

Task 1. Монополия

Дени очень нравится играть в игру Монополия. Но продолжительность одной игры очень большая — от 5 до 7 часов. Итак, Дени задумалась об изменении классических правил. Игроки в Монополии обычно перемещаются в одном направлении из клетки в клетку и через некоторое время возвращаются к началу, начинают снова и так далее. В новой версии движение тоже будет из клетки в клетку, но, в отличие от текущей версии, может быть несколько возможностей для следующего хода. Дени хочет найти такие направленные переходы между клетками, чтобы, начав игру в любой клетке, игрок никогда не мог вернуться в клетку, где он был, независимо от того, как двигается (следуя правилам, конечно). Таким образом, продолжительность игры будет короче.

Она уже начала делать новое поле — выбрала количество клеток N (клетки пронумерованы от 1 до N) и составила список с M переходами (каждый переход имеет направление, и нет перехода, который соединяет клетку с собой). Если клетка i связана с клеткой j , то нет прямого перехода в противоположном направлении (из клетки j в клетку i) а также нет других прямых переходов из клетки i в клетку j . Дени уже решила, что новое поле готово, но внезапно заметила, что условие, которого она добивается (при перемещении по переходам из клетки в клетку нельзя вернуться в ранее посещенную клетку), не подходит для ее списка переходов. Сначала она решила удалить некоторые переходы, но это привело бы к переписыванию списка переходов, который может быть очень длинным. Вот почему Дени решила обратить направление некоторых направленных переходов.

Task

Вы часто играете в Монополию с Дени. Поэтому вы хотите помочь ей, написав программу **monopoly**, которая должна указывать, какие переходы надо обратить, чтобы описанное условие было выполнено. Программа должна содержать функцию *find_reverse*, которая будет скомпилирована с программой жюри.

Implementation details

У функции *find_reverse* должен быть следующий прототип:

```
std::string find_reverse (int N, int M, int connections[][2]);
```

Он вызывается только один раз программой жюри с тремя параметрами: N — количество клеток на новом поле, M — количество направленных переходов в списке Дени и массив *переходов*, у которого M строк, каждая состоит из двух чисел x и y — начальная и конечная клетки для направленного перехода. Эта функция должна возвращать двоичную строку длиной M — в порядке массива *переходов*, для каждого перехода вы должны указать «1» в строке, если переход нужно обратить, и «0» в противном случае. Если существует несколько решений, вы можете вернуть любое из них.

Вам надо отправить в систему файл **monopoly.cpp**, который содержит реализацию функции *find_reverse*. Файл может содержать другие функции и код, необходимые для вашей программы, но он **не должен содержать** главную функцию — *main*. Кроме того, вы не должны пытаться читать из стандартного ввода или пытаться писать в стандартный вывод!

Constraints

- ♣ $3 \leq N \leq 5 \times 10^5$
- ♣ $3 \leq M \leq 1.5 \times 10^6$

Subtasks

| Subtask | Points | N | M | Further constraints |
|---------|--------|----------------------|------------------------|---------------------|
| 1 | 0 | — | — | Пример |
| 2 | 15 | ≤ 7 | ≤ 21 | — |
| 3 | 40 | $\leq 10^3$ | $\leq 5 \times 10^3$ | — |
| 4 | 25 | $\leq 10^5$ | $\leq 5 \times 10^5$ | — |
| 5 | 20 | $\leq 5 \times 10^5$ | $\leq 1.5 \times 10^6$ | — |

Чтобы получить баллы за подзадачу, решение должно пройти все тесты подзадачи.

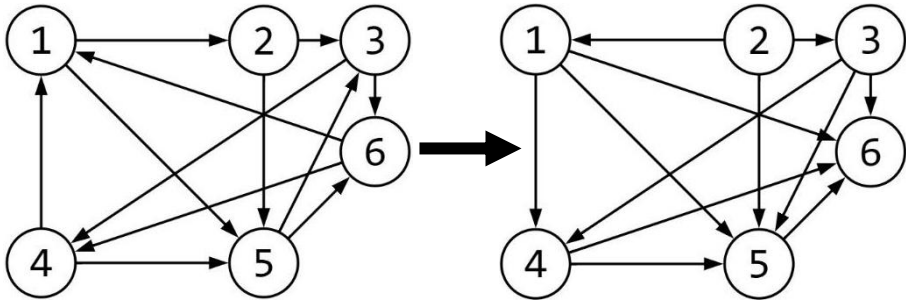
Local testing

Для локального тестирования вам предоставляется файл **Lgrader.cpp**. Вы должны поместить его в ту же папку, что и вашу программу **monopoly.cpp**, и скомпилировать вместе файлы **Lgrader.cpp** и **monopoly.cpp**. Таким образом вы создадите программу для проверки корректности вашей функции. Программе потребуется следующая последовательность данных из стандартного ввода:

- в первой строке: два целых числа — количество пробелов N и количество переходов M на новом поле.
- в оставшихся M строках: в каждой строке должно быть два целых числа x и y — начальная и конечная клетки для каждого направленного перехода.

Выводом программы будет найденная вами двоичная строка.

Example of local testing

| Input | Output | Explanation |
|--|--------------|---|
| 6 12 1 2 1 5 2 3 2 5 3 4 3 6 4 1 4 5 5 3 5 6 6 1 6 4 | 100000101011 | <p>Обращенные переходы: (1,2), (4, 1), (5, 3), (6, 1), (6, 4).</p>  <p>На рисунке выше показаны клетки. Сначала — с направленными переходами из списка Дени, а потом — после изменения направления переходов с соответствующей «1» на выходе. Действительно, вначале, используя направленные переходы (1, 2), (2, 3), (3, 4) и (4, 1), можно начать с клетки 1 и, двигаясь по этим направленным переходам, вернуться к этой же клетке. Видно, что на новом поле условие Дени выполнено. Обратите внимание, что есть и другие допустимые решения:</p> <p>(4, 1), (5, 3), (6, 1), (6, 4) (1, 5), (2, 3), (2, 5), (3, 4), (3, 6), (4, 5), (5, 6)</p> |