

## Содержание

<b>Must have</b>	<b>2</b>
Задача 31A. Ретроанализ для маленьких [0.5 sec, 256 mb]	2
Задача 31B. Одна кучка [0.3 sec, 256 mb]	3
<b>Задачи здорового человека</b>	<b>4</b>
Задача 31C. Жестокая задача [0.2 sec, 256 mb]	4
Задача 31D. Две кучки [0.5 sec, 256 mb]	5
Задача 31E. Произведение графов [0.4 sec, 256 mb]	6
Задача 31F. Choco. Шоколадка [0.1 sec, 256 mb]	7
<b>Для искателей острых ощущений</b>	<b>8</b>
Задача 31G. Игры на графе [0.3 sec, 256 mb]	8
Задача 31H. Не-не-не-нечестная игра [0.2 sec, 256 mb]	10
Задача 31I. Битва за кольцо [0.1 sec, 256 mb]	11
Задача 31J. Терминатор [0.2 sec, 256 mb]	12
<b>Для мастеров AI</b>	<b>13</b>
Задача 31K. Игра в пешки [0.2 sec, 256 mb]	14
Задача 31L. Smith [0.2 sec, 256 mb]	15

---

У вас не получается читать/выводить данные?

Воспользуйтесь примерами (**c++**) (**python**).

Обратите внимание, входные данные лежат в **стандартном потоке ввода** (он же stdin), вывести ответ нужно в **стандартный поток вывода** (он же stdout).

Обратите внимание на GNU C++ компиляторы с суффиксом inc.

Подни можно пользоваться **дополнительной библиотекой** (optimization.h).

То есть, использовать быстрый ввод-вывод: **пример про числа и строки**.

И быструю аллокацию памяти (ускоряет vector-set-map-весь-STL): **пример**.

Для тех, кто хочет разобраться, как всё это работает.

Короткая версия быстрого ввода-вывода (**тык**) и короткая версия аллокатора (**тык**).

## Must have

### Задача 31А. Ретроанализ для маленьких [0.5 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  $n$  ( $1 \leq n \leq 300\,000$ ) и  $m$  ( $1 \leq m \leq 300\,000$ ). 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	DRAW
1 2	DRAW
2 3	DRAW
3 1	FIRST
1 4	SECOND
4 5	FIRST
2 1	SECOND
1 2	FIRST
4 4	FIRST
1 2	SECOND
2 3	SECOND
3 1	
1 4	

#### Замечание

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

**Задача 31В. Одна кучка [0.3 sec, 256 mb]**

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

**Формат входных данных**

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

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

**Формат выходных данных**

Выведите  $m$  строк, в каждой ответ на вопрос “кто выиграет” — **First** или **Second**.

**Пример**

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

## Задачи здорового человека

### Задача 31С. Жестокая задача [0.2 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  $n$  ( $2 \leq n \leq 5000$ ) — 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 31D. Две кучки [0.5 sec, 256 mb]**

Два игрока играют в игру. На столе лежат две кучки: в первой  $a$  камней, во второй —  $b$ . Игроки ходят по очереди. Каждым ходом игрок выбирает одну кучку и берет какое-то количество камней из нее. Первый игрок может брать  $a_1, a_2, \dots, a_k$  камней, второй —  $b_1, b_2, \dots, b_l$ . Проигрывает тот, кто не может сделать ход. Определите победителя!

**Формат входных данных**

В первой строке записаны  $a$  и  $b$ . Во второй строке записаны  $k$  и последовательность  $a_i$ , на третьей —  $l$  и  $b_i$ .  $1 \leq a, b \leq 1000$ ,  $1 \leq k, l \leq 10$ ,  $1 \leq a_i, b_j \leq 1000$ .

**Формат выходных данных**

Если выигрывает первый игрок, выведите **First**. Иначе выведите **Second**.

**Пример**

stdin	stdout
2 2 2 1 2 1 1	First
2 2 1 1 2 1 2	Second

### Задача 31Е. Произведение графов [0.4 sec, 256 mb]

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 входных данных

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 выходных данных

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

#### Пример

stdin	stdout
3 2	first second
1 2	
2 3	
2 1	
1 2	
2	
1 1	
3 2	

### Задача 31F. Choso. Шоколадка [0.1 sec, 256 mb]

Двое играют в такую игру: перед ними лежит шоколадка размера  $N \times M$ . Игроки ходят по очереди. За ход разломить любой имеющийся кусок шоколадки на 2 «непустых» куска, при этом запрещено ломать куски размером не больше, чем  $1 \times S$  (т.е. нельзя ломать куски, у которых один размер равен 1, а другой не превосходит  $S$ ), куски можно поворачивать. Ломать, конечно, можно только вдоль линий, нанесенных на шоколадке, т.е. после разлома должны получаться два прямоугольника с целочисленными ненулевыми сторонами.

Проигрывает тот, кто не может сделать хода.

#### Формат входных данных

Во входном файле находятся три целых числа  $N$ ,  $M$  и  $S$  ( $0 < N, M, S \leq 100$ ).

#### Формат выходных данных

Выведите в выходной файл одно число 1 или 2 — номер игрока, который выигрывает при правильной игре.

#### Пример

stdin	stdout
2 2 1	1

## Для искателей острых ощущений

### Задача 31G. Игры на графе [0.3 сек, 256 mb]

Противник начал e2–e4. Я  
проанализировал его архитектуру и  
сдался

---

Из мемуаров 20-го чемпиона мира  
Фрица Рыбкина

Прибыв на место, Ааз тут же потребовал организовать совещание букмекеров, на котором он изложит свой план.

— Главная задача, — начал Ааз своё выступление перед букмекерами, — научиться использовать достижения прогресса. Мы планируем запуск множества новых видов соревнований, что — вполне возможно — приведёт к тому, что появятся какие-то игры по правилам, придуманным не нами. А значит, необходимо уметь быстро выяснять, насколько эти правила могут быть нам полезны.

— А можно ли хотя бы в общем пояснить, как это будет делаться? — последовал вопрос из зала.

— Вот пример задачи, решив которую, мы сможем разобраться с целым классом игр. Дан ориентированный граф некоторой игры для двух игроков и начальная позиция в ней. Напомним, что в игре на графе игрок имеет право переходить из позиции в любую позицию, в которую есть ребро из текущей. Игроки ходят по очереди; проигрывает тот, кто не может сделать ход. Требуется проверить, верно ли, что при любой игре сторон всегда выигрывает первый игрок.

#### Формат входных данных

Во входном файле содержится описание одного или нескольких тестов. В первой строке каждого теста заданы число вершин  $V$  и число рёбер  $E$  ( $1 \leq V \leq 100\,000$ ,  $1 \leq E \leq 100\,000$ ), а также номер начальной позиции  $a$  ( $1 \leq a \leq V$ ). Далее следуют  $E$  строк — описания рёбер в формате  $u_i v_i$  ( $1 \leq u_i, v_i \leq V$ ), что означает наличие ребра, направленного из вершины  $u_i$  в вершину  $v_i$ . Файл завершается тремя нулями. Сумма всех  $E$  по всем тестам не превосходит 100 000, количество тестов в файле не превосходит 1000.

#### Формат выходных данных

Следуйте формату примера максимально точно — проверка производится автоматически.



**Пример**

stdin
3 2 1 1 2 1 3 1 1 1 1 1 4 3 1 1 2 1 3 3 4 0 0 0
stdout
First player always wins in game 1. Players can avoid first player winning in game 2. Players can avoid first player winning in game 3.

### Задача 31Н. 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  $n$  ( $1 \leq n \leq 10^{12}$ ).

#### 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 31I. Битва за кольцо [0.1 sec, 256 mb]

Саруман Белый и Гэндальф Серый решили сыграть в игру. Победителю достаётся Кольцо Всевластия. Перед игроками лежат кольца, соединённые в  $K$  цепочек. Для каждого кольца известно содержание золота в нём в процентах – целое число от 1 до 100. Ходят по очереди. За ход разрешается выбрать одну из цепочек и какое-то кольцо из этой цепочки и дематериализовать все кольца из данной цепочки с процентным содержанием золота не больше, чем у выбранного. При этом, понятно, цепочка может распасться на несколько. Игра продолжается на оставшихся цепочках. Тот, кто дематериализовал последнее кольцо, выиграл. Первым ходит Гэндальф. Определите, может ли Гэндальф выиграть и, если может, какой первый ход он должен для этого сделать.

#### Формат входных данных

В первой строке дано целое число  $K$  ( $1 \leq K \leq 50$ ). В следующих  $K$  строках приведены описания цепочек в следующем формате: сперва дана длина цепочки – целое число от 1 до 100, затем – процентные содержания золота в кольцах цепочки. Числа в строке разделены пробелом.

#### Формат выходных данных

Выведите “S”, если Кольцо Всевластия достанется Саруману. В противном случае выведите в первой строке “G”, а во второй пару чисел, описывающих выигрышный первый ход Гэндальфа – номер цепочки и номер кольца в ней. Цепочки и кольца внутри цепочек нумеруются с 1. Если существует несколько выигрышных первых ходов, выведите ход с наименьшим номером цепочки, если и таких несколько – с наименьшим номером кольца.

#### Примеры

stdin	stdout
2 3 1 2 1 1 1	G 1 1
2 3 2 1 2 1 1	S

### Задача 31J. Терминатор [0.2 sec, 256 mb]

Два игрока играют в настольную игру. Игровое поле представляет собой квадратный лабиринт,  $8 \times 8$  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 входных данных

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 выходных данных

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

#### Примеры

stdin	stdout
01000000 10100000 31100000 00020000 00000000 00000000 00000000 00000000	-1

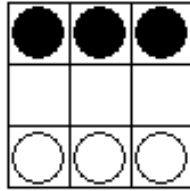
## Для мастеров AI

### *Правила.*

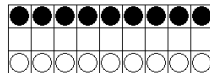
Это блок задач про промтинг. Пользоваться можно только бесплатными версиями ИИ. В шапке исходника указывать последовательность промтов, и сайты, куда они были отправлены. Если в процессе использования ИИ вы получили какие-то важные идеи для решения, это тоже часть решения. Если с помощью ИИ вы написали генератор тестов или стресс-тест, это тоже следует указывать. ЗадOCUMENTИРУЙТЕ, пожалуйста, сделанную работу в шапке отосланного решения. Если это нужно, вы можете писать часть кода сами.

### Задача 31К. Игра в пешки [0.2 сек, 256 mb]

В свободное время Дед Мороз Петрович и Дед Мороз Егорыч играют в следующую игру. На доске  $3 \times 3$  пешки расставляются следующим образом:



Пешки ходят и бьют по обычным шахматным правилам, к которым добавляется ещё одно: бить обязательно. Проигрывает тот, кто не может сделать ход. Первыми ходят белые. В течение 100 лет Петрович играл белыми и всегда выигрывал. Однажды Егорычу это надоело, и он принёс доску  $3 \times 5$ . Но и теперь он, играя чёрными, до сих пор проигрывает. «Как так?» — подумал Егорыч и решил купить доску  $3 \times N$ :



Вот тут Петровичу надо подумать, какими играть, чтобы выиграть. Вам нужно помочь ему это сделать.

#### Формат входных данных

Целое число  $N$  ( $1 \leq N \leq 10^9$ ).

#### Формат выходных данных

«White», если Петровичу надо играть белыми, и «Black», если надо играть чёрными. Егорыч и Петрович каждый раз ходят по наилучшей для себя стратегии.

#### Пример

stdin	stdout
3	White
4	Black
5	White

### Задача 31L. Smith [0.2 sec, 256 mb]

Do you know what Grundy's function is?

And do you know what Smith's function is? It's something like Grundy's function, but for **cyclic directed** graphs.

Now we will describe how to calculate Smith's function for **directed** graph, and you have to implement this algorithm. Of course, your implementation should be correct and fast.

#### Algorithm:

1. At first, value of each vertex is undefined
2. **while** (**true**)
3.     **for** each vertex  $v$ , calculate  $m(v) = mex(f_1, f_2, \dots, f_k)$ .  
      where  $f_1, f_2, \dots, f_k$ —values of children of  $v$ .  
       $mex(F)$  is minimal nonnegative integer which does not belong to  $F$ .
4.     **if** there is a vertex  $v$  such that for any edge  $(v, x)$  (from  $v$  to  $x$ )  
      we have already defined value of vertex  $x$  **OR**  
      from  $x$ , there is an edge to a vertex with value  $m(v)$  **then**
5.       Value of  $v$  is defined as  $m(v)$ .
6.     **if** nothing has changed **then break**

We define Smith's function as the result of this algorithm. If for some  $v$ , its value is still undefined, we set it to  $-1$ . Note that the algorithm is here only for purpose of the definition, and you may calculate Smith's function in any other way if it gives the same result. Also note that the definition is correct (the algorithm will not hang up, and result does not depend on the order in which we process the vertices).

#### Формат входных данных

The first line of file contains numbers  $N$  and  $M$ —number of vertices and edges in the graph ( $1 \leq N \leq 100\,000$ ,  $0 \leq M \leq 100\,000$ ). Next  $M$  lines contain two numbers  $x$  and  $y$  each ( $1 \leq x, y \leq N$ ); each of them means that there is an edge directed from vertex  $x$  to vertex  $y$ .

#### Формат выходных данных

Output  $N$  numbers—values of Smith's function for each vertex.

#### Пример

stdin	stdout
5 5 2 1 3 2 4 3 2 4 5 5	0 2 0 1 -1