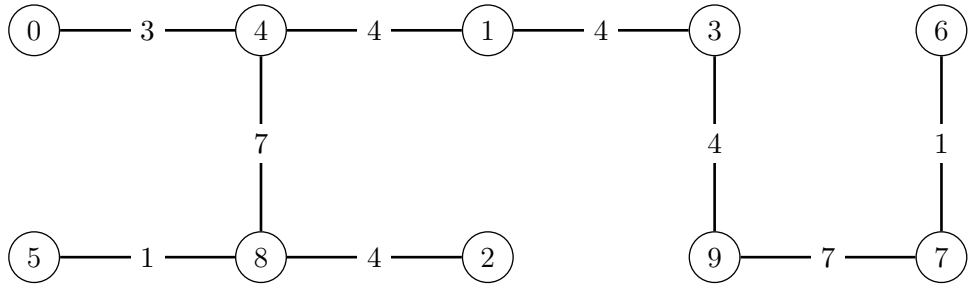
$P = [3, 1, 6, 2, 3, 3, 2, 6, 1, 2]$, $V1 = [5, 7, 9, 7, 5, 10, 8, 8, 4, 8]$,

$T = [0, 7, 8, 0, 2, 0, 0, 7, 8, 5]$, $V2 = [2, 2, 5, 5, 4, 5, 7, 2, 2, 7]$인 경우를 생각해 보자.

그레이더는 다음과 같이 함수를 호출한다.

```
police_thief([6, 8, 8, 3, 4, 9, 0, 1, 8], [7, 5, 2, 9, 1, 7, 4, 3, 4], [1, 1, 4, 4, 4, 7, 3, 4, 7], [3, 1, 6, 2, 3, 3, 2, 6, 1, 2], [5, 7, 9, 7, 5, 10, 8, 8, 4, 8], [0, 7, 8, 0, 2, 0, 0, 7, 8, 5], [2, 2, 5, 5, 4, 5, 7, 2, 2, 7])
```

아래 그림은 KOI 마을의 구조를 나타낸다.



함수의 반환값으로 가능한 값은 $[[11, 5], [16, 7], [31, 9], [4, 1], [19, 5], [11, 10], [31, 8], [1, 6], [15, 4], [3, 1]]$가 있다.

Sample grader

Sample grader는 아래와 같은 형식으로 입력을 받는다.

- Line 1: N Q
- Line $2 + i$ $(0 \le i \le N - 2)$: $A[i]$ $B[i]$ $D[i]$
- Line $1 + N + j$ $(0 \le j \le Q - 1)$: $P[j]$ $V1[j]$ $T[j]$ $V2[j]$

`police_thief`가 반환한 배열을 C라고 하자. Sample grader는 다음을 출력한다.

- Line $1 + j$ $(0 \le j \le Q - 1)$: $C[j][0]$ $C[j][1]$

Sample grader는 실제 채점에서 사용하는 그레이더와 다를 수 있음에 유의하라.