

Navigáció

 3 sec.  512 MB

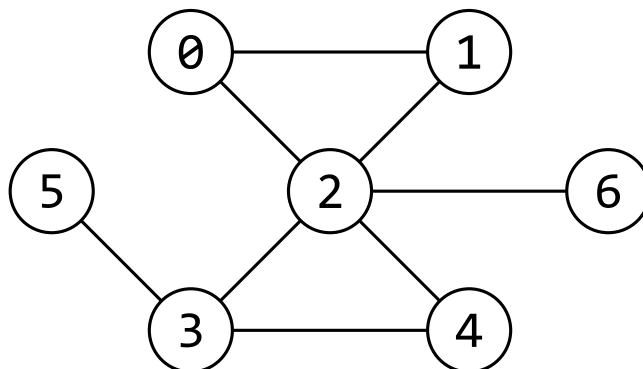
Van egy **egyszerű, összefüggő, irányítatlan kaktuszgráf**¹, $N \leq 1000$ csúccsal és M éllel. A csúcsok (0 és 1499 közötti nemnegatív egész számokkal) színezettek. Kezdetben minden csúcs színe 0. Egy **determinisztikus memória nélküli robot**² csúcsról csúcsra haladva fedezi fel a gráfot. Minden csúcsot legalább egyszer meg kell látogatnia, majd le kell állnia.

A robot a gráf egy adott csúcsából indul, ami a gráf bármelyik csúcsa lehet. Minden lépésben látja a jelenlegi és a szomszédos csúcsok színeit egy, az adott csúcsnál rögzített sorrendben (azaz, ha újból a csúcshoz ér, ugyanabban a sorrendben fogja látni a szomszédos csúcsokat, akkor is, ha közben a színük megváltozott). A robot minden lépésben az alábbi műveletek valamelyikét hajtja végre:

1. Úgy dönt, hogy leáll.
2. Kiszínezi a jelenlegi csúcsot valamilyen színnel (ez meg is egyezhet a korábbi színével), majd átmozog egy szomszédos csúcsba. A szomszédos csúcsot egy 0 és $D - 1$ közötti sorszámmal azonosítja, ahol D a szomszédos csúcsok száma.

A második esetben az aktuális csúcs átszíneződik (vagy esetleg változatlan marad a színe), és a robot a kiválasztott szomszédos csúcsba mozog. Ez addig ismétlődik, amíg a robot le nem áll, vagy amíg el nem éri a lépésszám határát. A robot akkor nyer, ha az összes csúcsot meglátogatja, majd az $L = 3000$ lépéses határon belül leáll (ellenkező esetben veszít).

¹Az egyszerű, összefüggő, irányítatlan kaktuszgráf egy olyan egyszerű, összefüggő irányítatlan gráf (nincsenek hurokélek és többszörös élek; bármely csúcs elérhető bármelyik másiktól; az élek kétirányúak), melyben minden él legfeljebb egy egyszerű körhöz tartozik (az egyszerű kör egy olyan kör, amely minden csúcsot legfeljebb egyszer tartalmaz). A lenti példa egy ilyen gráfot ábrázol.



²Egy robot determinisztikus és memória nélküli, ha a viselkedése kizárólag a jelenlegi bemenetétől függ (azaz, nem ment el semmilyen adatot a lépések között), és azonos bemenet esetén mindig ugyanazt a kimenetet adja.



Olyan stratégiát kell tervezned a robot számára, amely képes megoldani a feladatot bármely kaktuszgráfban. Ezen kívül törekedj arra, hogy a megoldásod a lehető legkevesebb különböző színt használja. Itt a 0 szín mindig használnak számít.



Megvalósítás

A robot stratégiáját az alábbi függvényben kell megvalósítanod:

```
std::pair<int, int> navigate(int currColor, std::vector<int> adjColors)
```

A függvény paraméterként megkapja az aktuális csúcs színét és az összes szomszédos csúcs színét (sorrendben). Egy olyan értékpárt kell visszaadnia, amelynek első eleme az aktuális csúcs új színe, második eleme pedig annak a csúcshozmészedjének sorszáma, ahová a robotnak át kell lépnie. Ha ehelyett a robotnak le kell állnia, a függvénynek a $(-1, -1)$ értékpárt kell visszaadnia.

Ez a függvény hívódik meg újra és újra, hogy kiválassza a robot lépéseit. Mivel determinisztikus, ezért ha a `navigate` függvényt már meghívtuk bizonyos paraméterekkel, akkor soha többé nem hívjuk meg ugyanazokkal a paraméterekkel; ehelyett a korábbi visszatérési értékét újra felhasználjuk. Ezen kívül minden teszt tartalmazhat $T \leq 5$ altesztet (különböző gráfokat és/vagy kiindulási csúcsokat), és ezek párhuzamosan is futhatnak (azaz a programunk felváltva kaphat hívásokat a különböző altesztektől). Végül, a `navigate` hívások a program **különböző végrehajtásaitól** is jöhetnek (vagy ugyanabban a végrehajtásban is történhetnek). A programvégrehajtások teljes száma $P = 100$. Ezek miatt a programodnak nem szabad megpróbálnia információt átadni a különböző hívások között.



Korlátok

- $3 \leq N \leq 1000$
- $0 \leq \text{szín} < 1500$
- $L = 3000$
- $T \leq 5$
- $P = 100$



Pontozás

Az egy részfeladatért kapott pontok S hányada C -től függ, ami a megoldásod által használt színek maximális száma (a 0 szint is beleértve) az adott részfeladat vagy bármelyik, ehhez a részfeladathoz szükséges részfeladat bármely tesztjénél:

- Ha a programod elbukik bármelyik teszten, akkor $S = 0$.
- Ha $C \leq 4$, akkor $S = 1.0$.
- Ha $4 < C \leq 8$, akkor $S = 1.0 - 0.6 \frac{C-4}{4}$.
- Ha $8 < C \leq 21$, akkor $S = 0.4 \frac{8}{C}$.
- Ha $C > 21$, akkor $S = 0.15$.



Részfeladatok

Részfeladat	Pontszám	Szükséges részfeladatok	N	További korlátok
0	0	—	≤ 300	A példa.
1	6	—	≤ 300	A gráf egy kör. ¹
2	7	—	≤ 300	A gráf egy csillag. ²
3	9	—	≤ 300	A gráf egy útvonalgráf. ³
4	16	2 – 3	≤ 300	A gráf egy fagráf. ⁴
5	27	—	≤ 300	Minden csúcsnak legfeljebb 3 szomszédja van és a kezdőcsúcsnak 1 szomszédja van.
6	28	0 – 5	≤ 300	—
7	7	0 – 6	—	—

¹Egy körgráf élei: $(i, (i + 1) \bmod N)$ minden $0 \leq i < N$ esetén.

²Egy csillaggráf élei: $(0, i)$ minden $1 \leq i < N$ esetén.

³Egy útvonalgráf élei: $(i, i + 1)$ minden $0 \leq i < N - 1$ esetén.

⁴A fagráf egy körmentes gráf.



Példa

Tekintsük példaként a fenti ábrán látható gráfot, amelyben $N = 7$, $M = 8$, és amelynek élei $(0, 1)$, $(1, 2)$, $(2, 0)$, $(2, 3)$, $(3, 4)$, $(4, 2)$, $(3, 5)$ és $(2, 6)$. Továbbá, mivel a csúcsok szomszédsági listáiban az elemek sorrendje is fontos, megadjuk őket ebben a táblázatban:



Csúcs	Szomszédos csúcsok
0	2, 1
1	2, 0
2	0, 3, 4, 6, 1
3	4, 5, 2
4	2, 3
5	3
6	2

Tegyük fel, hogy a robot az 5-ös csúcsnál kezd. Ekkor az alábbi egy lehetséges (sikertelen) lépéssorozat:

#	Színek	Csúcs	navigate függvényhívás	Visszatérési érték
1	0, 0, 0, 0, 0, 0, 0	5	navigate(0, {0})	{1, 0}
2	0, 0, 0, 0, 0, 1, 0	3	navigate(0, {0, 1, 0})	{4, 2}
3	0, 0, 0, 4, 0, 1, 0	2	navigate(0, {0, 4, 0, 0, 0})	{0, 3}
4	0, 0, 0, 4, 0, 1, 0	6	¹ navigate(0, {0})	{1, 0}
5	0, 0, 0, 4, 0, 1, 1	2	navigate(0, {0, 4, 0, 1, 0})	{8, 0}
6	0, 0, 8, 4, 0, 1, 1	0	navigate(0, {8, 0})	{3, 0}
7	3, 0, 8, 4, 0, 1, 1	2	navigate(8, {3, 4, 0, 1, 0})	{2, 2}
8	3, 0, 2, 4, 0, 1, 1	4	navigate(0, {2, 4})	{1, 1}
9	3, 0, 2, 4, 1, 1, 1	3	navigate(4, {1, 1, 2})	{-1, -1}

Itt a robot összesen 6 különböző színt használt: 0, 1, 2, 3, 4 és 8 (megjegyzendő, hogy a 0 akkor is használnak számítana, ha a robot soha nem tért volna vissza a 0 színnel, mivel minden csúcs színe kezdetben 0). A robot 9 lépést tett meg, mielőtt leállt. Azonban kudarcot vallott, mivel anélkül állt le, hogy meglátogatta volna az 1-es csúcsot.

¹Megjegyezzük, hogy a navigate hívása a 4. lépésnél valójában nem történik meg. Ez azért van, mert ez megegyezik az 1. lépésnél történő hívással, így az értékelő egyszerűen újra felhasználja a függvényed visszatérési értékét a korábbi hívásból. Ez azonban továbbra is egy lépésnek számít a robotnál.



Mintaértékelő

A mintaértékelő nem futtatja a programodat többször, így minden `navigate` hívás egy programfuttatáson belül történik.

A bemenet formátuma a következő: Először beolvassuk T -t, a résztesztek számát. Ezután minden résztesztre:

- 1. sor: két egész szám - N és M ;
- $2 + i$. sor (minden $0 \leq i < M$ esetén): két egész szám - A_i és B_i , az i . él által összekötött csúcsok sorszáma ($0 \leq A_i, B_i < N$).

A mintaértékelő ezután kiírja a megoldás által használt különböző színek számát és a megoldás befejezéséhez szükséges lépések számát. Alternatívaként hibaüzenetet ír ki, ha a megoldásod sikertelen volt.

Alapértelmezés szerint a mintaértékelő részletes információt ír ki arról, hogy a robot mit lát és mit csinál minden egyes lépésben. Ezt kikapcsolhatod, ha a `DEBUG` értékét `true`-ról `false`-ra változtatod.