



Задача Навигации

 3 sec.  512 MB

Имаме **свързан неориентиран прост граф тип кактус**¹ с $N \leq 1000$ върха и M ребра. Неговите върхове имат цветове (обозначени с цели числа от 0 до 1499). Първоначално всички върхове имат цвят 0. **Детерминистичен робот без памет**² изследва графа, като се движи от връх на връх. Той трябва да посети всички върхове поне веднъж и след това да приключи.

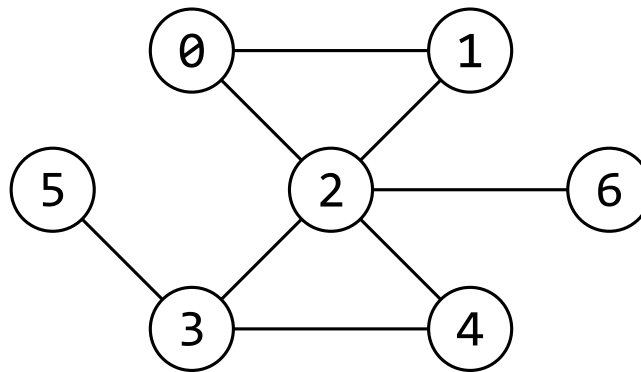
Роботът започва от някакъв връх, който може да бъде всеки от върховете в графа. На всяка стъпка той вижда цвета на текущия си връх и цветовете на всички съседни върхове **в някакъв фиксиран за текущия ред на върховете** (т.е. повторното посещение на връх ще даде на робота същата последователност от съседни върхове, дори ако цветовете им са различни от предишните). Роботът извършва едно от следните две действия:

1. Решава да прекрати работата си.
2. Избира нов (или евентуално същия) цвят за текущия връх и към кой съседен връх да се придвижи. Съседният връх се идентифицира с индекс от 0 до $D - 1$, където D е броя на съседни върхове.

Във втория случай текущият връх се преоцветява (или евентуално остава в същия цвят) и роботът се придвижва към избрания съседен връх. Това се повтаря, докато роботът не приключи или докато достигне лимита за итерации. Роботът печели, ако посети всички върхове и след това приключи в рамките на лимита за итерации от $L = 3000$ стъпки (в противен случай губи).

Трябва да разработите стратегия за робота, която може да реши задачата за всеки граф тип кактус. Освен това, трябва да се опитате да минимизирате броя на различните цветове, които вашето решение използва. Тук цвят 0 винаги се брои за използван.

¹Свързан неориентиран прост граф тип кактус е свързан неориентиран прост граф (всеки връх е достижим от всеки друг връх; ребрата са двупосочни; няма примки или мултиребра), в който всяко ребро принадлежи на най-много един прост цикъл (прост цикъл е цикъл, който съдържа всеки връх най-много веднъж). Изображението по-долу е пример.



²Роботът е детерминистичен и без памет, ако действието му зависи само от текущите му входни данни (т.е. не съхранява данни от стъпка на стъпка) и винаги избира едно и също действие, когато му бъдат дадени едни и същи входни данни.



Детайли по имплементацията

Стратегията на робота трябва да бъде реализирана чрез следната функция:

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

Като параметри получава цвета на текущия връх и цветовете на всички съседни върхове (по ред). Трябва да върне двойка, чийто първи елемент е новия цвят за текущия връх, а вторият елемент е индекса на съседния връх, към който роботът трябва да се премести. Ако вместо това роботът трябва да прекрати работата си, функцията трябва да върне двойката $(-1, -1)$.

Тази функция ще бъде извикана многократно, за да се изберат действията на робота. Тъй като е детерминистична, ако `navigate` вече е била извикана с някои параметри, тя никога повече няма да бъде извикана със същите тези параметри; вместо това предишната ѝ върната стойност ще бъде използвана повторно. Освен това, всеки тест може да съдържа $T \leq 5$ подтестове (отделни графи и/или начални позиции) и те могат да се изпълняват едновременно (т.е. вашата програма може да получава редуващи се извиквания за различни подтестове). И накрая, извикванията на `navigate` могат да се случат в **отделни изпълнения** на вашата програма (но понякога могат да се случат и в едно и също изпълнение). Общият брой изпълнения на вашата програма е $P = 100$. Поради всичко това, вашата програма не трябва да се опитва да предава информация между различните извиквания.



Ограничения

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



Оценяване

Частта от точките S за дадена подзадача, която получавате, зависи от C - максималният брой различни цветове, които вашето решение използва (включително цвят 0) във всеки тест в тази подзадача или във всяка друга задължителна подзадача:

- Ако решението ви е неуспешно на някой подтест, то $S = 0$
- Ако $C \leq 4$, то $S = 1.0$
- Ако $4 < C \leq 8$, то $S = 1.0 - 0.6 \frac{C-4}{4}$
- Ако $8 < C \leq 21$, то $S = 0.4 \frac{8}{C}$
- Ако $C > 21$, то $S = 0.15$



Подзадачи

Подзадача	Точки	Необходими подзадачи	N	Допълнителни ограничения
0	0	—	≤ 300	Примерът.
1	6	—	≤ 300	Графът е цикъл. ¹
2	7	—	≤ 300	Графът е звезда. ²
3	9	—	≤ 300	Графът е пръчка. ³
4	16	2 – 3	≤ 300	Графът е дърво. ⁴
5	27	—	≤ 300	Всички върхове имат най-много 3 съседни върха, а върхът, от който започва роботът, има 1 съседен връх.
6	28	0 – 5	≤ 300	—
7	7	0 – 6	—	—

¹Графът е цикъл, ако има ребра : $(i, (i + 1) \bmod N)$ for $0 \leq i < N$.

²Графът е звезда, ако има ребра: $(0, i)$ for $1 \leq i < N$.

³Графът е пръчка, ако има ребра: $(i, i + 1)$ for $0 \leq i < N - 1$.

⁴Графът е дърво, ако няма цикли.



Пример

Разгледайте примерния граф от изображението в условието, за който $N = 7$, $M = 8$ и ребрата са $(0, 1)$, $(1, 2)$, $(2, 0)$, $(2, 3)$, $(3, 4)$, $(4, 2)$, $(3, 5)$ и $(2, 6)$. Тъй като редът на елементите в списъците за съседство на върховете имат значение, те са дадени в тази таблица:

Връх	Съседни върхове
0	2, 1
1	2, 0
2	0, 3, 4, 6, 1
3	4, 5, 2
4	2, 3
5	3
6	2



Да предположим, че роботът започва от връх 5. Тогава следната е една възможна (неуспешна) последователност от действия:

#	Цветовете	Връх	Извикване на <code>navigate</code>	Върнатата стойност
1	0, 0, 0, 0, 0, 0, 0	5	<code>navigate(0, {0})</code>	{1, 0}
2	0, 0, 0, 0, 0, 1, 0	3	<code>navigate(0, {0, 1, 0})</code>	{4, 2}
3	0, 0, 0, 4, 0, 1, 0	2	<code>navigate(0, {0, 4, 0, 0, 0})</code>	{0, 3}
4	0, 0, 0, 4, 0, 1, 0	6	¹ <code>navigate(0, {0})</code>	{1, 0}
5	0, 0, 0, 4, 0, 1, 1	2	<code>navigate(0, {0, 4, 0, 1, 0})</code>	{8, 0}
6	0, 0, 8, 4, 0, 1, 1	0	<code>navigate(0, {8, 0})</code>	{3, 0}
7	3, 0, 8, 4, 0, 1, 1	2	<code>navigate(8, {3, 4, 0, 1, 0})</code>	{2, 2}
8	3, 0, 2, 4, 0, 1, 1	4	<code>navigate(0, {2, 4})</code>	{1, 1}
9	3, 0, 2, 4, 1, 1, 1	3	<code>navigate(4, {1, 1, 2})</code>	{-1, -1}

Тук роботът използва общо 6 различни цвята: 0, 1, 2, 3, 4 и 8 (имайте предвид, че 0 би се броило за използвано, дори ако роботът никога не е върнал цвят 0, тъй като всички върхове започват с цвят 0). Роботът е работил 9 итерации преди да прекрати работата си. Въпреки това, той е губи, тъй като е прекратил работата си, без да е посетил връх 1.

¹Обърнете внимание, че извикването на `navigate` на итерация 4 всъщност няма да се случи. Това е така, защото е еквивалентно на извикването на итерация 1, така че грейдърът просто ще използва повторно върнатата стойност на вашата функция от това извикване. Това обаче все още се брои за итерация на работа.



Примерен грейдър

Примерният грейдър не изпълнява многократно вашата програма, така че всички извиквания на `navigate` ще бъдат в едно и също изпълнение на вашата програма.

Входният формат е следният: Първо се прочита T (броят на подтестовите). След това за всеки подтест:

- ред 1: две цели числа – N и M ;
- ред $2 + i$ (за $0 \leq i < M$): две цели числа – A_i и B_i , което са двата върха, свързани от ребро i ($0 \leq A_i, B_i < N$).



След това примерният грейдър ще отпечата броя на различните цветове, използвани от вашето решение, и броя на итерациите, направени преди прекратяване на работата. Като алтернатива, ще отпечата съобщение за грешка, ако решението ви е неуспешно.

По подразбиране, примерният грейдър отпечатва подробна информация за това какво вижда и прави роботът при всяка итерация. Можете да деактивирате това, като промените стойността на `DEBUG` от `true` на `false`.