Содержание

В некоторых задачах большой ввод и вывод, в некоторых других — STL, который активно использует динамическую память (set-ы, map-ы). Смотрите как пользоваться быстрым вводом-выводом и переопределять стандартный аллокатор.

Задачи

Задача 10A. Одна кучка [0.1 sec, 256 mb]

Два игрока играют в игру. На столе лежит кучка из N камней. Двое ходят по очереди. За ход можно взять $a_1, a_2, \ldots, a_k$ камней. Проигрывает тот, кто не может сделать ход. Определите победителя!

Формат входных данных

В первой строке записано число k. Во второй строке k чисел — $a_1, a_2, \cdots, a_k$. В третьей строке идет число m — количество различных N, для каждого из которых требуется определить победителя. В четвертой строке m чисел — $N_1, N_2, \cdots, N_m$.

Ограничения: $1 \leqslant k \leqslant 20, m \leqslant 10^4, 1 \leqslant N_i, a_i \leqslant 10^6$.

Формат выходных данных

Выведите m строк, в каждой ответ на вопрос "кто выиграет" — `First` или `Second`.

Пример

stdin	stdout
3 1 2 3 8 1 2 3 4 5 6 7 8	First First First Second First First First Second

Задача 10B. Ретроанализ для маленьких [0.4 sec, 256 mb]

Дан ориентированный весёлый граф из n вершин и m ребер. Оля и Коля в игру. Изначально фишка стоит в вершине i. За ход можно передвинуть фишку по любому из исходящих ребер. Тот, кто не может сделать ход, проигрывает. Ваша задача — для каждой вершины i определить, кто выиграет при оптимальной игре обоих.

Формат входных данных

Входные данные состоят из одного или нескольких тестов. Каждый тест содержит описание весёлого ориентированного графа. Граф описывается так: на первой строке два целых числа n ($1 \leqslant n \leqslant 300\,000$) и m ($1 \leqslant m \leqslant 300\,000$). Следующие m строк содержат ребра графа, каждое описывается парой целых чисел от 1 до n. Пара a b обозначает, что ребро ведет из вершины a в вершину b. В графе могут быть петли, могут быть кратные ребра. Сумма n по всем тестам не превосходит 300 000, сумма m по всем тестам также не превосходит 300 000.

Формат выходных данных

Для каждого теста выведите для каждой вершины `FIRST`, `SECOND` или `DRAW` в зависимости от того, кто выиграет при оптимальной игре из этой вершины. Ответы к тестам разделяйте пустой строкой.

Примеры

stdin	stdout
5 5 1 2 2 3 3 1 1 4 4 5 2 1 1 2 4 4 1 2 2 3 3 1 1 4	DRAW DRAW DRAW FIRST SECOND FIRST SECOND FIRST FIRST SECOND SECOND

Замечание

Помните, как писать `bfs`? Кстати, можно и `dfs`...

Задача 10C. Вариация Нима [0.1 sec, 256 mb]

На столе лежат n кучек камней: a_1 камней в первой кучке, a_2 камней во второй, ..., 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 входных данных

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

Формат выходных данных

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

Пример

stdin	stdout
3 1 1 2 1 1 3 1 2 3	FIRST SECOND FIRST

Задача 10D. Комфортабельная рассадка [0.1 sec, 256 mb]

В парке стоит скамейка на n мест. Изначально она пуста. Две команды играют в игру, по очереди *комфортабельно* сажая на скамейку по одному человеку. Человек садится *комфортабельно*, если на соседних местах ни справа, ни слева от него непосредственно никого нет. Проигрывает та команда, которая не может *комфортабельно* посадить очередного человека на своём ходу.

Дано n — количество мест на скамейке. Какая команда выигрывает при правильной игре?

Формат входных данных

В первой строке входного файла задано целое число n ($1 \leqslant n \leqslant 100$).

Формат выходных данных

В первой строке выходного файла выведите "`FIRST`", если при правильной игре выигрывает первая команда, и "`SECOND`", если вторая.

Примеры

stdin	stdout
2	FIRST
4	SECOND

Бонусные задачи

Задача 10E. Не-не-не-нечестная игра [0.2 sec, 256 mb]

Дэвид Блейн и Вася сидят на уроке математики и скучают. Через некоторое время Дэвид Блейн предлагает поиграть в следующую игру:

- он записывает на листочке некоторое большое целое число и ходит первым;
- игроки ходят по очереди;
- на каждом ходу игрок обязан разделить текущее число на его простой делитель p. На этом же ходу игрок может умножить (а может и нет) результат деления на простое число q $(1 < q < p)$;
- проигрывает тот, кто не может сделать ход.

Мягко говоря, Вася не очень доверяет Дэвиду Блейну и боится, что тот выписывает только проигрышные для Васи начальные числа. Помогите ему определить, так это или нет.

Формат входных данных

Первая строка входного файла содержит единственное число n $(1 \leqslant n \leqslant 10^{12})$.

Формат выходных данных

В выходной файл выведите `Vasya`, если у Васи есть выигрышная стратегия, независимо от ходов Дэвида Блейна. Иначе выведите `David`.

Пример

stdin	stdout
16	Vasya

Задача 10F. Игры на графе [0.6 sec, 256 mb]

Коля и Петя любят играть в следующую игру на лекциях по теории сложности. Они рисуют двудольный граф G на листке бумаги и ставят фишку в одну из вершин графа. Далее они ходят по очереди, Коля ходит первым.

Ход состоит из перемещения фишки по ребру в графе. После перемещения фишки вершина, из которой фишка только что ушла, удаляется из графа вместе со всеми инцидентными ей рёбрами. Проигрывает игрок, который не может ходить.

Вам дан граф, который нарисовали Коля и Петя. Для каждой вершины графа определите, кто выиграет, если изначально фишка находится в этой вершине. Предполагайте, что и Коля, и Петя играют оптимально.

Формат входных данных

Первая строка файла содержит три целых числа n_1, n_2, m – число вершин в одной доли, второй доли и число рёбер, соответственно ($1 \leqslant n_1, n_2 \leqslant 500$, $0 \leqslant m \leqslant 50\,000$). Следующие m строк описывают рёбра графа – каждая срока содержит номера вершин, которые рёбро соединяет. Вершины в каждой из долей нумеруются с 1.

Формат выходных данных

Выведите две строке. На первой строке выведите n_1 символов, i-й символ должен быть '`N`', если при начале игры из i-й вершины первой доли выиграет Коля, или '`P`', если выиграет Петя. Аналогично во второй строке выведите n_2 символов – результаты игры для второй доли графа.

Пример

stdin	stdout
3 3 5 1 1 1 2 1 3 2 1 3 1	NPP NPP

Задача 10G. Вас снова замяукали! [0.1 sec, 256 mb]

Два котёнка попали в запутанный лабиринт со множеством комнат и переходов между ними. Котята 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 котята используют в своих играх.

Чаще всего котята играют в следующую игру: начиная в какой-то комнате лабиринта, котята поочерёдно выбирают, в какую из комнат им перейти. Котята 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 котята будут ходить бесконечно долго, но никто из них не сможет выиграть, то котята 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 входных данных

В первой строке ввода находятся два числа n и m — число комнат и переходов между комнатами в лабиринте ($1 \leqslant n \leqslant 100\,000$, $0 \leqslant m \leqslant 100\,000$). 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 выходных данных

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 выиграет, 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

stdin	stdout
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