Игра с перестановкой

Алиса и Боб являются друзьями с детства, и они часто играют в различные настольные игры. Сегодня они играют в новую игру на графах.

Игровое поле представляет собой **связный** неориентированный граф с m вершинами, занумерованными последовательными целыми числами от 0 до $m-1$, и e рёбрами, занумерованными последовательными целыми числами от 0 до $e-1$. Ребро с номером i соединяет вершины $u[i]$ и $v[i]$.

Кроме графа, имеется перестановка $p[0], p[1], \ldots, p[n-1]$ длины n, где $m \leq n$. Перестановка -- это произвольный массив, в котором каждое число от 0 до $n-1$ встречается ровно один раз. Определим **оценку** перестановки p как количество таких индексов i, что $p[i] = i$.

Игра длится не более 10^{100} ходов. Каждый ход происходит следующее:

1. Алиса может принять решение остановить игру. В этом случае игра заканчивается.
2. В противном случае Алиса выбирает массив из **различных индексов** $t[0], t[1], \ldots, t[m-1]$, где $0 \le t[i] \lt n. Отметим, что правила игры **не** требуют, чтобы индексы шли возрастанию: **не обязательно** $t[0] < t[1] < \ldots < t[m-1]$.
3. Боб выбирает ребро графа с номером $0 \leq j < e$ и меняет местами $p[t[u[j]]]$ и $p[t[v[j]]]$.

Задача Алисы максимизировать оценку перестановки в конце игры, в то время как задача Боба — минимизировать эту оценку.

Ваша задача — сыграть за Алису, за Боба будет играть грейдер жюри.

Определим *оптимальную оценку* как финальную оценку перестановки в случае, если Алиса и Боб играют оптимально.

Вы должны определить оптимальную оценку перестановки, после чего сыграть против Боба, чтобы достичь после некоторого числа ходов **как минимум** этой оценки.

Обратите внимание, что стратегия Алисы должна работать вне зависимости от того, какие ходы делает Боб, включая в том числе и ситуации, когда Боб играет неоптимально.

Детали реализации

Вы должны реализовать следующую функцию:

```
int Alice(int m, int e, std::vector<int> u, std::vector<int> v,
          int n, std::vector<int> p)
```

- m: количество вершин графа.
- e: количество рёбер графа.
- u и v: массивы длины e, задающие рёбра графа.
- n: длина перестановки.
- p: массив длины n, задающий перестановку.
- Эта функция вызывается один раз.
- Функция должна вернуть целое число — оценку перестановки в конце игры при условии, что Алиса и Боб играют оптимально.

Внутри этой функции, вы должны реализовать следующую функцию:

```
int Bob(std::vector<int> t)
```

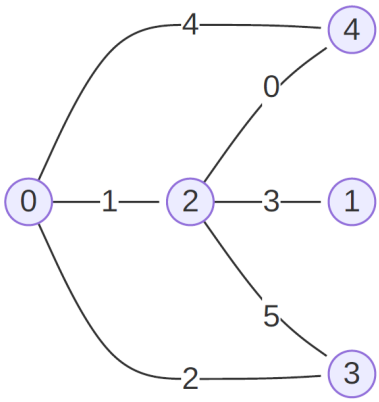
- t: массив длины m, содержащий различные индексы, для которого $0 \le t[i] < n$ и $t[i] \ne t[j]$ для любых $i \ne j$.
- Эта функция возвращает одно целое число j, для которого $0 \le j < m$.
- Эта функция может быть вызвана многократно.

Пример

Рассмотрим следующий вызов:

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

Граф имеет следующий вид:



а перестановка p изначально равна $[8, 2, 7, 6, 1, 5, 0, 9, 3, 4]$.

При заданных выше входных данных можно доказать, что оценка перестановки при оптимальной игре равна 1.

Предположим, что игроки делают следующие 4 хода :

Значение t при вызове `Bob`	`Bob` вернул	Соответствующие индексы p	p после действий Боба
$[3,1,5,2,0]$	5	$5,2$	$[8,2,5,6,1,7,0,9,3,4]$
$[9,3,7,2,1]$	0	$1,7$	$[8,9,5,6,1,7,0,2,3,4]$
$[5,6,7,8,9]$	1	$5,7$	$[8,9,5,6,1,2,0,7,3,4]$
$[7,5,2,3,6]$	3	$5,2$	$[8,9,2,6,1,5,0,7,3,4]$

Обратите внимание, что ходы Алисы и Боба в этой игре не обязательно являются оптимальными, примеры приведены исключительно с целью демонстрации. Также обратите внимание, что Алиса может закончить игру сразу же, так как оценка первоначальной перестановки изначально равна 1.

После того, как были сделаны все ходы, указанные выше, оценка перестановки равна 3 ($p[2]=2, p[5]=5, p[7]=7$).

Функция `Alice` должна вернуть 1 --- оптимальную оценку перестановки.

Заметьте, что, хотя Алиса в данном примере смогла получить оценку 3 при фактической игре против Боба, если вы вернёте 3 вместо 1 как результат вызова `Alice`, вы получите 0 баллов.

Ограничения

- $2 \le m \le 400$
- $m-1 \le e \le 400$
- $0 \le u[i], v[i] < m$
- $m \le n \le 400$
- $0 \le p[i] < n$
- Граф является связным и не содержит петель и кратных рёбер.
- p представляет собой перестановку, то есть. $p[i] \ne p[j]$ для всех $i \ne j$.

Подзадачи

1. (6 баллов) $m = 2$
2. (6 баллов) $e > m$
3. (10 баллов) $e = m - 1$
4. (24 баллов) $e = m = 3$

5. (24 баллов) $e = m = 4$
6. (30 баллов) $e = m$

Для каждой подзадачи возможен частичный балл. Пусть r --- максимальное значение отношения $\frac{k}{n}$ среди всех тестов подзадачи, где k --- количество ходов (то есть вызовов функции `Bob()`). В этом случае балл за эту подзадачу умножается на коэффициент, вычисляемый по следующим правилам:

Условие	Коэффициент
$12 \le r$	0
$3 < r < 12$	$1 - \log_{10}(r-2)$
$r \le 3$	1

В частности, если подзадача решается за $3n$ или менее ходов, вы получаете за неё полный балл, а если решается более, чем за $12n$ ходов, вы получаете за эту подзадачу 0 баллов (и вердикт `Wrong answer`).

Пример грейдера

Выданный вам грейдер считывает входные данные в следующем формате:

- строка 1: $m\ e$
- строка $2+i$ $(0 \le i \le e-1)$: $u[i]\ v[i]$
- строка $2+e$: n
- строка $3+e$: $p[0]\ p[1]\ \ldots\ p[n-1]$

И выводит данные в следующем формате:

- строка 1: итоговая перестановка p
- строка 2: значение, возвращаемое функцией `Alice()`
- строка 3: фактическая оценка итоговой перестановки
- строка 4: количество ходов