Perjalanan Terpanjang

Penyelenggara IOI 2023 sedang dalam masalah besar! Mereka lupa merencanakan perjalanan ke Ópusztaszer untuk hari yang akan datang. Tapi mungkin belum terlambat ...

Terdapat N titik kenal di Ópusztaszer yang dinomori dari 0 sampai $N-1$. Beberapa pasang titik kenal ini dihubungkan oleh **jalan** *dua arah*. Setiap pasang titik kenal dihubungkan oleh paling banyak satu jalan. Penyelenggara *tidak tahu* titik kenal mana saja yang dihubungkan oleh jalan tersebut.

Kita katakan **kepadatan** pada sebuah jaringan jalan Ópusztaszer adalah **setidaknya** δ jika setiap 3 titik kenal berbeda memiliki setidaknya δ jalan di antara mereka. Dengan kata lain, untuk setiap tripel titik kenal (u, v, w) sedemikian sehingga $0 \leq u < v < w < N$, di antara pasangan-pasangan titik kenal $(u, v), (v, w)$ dan (u, w) setidaknya δ pasangan terhubung dengan sebuah jalan.

Penyelenggara *mengetahui* sebuah bilangan bulat positif D sedemikian sehingga kepadatan pada jaringan jalan adalah setidaknya D. Perhatikan bahwa nilai D tidak mungkin lebih dari 3.

Penyelanggara dapat melakukan beberapa **panggilan** kepada operator telepon di Ópusztaszer untuk mencari informasi mengenai hubungan jalan antara titik-titik kenal tertentu. Dalam setiap panggilan, dua *array* tak kosong yang berisi titik-titik kenal $[A[0], \ldots, A[P-1]]$ dan $[B[0], \ldots, B[R-1]]$ harus diberikan. Titik-titik kenal tersebut haruslah berbeda, yaitu

- $A[i] \neq A[j]$ untuk setiap i dan j sedemikian sehingga $0 \leq i < j < P$;
- $B[i] \neq B[j]$ untuk setiap i dan j sedemikian sehingga $0 \leq i < j < R$;
- $A[i] \neq B[j]$ untuk setiap i dan j sedemikian sehingga $0 \leq i < P$ dan $0 \leq j < R$.

Untuk setiap panggilan, operator melaporkan apakah terdapat sebuah jalan yang menghubungkan sebuah titik kenal dari A dan sebuah titik kenal dari B. Lebih jelasnya, operator mengiterasikan semua pasangan i dan j sedemikian sehingga $0 \leq i < P$ dan $0 \leq j < R$. Jika terdapat titik-titik kenal $A[i]$ dan $B[j]$ yang terhubung oleh jalan, operator mengembalikan `true`. Sebaliknya, operator mengembalikan `false`.

Sebuah **perjalanan** sepanjang l adalah sebuah barisan dari titik-titik kenal berbeda $t[0], t[1], \ldots, t[l-1]$, untuk setiap i di antara 0 dan $l-2$, inklusif, titik kenal $t[i]$ dan titik kenal $t[i+1]$ dihubungkan dengan sebuah jalan. Sebuah perjalanan dengan panjang l dikatakan **perjalanan terpanjang** jika tidak terdapat perjalanan lain dengan panjang setidaknya $l+1$.

Tugas Anda adalah untuk membantu penyelenggara untuk mencari perjalanan terpanjang di Ópusztaszer dengan melakukan panggilan-panggilan ke operator.

Detail Implementasi

Anda harus mengimplementasikan prosedur berikut:

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

- N: banyaknya titik kenal di Ópusztaszer.
- D: kepadatan minimum pada jaringan jalan.
- Prosedur ini harus mengembalikan sebuah *array* $t = [t[0], t[1], \ldots, t[l-1]]$, yang menyatakan perjalanan terpanjang.
- Prosedur ini dapat dipanggil **berkali-kali** dalam setiap kasus uji.

Prosedur di atas dapat melakukan panggilan pada prosedur berikut:

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

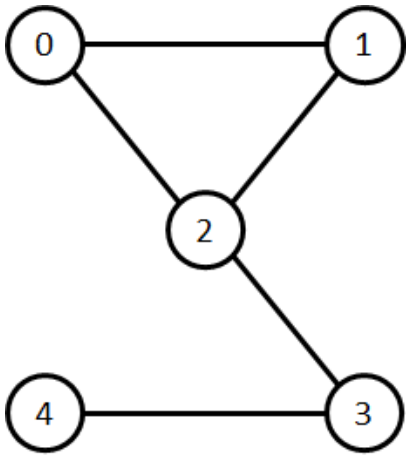
- A: *array* tak kosong dari titik-titik kenal berbeda.
- B: *array* tak kosong dari titik-titik kenal berbeda.
- A dan B haruslah saling lepas.
- Prosedur ini mengembalikan `true` jika terdapat sebuah titik kenal dari A dan sebuah titik kenal dari B yang dihubungkan oleh sebuah jalan. Sebaliknya, prosedur ini mengembalikan `false`.
- Prosedur ini dapat dipanggil paling banyak $32\,640$ kali dalam setiap pemanggilan `longest_trip`, dan paling banyak $150\,000$ kali secara total.
- Panjang total panjang-panjang *array* A dan B yang diberikan ke prosedur ini tidak boleh melebihi $1\,500\,000$ pada semua panggilannya.

Grader bersifat **tidak adaptif**. Setiap submisi dinilai dengan menggunakan kasus-kasus uji yang sama. Dengan kata lain, nilai dari N dan D, serta pasangan-pasangan titik kenal yang terhubung sudah ditetapkan sebelum panggilan `longest_trip` dilakukan.

Contoh

Contoh 1

Perhatikan sebuah skenario dengan $N = 5$, $D = 1$, dan hubungan-hubungan jalan ditunjukkan di gambar berikut:



Prosedur `longest_trip` dipanggil sebagai berikut:

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

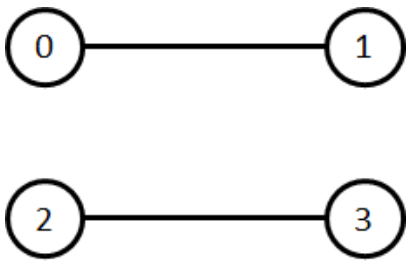
Prosedur dapat melakukan panggilan-panggilan pada `are_connected` sebagai berikut.

Panggilan	Pasangan terhubung oleh jalan	Nilai kembali
`are_connected([0], [1, 2, 4, 3])`	$(0, 1)$ dan $(0, 2)$	`true`
`are_connected([2], [0])`	$(2, 0)$	`true`
`are_connected([2], [3])`	$(2, 3)$	`true`
`are_connected([1, 0], [4, 3])`	tidak ada	`false`

Setelah pemanggilan keempat, ternyata *tidak ada* dari pasangan $(1, 4)$, $(0, 4)$, $(1, 3)$ dan $(0, 3)$ yang terhubung oleh sebuah jalan. Karena kepadatan pada jaringan adalah setidaknya $D = 1$, kita tahu bahwa tripel $(0, 3, 4)$, pasangan $(3, 4)$ pasti dihubungkan oleh sebuah jalan. Serupa dengan ini, titik kenal 0 dan 1 pasti terhubung.

Pada tahap ini, dapat disimpulkan bahwa $t = [1, 0, 2, 3, 4]$ adalah sebuah perjalanan dengan panjang 5, dan tidak ada perjalanan dengan panjang yang lebih besar dari 5. Oleh karena itu, prosedur `longest_trip` harus mengembalikan $[1, 0, 2, 3, 4]$.

Perhatikan skenario lain dengan $N = 4$, $D = 1$, dan jalan-jalan antara titik-titik kenal ditunjukkan dengan gambar berikut:



Prosedur `longest_trip` dipanggil sebagai berikut:

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

Pada skenario ini, panjang dari perjalanan terjauh adalah 2. Dengan kata lain, setelah beberapa pemanggilan ke prosedur `are_connected`, prosedur `longest_trip` dapat mengembalikan salah satu dari $[0, 1]$, $[1, 0]$, $[2, 3]$ atau $[3, 2]$.

Contoh 2

Subsoal 0 mengandung kasus uji tambahan dengan $N = 256$ titik kenal. Kasus uji ini disertakan dalam paket lampiran yang dapat Anda unduh dari sistem kontes.

Batasan

- $3 \leq N \leq 256$
- Jumlah dari N pada seluruh panggilan `longest_trip` tidak melebihi $1\,024$.
- $1 \leq D \leq 3$

Subsoal

1. (5 poin) $D = 3$
2. (10 poin) $D = 2$
3. (25 poin) $D = 1$. Misalkan $l^\star$ sebagai panjang dari perjalanan terpanjang. Prosedur `longest_trip` tidak harus mengembalikan sebuah perjalanan dengan panjang $l^\star$. Tetapi, prosedur dapat mengembalikan sebuah perjalanan dengan panjang setidaknya $\left\lceil \frac{l^\star}{2} \right\rceil$.
4. (60 poin) $D = 1$

Di subsoal 4, nilai Anda ditentukan berdasarkan jumlah panggilan pada prosedur `are_connected` pada setiap panggilan `longest_trip`. Anggap q sebagai maksimum banyaknya panggilan pada setiap pemanggilan `longest_trip` pada setiap kasus uji dari subsoal tersebut. Nilai Anda dihitung berdasarkan tabel berikut:

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

Contoh *Grader*

Misalkan C sebagai banyaknya skenario. Dengan kata lain, banyaknya pemanggilan `longest_trip`. Contoh *grader* membaca masukan dengan format berikut:

- baris 1: C

Deskripsi dari C skenario akan mengikuti.

Contoh *grader* membaca deskripsi dari setiap skenario dengan format berikut:

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

Di sini, setiap U_i ($1 \le i < N$) adalah sebuah *array* dengan ukuran i, yang menjelaskan pasangan titik kenal mana yang dihubungkan oleh sebuah jalan. Untuk setiap i dan j sedemikian sehingga $1 \le i < N$ dan $0 \le j < i$:

- jika titik kenal j dan i dihubungkan dengan sebuah jalan, maka nilai dari $U_i[j]$ haruslah 1;
- jika tidak terdapat jalan yang menghubungi titik kenal j dan i, maka nilai dari $U_i[j]$ haruslah 0.

Pada setiap skenario, sebelum pemanggilan `longest_trip`, contoh *grader* akan memeriksa apakah kepadatan pada jaringan jalan adalah setidaknya D. Jika kondisi ini tidak terpenuhi, *grader* akan mencetak sebuah pesan `Insufficient Density` dan berhenti.

Jika contoh *grader* menemukan adanya pelanggaran protokol, keluaran dari contoh *grader* adalah `Protocol Violation: <MSG>`, dengan `<MSG>` adalah salah satu dari pesan error berikut:

- `invalid array`: dalam sebuah pemanggilan ke `are_connected`, setidaknya salah satu dari *array* A dan B
  - adalah kosong, atau
  - mengandung sebuah elemen yang bukan merupakan bilangan bulat antara 0 dan $N-1$, inklusif, atau
  - mengandung elemen yang sama.
- `non-disjoint arrays`: dalam sebuah panggilan pada `are_connected`, *array* A dan B tidak saling lepas.
- `too many calls`: banyaknya panggilan yang dilakukan ke `are_connected` melebihi $32\,640$ selama pemanggilan `longest_trip`, atau melebihi $150\,000$ secara keseluruhan.
- `too many elements`: total titik kenal yang diberikan ke `are_connected` pada semua panggilan melebihi $1\,500\,000$.

Sebaliknya, anggap elemen-elemen pada *array* yang dikembalikan oleh `longest_trip` dalam sebuah skenario sebagai $t[0], t[1], \ldots, t[l-1]$ untuk l non-negatif. Contoh grader mencetak tiga baris untuk skenario ini dalam format berikut:

- baris 1: l
- baris 2: $t[0]\ t[1]\ \ldots\ t[l-1]$

- baris 3: banyaknya panggilan `are_connected` pada skenario ini

Pada akhirnya, contoh *grader* mencetak:

- baris $1 + 3 \cdot C$: maksimum banyaknya pemanggilan `are_connected` pada semua pemanggilan `longest_trip`