Tezej

> *Ako se svi delovi Tezejevog broda vremenom zamenjuju jedan po jedan, u kom trenutku — ako uopšte — prestaje da bude isti brod?*

Kada ne razmišlja duboko o apstraktnom, Tezej u slobodno vreme ubija minotaure. Ovog puta, međutim, prvo mora da prođe kroz mračan i uvijen lavirint. Pošto to nije lak zadatak, on traži pomoć od Arijadne da ga vodi. Lavirint se može posmatrati kao neusmereni povezan graf koji ima n čvorova (označenih od 1 do n) i m grana, i specijalni čvor t, u kome sedi Minotaur.

Tezej uopšte ne može da vidi graf, ali Arijadna može. Ona i Tezej odlučuju da osmisle strategiju kako bi on bezbedno stigao do čvora gde se nalazi Minotaur: ona će staviti oznaku 0 ili 1 na svaku od m grana. Nakon toga, Tezej će ući u lavirint kroz čvor s koji Arijadna ne zna unapred.

Pošto je u lavirintu veoma mračno, u svakom trenutku on može da vidi samo indeks čvora u kome se nalazi, indekse susednih čvorova i oznake susednih grana. Takođe, zbog uvijene prirode lavirinta, on se **nikada ne može setiti** bilo kakvih informacija o čvorovima koje je prethodno posetio.

Da bi bezbedno stigao do Minotaura, Tezej se mora pomeriti najviše $min + C$ puta, gde je min minimalan broj grana na putu od s do t, a C je konstanta.

Detalji implementacije

Ti treba da implementiraš dve funkcije:

```
std::vector<int> label(int n, std::vector<std::pair<int,int>> edges, int t);
```

gde je:

- n: broj čvorova u grafu (lavirintu)
- $edges$: lista dužine m koja opisuje grane grafa
- t: čvor koji predstavlja cilj, tj. čvor u kome sedi Minotaur
- Ova funkcija treba da vrati listu oznaka dužine m tako da je i-ti element liste broj (oznaka) 0 ili 1 i on predstavlja oznaku i-te grane za svako $0 \leq i < m$.
- Svaka grana **mora** biti označena ili brojem 0 ili brojem 1. Označavanje grane nekom drugom oznakom dovodi do **nedefinisanog ponašanja**.
- Ova funkcija će biti pozvana **tačno jednom** za svaki test primer.

```
int travel(int n, int u, std::vector<std::pair<int,int>> neighbours);
```

gde je:

- n: broj čvorova u grafu
- u: trenutni čvor
- $neighbours$: lista parova (v, e) koj označava da je grana između čvorova u i v označena oznakom e
- Ova funkcija vraća susedni čvor u koji se pomera Tezej. Ako je susedni čvor baš čvor t, izvršavanje programa se automatski završava (prekida).
- Garantuje se da pri svakom pozivu ove funkcije, čvor u neće biti specijalni čvor t.
- Pozivanje ove funkcije predstavlja kretanje kroz lavirint. Tako će za svaki test primer ova funkcija biti pozvana **onoliko puta koliko je potrebno da se stigne** do ciljnog čvora.

Pažnja! Nije dozvoljeno deklarisanje bilo koje globalne/statičke promenljive i korišćenje za komunikaciju između različitih poziva funkcija `label` ili `travel`. Bilo koji pokušaj da se ovo zaobiđe doveo bi do **nedefinisanog ponašanja**.

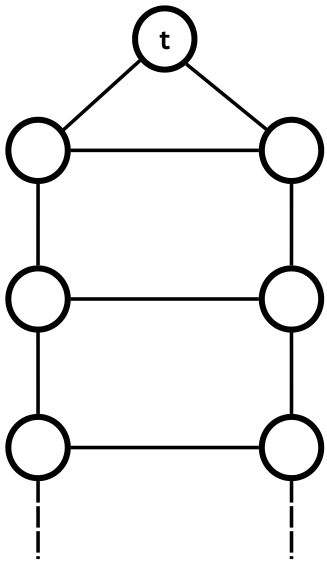
Ograničenja

- $1 \leq n \leq 10000$
- $1 \leq m \leq 50000$
- $C = 14$
- Početni čvor s je fiksiran za svaki test primer pre poziva funkcije `label`.

Podzadaci

1. (4 boda) Graf predstavlja kliku, tj. kompletan graf (odnosno postoji ivica između bilo koja dva čvora $1 \leq u < v \leq n$).
2. (10 bodova) Rastojanje između cilja i bilo kog čvora u grafu je najviše 2 (grane).
3. (11 bodova) Graf je stablo.
4. (13 bodova) Graf je bipartitan (tj, skup čvorova grafa se može podeliti u dva disjunktna podskupa tako da ne postoji grana između neka dva čvora iz istog podskupa).
5. (12 bodova) Graf će biti merdevine (videti definiciju ispod).
6. (50 bodova) Nema dodatnih ograničenja.

Napomena: Merdevine su graf koji se sastoji od dva paralelna puta (lanca) iste dužine, tako da između svaka dva odgovarajuća čvora na tim putevima postoji grana, koje obrazuju prečke merdevina. Dodatno, na jednom kraju merdevina je specijalni čvor $-$ tj, čvor-cilj t $-$ koji je povezan sa oba krajnja čvora merdevina, i predstavlja njihovog zajedničkog roditelja. Garantovano je da će n biti neparan za bilo koji takav graf. Pogledaj sliku ispod.



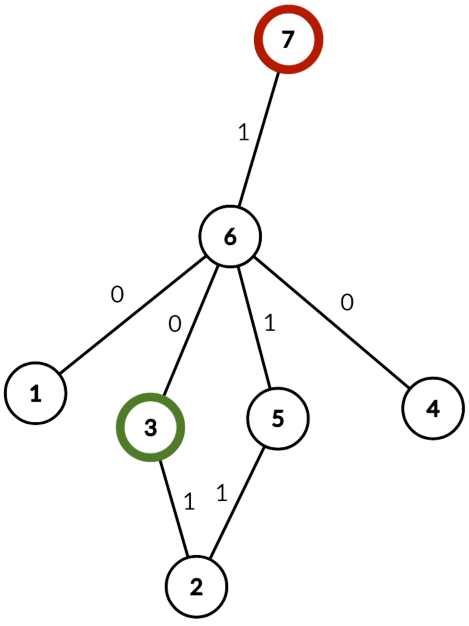
Primer

Primer 1

Posmatrajmo graf koji ima 7 čvorova i 7 grana (izlistane su ispod). Polazni čvor će biti čvor 3 (obojen **zeleno**) , a ciljni čvor je čvor 7 (obojen **crveno**). Grejder će prvo izršiti sledeći pozive:

```
label(7, {{1, 6}, {7, 6}, {2, 5}, {3, 2}, {3, 6}, {6, 5}, {6, 4}}, 7)
```

Pretpostavimo da je taj poziv funkcije `label` vratio listu `{0, 1, 1, 1, 0, 1, 0}`. Tada rezultujući graf izgleda kao što je prikazano na slici:



Jedan mogući niz poziva funkcije `travel` koji dovodi do ispravnog rešenja je prikazan u sledećoj tabeli:

Poziv	Vrednost koja je vraćena
`travel(7, 3, {{2, 1}, {6, 0}}`	2
`travel(7, 2, {{5, 1}, {3, 1}}`	5
`travel(7, 5, {{6, 1}, {2, 1}}`	6
`travel(7, 6, {{3, 0}, {5, 1}, {1, 0}, {4, 0}, {7, 1}}`	4
`travel(7, 4, {{6, 0}}`	6
`travel(7, 6, {{3, 0}, {5, 1}, {1, 0}, {4, 0}, {7, 1}}`	7

U trenutku kada je vraćena vrednost 7 (tj. ciljni čvor), program se zaustavlja.

Priloženi grejder

Priloženi grejder čita ulazne podatke u sledećem formatu:

- Red 1: $n\ m$
- Red 2: $s\ t$
- Red $3 + i$: $a\ b$ što označava da postoji grana između čvorova a i b.

Grejder će prvo pozvati funkciju `label` sa odgovarajućim argumentima i obeležiće ivice grafa prema vraćenom vektoru.

Zatim će pozvati `travel` sa argumentima n, s i susedima čvora s. Nakon prvog poziva, nastaviće da vrši uzastopne pozive `travel`, pri čemu je trenutni čvor onaj koji je vratio prethodni poziv, sve dok se ne stigne do cilja t.

Na kraju će ispisati broj poteza koje je vaše rešenje napravilo i čvorove kroz koje je prošlo u redosledu prolaska.