



Task Navigation

 3 sec.  512 MB

Существует **связный неориентированный простой кактус**¹ с $N \leq 1000$ вершинами и M ребрами. Его вершины имеют цвета (обозначенные неотрицательными целыми числами от 0 до 1499). Изначально все вершины имеют цвет 0. **Детерминированный робот без памяти**² исследует граф, перемещаясь от вершины к вершине. Он должен посетить все вершины по крайней мере один раз, а затем завершить работу.

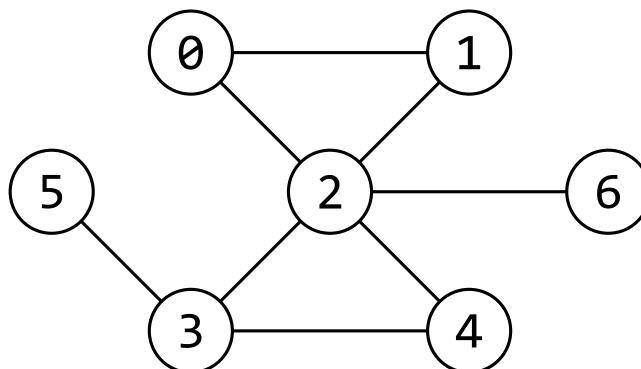
Робот начинает с какой-то вершины, которая может быть любой из графа. На каждом шаге он видит цвет своей текущей вершины и цвета всех соседних вершин **в некотором фиксированном для текущей вершины порядке** (т. е. повторное посещение вершины даст роботу ту же последовательность соседних вершин, даже если их цвета отличаются от прежних). Робот выполняет одно из следующих двух действий:

1. Завершает работу.
2. Выбирает новый (или, возможно, тот же) цвет для текущей вершины и соседнюю вершину, к которой необходимо перейти. Соседняя вершина выбирается индексом от 0 до $D - 1$, где D — количество соседних вершин.

Во втором случае текущая вершина перекрашивается (или, возможно, остается того же цвета), и робот переходит к выбранной соседней вершине. Это повторяется до тех пор, пока робот не завершит работу или не достигнет предела итераций. Робот выигрывает, если он посещает все вершины и затем завершает работу в пределах лимита итераций $L = 3000$ шагов (в противном случае он проигрывает).

Вы должны разработать стратегию для робота, которая сможет решить задачу на любого кактуса. Кроме того, вы должны постараться минимизировать количество различных цветов, используемых в вашем решении. Здесь цвет 0 всегда считается использованным.

¹Связный неориентированный простой кактус — это связный неориентированный простой граф (каждая вершина достижима из любой другой вершины; ребра двунаправлены; нет петель и кратных ребер), в котором каждое ребро принадлежит не более чем одному простому циклу (простой цикл — это цикл, в котором каждая вершина встречается не более одного раза). Пример показан на рисунке ниже.



²Робот называется детерминированным и без памяти, если его действия зависят только от текущих входных данных (т.е. он не сохраняет данные между итерациями) и он всегда выбирает одно и то же действие при одинаковых входных данных.



Implementation details

Стратегия робота должна быть реализована в виде следующей функции:

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

Она принимает в качестве аргументов цвет текущей вершины и цвета всех соседних вершин (в фиксированном порядке). Она должна возвращать пару, первый элемент которой является новым цветом для текущей вершины, а второй элемент — индексом соседней вершины, к которой должен перейти робот. Если же робот должен завершить работу, функция должна возвращать пару $(-1, -1)$.

Эта функция будет вызываться повторно для выбора следующих действий робота. Поскольку она детерминирована, если `navigate` уже вызывалась с некоторыми аргументами, она никогда не будет вызвана с теми же аргументами снова; вместо этого будет использовано значение, которое оно уже вернула до этого. Кроме того, каждый тест может содержать $T \leq 5$ подтестов (различные графы и/или начальные позиции), и они могут выполняться одновременно (т. е. ваша программа может получать поочередные вызовы по разным подтестам). Наконец, вызовы `navigate` могут происходить в **отдельных выполнениях** вашей программы (но иногда они могут происходить и в одном и том же выполнении). Общее количество выполнений вашей программы составляет $P = 100$. Это сделано для того чтобы ваша программа не могла передавать информацию между разными вызовами функции.



Constraints

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



Scoring

Часть S баллов за подзадачу, которую вы получаете, зависит от C — максимального числа различных цветов, используемых в вашем решении (включая цвет 0) в любом тесте этой подзадачи или любой другой требуемой подзадачи:

- Если ваше решение не проходит какой-либо подтест, то $S = 0$.
- Если $C \leq 4$, то $S = 1,0$.
- Если $4 < C \leq 8$, то $S = 1,0 - 0,6 \frac{C-4}{4}$.
- Если $8 < C \leq 21$, то $S = 0,4 \frac{8}{C}$.
- Если $C > 21$, то $S = 0,15$.



Subtasks

Подзадача	Баллы	Требуемые подзадачи	N	Дополнительные ограничения
0	0	—	≤ 300	Примеры.
1	6	—	≤ 300	Граф является циклом. ¹
2	7	—	≤ 300	Граф является звездой. ²
3	9	—	≤ 300	Граф является путем. ³
4	16	2 – 3	≤ 300	Граф является деревом. ⁴
5	27	—	≤ 300	Все вершины имеют не более 3 соседних вершин, а вершина, с которой начинает робот, имеет ровно 1 соседнюю вершину.
6	28	0 – 5	≤ 300	—
7	7	0 – 6	—	—

¹Граф является циклом если имеет ребра: $(i, (i + 1) \bmod N)$ для $0 \leq i < N$.

²Граф является звездой если имеет ребра: $(0, i)$ для $1 \leq i < N$.

³Граф является путем если имеет ребра: $(i, i + 1)$ для $0 \leq i < N - 1$.

⁴Дерево — это связный граф без циклов.



Example

Рассмотрим пример графа из изображения в условии, который имеет $N = 7$, $M = 8$ и ребра $(0, 1)$, $(1, 2)$, $(2, 0)$, $(2, 3)$, $(3, 4)$, $(4, 2)$, $(3, 5)$ и $(2, 6)$. Кроме того, поскольку порядок соседних вершин имеет значение, мы приводим его в этой таблице:

Node	Adjacent nodes
0	2, 1
1	2, 0
2	0, 3, 4, 6, 1
3	4, 5, 2
4	2, 3
5	3
6	2

Предположим, что робот начинает с вершины 5. Тогда ниже приведена одна из



возможных (неудачных) последовательностей вызовов:

#	Цвета	Текущая вершина	Вызов <code>navigate</code>	Ответ функции
1	0, 0, 0, 0, 0, 0, 0	5	<code>navigate(0, {0})</code>	<code>{1, 0}</code>
2	0, 0, 0, 0, 0, 1, 0	3	<code>navigate(0, {0, 1, 0})</code>	<code>{4, 2}</code>
3	0, 0, 0, 4, 0, 1, 0	2	<code>navigate(0, {0, 4, 0, 0, 0})</code>	<code>{0, 3}</code>
4	0, 0, 0, 4, 0, 1, 0	6	¹ <code>navigate(0, {0})</code>	<code>{1, 0}</code>
5	0, 0, 0, 4, 0, 1, 1	2	<code>navigate(0, {0, 4, 0, 1, 0})</code>	<code>{8, 0}</code>
6	0, 0, 8, 4, 0, 1, 1	0	<code>navigate(0, {8, 0})</code>	<code>{3, 0}</code>
7	3, 0, 8, 4, 0, 1, 1	2	<code>navigate(8, {3, 4, 0, 1, 0})</code>	<code>{2, 2}</code>
8	3, 0, 2, 4, 0, 1, 1	4	<code>navigate(0, {2, 4})</code>	<code>{1, 1}</code>
9	3, 0, 2, 4, 1, 1, 1	3	<code>navigate(4, {1, 1, 2})</code>	<code>{-1, -1}</code>

Здесь робот использовал в общей сложности 6 различных цветов: 0, 1, 2, 3, 4 и 8 (обратите внимание, что 0 считался бы использованным, даже если бы робот никогда не возвращал цвет 0, поскольку все узлы начинают с цвета 0). Робот проработал 9 итераций, прежде чем завершить работу. Однако он провалился, поскольку завершил работу, не посетив узел 1.

¹Обратите внимание, что вызов `navigate` на итерации 4 на самом деле не произойдет. Это потому, что он эквивалентен вызову на итерации 1, поэтому грейдер просто повторно использует возвращаемое значение вашей функции из этого вызова. Однако это все равно считается итерацией робота.



Sample grader

Грейдер не запускает несколько запусков вашей программы, поэтому все вызовы `navigate` будут находиться в одном запуске вашей программы.

- строка 1: два целых числа — N и M ;
- строка $2 + i$ (для $0 \leq i < M$): два целых числа — A_i и B_i , вершины соединяемыми ребром i ($0 \leq A_i, B_i < N$).

Затем грейдер выведет количество различных цветов, использованных в вашем решении, и количество итераций, необходимых для его завершения. В противном случае, если ваше решение не прошло проверку, будет выведено сообщение об ошибке.

По умолчанию, грейдер выводит подробную информацию о том, что робот видит и делает на каждой итерации. Вы можете отключить эту функцию, изменив значение `DEBUG` с `true` на `false`.