

Задача А. Произведение графов

Имя входного файла: `graphprod.in`
Имя выходного файла: `graphprod.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Пусть дан ориентированный ациклический граф. Стандартная игра на графе заключается в следующем: изначально на одной из вершин графа (называемой начальной позицией) стоит фишка. Двое игроков по очереди двигают её по рёбрам. Проигрывает тот, кто не может сделать ход.

В теории игр часто рассматриваются более сложные игры. Например, прямое произведение двух игр на графах. Прямое произведение игр — это следующая игра: изначально на каждом графе в начальной позиции стоит по фишке. За ход игрок двигает обе фишки по рёбрам (каждую фишку двигает в собственном графе). Проигрывает тот, кто не может сделать ход. То есть тот, кто не может сделать ход хотя бы в одной игре.

Ваша задача — опеределить, кто выиграет при правильной игре.

Формат входного файла

На первой строке будут даны числа N_1 и M_1 — количество вершин и рёбер в первом графе ($1 \leq N_1, M_1 \leq 100\,000$). На следующих M_1 строках содержится по два числа x и y ($1 \leq x, y \leq N_1$).

В следующих $M_2 + 1$ строках задан второй граф в том же формате.

Заканчивается входной файл списком пар начальных вершин, для которых нужно решить задачу. На первой строке задано число T ($1 \leq T \leq 100\,000$) — количество пар начальных вершин. В следующих T строках указаны пары вершин v_1 и v_2 ($1 \leq v_1 \leq N_1, 1 \leq v_2 \leq N_2$).

Учтите, что в графах могут быть кратные рёбра.

Формат выходного файла

На каждую из T пар начальных вершин выведите строку “**first**”, если при правильной игре выиграет первый, и “**second**”, если второй.

Примеры

<code>graphprod.in</code>	<code>graphprod.out</code>
3 2	first
1 2	second
2 3	
2 1	
1 2	
2	
2 1	
3 2	

Задача В. Терминатор

Имя входного файла: `terminator.in`
Имя выходного файла: `terminator.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Два игрока играют в настольную игру. Игровое поле представляет собой квадратный лабиринт, 8×8 клеток. В некоторых клетках располагаются стенки. Один игрок управляет фишкой-терминатором, а второй — фишкой-беглецом. Игроки ходят по очереди, ходы пропускать нельзя (гарантируется, что ход всегда возможен). За один ход игрок может переместить свою фишку в любую из свободных клеток, расположенных рядом с исходной по горизонтали, вертикали или по диагонали (то есть ходом короля). Терминатор, кроме того, может стрелять в беглеца ракетами. Выстрел идет по прямой в любом направлении по горизонтали, вертикали или диагонали. Если беглец оказывается на линии выстрела терминатора и не прикрыт стенками, то терминатор незамедлительно делает выстрел (вне зависимости от того, чей ход), и беглец проигрывает. Начальное положение фишек задано. Первый ход делает беглец. Он выигрывает, если сделает ход с восьмой строки за пределы игрового поля, так как остальные границы поля окружены стенками.

Вопрос задачи: может ли беглец выиграть при оптимальной игре обеих сторон?

Формат входного файла

Во входном файле задано игровое поле. Свободная клетка обозначена цифрой 0, а клетка со стенкой — цифрой 1. Клетка, в которой находится беглец, обозначена цифрой 2, а клетка с терминатором — цифрой 3.

Формат выходного файла

В выходной файл выведите число 1, если беглец выигрывает, и -1 — в противном случае.

Примеры

<code>terminator.in</code>	<code>terminator.out</code>
01000000 10100000 31100000 00020000 00000000 00000000 00000000 00000000	-1

Задача С. Вас снова замяукали!

Имя входного файла:	labyrinth.in
Имя выходного файла:	labyrinth.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Два котёнка попали в запутанный лабиринт со множеством комнат и переходов между ними. Котятя долго по нему плутали, обошли все комнаты по много раз, нашли выход (да даже и не один, а несколько), в общем, изучили там всё, что смогли. Теперь этот лабиринт котятя используют в своих играх.

Чаще всего котятя играют в следующую игру: начиная в какой-то комнате лабиринта, котятя поочерёдно выбирают, в какую из комнат им перейти. Котятя изначально находятся в одной комнате и ходят вместе. Как только котёнок, который должен выбрать следующую комнату, не может этого сделать, он признаётся проигравшим. Обычно в таких играх выигрывающий игрок стремится выиграть как можно быстрее, а проигрывающий стремится как можно дольше оттянуть свое поражение. Но у котят свои представления о победе и поражении. Если котёнок знает, что, начиная из текущей комнаты, он выиграет (вне зависимости от действий другого котёнка), то он стремится играть как можно дольше, чтобы продлить себе удовольствие от выигрыша (естественно, при этом выигрывающий котёнок должен гарантировать себе, что будет постоянно уверен в выигрыше). Котёнок, который знает, что проиграет (при условии, конечно, что другой котёнок будет действовать оптимально), старается проиграть как можно быстрее, чтобы начать новую игру, в которой и взять реванш.

Если котятя будут ходить бесконечно долго, но никто из них не сможет выиграть, то котятя считают игру завершившейся вничью и замяукивают Вас.

Вас попросили для каждой комнаты в лабиринте узнать, выиграет или проиграет котёнок, начинающий ходить из данной комнаты. Если котёнок, начинающий из этой комнаты, выигрывает, требуется узнать максимальное количество ходов, которое он сможет играть, если же проигрывает — минимальное количество, которое ему придётся играть.

Формат входного файла

В первой строке ввода находятся два числа n и m — число комнат и переходов между комнатами в лабиринте ($1 \leq n \leq 100\,000$, $0 \leq m \leq 100\,000$). Далее следует m строк с описаниями переходов. Описание перехода состоит из двух чисел a и b , означающих, что котёнок, начинающий игру в комнате с номером a , может выбрать комнату b в качестве следующей.

Формат выходного файла

Выведите n строк — для каждой комнаты результат игры для котёнка, который начнет игру из этой комнаты. Если игра закончится вничью, выведите «DRAW». Если начинающий котёнок выигрывает, выведите «WIN K », где K — количество ходов, которые сможет играть выигрывающий котёнок. Если котёнок сможет играть сколь угодно долго, сохраняя возможность в любой момент выиграть, выведите «WIN INF». Если котёнок, начинающий из этой комнаты, проиграет, выведите «LOSE K », где K — количество ходов, которые придется играть проигрывающему котёнку. Если же котёнку придется играть сколь угодно долго, при том, что его соперник сможет в любой момент выиграть, выведите «LOSE INF».

Примеры

labyrinth.in	labyrinth.out
4 4 1 2 1 3 2 4 3 4	LOSE 2 WIN 1 WIN 1 LOSE 0
6 6 1 2 2 3 3 4 4 1 4 5 5 6	DRAW DRAW DRAW DRAW WIN 1 LOSE 0
6 6 1 2 2 3 3 4 4 1 2 6 4 5	LOSE INF WIN INF LOSE INF WIN INF LOSE 0 LOSE 0

Задача D. Вариация нима

Имя входного файла: `varnim.in`
Имя выходного файла: `varnim.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

На столе лежат n кучек камней: a_1 камней в первой кучке, a_2 камней во второй, \dots , a_n в n -ой. Двое играют в игру, делая ходы по очереди. За один ход игрок может либо взять произвольное ненулевое количество камней (возможно, все) из одной любой кучки, либо произвольным образом разделить любую существующую кучку, в которой не меньше двух камней, на две непустые кучки. Проигрывает тот, кто не может сделать ход. Кто выигрывает при правильной игре?

Формат входного файла

В первой строке задано целое число t — количество тестов ($1 \leq t \leq 100$). Следующие t строк содержат сами тесты. Каждая из них начинается с целого числа n — количества кучек ($1 \leq n \leq 100$). Далее следует n целых чисел a_1, a_2, \dots, a_n через пробел — количество камней в кучках ($1 \leq a_i \leq 10^9$).

Формат выходного файла

Выведите t строк; в i -ой строке выведите “FIRST”, если в i -ом тесте при правильной игре выигрывает первый игрок, и “SECOND”, если второй.

Примеры

<code>varnim.in</code>	<code>varnim.out</code>
3	FIRST
1 1	SECOND
2 1 1	FIRST
3 1 2 3	

Задача Е. Сумма игр

Имя входного файла: `smith.in`
Имя выходного файла: `smith.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Пусть дан ориентированный граф. Стандартная игра на графе заключается в следующем: изначально на одной из вершин графа (называемой начальной позицией) стоит фишка. Двое игроков по очереди двигают её по рёбрам. Проигрывает тот, кто не может сделать ход.

В теории игр часто рассматриваются более сложные игры. Например, прямая сумма двух игр на графах. Прямая сумма игр — это следующая игра: изначально на каждом графе в начальной позиции стоит по фишке. За ход игрок выбирает любую фишку и двигает по ребру соответствующего графа. Проигрывает тот, кто не может сделать ход.

Ваша задача — опеределить, кто выиграет при правильной игре.

Формат входного файла

На первой строке будут даны числа N_1 и M_1 — количество вершин и рёбер в первом графе ($1 \leq N_1, M_1 \leq 10\,000$). На следующих M_1 строках содержится по два числа x и y ($1 \leq x, y \leq N_1$).

В следующих $M_2 + 1$ строках задан второй граф в том же формате.

Заканчивается входной файл списком пар начальных вершин, для которых нужно решить задачу. На первой строке задано число T ($1 \leq T \leq 100\,000$) — количество пар начальных вершин. В следующих T строках указаны пары вершин v_1 и v_2 ($1 \leq v_1 \leq N_1, 1 \leq v_2 \leq N_2$).

Формат выходного файла

На каждую из T пар начальных вершин выведите строку “**first**”, если при правильной игре выиграет первый, “**second**”, если второй, или “**draw**”, если будет ничья.

Примеры

smith.in	smith.out
3 2	first
1 2	second
2 3	
2 1	
1 2	
2	
1 1	
3 2	