Race

In conjunction with the IOI, Pattaya City will host a race: the *International Olympiad in Racing* (IOR) 2011. As the host, we have to find the best possible course for the race.

In the Pattaya-Chonburi metropolitan area, there are **N** cities connected by a network of **N-1** highways. Each highway is bidirectional, connects two different cities, and has an integer length in kilometers. Furthermore, there is *exactly one* possible path connecting any pair of cities. That is, there is exactly one way to travel from one city to another city by a sequence of highways without visiting any city twice.

The IOR has specific regulations that require the course to be a path whose total length is *exactly* **K** kilometers, starting and ending in different cities. Obviously, no highway (and therefore also no city) may be used twice on the course to prevent collisions. To minimize traffic disruption, the course must contain as few highways as possible.

Your task

Write a procedure **best_path(N,K,H,L)** that takes the following parameters:

- **N** – the number of cities. The cities are numbered **0** through **N-1**.
- **K** – the required distance for the race course.
- **H** – a two-dimensional array representing highways. For **0** ≤ **i** < **N-1**, highway **i** connects the cities **H[i][0]** and **H[i][1]**.
- **L** – a one-dimensional array representing the lengths of the highways. For **0** ≤ **i** < **N-1**, the length of highway **i** is **L[i]**.

You may assume that all values in the array **H** are between **0** and **N-1**, inclusive, and that the highways described by this array connect all cities as described above. You may also assume that all values in the array **L** are integers between **0** and **1 000 000**, inclusive.

Your procedure must return the *minimum number of highways* on a valid race course of length exactly **K**. If there is no such course, your procedure must return **-1**.

Examples

Example 1

Consider the case shown in Figure 1, where **N=4**, **K=3**,

H=
0 1
1 2
1 3

L=
1
2
4

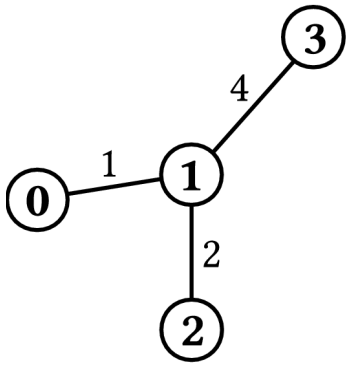

Figure 1.

The course can start in city 0, go to city 1, and terminate in city 2. Its length will be exactly 1 km + 2 km = 3 km, and it consists of two highways. This is the best possible course; therefore **best_path(N,K,H,L)** must return **2**.

Example 2

Consider the case shown in Figure 2, where **N=3**, **K=3**,

H=
0 1
1 2

L=
1
1

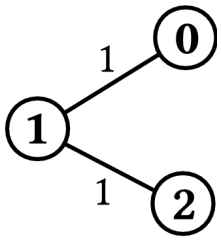

Figure 2.

There is no valid course. In this case, **best_path(N,K,H,L)** must return **-1**.

Example 3

Consider the case shown in Figure 3, where **N=11**, **K=12**,

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

L=
3
4
5
4
6
3
2
5
6
7

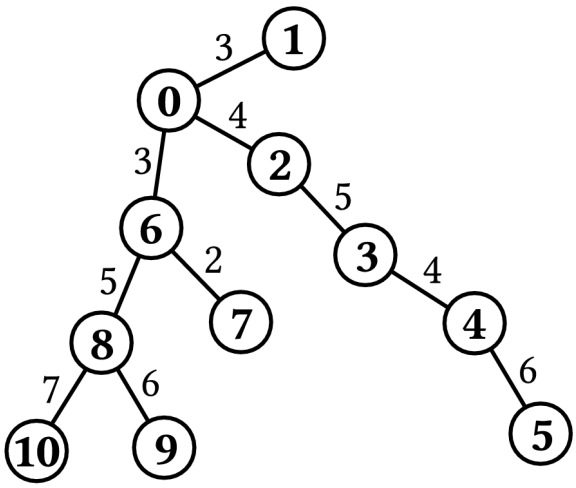

Figure 3.

One possible course consists of 3 highways: from city 6 via city 0 and city 2 to city 3. Another course starts in city 10 and goes via city 8 to city 6. Both of these courses have length exactly 12 km, as required. The second one is optimal, as there is no valid course with a single highway. Hence, **best_path(N,K,H,L)** must return **2**.

Subtasks

Subtask 1 (9 points)

- $1 \le N \le 100$
- $1 \le K \le 100$
- The network of highways forms the simplest possible line: For $0 \le i < N-1$, highway i connects cities i and $i+1$.

Subtask 2 (12 points)

- $1 \le N \le 1\,000$
- $1 \le K \le 1\,000\,000$

Subtask 3 (22 points)

- $1 \le \mathbf{N} \le 200\,000$
- $1 \le \mathbf{K} \le 100$

Subtask 4 (57 points)

- $1 \le \mathbf{N} \le 200\,000$
- $1 \le \mathbf{K} \le 1\,000\,000$

Implementation details

Limits

- CPU time limit: 3 seconds
- Memory limit: 256 MB
  Note: There is no explicit limit for the size of stack memory. Stack memory counts towards the total memory usage.

Interface (API)

- Implementation folder: race/
- To be implemented by contestant: race.c or race.cpp or race.pas
- Contestant interface: race.h or race.pas
- Grader interface: race.h or racelib.pas
- Sample grader: grader.c or grader.cpp or grader.pas
- Sample grader input: grader.in.1, grader.in.2, ...

  Note: The sample grader reads the input in the following format:

  - Line 1: **N** and **K**.
  - Lines 2 to **N**: information on the highways; i.e., line **i+2** contains **H[i][0]**, **H[i][1]**, and **L[i]**, separated by a space, for **0** ≤ **i** < **N-1**.
  - Line **N**+1: the expected solution.
- Expected output for sample grader input: grader.expect.1, grader.expect.2, ...
  For this task, each one of these files should contain precisely the text "**Correct.**"