Longest Trip

De IOI 2023 organisatie zit flink in de problemen! Men is vergeten een trip naar Ópusztaszer te plannen voor de volgende dag. Maar misschien is het nog niet te laat...

Er zijn n bezienswaardigheden in Ópusztaszer genummerd vanaf 0 tot en met $N-1$. Sommige paren bezienswaardigheden zijn verbonden met *tweerichtingsverkeer*-**wegen**. Elk paar bezienswaardigheden is verbonden via hoogstens één weg. De organisatie *weet niet* welke bezienswaardigheden met elkaar verbonden zijn via wegen.

We zeggen dat de **dichtheid** van het wegennetwerk van Ópusztaszer **minstens** δ is, als voor elke 3 verschillende bezienswaardigen er minstens δ wegen tussen liggen. Met andere woorden, voor elk drietal van bezienswaardigheden (u, v, w) dat voldoet aan $0 \leq u < v < w < N$, zijn er onder de paren $(u, v), (v, w)$ en (u, w) minstens δ paren die zijn verbonden via een weg.

De organisatie *kent* een positief geheel getal D zodanig dat de dichtheid van het wegennetwerk minstens D is. Merk op dat de waarde van D niet meer dan 3 kan zijn.

De organisatie kan de telefooncentrale van Ópusztaszer **aanroepen** om informatie te verzamelen over de wegenverbindingen tussen sommige bezienswaardigheden. Voor elke aanroep moet je twee niet-lege arrays van bezienswaardigheden $[A[0], \ldots, A[P-1]]$ en $[B[0], \ldots, B[R-1]]$ specificeren. De bezienswaardigheden moeten paarsgewijs verschillend zijn, oftewel,

- $A[i] \neq A[j]$ voor alle i en j waarbij $0 \leq i < j < P$;
- $B[i] \neq B[j]$ voor alle i en j waarbij $0 \leq i < j < R$;
- $A[i] \neq B[j]$ voor alle i en j waarbij $0 \leq i < P$ en $0 \leq j < R$.

Bij elke aanroep geeft de telefooncentrale aan of er een weg een bezienswaardigheid uit A verbindt met een bezienswaardigheid uit B. Specifieker, de telefooncentrale loopt over alle paren i en j zodanig dat $0 \leq i < P$ en $0 \leq j < R$. Als voor één ervan de bezienswaardigheden $A[i]$ en $B[j]$ verbonden zijn via een weg, dan geeft de telefooncentrale `true` terug. Zoniet, geeft het `false` terug.

Een **trip** van lengte l is een rij van *verschillende* bezienswaardigheden $t[0], t[1], \ldots, t[l-1]$, waarbij voor elke i vanaf 0 tot en met $i-2$, de bezienswaardigheden $t[i]$ is verbonden via een weg met $t[i+1]$. Een trip van lengte l is een zogenaamde **langste trip** als er geen trip bestaat van lengte minstens $l+1$.

Jouw taak is de organisatie te helpen met het vinden van de langste trip in Ópusztaszer door de telefooncentrale aan te roepen.

Implementatiedetails

Je moet de volgende functie implementeren:

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

- N: het aantal bezienswaardigheden in Ópusztaszer.
- D: de gegarandeerde minimale dichtheid van het wegennetwerk.
- Deze procedure moet een array teruggeven $t = [t[0], t[1], \ldots, t[l-1]]$, dat de langste trip representeert.
- Deze procedure kan **meerdere keren** aangeroepen worden binnen één enkele testcase.

De procedure hierboven kan de volgende functie aanroepen:

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

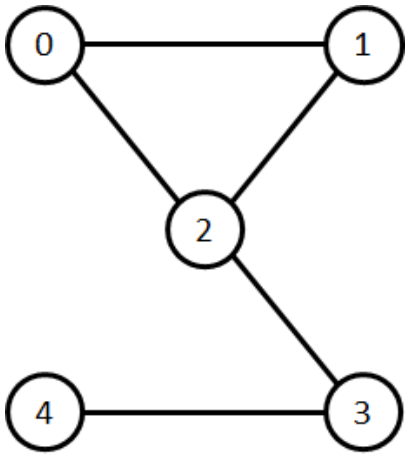
- A: een niet-lege array van verschillende bezienswaardigheden.
- B: een niet-lege array van verschillende bezienswaardigheden.
- A en B moeten niet overlappen.
- Deze functie geeft `true` terug als er een bezienswaardigheid uit A en een bezienswaardigheid uit B zijn verbonden via een weg. Zoniet, geeft de functie `false` terug.
- De functie kan maar maximaal $32\,640$ keer aangeroepen worden in elke aanroep van `longest_trip`, en in totaal hoogstens $150\,000$ keer.
- De totale lengte van de arrays A en B die aan deze functie meegegeven worden, mag in totaal over alle aanroepen niet meer dan $1\,500\,000$ zijn.

De grader is **niet adaptief**. Elke inzending wordt beoordeeld op dezelfde verzameling testcases. Oftewel, de waardes van N en D, en de paren van bezienswaardigheden die verbonden zijn via wegen, worden gefixeerd voordat `longest_trip` wordt aangeroepen binnen een testcase.

Voorbeelden

Voorbeeld 1

Bekijk een scenario waarin $N = 5$, $D = 1$, en de wegverbindingen zoals in het volgend figuur staan:



De functie `longest_trip` wordt op de volgende manier aangeroepen:

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

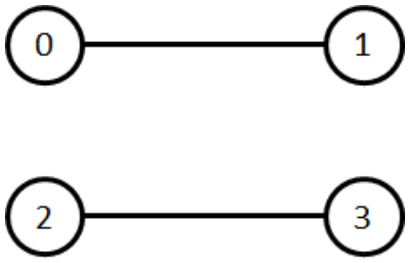
De functie mag als volgt aanroepen maken naar `are_connected`.

Aanroep	Paren verbonden via een weg	Return value
`are_connected([0], [1, 2, 4, 3])`	$(0,1)$ en $(0,2)$	`true`
`are_connected([2], [0])`	$(2,0)$	`true`
`are_connected([2], [3])`	$(2,3)$	`true`
`are_connected([1, 0], [4, 3])`	geen	`false`

Na de vierde aanroep, blijkt dat *geen enkele* van de paren $(1,4)$, $(0,4)$, $(1,3)$ en $(0,3)$ verbonden is via een weg. Omdat de dichtheid van het netwerk minstens $D = 1$ is, zien we dat uit het drietal $(0,3,4)$, het paar $(3,4)$ verbonden moet zijn via een weg. Op een vergelijkbare manier moeten bezienswaardigheden 0 en 1 verbonden zijn.

Nu kunnen we concluderen dat $t = [1,0,2,3,4]$ een trip is van lengte 5, en dat er geen trip bestaat met een lengte van meer dan 5. Daarom mag de functie `longest_trip` $[1,0,2,3,4]$ teruggeven.

Bekijk een ander scenario waarin $N = 4$, $D = 1$, en de wegverbindingen tussen de bezienswaardigheden zoals in het volgend figuur staan:



De functie `longest_trip` wordt op de volgende manier aangeroepen:

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

In dit scenario is de lengte van de langste trip 2. Na een aantal aanroepen naar `are_connected`, mag de functie `longest_trip` dus $[0, 1]$, $[1, 0]$, $[2, 3]$ of $[3, 2]$ teruggeven.

Voorbeeld 2

Subtask 0 bevat ook een ander voorbeeld testcase met $N = 256$ bezienswaardigheden. Deze testcase vind je in de bijlage die je kunt downloaden vanuit het wedstrijdsysteem.

Randvoorwaarden

- $3 \leq N \leq 256$
- De som van alle N van alle aanroepen naar `longest_trip` is hoogstens $1\,024$ in elke testcase.
- $1 \leq D \leq 3$

Subtasks

1. (5 punten) $D = 3$
2. (10 punten) $D = 2$
3. (25 punten) $D = 1$. Schrijf $l^\star$ voor de lengte van de langste trip. De functie `longest_trip` hoeft geen trip van lengte $l^\star$ terug te geven. Daarentegen, moet hij een trip van lengte minstens $\left\lceil \frac{l^\star}{2} \right\rceil$ teruggeven.
4. (60 punten) $D = 1$

In subtask 4 wordt je score bepaald op basis van het aantal keer dat je de functie `are_connected` aanroept binnen een enkele aanroep naar `longest_trip`. Laat q het maximale aantal zijn dat je de functie aanroept over alle aanroepen van `longest_trip` en alle testcases van een subtask. Je score voor deze subtask wordt berekend op basis van de volgende tabel:

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

Als in een van de testcases, de aanroepen naar `are_connected` niet aan de randvoorwaarden voldoen, zoals beschreven in de Implementatiedetails, of de array die `longest_trip` teruggeeft onjuist is, zal je oplossing 0 punten scoren op deze subtask.

Sample Grader

Laat C het aantal scenario's zijn, oftewel het aantal keer dat `longest_trip` wordt aangeroepen. De sample grader leest de invoer in het volgende formaat:

- regel 1: C

De beschrijving van C scenario's volgt.

De sample grader leest de beschrijving van elk scenario in het volgende formaat:

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

Hierin is elke U_i ($1 \leq i < N$) een array van lengte i, dat beschrijft welke bezienswaardigheden verbonden zijn via een weg. Voor elke i en j zodanig dat $1 \leq i < N$ and $0 \leq j < i$:

- als bezienswaardigheden j en i zijn verbonden via een weg, dan is $U_i[j] = 1$;
- als bezienswaardigheden j en i niet zijn verbonden via een weg, dan is $U_i[j] = 0$.

In elk scenario, controleert de sample grader dat de dichtheid van het wegennetwerk minstens D is, voordat het `longest_trip` aanroept. Als hier niet aan voldaan wordt, print deze het bericht `Insufficient Density` en sluit het zichzelf af.

Als de sample grader een schending van het protocol detecteert, geeft de sample grader `Protocol Violation: <MSG>` aan, waarbij `<MSG>` een van de volgende foutmeldingen is:

- `invalid array`: in een aanroep naar `are_connected`, is A en/of B
  - leeg, of
  - een array met een element wat geen geheel getal is tussen 0 en $N-1$, inclusief, of
  - een array wat hetzelfde element minstens twee keer bevat.
- `non-disjoint arrays`: in een aanroep naar `are_connected`, overlappen de arrays A en B.
- `too many calls`: de functie `are_connected` is meer dan $32\,640$ keer aangeroepen tijdens één aanroep naar `longest trip`, of meer dan $150\,000$ in totaal.
- `too many elements`: het totale aantal bezienswaardigheden wat aan `are_connected` gegeven is, genomen over alle aanroepen, is meer dan $1\,500\,000$.

Zoniet, noem de elementen van de array die `longest_trip` teruggeeft $t[0], t[1], \ldots, t[l-1]$ voor een niet-negatief getal l. De sample grader print drie regels voor dit scenario in het volgende formaat:

- regel 1: l
- regel 2: $t[0]$ $t[1]$ $\ldots$ $t[l-1]$
- regel 3: het aantal aanroepen naar `are_connected` tijdens dit scenario

Tot slot, print de sample grader:

- regel $1 + 3 \cdot C$: het maximum aantal aanroepen naar `are_connected`, genomen over alle aanroepen naar `longest_trip`