Найдовший маршрут

В організаторів IOI 2023 проблема! Вони забули організувати мандрівку до Національного парку спадщини Опускасер, яка має відбутися завтра. Можливо ще не пізно...

У парку N пам'яток пронумерованих від 0 до $N-1$. Деякі пари з'єднані *двосторонніми* **дорогами**. Кожна пара з'єднана не більше однією дорогою. Організатори *не знають* які пам'ятки з'єднані дорогою.

Вважаємо, що **щільність** мережі доріг у парку **принаймні** δ, якщо між кожними 3 різними пам'ятками є принаймні δ доріг, що їх сполучають. Іншими словами, для кожної трійки пам'яток (u, v, w) таких, що $0 \leq u < v < w < N$, серед пар пам'яток $(u, v), (v, w)$ та (u, w) принаймні δ пар з'єднані дорогою.

Організатори *знають* таке додатне ціле число D, що щільність мережі доріг принаймні D. Зауважимо, що значення D не може перевищувати 3.

Організатори можуть робити **виклики** на телефон диспетчера парку для збору інформації про шляхи між конкретними пам'ятками. У кожному виклику, мають визначатись два непорожні масиви пам'яток $[A[0], \ldots, A[P-1]]$ і $[B[0], \ldots, B[R-1]]$. Пам'ятки мають бути попарно різними, тобто,

- $A[i] \neq A[j]$ для всіх i і j таких, що $0 \leq i < j < P$;
- $B[i] \neq B[j]$ для всіх i і j таких, що $0 \leq i < j < R$;
- $A[i] \neq B[j]$ для всіх i і j таких, що $0 \leq i < P$ і $0 \leq j < R$.

Для кожного виклику диспетчер повідомляє чи пам'ятки з A та з B з'єднані дорогою. Тобто, диспетчер повертає `true`, якщо існують i та j такі, що $0 \leq i < P$ та $0 \leq j < R$, і $A[i]$ та $B[j]$ з'єднані дорогою. Інакше, диспетчер повертає `false`.

Маршрут довжиною l - це послідовність *різних* пам'яток $t[0], t[1], \ldots, t[l-1]$, де для всіх i від 0 до $l-2$ включно, пам'ятки $t[i]$ та $t[i+1]$ з'єднані дорогою. Маршрут довжиною l називається **найдовшим маршрутом**, якщо не існує жодного маршруту, довжиною принаймні $l+1$.

Ваше завдання - допомогти організаторам знайти найдовший маршрут у парку за допомогою викликів диспетчера.

Деталі імплементації

Вам потрібно імплементувати наступну функцію:

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

- N: кількість пам'яток в парку.
- D: гарантована мінімальна щільність дорожньої мережі.
- Ця функція повинна повернути масив $t = [t[0], t[1], \ldots, t[l-1]]$, який позначає найдовший маршрут.
- Цю функцію можуть викликати **кілька разів** для кожного тесту.

Описана вище функція може викликати наступну функцію:

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

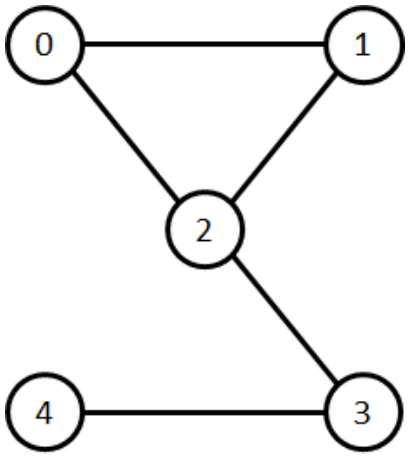
- A: непорожній масив різних пам'яток.
- B: непорожній масив різних пам'яток.
- A і B не мають перетинатися.
- Ця функція повертає `true`, якщо є пам'ятка з A та пам'ятка з B, які з'єднані дорогою. Інакше, вона повертає `false`.
- Ця функція може викликатись не більше $32\,640$ разів при кожному виклику `longest_trip`, а також не більше ніж $150\,000$ разів в сумі.
- Сумарна довжина масивів A і B, що передається до функції при всіх її викликах не може перевищувати $1\,500\,000$.

Градер **не адаптивний**. Величини N і D, як і пари пам'яток, з'єднаних дорогами, зафіксовані до викликів `longest_trip`.

Приклади

Приклад 1

Розглянемо сценарій, в якому $N = 5$, $D = 1$, і мережа доріг має наступний вигляд:



Функція `longest_trip` викликається наступним чином:

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

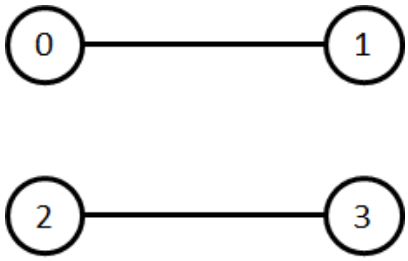
Функція має робити виклики `are_connected` наступним чином.

Виклик	Пари, з'єднані дорогою	Значення
`are_connected([0], [1, 2, 4, 3])`	$(0,1)$ і $(0,2)$	`true`
`are_connected([2], [0])`	$(2,0)$	`true`
`are_connected([2], [3])`	$(2,3)$	`true`
`are_connected([1, 0], [4, 3])`	жодні	`false`

Після четвертого виклику виявляється, що *жодна* з пар $(1,4)$, $(0,4)$, $(1,3)$ і $(0,3)$ не з'єднана дорогою. Оскільки щільність мережі доріг принаймні $D = 1$, бачимо, що з трійки $(0,3,4)$, пара $(3,4)$ повинна бути з'єднаною дорогою. Аналогічно мають бути з'єднані і 0 та 1.

Таким чином можна зробити висновок, що $t = [1,0,2,3,4]$ - маршрут довжиною 5, і що не існує маршруту, довжиною більше 5. Таким чином функція `longest_trip` повинна повернути $[1,0,2,3,4]$.

Розглянемо інший сценарій, в якому $N = 4$, $D = 1$, мережа доріг має такий вигляд:



Функція `longest_trip` викликається наступним чином:

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

У цьому сценарії довжина найбільшого маршруту становить 2. Тому після кількох викликів функції `are_connected`, функція `longest_trip` може повернути один із наступних варіантів: $[0,1]$, $[1,0]$, $[2,3]$ або $[3,2]$.

Приклад 2

Підзадача 0 містить додатковий тест з $N = 256$. Цей тест включено до вкладення, яке можна завантажити із тестувальної системи.

Обмеження

- $3 \leq N \leq 256$
- Сума N на всіх викликах `longest_trip` не перевищує $1\,024$.
- $1 \leq D \leq 3$

Підзадачі

1. (5 балів) $D = 3$
2. (10 балів) $D = 2$
3. (25 балів) $D = 1$. Нехай $l^\star$ позначає довжину найдовшого маршруту. Функція `longest_trip` не забов'язана повернути маршрут довжиною $l^\star$. Натомість вона повинна повернути маршрут довжини принаймні $\left\lceil \frac{l^\star}{2} \right\rceil$.
4. (60 балів) $D = 1$

Якщо в процесі тестування функція `are_connected` буде викликана з некоректними параметрами, або масив, що повернула `longest_trip` невірний - ваш розв'язок невірний і отримає 0 балів.

У підзадачі 4 ваш бал визначається з врахуванням кількості викликів функції `are_connected` під час одного виклику `longest_trip`. Нехай q - максимальне число викликів серед всіх викликів `longest_trip` серед всіх тестів підзадачі. Ваш бал за підзадачу обраховується у відповідності до таблички:

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

Приклад градера

Нехай C позначає кількість сценаріїв, що є кількістю викликів `longest_trip`. Градер зчитує вхідні дані у наступному форматі:

- рядок 1: C

Розглянемо опис C сценаріїв.

Градер зчитує опис кожного сценарію в наступному форматі:

- рядок 1: N D

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

Тут, кожен U_i ($1 \leq i < N$) - масив довжини i, що показує, які пари пам'яток з'єднані дорогою. Для всіх i та j таких, що $1 \leq i < N$ і $0 \leq j < i$:

- якщо пам'ятки j та i з'єднані дорогою, то значення $U_i[j]$ повинно бути 1;
- якщо немає дороги, що сполучає j та i, то значення $U_i[j]$ повинно бути 0.

У кожному сценарію до викликів `longest_trip`, градер перевіряє, чи щільність мережі доріг дорівнює принаймні D. Якщо ці умови не виконані, він виводить повідомлення `Insufficient Density` і вимикається.

Якщо градер помічає порушення протоколу він виводить `Protocol Violation: <MSG>`, де `<MSG>` - один з наступних повідомлень:

- `invalid array`: при виклику `are_connected`, принаймні один з масивів A і B
  - порожній, або
  - містить елементи, що не є цілими числами в межах між 0 та $N - 1$, включно, або
  - містить однакові елементи принаймні двічі.
- `non-disjoint arrays`: при виклику `are_connected`, масиви A і B перетинаються.
- `too many calls`: кількість викликів `are_connected` перевищує $32\,640$ під час поточного виклику `longest trip`, або $150\,000$ загалом.
- `too many elements`: загальна кількість пам'яток переданих до `are_connected` за всі виклики перевищує $1\,500\,000$.

В іншому випадку, нехай елементи масива, що повернула `longest_trip` під час сценарію будуть $t[0], t[1], \ldots, t[l - 1]$ для деякого невід'ємного l. Градер виводить три рядки для цього сценарію у наступному форматі:

- рядок 1: l
- рядок 2: $t[0]$ $t[1]$ $\ldots$ $t[l - 1]$
- рядок 3: кількість викликів `are_connected` в межах цього сценарію

Нарешті градер виводить:

- рядок $1 + 3 \cdot C$: максимальна кількість викликів `are_connected` серед усіх викликів `longest_trip`