Содержание

В некоторых задачах большой ввод и вывод. Пользуйтесь быстрым вводом-выводом.

В некоторых задачах нужен STL, который активно использует динамическую память (set-ы, map-ы) переопределение стандартного аллокатора ускорит вашу программу.

Задачи

Задача 12A. Альфа Дерево [1.5 sec, 256 mb]

У вас есть полное бинарное дерево глубины n ($0 \leqslant n \leqslant 30$).

В дереве 2^n листьев, они пронумерованы слева направо числами от 0 до 2^n-1.

В i-м листе записано число $x_i = (ai^2 + bi + c) \bmod m$.

Есть фишка, которая изначально находится в корне дерева. Двое играют в игру, двигая фишку вниз по дереву. Когда фишка достигает листа дерева, игра заканчивается. Цель первого игрока – максимизировать число в листе, цель второго – минимизировать.

Формат входных данных

Числа n, a, b, c, m. При этом $10 \leqslant m \leqslant 10^9$.

Все a, b, c сгенерированы равномерным распределением на $[0, m)$.

Формат выходных данных

Выведите результат игры при оптимальной игре обоих.

Примеры

stdin	stdout
3 10 7 9 20	11

Замечание

Взятие остатка по модулю – небыстрая операция. Чем их меньше, тем лучше.

Задача 12B. 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

Задача 12C. Жестокая задача [0.1 sec, 256 mb]

Штирлиц и Мюллер стреляют по очереди. В очереди n человек, стоящих друг за другом. Каждым выстрелом убивается один из стоящих. Кроме того, если у кого-то из стоящих в очереди убиты все его соседи, то этот человек в ужасе убегает. Проигрывает тот, кто не может ходить. Первым стреляет Штирлиц. Требуется определить, кто выиграет при оптимальной игре обеих сторон, и если победителем будет Штирлиц, то найти все возможные первые ходы, ведущие к его победе.

Формат входных данных

Входной файл содержит единственное число n ($2 \leqslant n \leqslant 5\,000$) — количество человек в очереди.

Формат выходных данных

Если выигрывает Мюллер, выходной файл должен состоять из единственного слова `Mueller`. Иначе в первой строке необходимо вывести слово `Schtirlitz`, а в последующих строках — номера людей в очереди, которых мог бы первым ходом убить Штирлиц для достижения своей победы. Номера необходимо выводить в порядке возрастания.

Пример

stdin	stdout
3	Schtirlitz 2
4	Mueller
5	Schtirlitz 1 3 5

Задача 12D. 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

Гробы

Задача 12E. Игры на графе [0.5 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