경주 (Race)

IOI 를 개최하는 파타야 시는 경주대회인 IOR 2011 도 함께 개최하며 이를 위해 가장 적합한 경주코스를 찾고 있다.

파타야 인근 지역에는 **N** 개의 도시가 있고 **N-1** 개의 고속도로가 이 도시들을 연결하고 있다. 각 고속도로는 양방향이며 서로 다른 두 개의 도시를 연결한다. 각 고속도로의 길이는 킬로미터 단위로 나타내며 정수 값이다. 그리고 임의의 두 도시는 직접 고속도로로 연결되지 않더라도 **단 하나의 경로**에 의해 연결된다. 즉, 같은 도시를 두 번 이상 방문하지 않고 한 도시에서 출발하여 다른 도시에 도착하는 방법은 유일하다**.**

IOR 에 사용되는 경주코스는 출발 도시와 도착 도시가 서로 달라야 하며 길이는 정확하게 **K** 킬로미터인 경로이다. 그리고 충돌을 방지하기 위해 한 고속도로를 두 번 이상 사용하지 않는다. (따라서 한 도시도 두 번 이상 방문하지 않는다.) 또한 교통체증을 줄이기 위해 되도록 가장 작은 수의 고속도로를 사용하여 경주코스를 구성하려고 한다.

태스크

다음의 파라미터를 받는 best_path(N,K,H,L) 함수를 작성하라.

- **N** – 도시의 수. 각 도시는 **0** 번부터 **N-1** 번까지 정수로 나타낸다.
- **K** – 경주코스의 길이.
- **H** – 각 고속도로를 나타내는 2 차원 배열. 고속도로 **i** (**0** ≤ **i** < **N-1**)는 도시 **H[i][0]**와 도시 **H[i][1]**를 연결하는 도로이다.
- **L** – 고속도로의 길이를 나타내는 1 차원 배열. 고속도로 **i** (**0** ≤ **i** < **N-1**)의 길이는 **L[i]**이다.

배열 **H** 에 저장된 값은 **0** 이상 **N-1** 이하이다. 또한 배열 **L** 에 저장된 값은 **0** 이상 **1 000 000** 이하의 정수이다. 그리고 모든 도시들은 연결되어 있다.

당신이 작성한 함수는 길이가 **K** 인 경주코스 중에서 고속도로 수가 가장 작은 경주코스의 고속도로 수를 반환한다. 만약 길이가 **K** 인 경주코스가 없다면 -1 을 반환하라.

예제

예제 1

그림 1 은 **N=4** 이고 **K=3** 인 경우이다. 이경우 배열 H 와 L 은 다음과 같다.

H=
0 1
1 2
1 3

L=
1
2
4

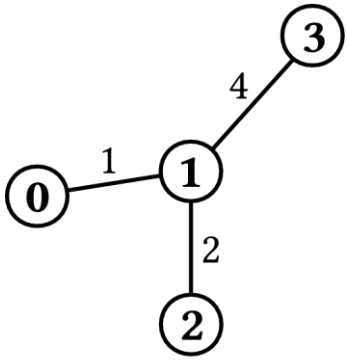

그림 1.

가능한 경주코스는 도시 0 에서 출발하여 도시 1 을 거쳐 도시 2 에 도착하는 코스이다. 이 코스의 총 길이는 1 km + 2 km = 3 km 이며 2 개의 고속도로를 사용한다. 이 코스가 최적의 코스이므로 best_path(N,K,H,L)는 2 를 반환해야 한다.

예제 2

그림 2 는 **N=3** 이고 **K=3** 인 경우이다.

H=
0 1
1 2

L=
1
1

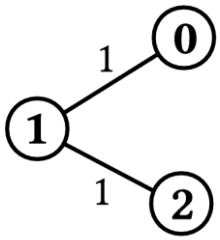

그림 2.

이 경우 가능한 경주코스가 없으므로 best_path(N,K,H,L)는 **-1** 을 반환해야 한다.

예제 3

그림 3 은 **N=11 이고 K=12** 인 경우이다.

H=

0 1	3
0 2	4
2 3	5
3 4	4
4 5	6
0 6	3
6 7	2
6 8	5
8 9	6
8 10	7

L=

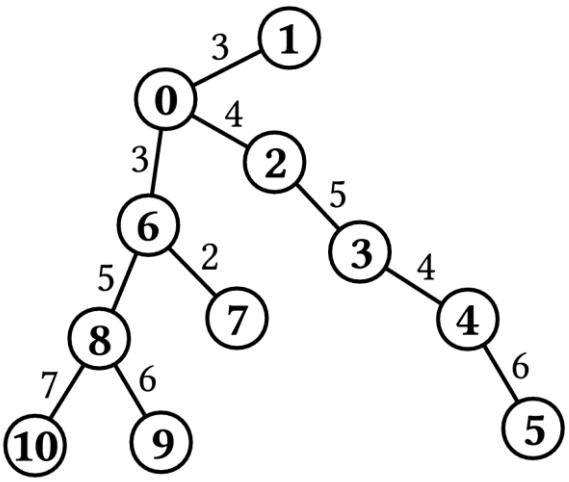

그림 3.

이 경우 여러 개의 경주코스가 존재한다. 그 중 하나는 도시 6 에서 출발하여 도시 0 과 2 를 거쳐 도시 3 에 도착하는 코스이다. 또 다른 하나는 도시 10 에서 출발하여 도시 8 을 거쳐 도시 6 에 도착하는 코스이다. 두 코스 모두 길이가 12km 이지만 두 번째 코스는 2 개의 고속도로만 이용하며 이는 최적의 코스이다. 왜냐하면 한 개의 고속도로만 이용하는 코스가 없기 때문이다. 따라서 best_path(N,K,H,L)는 2 를 반환해야 한다.

서브태스크

서브태스크 1 (9 점)

- $1 \le \mathbf{N} \le 100$
- $1 \le \mathbf{K} \le 100$
- 고속도로들이 직선을 이룬다: 고속도로 **i** 는 ($\mathbf{0} \le \mathbf{i} < \mathbf{N-1}$) 도시 **i** 와 도시 **i+1** 을 연결한다.

서브태스크 2 (12 점)

- 1 ≤ **N** ≤ 1 000
- 1 ≤ **K** ≤ 1 000 000

서브태스크 3 (22 점)

- 1 ≤ **N** ≤ 200 000
- 1 ≤ **K** ≤ 100

서브태스크 4 (57 점)

- 1 ≤ **N** ≤ 200 000
- 1 ≤ **K** ≤ 1 000 000

구현시 유의사항

제약조건

- CPU 시간 제한: 3 초
- 메모리 제한: 256 MB

  유의사항: 스택 메모리 크기에 대한 제한은 명시되지 않는다. 스택 메모리는 전체 메모리 사용량에 포함된다.

인터페이스 (API)

- 프로그램 작업폴더: race/
- 참가자가 작성할 파일: race.c 또는 race.cpp 또는 race.pas
- 참가자 인터페이스: race.h 또는 race.pas
- 채점프로그램(grader) 인터페이스: race.h 또는 racelib.pas
- 견본 채점프로그램 (sample grader): grader.c 또는 grader.cpp 또는 grader.pas
- 견본 채점프로그램 입력 (sample grader input): grader.in.1, grader.in.2, ...

  유의사항: 견본 채점프로그램은 다음과 같은 양식으로 입력을 읽는다:

  - 1 번째 줄: **N** 과 **K**.
  - 2 번째 줄부터 **N** 번째 줄까지: 고속도로에 대한 정보; 즉, **i**+2 번째 줄에는 H[i][0], H[i][1], 그리고 L[i]가 공백을 사이에 두고 주어진다 (0 ≤ **i** < **N**-1).
  - **N**+1 번째 줄: 예상되는 해.

- 견본 채점프로그램 입력에 대하여 예상되는 출력: grader.expect.1, grader.expect.2, ...
- 이 태스크에 대해서 각각의 파일은 정확히 "**Correct.**"를 포함해야 한다.