Longest Trip

Организаторы IOI 2023 попали в сложное положение! Они забыли спланировать экскурсию в Опусташер на завтра. Но возможно еще не поздно...

В Опусташере находятся N достопримечательностей, пронумерованных от 0 до $N-1$. Некоторые пары этих достопримечательностей соединены *двусторонними* **дорогами**. Каждая пара достопримечательностей соединена не более чем одной дорогой. Организаторы *не знают*, какие пары достопримечательностей соединены дорогами.

Будем говорить, что **плотность** дорожной сети Опусташера **не меньше** δ, если для любых 3 различных достопримечательностей между ними существует хотя бы δ дорог. Другими словами, для каждой тройки достопримечательностей (u, v, w), таких что $0 \leq u < v < w < N$, среди пар достопримечательностей $(u, v), (v, w)$ и (u, w) хотя бы δ пар соединены дорогой.

Организаторы *знают* положительное число D, такое что плотность дорожной сети не меньше D. Обратите внимание, что D не может превышать 3.

Организаторы могут совершать **звонки** по телефону диспетчеру в Опусташере, чтобы собрать информацию о дорогах, соединяющих определенные достопримечательности. Во время каждого звонка они передают два непустых массива достопримечательностей $[A[0], \ldots, A[P-1]]$ и $[B[0], \ldots, B[R-1]]$. Все достопримечательности в массивах различны, а именно

- $A[i] \neq A[j]$ для каждого i и j, таких что $0 \leq i < j < P$;
- $B[i] \neq B[j]$ для каждого i и j, таких что $0 \leq i < j < R$;
- $A[i] \neq B[j]$ для каждого i и j, таких что $0 \leq i < P$ и $0 \leq j < R$.

Для каждого звонка диспетчер сообщает, есть ли дорога от какой-либо достопримечательности из массива A до какой-либо достопримечательности из массива B. А именно, диспетчер перебирает все пары i и j, такие что $0 \leq i < P$ и $0 \leq j < R$. Если для какой-то из них $A[i]$ и $B[j]$ соединены дорогой, диспетчер возвращает `true`. Иначе диспетчер возвращает `false`.

Путешествие длины l — это последовательность *различных* достопримечательностей $t[0], t[1], \ldots, t[l-1]$, где для каждого i от 0 до $l-2$, включительно, достопримечательности

$t[i]$ и $t[i+1]$ соединены дорогой. Путешествие длины l называется **самым длинным путешествием**, если не существует путешествия длиной хотя бы $l+1$.

Ваша задача — помочь организаторам найти самое длинное путешествие, выполняя звонки диспетчеру.

Implementation Details

Вам следует реализовать следующую функцию:

```
int[] longest_trip(int N, int D)
```

- N: количество достопримечательностей в Опусташере.
- D: гарантированная минимальная плотность дорожной сети.
- Функция должна вернуть массив $t = [t[0], t[1], \ldots, t[l-1]]$, содержащий самое длинное путешествие.
- Функция может быть вызвана **несколько раз** в одном тесте.

Эта функция может делать вызовы следующей функции:

```
bool are_connected(int[] A, int[] B)
```

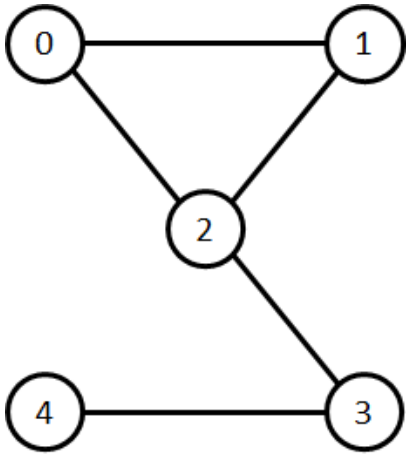
- A: непустой массив различных достопримечательностей.
- B: непустой массив различных достопримечательностей.
- A и B не должны иметь общих элементов.
- Эта функция возвращает `true`, если есть достопримечательность в массиве A и достопримечательность в массиве B, соединенные дорогой. Иначе она возвращает `false`.
- Эту функцию можно вызать не более $32\,640$ раз в каждом вызове `longest_trip`, а также суммарно не более $150\,000$ раз.
- Суммарная длина массивов A и B, переданных этой функции во всех вызовах, не должна превышать $1\,500\,000$.

Грейдер в этой задаче **не адаптивный**. Каждое решение тестируется на одном и том же наборе тестов. При этом в каждом тесте значения N и D, а также все пары достопримечательностей, соединенных дорогами, зафиксированы до того, как выполняется соответствующий вызов `longest_trip`.

Examples

Example 1

Рассмотрим сценарий, в котором $N = 5$, $D = 1$, и дороги соединяют достопримечательности, как показано на следующем рисунке:



Функция `longest_trip` вызывается следующим образом:

```
longest_trip(5, 1)
```

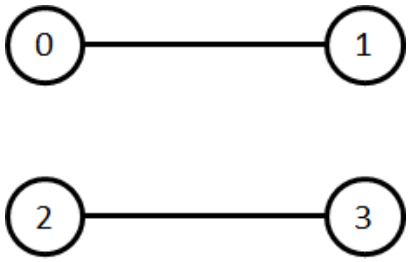
Эта функция может, например, вызывать функцию `are_connected` следующим образом.

Вызов	Пары, соединенные дорогой	Возвращенное значение
`are_connected([0], [1, 2, 4, 3])`	$(0, 1)$ и $(0, 2)$	`true`
`are_connected([2], [0])`	$(2, 0)$	`true`
`are_connected([2], [3])`	$(2, 3)$	`true`
`are_connected([1, 0], [4, 3])`	нет	`false`

В результате четвертого вызова выясняется, что *ни одна* пара из $(1, 4)$, $(0, 4)$, $(1, 3)$ и $(0, 3)$ не соединена дорогой. Поскольку плотность дорожной сети не меньше $D = 1$, можно сделать вывод, что в тройке $(0, 3, 4)$ пара $(3, 4)$ должна быть соединена дорогой. Аналогично достопримечательности 0 и 1 должны быть соединены дорогой.

В этот момент можно сделать вывод, что $t = [1, 0, 2, 3, 4]$ — это путешествие длины 5, и что никакое путешествие не имеет длину больше 5. Следовательно функция `longest_trip` может вернуть $[1, 0, 2, 3, 4]$.

Рассмотрим еще один сценарий, где $N = 4$, $D = 1$, и дороги между достопримечательностями показаны на следующем рисунке:



Функция `longest_trip` будет вызывана следующим образом:

```
longest_trip(4, 1)
```

В этом сценарии длина самого длинного путешествия равна 2. Таким образом, сделав те или иные вызовы функции `are_connected`, функция `longest_trip` может вернуть один из массивов $[0, 1]$, $[1, 0]$, $[2, 3]$ или $[3, 2]$.

Example 2

Подзадача 0 содержит дополнительный пример с $N = 256$ достопримечательностями. Этот тест можно скачать в приложении к задаче из тестирующей системы.

Constraints

- $3 \leq N \leq 256$
- Сумма значений N по всем вызовам `longest_trip` не превышает $1\,024$ в каждом тесте.
- $1 \leq D \leq 3$

Subtasks

1. (5 балла) $D = 3$
2. (10 баллов) $D = 2$
3. (25 баллов) $D = 1$. Пусть $l^\star$ обозначает длину самого длинного путешествия. Функция `longest_trip` не обязательно должна вернуть путешествие длины $l^\star$. Вместо этого она должна вернуть путешествие длиной не меньше $\left\lceil \frac{l^\star}{2} \right\rceil$.
4. (60 баллов) $D = 1$

В подзадаче 4 ваши баллы зависят от числа вызовов функции `are_connected` в одном вызове `longest_trip`. Пусть q равно максимальному количеству вызовов `are_connected` среди всех вызовов функции `longest_trip` по всем тестам этой подзадачи. Ваш балл за эту подзадачу определяется по следующей таблице:

Условие	Баллы
$2\,750 < q \leq 32\,640$	20
$550 < q \leq 2\,750$	30
$400 < q \leq 550$	45
$q \leq 400$	60

Если в одном из тестов вызовы функции `are_connected` не соответствуют ограничениям, указанным в разделе Implementation Details, или возвращенный функцией `longest_trip` массив неверный, баллы за соответствующую подзадачу равны 0.

Sample Grader

Пусть C обозначает число сценариев в тесте, иначе говоря, число вызовов функции `longest_trip`. Грейдер читает данные в следующем формате:

- строка 1: C

Затем следует C описаний сценариев.

Грейдер читает описание сценария в следующем формате:

- строка 1: $N\ D$
- строка $1 + i$ ($1 \leq i < N$): $U_i[0]\ U_i[1]\ \ldots\ U_i[i-1]$

Здесь каждое U_i ($1 \leq i < N$) это массив размера i, задающий пары достопримечательностей, соединенных дорогами. Для каждого i и каждого j, таких что $1 \leq i < N$ и $0 \leq j < i$:

- если достопримечательности j и i соединены дорогой, значение $U_i[j]$ равно 1;
- если достопримечательности j и i не соединены дорогой, значение $U_i[j]$ равно 0.

В каждом сценарии перед вызовом `longest_trip`, грейдер проверяет, что плотность дорожной сети не менее D. Если это условие не выполнено, он выводит сообщение `Insufficient Density` и завершает работу.

Если грейдер замечает нарушение протокола взаимодействия, он выводит `Protocol Violation: <MSG>`, где `<MSG>` — одно из следующих сообщений:

- `invalid array`: в вызове `are_connected`, хотя бы один из массивов A или B
  - пустой, или
  - содержит элементы, которые не являются целыми числами от 0 до $N-1$, включительно, или
  - содержит один и тот же элемент хотя бы дважды.

- `non-disjoint arrays`: в вызове `are_connected` массивы A и B содержат общий элемент.
- `too many calls`: число вызовов `are_connected` превышает $32\,640$ в текущем вызове `longest trip`, или суммарно превышает $150\,000$.
- `too many elements`: суммарное число достопримечательностей, переданных всем вызовам `are_connected`, превышает $1\,500\,000$.

Иначе, пусть элементы массива, которые вернул вызов `longest_trip` в этом сценарии, равны $t[0], t[1], \ldots, t[l-1]$ для некоторого неотрицательного l. Грейдер выводит три строки в следующем формате:

- строка 1: l
- строка 2: $t[0]\ t[1]\ \ldots\ t[l-1]$
- строка 3: число вызовов `are_connected` в этом сцерании

В конце грейдер выводит:

- строка $1 + 3 \cdot C$: максимальное число вызовов `are_connected` по всем вызовам `longest_trip`