# Задача A. Вычисление сложности программы

| | |
|---|---|
| Имя входного файла: | `complex.in` |
| Имя выходного файла: | `complex.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

Аня решила стать чемпионкой мира по программированию. Для этого ей, кроме всего прочего, потребовалось научиться вычислять время работы программ. Во многих программах основное время работы занимают вложенные циклы. Поэтому Аня хочет научиться узнавать, сколько операций будет выполнено внутри последовательности вложенных циклов.

Пока Аня еще только учится, поэтому все программы выглядят как последовательность вложенных циклов, причем все циклы выполняются от некоторого числа или переменной до некоторого другого числа. Ей требуется найти, сколько раз будет выполнена операция внутри самого вложенного цикла. Для этого она создала счетчик, проинициализировала его нулем, и увеличивает его на единицу каждый раз, входя в самый вложенный цикл.

Заметьте, что Аня пока не знает, что в качестве верхней границы в цикле может служить переменная. В качестве нижней границы может выступать как переменная, так и фиксированное значение. Если в каком-то цикле левая граница больше, чем правая, то цикл не будет выполняться вообще.

Так как программы у Ани иногда работают очень долго, то она попросила своего друга узнать ответ, когда же и он не смог этого посчитать, она попросила об этом вас. Помогите ей.

Напишите программу, которая находит, какое значение будет в счетчике в конце работы Аниной программы, если счетчик хранился в переменной целого 32-битного беззнакового типа.

## Формат входного файла

В первой строке находится число $1 \le N \le 3\,000$ — количество циклов. Далее в $N$ строках написано по два целых числа в каждой. В $i+1$-й строке входного файла записаны левая и правая граница $i$-го цикла: $L_i$ и $R_i$ ($1 - i \le L_i \le 3\,000$, $0 \le R_i \le 3\,000$). При этом если $L_i < 0$, то это обозначает, что цикл начинается со значения переменной $(-L_i)$-го цикла, в противном случае цикл начинается со значения $L_i$.

## Формат выходного файла

Выведите значение, которое будет в счетчике в конце работы программы.

## Пример

| complex.in | complex.out |
|---|---|
| 2<br>1 100<br>-1 100 | 5050 |
| 2<br>4 6<br>-1 3 | 0 |
| 4<br>1 256<br>1 256<br>1 256<br>1 256 | 0 |
| 4<br>1 1000<br>1 1000<br>1 1000<br>1 1000 | 3567587328 |

# Задача B. Ложь!

| | |
|---|---|
| Имя входного файла: | `lie.in` |
| Имя выходного файла: | `lie.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

— Ты врешь, Коля! На Марсе жизни нет! Кто тебе такую чушь сказал?

— Петя. А ему сказал Саша.

— Да от Пети я в жизни правдивого слова не слышал! Ему что ни скажут, он все переврет. Да и Саше откуда знать?

— А Саше рассказал про это Владимир Алексеевич, наш учитель биологии.

— Ну, Владимиру Алексеевичу-то можно верить... Только вряд ли он так сказал, это либо Саша, либо Петя придумал. А может, это ты меня разыгрываешь?..

— Минуточку, ребята, — вмешался подошедший к спорящим учитель математики, Глеб Тимофеевич, — давайте подойдем к проблеме формально.

Допустим, что все диалоги — Владимира Алексеевича с Сашей, Саши с Петей и Пети с Колей — действительно имели место. Пронумеруем ребят числами 1, 2 и 3. Предположим также, что каждый из ребят независимо друг от друга передал полученную информацию относительно жизни на Марсе верно с вероятностью $p_i$, а соврал с вероятностью $q_i = 1 - p_i$ для $i = 1, 2, 3$. Вероятности — это вещественные числа от нуля до единицы; событие, имеющее вероятность 0, никогда не произойдет, событие же с вероятностью 1 произойдет без всякого сомнения. Зная, что Коля после этого объявил, что жизнь на Марсе все-таки есть, найдите по данным $p_i$ вероятность того, что так действительно сказал Владимир Алексеевич.

— А как искать эту вероятность? И что значит «независимо друг от друга»? — растерялись ребята.

— Независимость означает, что действие одного из ребят никак не отражается на том, как поступят другие. К примеру, Пете неважно, соврал ли Саша — в любом случае он передаст сказанное Сашей правильно с вероятностью ровно $p_2$.

Задача несложная, и можно рассмотреть все восемь возможных случаев. Первый случай — все ребята говорили правду, и вероятность этого случая равна $p_1 \cdot p_2 \cdot p_3$. В этом случае жизнь на Марсе, без сомнения, есть — Владимиру Алексеевичу мы верим, а ребята передали его слова правильно. Второй случай, когда соврал только Саша, имеет место с вероятностью $q_1 \cdot p_2 \cdot p_3$, и в этом случае жизни на Марсе нет. Далее переберем остальные шесть случаев, каждый раз перемножая соответствующие вероятности, а потом просуммируем вероятности тех случаев, в которых слова учителя переданы правильно. То, что вероятности для отдельных ребят в каждом случае надо перемножить — это и есть формальное определение независимости. Ну, в скольких случаях будет передано именно то, что говорил Владимир Алексеевич?

— В одном...

— А вот и нет. Например, если Петя и Коля соврали, а Саша сказал правду, то истина, дважды исказившись, дойдет до нас в неизменном виде. И вообще, четное количество отрицаний, примененных к утверждению, дает само утверждение. В нашей задаче случаев с четным количеством отрицаний — четыре, и итоговая вероятность равна $p_1 \cdot p_2 \cdot p_3 + q_1 \cdot q_2 \cdot p_3 + q_1 \cdot p_2 \cdot q_3 + p_1 \cdot q_2 \cdot q_3$.

— То есть если Петя и Коля точно соврут, а Саша точно скажет правду, то от Коли мы услышим в точности то, что говорил учитель?

— Совершенно верно. А теперь решите-ка задачу для общего случая, когда ребят не трое, а $n$. Первому, кто решит — пятерка на следующей контрольной!

## Формат входного файла

В первой строке входного файла записано целое число $n$ ($1 \le n \le 100$). Во второй строке через пробел записаны $n$ вещественных чисел — это числа $p_1$, $p_2$, ..., $p_n$ ($0 \le p_i \le 1$). Числа даны с не более чем шестью десятичными знаками после запятой.

## Формат выходного файла

Выведите в выходной файл одно вещественное число, округленное до шести знаков после запятой — вероятность существования жизни на Марсе.

## Пример

| lie.in | lie.out |
|---|---|
| 3<br>1 0.1 0.9 | 0.180000 |

# Задача C. Матрицы

| | |
|---|---|
| Имя входного файла: | `matrix.in` |
| Имя выходного файла: | `matrix.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

Аня недавно узнала, что такое квадратная матрица размерности $n$. Это таблица $n \times n$ с целыми числами в ячейках. Число, стоящее на пересечении $i$-ой строки и $j$-ого столбца матрицы $A$, кратко обозначается $A[i, j]$. Матрицы можно умножать, и Аня быстро освоила, как запрограммировать эту операцию с помощью циклов. Результатом умножения двух матриц $A$ и $B$ будет матрица $C$, элементы которой определяются следующим образом:

$$C[i, j] = \sum_{k=1}^{n} A[i, k]B[k, j].$$

Матрицы ей понадобились для конкретной задачи, в которой надо узнать определенный элемент произведения нескольких матриц. Это уже достаточно сложная задача для Ани, но она усложняется тем, что все вычисления ведутся по модулю некоторого простого числа $p$, то есть если при арифметических операциях получается число, большее, либо равное $p$, оно заменяется на остаток при делении на $p$.

Помогите Ане вычислить нужный ей элемент.

## Формат входного файла

В первой строчке входного файла стоят два числа: $m$ — количество матриц, $n$ — размер каждой из матриц ($1 \le m \le 200$, $1 \le n \le 200$). В следующей строчке содержатся номер строки и столбца интересующего Аню элемента $1 \le a \le n$, $1 \le b \le n$. В третьей строке содержится простое число $p \le 1\,000$. Далее следует описание $m$ матриц. Описание каждой матрицы состоит из $n$ строк. В каждой из строк содержится $n$ неотрицательных целых чисел, меньших $p$. Соседние числа в строке разделены пробелом, а перед каждой матрицей пропущена строка.

## Формат выходного файла

Требуется вывести единственное число: нужный Ане элемент произведения матриц.

## Пример

<table>
<tr><th>matrix.in</th><th>matrix.out</th></tr>
<tr><td><pre>3 2
1 2
239

1 2
3 4

4 2
1 3

1 2
2 1</pre></td><td><pre>20</pre></td></tr>
</table>

# Задача D. Упаковка

| | |
|---|---|
| Имя входного файла: | `packing.in` |
| Имя выходного файла: | `packing.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

Вот уже неделю Ваня решает сложную задачу: по данному числу $n$ необходимо расположить на плоскости $n$ кругов с радиусами 1, 2, ..., $n$ таким образом, чтобы, во-первых, каждая пара кругов не пересекалась (возможно, касаясь), а во-вторых, все эти круги помещались в большой объемлющий круг как можно меньшего радиуса. В процессе решения Ваня заметил, что, расположив достаточное количество больших кругов, можно сразу начинать искать объемлющий круг, поскольку все оставшиеся маленькие круги можно поместить в промежутках, оставшихся между большими.

Теперь Ваня хочет по данному множеству радиусов оценить, как часто между тремя попарно касающимися кругами с радиусами из этого множества можно поместить четвертый круг. Для этого он ввел *рейтинг упаковываемости* $P$: для множества радиусов $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ рейтинг $P(\mathcal{R})$ равен количеству таких упорядоченных четверок индексов $(i, j, k, l)$, что $r_i > r_j > r_k > r_l$ и между тремя попарно касающимися кругами радиусов $r_i$, $r_j$ и $r_k$ можно поместить круг радиуса $r_l$ так, чтобы он не пересекался с ними, возможно, касаясь. Выражение «поместить между» означает, что центр четвертого круга должен лежать внутри треугольника с вершинами в центрах первых трех кругов.

Помогите Ване посчитать рейтинг упаковываемости данного множества.

## Формат входного файла

В первой строке входного файла записано целое число $n$ ($1 \le n \le 250$). Во второй строке через пробел записаны $n$ различных целых чисел $r_1$, $r_2$, ..., $r_n$ ($1 \le r_i \le 250$) — элементы множества $\mathcal{R}$. Гарантируется, что $\mathcal{R}$ непусто.

## Формат выходного файла

Выведите в выходной файл одно число — рейтинг упаковываемости $P(\mathcal{R})$.

## Пример

| packing.in | packing.out |
|---|---|
| 5<br>2 1 5 10 20 | 1 |

# Задача E. Сосиска в тесте

| | |
|---|---|
| Имя входного файла: | `sausage.in` |
| Имя выходного файла: | `sausage.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

Аня еще учится в школе, однако она уже много слышала о трудностях студенческой жизни. Ее это очень волнует, так как она задумывается о своем будущем и понимает, что после окончания школы ей придется учиться в ВУЗе. Из многочисленных рассказов она узнала, что излюбленной пищей студентов являются сосиски. А так как студенты часто испытывают недостатки в средствах, сосиски часто приходится делить. Однако никто не доверяет делению «на глаз», всегда найдется кто-нибудь, утверждающий, что его обделили.

Сосиска представляет собой практически плоскую фигуру (таковы уж студенческие столовые), которую можно описать следующим образом. Рассмотрим некоторый горизонтальный отрезок на плоскости, а также некоторый круг. Тогда, если центр круга провести по всем точкам отрезка от начальной до конечной, то любая точка, лежавшая хотя бы в какой-то момент в круге, принадлежит сосиске. Можно описать сосиску и другим способом. Рассмотрим тот же отрезок и два круга одинакового радиуса с центрами в его крайних точках. Тогда точка принадлежит сосиске ровно в том случае, если она лежит на некотором отрезке, у которого каждый из концов лежит в одном из кругов.

Подумав, Аня понимает, что при наличии линейки можно легко отмерить нужное расстояние от левого края сосиски и вертикально отрезать необходимую часть. Но если при этом отмерять одинаковые расстояния, то те, кому достанутся крайние куски, получат меньше. Помогите Ане разместить разрезы в нужных местах сосиски, чтобы все получили поровну.

## Формат входного файла

Сосиска в тесте задается двумя числами: $l$ — расстоянием между центрами крайних кругов и $r$ — радиусом каждой из них. Оба числа вещественные, положительные и не превосходят 10 000 и располагаются на первой строке входного файла. На второй строке находится $n$ — количество студентов, претендующих на сосиску ($2 \le n \le 1\,000$).

## Формат выходного файла

В выходном файле должно содержаться $n - 1$ строка. В $i$-ой строке должно содержаться расстояние от левого края сосиски до правого края $i$ куска (куски нумеруются слева направо). Все числа должны быть вычислены с точностью до 6 десятичных знаков после запятой.

## Пример

| sausage.in | sausage.out |
|---|---|
| 1 1<br>2 | 1.500000 |
| 1 1<br>4 | 0.856810<br>1.500000<br>2.143190 |

# Задача F. Остановки

| | |
|---|---|
| Имя входного файла: | `stations.in` |
| Имя выходного файла: | `stations.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

Представьте себе, что вы один из проектировщиков железной дороги. Вам нужно построить $N$ остановок, размеры которых довольно велики, но все равны между собой и равны некоторому числу $l$. Стоимость постройки меняется в зависимости от рода поверхности, по которой предстоит строить платформу. При этом остановки не должны располагаться друг от друга дальше чем на $R$ метров, иначе каким-то жителям области придется далеко ходить. Также остановки не должны располагаться друг к другу ближе чем на $r$ метров, иначе электрички не будут успевать разгоняться между остановками и будут медленно ездить. Руководство хочет потратить на постройку остановок минимальное количество денег. Вам известны стоимости постройки метра платформы на каждом из $L$ метров всей дороги. Стоимость $i$-го метра — $a_i$ (при $1 \le i \le N$). Вычислите минимально возможную стоимость расстановки всех остановок.

## Формат входного файла

В первой строке входного файла содержатся целые числа $N$, $l$ и $L$ ($1 \le N \le 100$, $1 \le l \le 100$, $1 \le L \le 10\,000$). Во второй строке содержится два целых числа: $r$ и $R$ ($0 \le r \le R \le 1\,000$). В третьей строке содержится $N$ целых чисел $a_i$, разделенных пробелами ($1 \le a_i \le 10\,000$).

## Формат выходного файла

Выведите искомый ответ (гарантируется, что он существует).

## Пример

| stations.in | stations.out |
|---|---|
| `3 1 5`<br>`0 1`<br>`3 7 1 5 2` | `6` |
| `2 3 9`<br>`1 2`<br>`8 1 2 3 7 4 3 1 11` | `14` |

# Задача G. Башни

| | |
|---|---|
| Имя входного файла: | `towers.in` |
| Имя выходного файла: | `towers.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

Башней называется выражение вида

$$a_0^{a_1^{a_2^{\cdot^{\cdot^{\cdot^{a_k}}}}}}, k \geq 1,$$

т.е. последовательное возведение в степень чисел $a_0, \ldots, a_k$. Отметим, что операция возведения в степень выполняется справа налево, т.е. выражение $a^{b^c}$ вычисляется как $a^{(b^c)}$.

Аня недавно изучила алгоритмы сортировок, и теперь она умеет сортировать целые числа, вещественные числа, и даже строки. Узнав об этом, Андрюша предложил ей написать программу для сортировки башен. Как же это сделать?

Аня привыкла учиться на примерах, и поэтому ей нужна ваша работающая программа. Напишите программу, которая по заданным во входном файле башням вычисляла бы порядок, в котором нужно их поставить, чтобы они оказались расположены по возрастанию.

## Формат входного файла

В первой строке входного файла задается число башен $N$ ($1 \leq N \leq 50\,000$). Далее следуют $N$ строк, каждая из которых содержит одну башню в формате $k_i$ $a_{i,0}$ $a_{i,1}$ $\ldots a_{i,k}$, здесь все числа разделены пробелами. Каждое из $a_{i,j}$ — целое число в пределах от 1 до 99, кроме того, $1 \leq k_i \leq 9$.

Известно, что среди башен во входном файле нет равных, кроме того, любые две башни отличаются достаточно сильно, чтобы их можно было различить. Например, числа $10^{10}$ и $10^{10} + 1$ отличаются достаточно сильно, а $10^{20} + 10^5$ и $10^{20}$ — недостаточно. Заметьте, что значения у башен могут быть весьма велики — например, даже число $2^{2^{2^{2^2}}} = 2^{65536}$ не помещается ни в какой вещественный тип.

## Формат выходного файла

Выведите перестановку номеров башен $b_1, b_2, \ldots, b_N$ в таком порядке, что если взять сначала башню с номером $b_1$, потом с номером $b_2$, и т.д., то они окажутся расположенными в порядке возрастания.

## Пример

| towers.in | towers.out |
|---|---|
| 10<br>4 2 2 2 2 2<br>1 2 2<br>1 3 2<br>1 2 3<br>3 2 2 2 2<br>2 2 2 2<br>1 3 3<br>3 3 3 3 3<br>2 4 3 3<br>2 2 3 4 | 2 4 3 6 7 5 9 10 1 8 |

# Задача H. Слабая $k$-связность

| | |
|---|---|
| Имя входного файла: | `weakconn.in` |
| Имя выходного файла: | `weakconn.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

Ане, как будущей чемпионке мира по программированию, поручили очень ответственное задание. Правительство Н-ской области вручает ей план постройки дорог между городами. По плану все дороги односторонние, но между двумя городами может быть больше одной дороги, возможно, в разных направлениях. Ане необходимо вычислить минимальное такое $k$, что данный ей план является слабо $k$-связным.

Правительство называет план слабо $k$-связным, если выполнено следующее условие: для любых двух различных городов можно проехать от одного до другого, нарушая правила движения не более $k$ раз. Нарушение правил — это проезд по существующей дороге в обратном направлении. Гарантируется, что между любыми двумя городами можно проехать, возможно, несколько раз нарушив правила.

## Формат входного файла

В первой строке входного файла даны два числа $2 \le n \le 300$ и $1 \le m \le 10^5$ — количество городов и дорог в плане. В последующих $m$ строках даны по два числа — номера городов, в которых начинается и заканчивается соответствующая дорога.

## Формат выходного файла

Выведите минимальное $k$, такое, что данный во входном файле план является слабо $k$-связным.

## Пример

| weakconn.in | weakconn.out |
|---|---|
| 3 2<br>1 2<br>1 3 | 1 |
| 4 4<br>2 4<br>1 3<br>4 1<br>3 2 | 0 |

# Задача I. Задача про XOR

| | |
|---|---|
| Имя входного файла: | `xor.in` |
| Имя выходного файла: | `xor.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

В рамках подготовки к чемпионату мира Кирилл придумал Ане задачу. Он написал $N$ знаковых 32-битных чисел ($1 \le N \le 10^5$) и попросил вычислить значение некоторого выражения $S$. Пусть $a_1, ...., a_N$ — все эти числа. Тогда выражение это

$$S = (a_1 \oplus a_2 \oplus \ldots \oplus a_n) \oplus (b_1 \oplus b_2 \oplus \ldots \oplus b_{(n-1)}),$$

где

$$b_i = F(a_i, a_{(i+1)}) \oplus F(a_i, a_{(i+2)}) \oplus \ldots \oplus F(a_i, a_n).$$

В этой формуле под знаком $\oplus$ понимается побитовое «исключающее или» (`XOR`), а $F(a, b) = x - 1$, где $x$ — максимальная степень двойки, на которую делится нацело $a - b$, если $a \ne b$, и $F(a, b) = -1$, если $a = b$. Все операции `XOR` выполняются слева направо, если скобки не указывают иной порядок.

Аня, как большая специалистка в области циклов, быстро написала требуемую программу, однако программа работала слишком долго. Чтобы лучше разобраться в этом вопросе, она попросила вас написать программу, которая бы укладывалась в отведенное время. Помогите ей это сделать.

## Формат входного файла

В первой строке входного файла содержится число $N$. Во второй строке содержится $N$ 32-битных знаковых целых чисел $a_i$, разделенных пробелами.

## Формат выходного файла

Выведите значение выражения.

## Пример

| xor.in | xor.out |
|---|---|
| 3<br>1<br>2<br>3 | 1 |
| 2<br>1<br>1 | -1 |
| 3<br>1<br>2<br>4 | 6 |