Nejdelší výlet

Organizátoři IOI 2023 jsou ve velké šlamastice! Zapomněli naplánovat zítřejší výlet do Ópusztaszeru. Ale možná není ještě příliš pozdě . . .

V Ópusztaszeru je N památek očíslovaných od 0 do $N-1$. Některé dvojice těchto památek jsou propojeny *oboustrannými* **silnicemi**. Každá dvojice památek je propojena nejvýše jednou silnicí. Organizátoři *neví*, které památky jsou propojeny silnicemi.

Říkáme, že **hustota** dopravní sítě v Ópusztaszeru je **alespoň** δ, pokud každé 3 památky mají alespoň δ silnic mezi sebou. Formálně pro každou trojici památek (u, v, w) $(0 \leq u < v < w < N)$ platí, že mezi páry památek $(u, v), (v, w)$ a (u, w) je alespoň δ párů propojených silnicí.

Organizátoři *ví* kladné číslo D takové, že hustota dopravní sítě je alespoň D. Všimněte si, že hodnota D nemůže být větší než 3.

Organizátoři můžou provádět **volání** na telefonní ústřednu Ópusztaszeru, aby získali informace o silnicích mezi památkami. V každém volání je zapotřebí poskytnout dvě neprázdná pole památek $[A[0], \ldots, A[P-1]]$ a $[B[0], \ldots, B[R-1]]$. Památky musí být po dvou rozdílné:

- $A[i] \neq A[j]$ pro každé i a j, pro které platí $0 \leq i < j < P$;
- $B[i] \neq B[j]$ pro každé i a j, pro které platí $0 \leq i < j < R$;
- $A[i] \neq B[j]$ pro každé i a j, pro které platí $0 \leq i < P$ and $0 \leq j < R$.

Pro každé volání, dispečeř řekne, zdali existuje silnice propojující památku z A a památku z B. To znamená, dispečer vrátí `true` pokud existuje i a j, tak že existuje silnice mezi $A[i]$ a $B[j]$. Jinak vrátí `false`.

Výlet délky l je posloupnost *různých* památek $t[0], t[1], \ldots, t[l-1]$, kde pro každé i mezi 0 a $l-2$ (včetně) památky $t[i]$ a $t[i+1]$ jsou propojeny silnicí. Výlet délky l je **nejdelší**, pokud neexistuje žádný výlet délky alespoň $l+1$.

Vaším úkolem je pomoct organizátorům najít nejdelší výlet v Ópusztaszeru potom, co (opakovaně) zavoláte dispečerovi.

Implementační detaily

Měli byste implementovat následující funkci:

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

- N: Počet památek v Ópusztaszeru.
- D: Slíbená minimální hustota dopravní sítě.
- Funkce by měla vrátit pole $t = [t[0], t[1], \ldots, t[l-1]]$: nejdelší výlet.
- Funkce může být zavolána **vícekrát** v každém vstupu.

Funkce výše může volat tuto funkci:

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

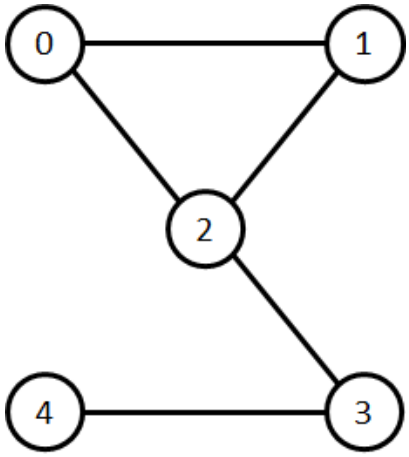
- A: Neprázdné pole různých památek.
- B: Neprázdné pole různých památek.
- Prvky polí A a B jsou disjunktní.
- Funkce vrátí `true` pokud existtuje památka z A a památka z B propojené silnicí. Jinak, vrátí `false`.
- Funkce může být volána nejvýše $32\,640$-krát v každém volání `longest_trip`, a nejvýše $150\,000$ celkem.
- Celková délka polí A a B předaná této funkci přes všechna volání nesmí převýšit $1\,500\,000$.

Grader **není adaptivní**. Hodnoty N a D a stejně tak páry památek propojené silnicemi jsou zafixované předtím, než je `longest_trip` zavolán.

Příklady

Příklad 1

Zvažte případ, kde $N = 5$, $D = 1$ a silnice jsou podle obrázku:



Funkce `longest_trip` je volána následujícím způsobem:

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

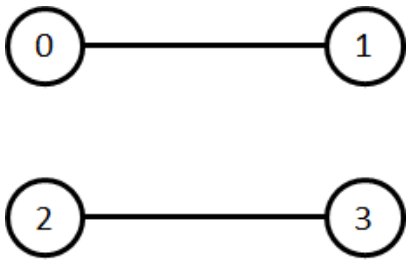
Tato funkce volá funkci `are_connected` takto:

Call	Páry propojené silnicí	Návratová hodnota
`are_connected([0], [1, 2, 4, 3])`	$(0,1)$ a $(0,2)$	`true`
`are_connected([2], [0])`	$(2,0)$	`true`
`are_connected([2], [3])`	$(2,3)$	`true`
`are_connected([1, 0], [4, 3])`	žádné	`false`

Po čtvrtém volání se ukáže, že žádné z párů $(1,4)$, $(0,4)$, $(1,3)$, $(0,3)$ nejsou propojené silnicí. Protože hustota sítě je alespoň $D = 1$, tak víme, že z trojice $(0,3,4)$, pár $(3,4)$ musí být propojen silnicí. Obdobně, památky 0 a 1 musí být propojeny.

Nyní lze dospět k závěru, že $t = [1,0,2,3,4]$ je výlet délky 5 a zároveň neexistuje delší výlet. Proto funkce `longest_trip` může vrátit $[1,0,2,3,4]$.

Zvažte další případ, ve kterém $N = 4$, $D = 1$, a silnice mezi památkami jsou podle obrázku:



Funkce `longest_trip` je zavolána následujícím způsobem:

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

V tomto případě, délka nejdelšího výletu je 2. Proto, po několika volání funkce `are_connected`, funkce `longest_trip` může vrátit jedno z $[0,1]$, $[1,0]$, $[2,3]$ or $[3,2]$.

Příklad 2

Podúloha 0 obsahuje další vstup s $N = 256$ památkami. Tento vstup najdete na stránce s přílohami v soutěžním systému.

Omezení

- $3 \leq N \leq 256$
- Součet N přes všechna volání `longest_trip` nepřesáhne $1\,024$.
- $1 \leq D \leq 3$

Podúlohy

1. (5 bodů) $D = 3$
2. (10 bodů) $D = 2$
3. (25 bodů) $D = 1$. Nechť $l^\star$ značí délku nejdelšího výletu. Funkce `longest_trip` nemusí vrátit výlet délky $l^\star$. Stačí když vrátí výlet délky alespoň $\left\lceil \frac{l^\star}{2} \right\rceil$.
4. (60 bodů) $D = 1$

Když ve kterémkoliv vstupu volání funkce `are_connected` neodpovídá omezením popsaným v implementačních detailech, nebo pole vrácené `longest_trip` je nesprávné, skóre vašeho řešení pro danou podúlohu bude 0.

V podúloze 4 vaše skóre je určeno na základě počtu volání funkce `are_connected` během jednoho spuštění `longest_trip`. Nechť q je maximální počet volání přes všechna spuštění `longest_trip` přes všechny vstupy v podúloze 4. Skóre se pak určí dle následující tabulky:

Podmínka	Počet bodů
$2\,750 < q \leq 32\,640$	20
$550 < q \leq 2\,750$	30
$400 < q \leq 550$	45
$q \leq 400$	60

Ukázkový Grader

Nechť C značí počet scénářů, tedy počet volání `longest_trip`. Grader čte vstup v následujícím formátu:

- řádek 1: C

Následuje C scénářů v následujícím formátu:

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

Kde U_i ($1 \leq i < N$) je seznam délky i popisující které dvojice památek jsou spojeny cestou. Pro každé přípustné i, j:

- Pokud jsou památky j a i spojeny cestou, hodnota $U_i[j]$ má být 1;
- V opačném případě 0

V každém scénáři před zavoláním `longest_trip` grader zkontroluje, že hustota dopravní sítě je alepsoň D. Pokud ne, vypíše `Insufficient Density` a skončí.

V případě porušení protokolu je výstup graderu `Protocol Violation: <MSG>`, kde `<MSG>` je jedna z následujících:

- `invalid array`: při volání `are_connected`, alespoň jedno z A a B
  - je prázdné, nebo
  - obsahuje prvek mimo interval 0 až $N-1$, nebo
  - obsahuje nějaký prvek více než jednou
- `non-disjoint arrays`: při volání `are_connected`, pole A a B obsahují stejný prvek.
- `too many calls`: počet volání `are_connected` překročil $32\,640$ v rámci jednoho spuštění `longest trip`, nebo $150\,000$ celkem.
- `too many elements`: celkový počet památek poslaný `are_connected` v jednom spuštění překročil $1\,500\,000$.

Pokud vše půjde dobře, označme návratovou hodnotu `longest_trip` ve scénáři jako $t[0], t[1], \ldots, t[l-1]$ pro nějaké l. Grader vypíše:

- řádek 1: l
- řádek 2: $t[0]\ t[1]\ \ldots\ t[l-1]$
- řádek 3: počet volání `are_connected` v daném scénáři

Nakonec grader vypíše:

- řádek $1 + 3 \cdot C$: maximální počet zavolání `are_connected` přes všechna spuštění `longest_trip`