Soľné bane

Išli sme na exkurziu do soľnej bane. Soľné bane su veľmi komplikované, skoro ako bludisko. Ak si človek nedá pozor, ľahko sa v nich stratí. Janči nemá veľmi dobrý orientačný zmysel ani veľmi dobrú pamäť. Naopak Hanka v týchto oblastiach exceluje a tak sa dohodli, že mu pomôže. Bane môžeme chápať ako súvislý neorientovaný graf s n vrcholmi (očíslovanými od 1 po n) a m hranami, so špeciálnym vrcholom t, kde je stánok so suvenírmi.

Za desať dvanásť sa treba stretnúť pri stánku su suvenírmi, takže Janči by sa z každého vrcholu veľmi rád vedel dostať do vrcholu t veľmi rýchlo. Hanka a Janči si vedia dohodnúť stratégiu. Hanka si vie potom pozrieť celý graf a na každú hranu buď nalepiť alebo nenalepiť magnetku (viditeľnosť magnetky je rovnaká z oboch strán hrany). Následne Janči vstúpi do soľnej bane cez neznámy štartovací vrchol s (ktorý dopredu nepozná ani Hanka) a snaží sa dostať do vrcholu t.

V každom okamihu Janči vidí len číslo vrcholu, v ktorom sa nachádza, čísla susedných vrcholov a prítomnosť magnetiek na priľahlých hranách. Navyše si **nemôže pamätať žiadne** informácie o predchádzajúcich navštívených vrcholoch.

Aby sa Janči načas dostal k stánku so suvenírmi, musí vykonať najviac $min + C$ krokov, kde min je minimálny počet hrán na ceste z s do t a C je konštanta.

Detaily implementácie

Treba implementovať dve funkcie:

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

- n: počet vrcholov
- $edges$: zoznam dĺžky m, opisujúci hrany grafu
- t: cieľový vrchol
- Táto funkcia by mala vrátiť zoznam dĺžky m, kde i-ty prvok môže byť buď 0 alebo 1 a reprezentuje, či je na i-tej hrane magnetka alebo nie, pre všetky $0 \leq i < m$.
- Každá hrana musí byť označená buď 0 alebo 1. Označenie inou hodnotou spôsobí **nedefinované správanie**.
- Táto procedúra sa volá **presne raz** pre každý vstup.

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

- n: počet vrcholov v grafe
- u: aktuálny vrchol
- $neighbours$: zoznam dvojíc (v, e), kde existuje hrana medzi u a v označená e. Poradie dvojíc nie je fixné, môže sa líšiť medzi volaniami tejto procedúry pre ten istý vrchol u.
- Táto procedúra by mala vrátiť číslo susedného vrcholu, do ktorého sa má Janči pohnúť. Ak je susedný vrchol rovný t, program sa automaticky ukončí.
- Je zaručené, že pre každý volaný prípad tejto funkcie bude $u \neq t$.
- Každé volanie tejto procedúry predstavuje krok v labyrinte. Preto sa táto procedúra môže volať **ľubovoľne veľakrát** v rámci jedného testu, pokiaľ Janči nedorazí do cieľa.

Pozor! Program **nesmie používať globálne ani statické premenné** na komunikáciu medzi volaniami funkcií `label` a `travel`. Akýkoľvek pokus o obídenie tohto obmedzenia vedie k **nedefinovanému správaniu**.

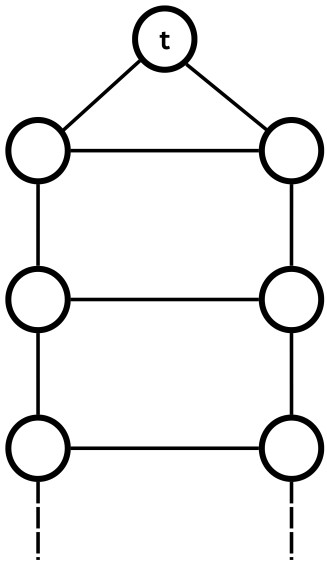
Obmedzenia

- $1 \leq n \leq 10000$
- $1 \leq m \leq 50000$
- $C = 14$
- Začiatočný vrchol s je pevne daný pre každý test ešte pred volaním funkcie `label`.

Podúlohy

1. (4 body) Graf je klika (t.j. existuje hrana medzi každou dvojicou vrcholov $1 \leq u < v \leq n$).
2. (10 bodov) Vzdialenosť medzi cieľovým vrcholom a ľubovoľným iným je najviac 2 hrany.
3. (11 bodov) Graf je strom.
4. (13 bodov) Graf je bipartitný (t.j. existuje rozdelenie vrcholov do dvoch množín tak, že medzi vrcholmi z tej istej množiny neexistuje žiadna hrana).
5. (12 bodov) Graf je rebrík (pozri definíciu nižšie).
6. (50 bodov) Žiadne ďalšie obmedzenia.

Poznámka: Rebríkový graf pozostáva z dvoch paralelných ciest rovnakej dĺžky, pričom každá dvojica zodpovedajúcich si vrcholov je spojená hranou (priečky rebríka). Na jednom konci rebríka sa nachádza špeciálny vrchol (cieľ t), ktorý je spojený s oboma koncovými vrcholmi rebríka, čím vytvára spoločný rodičovský uzol. Je zaručené, že n bude nepárne pre každý takýto graf. Pozri obrázok nižšie.



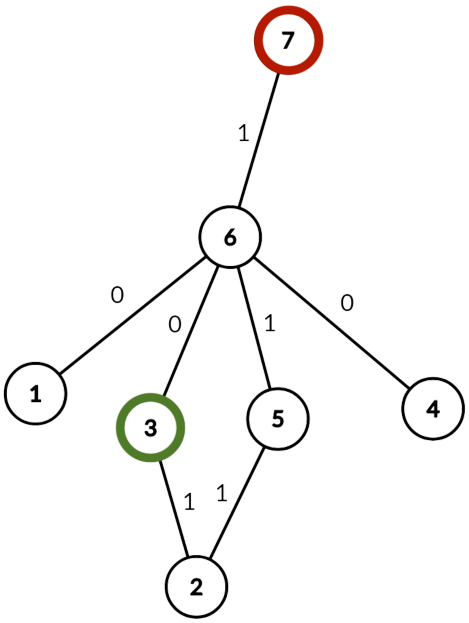
Príklady

Príklad 1

Majme graf so 7 vrcholmi a 7 hranami (uvedenými nižšie). Štartovací vrchol bude 3 (označený **zelenou**) a cieľový vrchol je 7 (označený **červenou**). Tester najprv zavolá:

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

Predpokladajme, že volanie `label` vráti `{0, 1, 1, 1, 0, 1, 0}`. Potom bude výsledný graf vyzerať nasledovne:



Nasleduje možná postupnosť volaní `travel`, ktorá vedie ku korektnému riešeniu:

Volanie	Návratová hodnota
`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

Keď návratová hodnota je 7 (t.j. cieľový vrchol), program sa ukončí.

Ukážkový tester

Ukážkový tester číta vstup vo formáte:

- riadok 1: n m
- riadok 2: s t
- riadky $3 + i$: a b znamená, že existuje hrana medzi vrcholmi a a b.

Najprv tester zavolá `label` s príslušnými parametrami a označí hrany grafu podľa vráteného zoznamu.

Potom zavolá `travel` s parametrami n, s a zoznamom susedov vrcholu s. Po prvom volaní bude pokračovať ďalšími volaniami `travel`, kde aktuálny vrchol bude ten, ktorý sa vrátil z predchádzajúceho volania, až kým sa nedosiahne cieľový vrchol t.

Na konci vypíše počet pohybov a poradie navštívených vrcholov.