



Задатак Navigation

⌚ 3 секунде 💾 512 MB

Дат је **повезан неусмерени једноставан кактус граф**¹ са $N \leq 1000$ чворова и M грана. Чворови су обојени (боје су означене ненегативним целим бројевима од 0 до 1499). На почетку, сви чворови имају боју 0. **Детерминистички робот без меморије**² истражује граф крећући се од једног чвора до другог. Мора посетити све чворове барем једном, а затим ће се искључити.

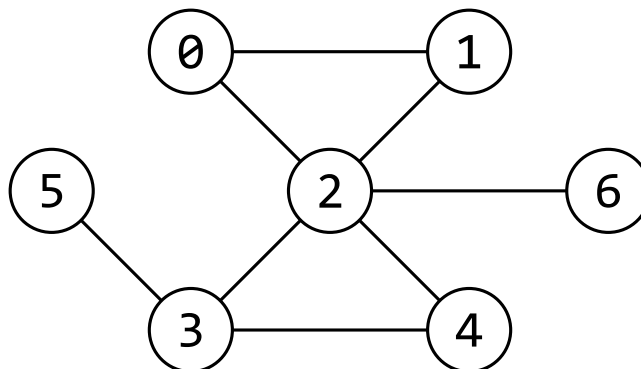
Робот почиње од произвољног чвора у графу. У сваком кораку, робот види боју тренутног чвора и боје свих суседних чворова у **неком фиксном редоследу за тренутни чвор** (тј. поновна посета чвору ће дати роботу исти низ суседних чворова, чак и ако су њихове боје другачије од претходних). Робот може да изврши једну од следеће две радње:

1. Доноси одлуку да се искључи.
2. Бира нову (можда исту) боју за тренутни чвор и бира суседни чвор на који ће се померити. Суседни чвор је јединствено означен индексом од 0 до $D - 1$, где је D број суседних чворова.

Ако изабере другу радњу, боја тренутног чвора се промени у другу (или се задржава постојећа) и робот се помера на изабрани суседни чвор. Ово се понавља док се робот не искључи или не достигне ограничење броја итерација. Робот ће победити ако посети све чворове, а затим се искључи у ограниченом броју итерација од $L = 3000$ корака (у супротном, робот губи).

Потребно је да одредите такву стратегију робота која ће решити проблем за било који кактус граф. Поред тога, требало би да минимизујете број различитих боја које ваше решење користи. Боја означена са 0 се увек рачуна као да је коришћена.

¹Повезани неусмерени једноставан кактус граф је повезани неусмерени једноставан граф (сваки чвор се може достићи из било ког другог чвора; гране су двосмерне; нема повратних петљи на чворовима или дуплих грана) у коме свака грана припада само једном простом циклусу (прост циклус је циклус који садржи сваки чвор највише једном). Пример је на слици испод.





²Робот се сматра детерминистичким и без меморије, ако његова радња зависи искључиво од тренутних улаза (тј. не чува информације између два корака) и увек бира исту радњу када му се дају исти улази.



Детаљи имплементације

Стратегија робота треба да буде имплементирана као следећа функција:

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

Функција узима као параметре боју тренутног чвора и боје свих суседних чворова (у одређеном редоследу). Функција треба да врати пар чији је први елемент нова боја тренутног чвора, а други елемент је индекс суседног чвора на који робот треба да се помери. Ако робот треба да се искључи, функција треба да врати пар $(-1, -1)$.

Позиви функција ће се понављати да би се изабрала акција робота. Пошто је робот детерминистички, ако је функција `navigate` претходно позвана са неким параметрима, неће бити поново позвана са истим параметрима; уместо тога, користиће се вредност коју је функција претходно вратила. Поред тога, сваки тест може да садржи $T \leq 5$ подтестова (различити графови и/или различите почетне позиције) који се могу извршавати истовремено (тј. ваш програм може да прима наизменичне позиве који се односе на различите подтестове). Коначно, позиви функције `navigate` могу се десити у **одвојеним извршавањима** вашег програма (али је могуће да се понекад догоде у оквиру истог извршавања). Укупан број извршавања вашег програма је $P = 100$. Због свега наведеног, ваш програм не би требало да преноси податке између различитих позива.



Ограничења

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



Бодовање

Део S поена које добијате за подзадатак зависи од C - максималног броја различитих боја које ваше решење користи (укључујући боју 0) на било ком тесту унутар тог подзадатка или било ког другог захтеваног подзадатка:

- Ако ваше решење не прође било који подтест, онда је $S = 0$.
- Ако је $C \leq 4$, онда је $S = 1.0$.
- Ако је $4 < C \leq 8$, онда је $S = 1.0 - 0.6 \frac{C-4}{4}$.
- Ако је $8 < C \leq 21$, онда је $S = 0.4 \frac{8}{C}$.
- Ако је $C > 21$, онда је $S = 0.15$.



Подзадаци

Подзадатак	Поени	Захтевани подзадаци	N	Додатна ограничења
0	0	—	≤ 300	Пример.
1	6	—	≤ 300	Граф је циклус. ¹
2	7	—	≤ 300	Граф је звезда. ²
3	9	—	≤ 300	Граф је пут. ³
4	16	2 – 3	≤ 300	Граф је стабло. ⁴
5	27	—	≤ 300	Сви чворови имају највише 3 суседна чвора, а чвор где робот почиње има 1 суседног чвора.
6	28	0 – 5	≤ 300	—
7	7	0 – 6	—	—

¹Граф који је циклус има гране: $(i, (i + 1) \bmod N)$ за $0 \leq i < N$.

²Граф који је звезда има гране: $(0, i)$ за $1 \leq i < N$.

³Граф који је пут има гране: $(i, i + 1)$ за $0 \leq i < N - 1$.

⁴Стабло је граф без циклуса.



Пример

Размотримо пример графа са слике у задатку, који има $N = 7$, $M = 8$ и гране $(0, 1)$, $(1, 2)$, $(2, 0)$, $(2, 3)$, $(3, 4)$, $(4, 2)$, $(3, 5)$ и $(2, 6)$. Поред тога, пошто је редослед елемената у листама суседних чворова битан, навешћемо их у следећој табели:



Чвор	Суседни чворови
0	2, 1
1	2, 0
2	0, 3, 4, 6, 1
3	4, 5, 2
4	2, 3
5	3
6	2

Претпоставимо да робот почиње на чвору 5. Онда имамо један могући (неуспешан) низ интеракција:

#	Боје	Чвор	Позив <code>navigate</code>	Повратна вредност
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}

Овде је робот користио укупно 6 различитих боја: 0, 1, 2, 3, 4 и 8 (имајте на уму да се боја 0 рачуна као коришћена чак и ако робот никада није вратио боју 0, јер су сви чворови на почетку обојени бојом 0). Робот је извршио 9 итерација пре него што се искључио. Међутим, није успео, јер се зауставио без посете чвору 1.

¹Имајте на уму да се позив функције `navigate` у итерацији 4 заправо неће догодити. Разлог је тај што је позив идентичан позиву у итерацији 1, тако да ће грејдер искористити вредност коју је вратила функција у том позиву. Међутим, ово се и даље рачуна као једна итерација робота.



Пример грејдера

Пример грејдера неће извршавати ваш програм више пута, тако да ће сви позиви на `navigate` бити у оквиру једног извршавања вашег програма.

Формат улаза је следећи: Прво се уноси T (број подтестова). Затим, за сваки подтест:

- линија 1: два цела броја - N и M ;
- линија $2 + i$ (за $0 \leq i < M$): два цела броја - A_i и B_i , који представљају два чвора повезана граном i ($0 \leq A_i, B_i < N$).

Пример грејдера ће затим исписати број различитих боја које је ваше решење користило и број итерација које су биле потребне пре него што се робот искључио. Ако је ваше решење било неуспешно, исписаће поруку о грешци.

По дифолту ће пример грејдера исписати детаљне информације о томе шта робот види и ради у свакој итерацији. Ово можете онемогућити променом вредности `DEBUG` из `true` у `false`.