Подземният град на крокодилите

Археоложката Бенджамас изследва мистериозния подземен град на крокодилите. Градът се състои от **N** стаи и **M** двупосочни коридора, всеки от които свързва различна двойка от различни стаи. Преминаването през всеки коридор изисква определено време за този коридор. Само **K** от **N**-те стаи са такива, че от тях може да се излезе от подземния град. Бенджамас се намира в стая 0. Тя иска да отиде възможно най-бързо в стая, откъдето може да излезе от града (т.е. да отиде в *стая-изход*).

Пазачът на кродилите иска да попречи на Бенджамас да избяга. От своето убежище, той управлява секретни врати, които могат да затворят всеки отделен коридор. Това означава, че когато той трябва да затвори нов коридор, трябва да отвори предишния затворен.

Положението на Бенджамас може да се опише по следния начин: Всеки път, когато тя се опита да излезе от стая, пазачът може да затвори един от коридорите, излизащ от стаята. Тогава Бенджамас избира да тръгне по коридор, който не е затворен, за да отиде в друга стая. Когато Бенджамас влезе в коридор, пазачът не може да затвори този коридор, докато тя е в коридора. Но когато тя отиде в следващата стая, пазачът вече може да затвори един от коридорите, свързващ тази стая (даже и коридора, по който тя е дошла).

Бенджамас иска да има предварително прост план за излизане от града. По-точно, тя би искала да има множество от инструкции, които да й казват какво да направи, когато се намира в стая. Нека **A** е една от стаите. Ако това е стая-изход, очевидно не е необходимо да има инструкция за нея. В противен случай, инструкцията за стая **A** трябва да има една от следните форми:

- „Ако си достигнала стая A, избери определен коридор водещ до стая B. Обаче, ако този коридор е затворен, тогава избери коридор, водещ до стая C.”
- „Не се интересувай от стая A; съгласно твоя план, ти не можеш въобще да я достигнеш”

Забележка: В някои случаи (например, ако вашият план води Бенджамас да се движи по цикъл) пазачът е способен да попречи на Бенджамас да излезе от града. Един план за излизане от града е *добър*, ако за Бенджамас се гарантира, че тя ще достигне стая-изход след крайно време, независимо от действията на пазача. Нека за един добър план да означим с **T** най-малкото време, такова че след това време Бенджамас (следвайки плана) гарантирано ще достигане стая-изход. В случая казваме, че този *добър план за изход има време* **T**.

Задача

Напишете функция travel_plan(N,M,R,L,K,P) със следните параметри:

- **N** – брой на стаите. Стаите са номерирани от **0** до **N-1**.
- **M** – брой на коридорите. Коридорите са номерирани от **0** до **M – 1**.
- **R** – двумерен масив от цели числа, задаващ коридорите. За $\mathbf{0 \le i < M}$, коридор **i** свързва двете различни стаи **R[i][0]** и **R[i][1]. Няма** два коридора, които да свързват една и съща двойка стаи.
- **L** – едномерен масив от цели числа, съдържащ времената за преминаване на коридорите. За $\mathbf{0 \le i < M}$, стойността $\mathbf{1 \le L[i] \le 1\ 000\ 000\ 000}$ е времето, за което Бенджамас изминава i-тия коридор.
- **K** – брой на стаите-изходи. Дадено е, че $\mathbf{1 \le K \le N}$.

- **P** – едномерен масив от цели числа с **K** различни стойности, съдържащи номерата на стаите-изходи. За $\mathbf{0} \leq \mathbf{i} < \mathbf{K}$ стойността **P[i]** е номерът на i-тата стая-изход. Стаята с номер 0 не е стая-изход.

Функцията трябва да върне най-малкото време **T**, което може да има един добър план за излизане от града.

Дадено е, че всяка стая, която не е стая-изход има поне два свързващи я коридора. Може да считате, че винаги има добър план за излизане от града, за който $\mathbf{T} \leq \mathbf{1\ 000\ 000\ 000}$.

Примери

Пример 1

Разгледайте случая, даден на Figure 1, където **N=5**, **M=4**, **K=3** и

R=
0 1
0 2
3 2
2 4

L=
2
3
1
4

P=
1
3
4

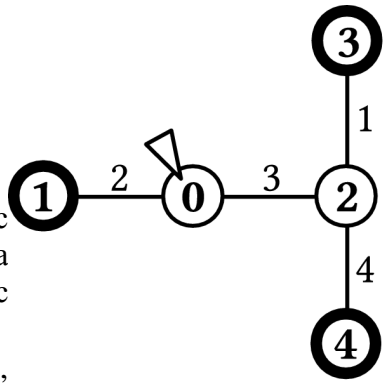

Figure 1.

Стаите са изобразени с кръгчета, а свързващите ги коридори – с отсечки от прави линии. Стаите-изходи са отбелязани с дебела линия на кръгчетата. Бенджамас започва в стая 0 (отбелязана с триъгълник). Оптимален план за излизане е следният:

- Ако достигнеш стая 0, избери коридора към стая 1. Но, ако този коридор е затворен, избери коридора водещ към стая 2.
- Ако достигнеш стая 2, избери коридора, водещ към стая 3. Но, ако този коридор е затворен, избери коридора, водещ към стая 4.

В най-лошия случай, Бенджамас ще достигне стая-изход за 7 единици време. Следователно, **travel_plan** трябва да върне 7.

Пример 2

Разгледайте случая, даден на Figure 2, където **N=5**, **M=7**, **K=2** и

R=
0 2
0 3
3 2
2 1
0 1
0 4
3 4

L=
4
3
2
10
100
7
9

P=
1
3

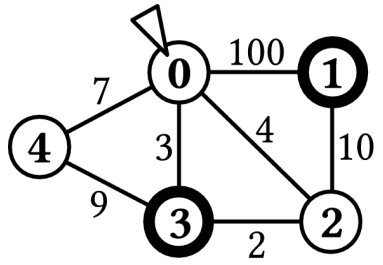

Figure 2.

Тук оптималният план е:

- Ако достигнеш стая 0, избери коридора към стая 3. Но, ако този коридор е затворен, избери коридора водещ към стая 2.
- Ако достигнеш стая 2, избери коридора, водещ към стая 3. Но, ако този коридор е затворен, избери коридора, водещ към стая 1.
- Не се интересувай от стая 4; не е възможно да я достигнеш, съгласно този план.

В най-лошия случай, Бенджамас ще достигне стая-изход за 14 единици време. Следователно, **travel_plan** трябва да върне 14.

Подзадачи

Подзадача 1 (46 точки)

- **3 ≤ N ≤ 1 000**.
- Подземният град е дърво. Т.е., **M = N**-1 и за всяка двойка стаи **i** и **j** съществува редица от коридори, свързващи **i** и **j**.
- Всяка стая-изход е свързана с точно една друга стая.
- Всяка друга стая е свързана директно с 3 или повече други стаи.

Подзадача 2 (43 точки)

- **3 ≤ N ≤ 1 000**.
- **2 ≤ M ≤ 100 000**.

Подзадача 3 (11 точки)

- **3 ≤ N ≤ 100 000**.
- **2 ≤ M ≤ 1 000 000**.

Детайли по реализацията

Ограничения

- Ограничение по време: 2 секунди
- Ограничение за памет: 256 MB
  Забележка: Не се задава явно ограничение за размера на стека. Паметта за стека зависи от общото ограничение за памет.

Интерфейс (API)

- Фолдер на имплементацията: crocodile/
- Да се имплементира от състезателя: crocodile.c или crocodile.cpp или crocodile.pas
- Интерфейс на състезателя: crocodile.h или crocodile.pas
- Интерфейс на грейдера: crocodile.h или crocodile.pas
- Примерен грейдер: grader.c или grader.cpp или grader.pas и crocodilelib.pas
- Вход на примерния грейдер: grader.in.1, grader.in.2, ...
  Забележка: Примерният грейдер чете входа си в следния формат:
  - Ред 1: **N**, **M** и **K**.
  - Редове 2 до **M**+1: за **0 ≤ i < M**, ред i+2 съдържа R[i][0], R[i][1] и L[i], разделени с интервал.
  - Ред **M**+2: списък от K цели числа P[0], P[1],..., P[K-1], разделени с интервали.
  - Ред **M**+3: очакваното решение
- Очакван изход за примерния грейдер: grader.expect.1, grader.expect.2, ...
  За тази задача, всеки един от тези файлове трябва да съдържа единствено текста "**Correct.**".