Най-дълъг път

Организаторите на IOI 2023 са в тежка ситуация. Те са забравили да планират екскурзията до Опустасер за следващия ден. Може би все пак не е твърде късно ...

В Опустасер има N забележителности, индексирани с числата от 0 до $N-1$. Някои двойки от забележителностите са свързани с *двупосочни* **шосета**. Всяка двойка забележителности е свързана с най-много едно шосе. Организаторите *не знаят* кои забележителности са свързани с шосета.

Ще казваме, че **гъстотата** на пътната мрежа в Опустасер е **поне** δ, ако всеки 3 различни забележителности имат поне δ шосета помежду си. С други думи, за всяка тройка от забележителности (u, v, w), такива че $0 \leq u < v < w < N$, измежду двойките (u, v), (v, w) и (u, w) поне δ двойки са свързани с шосе.

Организаторите *знаят* положително число D, такова че гъстотата на пътната мрежа е поне D. Забележете, че стойността на D не може да бъде по-голяма от 3.

Организаторите могат да правят **обаждания** към диспечера в Опустасер, за да си набавят информация за шосетата между забележителностите. За всяко обаждане организаторите избират два непразни масива от забележителности $[A[0], \ldots, A[P-1]]$ и $[B[0], \ldots, B[R-1]]$. Всички забележителности измежду двата масива трябва да са различни. Формално:

- $A[i] \neq A[j]$ за всяко i и j, такива че $0 \leq i < j < P$;
- $B[i] \neq B[j]$ за всяко i и j, такива че $0 \leq i < j < R$;
- $A[i] \neq B[j]$ за всяко i и j, такива че $0 \leq i < P$ и $0 \leq j < R$.

При всяко обаждане диспечера отговаря дали съществува шосе, свързващо някоя забележителност от A с някоя забележителност от B. По-специфично, диспечерът не може да отговори по-бързо от това да итерира през всяка двойка i и j, такива че $0 \leq i < P$, $0 \leq j < R$. Ако за някоя от двойките, $A[i]$ и $B[i]$ са свързани с шосе, то диспечерът отговаря `true`. В противен случай, диспечерът отговаря `false`.

Път с дължина l е редица от *различни* забележителности $t[0], t[1], \ldots, t[l-1]$, като за всяко i между 0 и $l-2$ (включително) забележителности $t[i]$ и $t[i+1]$ са свързани с шосе. Път с дължина l наричаме **най-дълъг път**, ако не съществува път с дължина $l+1$.

Вашата задача е да помогнете на организаторите да намерят най-дългия път в Опустасер като правите обаждания към диспечера.

Детайли по имплементацията

Трябва да имплементирате следната функция:

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

- N: броят забележителности в Опустасер.
- D: гарантираната минимална гъстота на пътната мрежа.
- Функцията трябва да върне масив $t = [t[0], t[1], \ldots, t[l-1]]$, описващ най-дългия път.
- Функцията може да бъде викана **повече от веднъж** в рамките на един тест.

Горната функция може да прави извиквания към следната функция:

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

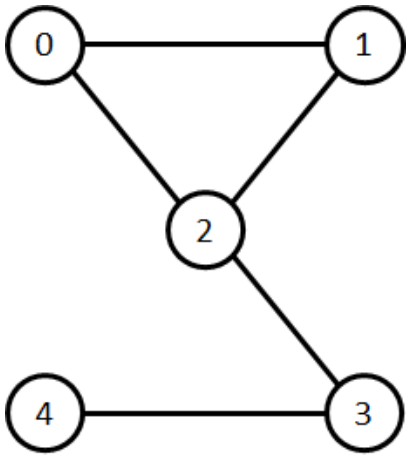
- A: непразен масив от различни забележителности.
- B: непразен масив от различни забележителности.
- A и B не трябва да имат общи елементи.
- Функцията връща `true`, ако съществува забележителност в A и забележителност в B, които са свързани с шосе. В противен случай връща `false`.
- Функцията може да бъде викана най-много $32\,640$ пъти за всяко едно извикване на `longest_trip`, и най-много $150\,000$ пъти общо.
- Общата дължина на масивите A и B, подадени на тази функция в рамките на всички извиквания, трябва да е най-много $1\,500\,000$.

Грейдърът **не е адаптивен**. Всеки събмит бива тестван на едни и същи тестове. Стойностите на N и D, както и двойките забележителности свързани с шосе, са фиксирани за всяко извикване към `longest_trip` във всеки тест.

Примери

Пример 1

Да разгледаме случай, в който $N = 5$, $D = 1$, и пътната мрежа е показана на фигурата:



Функцията `longest_trip` бива извикана по следния начин:

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

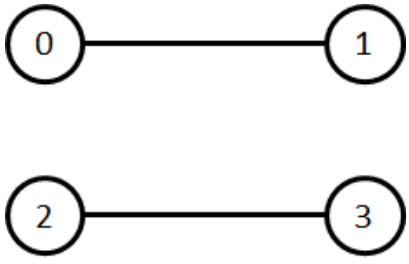
Функцията може да направи следните извиквания към `are_connected`.

Извикване	Двойки свързани с шосе	Върната стойност
`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])`	няма	`false`

След четвъртото извикване се разбира, че никои от двойките $(1, 4)$, $(0, 4)$, $(1, 3)$ и $(0, 3)$ не са свързани с шосе. Тъй като гъстотата на мрежата е поне $D = 1$, можем да заключим от тройката $(0, 3, 4)$, че двойката $(3, 4)$ със сигурност е свързана с шосе. С подобни разсъждения излиза, че забележителности 0 и 1 също са свързани с шосе.

В този момент може да се заключи, че $t = [1, 0, 2, 3, 4]$ е път с дължина 5, и че не съществува път с дължина по-голяма от 5. Съответно, функцията `longest_trip` може да върне $[1, 0, 2, 3, 4]$.

Да разгледаме друг случай, в който $N = 4$, $D = 1$, и шосетата са показани на фигурата:



Функцията `longest_trip` бива извикана по следния начин:

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

В този случай дължината на най-дългия път е 2. Съответно, след няколко извиквания на функцията `are_connected`, функцията `longest_trip` може да върне, който и да е от масивите $[0, 1]$, $[1, 0]$, $[2, 3]$ или $[3, 2]$.

Пример 2

Подзадача 0 съдържа допълнителен примерен тест с $N = 256$ забележителности. Този тест е включен в архива към задачата, който може да изтеглите от системата.

Ограничения

- $3 \leq N \leq 256$
- Сумата на N за всички извиквания на `longest_trip` не надхвърля $1\,024$ в рамките на един тест.
- $1 \leq D \leq 3$

Подзадачи и оценяване

1. (5 точки) $D = 3$
2. (10 точки) $D = 2$
3. (25 точки) $D = 1$. Нека $l^\star$ обозначава дължината на най-дългия път. Не е нужно функцията `longest_trip` да върне път с дължина $l^\star$. Вместо това, функцията може да върне, който и да е път с дължина поне $\left\lceil \frac{l^\star}{2} \right\rceil$
4. (60 точки) $D = 1$

В подзадача 4, точките Ви се определят на база на броя извиквания на функцията `are_connected` в рамките на едно извикване на `longest_trip`. Нека q е максималният брой извиквания на `are_connected` измежду всички извиквания на `longest_trip` на всички тестове от подзадачата. Точките ви за тази подзадача ще се смятат спрямо следната таблица:

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

Ако в който и да е от тестовете има извикване на функцията `are_connected`, което не отговаря на ограниченията, описани в секция Детайли по имплементацията, или масивът

върнат от `longest_trip` не е верен, то точките на вашето решение за тази подзадача ще бъдат 0.

Локално тестване

Нека C е броят случаи, които се разглеждат в дадения тест. С други думи, C е броят извиквания на `longest_trip` в рамките на теста. Локалният грейдър чете вход в следния формат:

- ред 1: C

Следват описанията на всеки от C-те случая.

Локалният грейдър чете всяко описание в следния формат:

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

Тук всяко U_i ($1 \le i < N$) е масив с дължина i, описващ кои двойки забележителности са свързани с шосе. За всяко i и j, такива че $1 \le i < N$ и $0 \le j < i$:

- ако забележителности j и i са свързани с шосе, то стойността $U_i[j]$ ще е 1;
- ако няма шосе, свързващо забележителности j и i, то стойността $U_i[j]$ ще е 0.

Преди да се извика `longest_trip`, локалният грейдър проверява дали гъстотата на пътната мрежа за дадения случай е поне D. Ако това не е изпълнено, то грейдърът принтира `Insufficient Density` и приключва.

Ако локалният грейдър открие нарушение на някое ограничение, то изходът на грейдъра е `Protocol Violation: <MSG>`, където `<MSG>` е една от следните грешки:

- `invalid array`: при извикване на `are_connected` поне един от масивите A и B
  - е празен, или
  - съдържа елемент който не е цяло число между 0 и $N-1$, включително, или
  - съдържа повторящ се елемент
- `non-disjoint arrays`: при извикване на `are_connected` масивите A и B имат поне един общ елемент
- `too many calls`: броят извиквания на `are_connected` надвишава $32\,640$ в рамките на текущото извикване на `longest_trip` или надвишава общо $150\,000$ извиквания в рамките на целия тест.
- `too many elements`: общият брой забележителности подадени на `are_connected` измежду всички извиквания за текущия тест надвишава $1\,500\,000$.

Ако локалният грейдър не открие нарушение, то нека масивът върнат от `longest_trip` за даден случай е $t[0], t[1], \ldots, t[l-1]$ за някое неотрицателно число l. Грейдърът извежда три

реда за този случай в следния формат:

- ред 1: l
- ред 2: $t[0]$ $t[1]$ $\ldots$ $t[l-1]$
- ред 3: броя извиквания към `are_connected` в рамките на този случай

Финално грейдърът извежда:

- ред $1 + 3 \cdot C$: максималния брой извиквания на `are_connected` измежду всички случаи