

## Задача 5. Игральные кубики

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

Юный математик Матвей интересуется теорией вероятностей, и по этой причине у него всегда есть с собой несколько стандартных шестигранных игральных кубиков. Стандартный шестигранный кубик имеет три противоположащих пары граней, которые размечены таким образом, что напротив грани с числом 1 находится грань с числом 6, напротив грани с числом 2 — грань с числом 5 и напротив грани с числом 3 — грань с числом 4.

Анализируя различные игры с шестигранными кубиками, Матвей придумал новую игру. В эту игру играют два игрока, и проходит она следующим образом: первый игрок бросает один или несколько стандартных кубиков (количество кубиков он определяет сам). После этого первому игроку начисляется количество очков, равное сумме чисел, оказавшихся на верхних гранях всех кубиков, а второму игроку — сумма чисел, оказавшихся на нижних гранях этих кубиков. Побеждает тот, кто набрал больше очков.

Например, если был брошен один кубик, и на верхней его грани выпало число два, то первый игрок получает два очка, а второй — пять. В свою очередь, если было брошено два кубика и на их верхних гранях выпало по единице, то первый игрок получает также два очка, а второй игрок — двенадцать очков, так как на нижних гранях этих кубиков оказались шестерки.

Матвей рассказал об этой игре своему другу, юному информатику Фоме, и они начали играть в неё через Интернет. Поскольку Фома не видит результат броска и не знает, сколько кубиков бросает Матвей как первый игрок, то о набранных каждым игроком очках он узнает только от Матвея. Чтобы проверить достоверность этой информации, Фома решил узнать, какое минимальное и максимальное количество очков мог получить он, как второй игрок, если известно, сколько очков набрал Матвей.

Требуется написать программу, которая по количеству очков, которые набрал первый игрок после броска, определяет наименьшее и наибольшее количество очков, которые может получить второй игрок за этот бросок.

### Формат входного файла

Первая строка входного файла содержит целое положительное число  $n$  — количество очков, которые получил первый игрок ( $1 \leq n \leq 10^{10}$ ).

### Формат выходного файла

Выходной файл должен содержать два разделенных пробелом целых числа: минимальное и максимальное количество очков, соответственно, которые мог набрать второй игрок при таком броске кубиков.

### Примеры входных и выходных файлов

<code>dices.in</code>	<code>dices.out</code>
2	5 12
36	6 216

### Система оценивания

Правильные решения для тестов, в которых  $1 \leq n \leq 1000$ , будут оцениваться из 50 баллов.

## Задача 6. Имена

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

На далекой планете Тау Кита есть непонятные нам обычаи. Например, таукитяне очень необычно для землян выбирают имена своим детям. Родители так выбирают имя ребенку, чтобы оно могло быть получено как удалением некоторого набора букв из имени отца, так и удалением некоторого набора букв из имени матери. Например, если отца зовут «*abacaba*», а мать — «*bbssaa*», то их ребенок может носить имена «*a*», «*bba*», «*bcaa*», но не может носить имена «*aaa*», «*ab*» или «*bbs*». Возможно, что имя ребенка совпадает с именем отца и/или матери, если оно может быть получено из имени другого родителя удалением нескольких (возможно, ни одной) букв.

Пусть отец по имени *X* и мать по имени *Y* выбирают имя своему новорожденному ребенку. Так как в таукитянских школах учеников часто вызывают к доске в лексикографическом порядке имен учеников, то есть в порядке следования имен в словаре, то они хотят выбрать своему ребенку такое имя, чтобы оно лексикографически следовало как можно позже.

Формально, строка *S* лексикографически больше строки *T*, если выполняется одно из двух условий:

- строка *T* получается из *S* удалением одной или более букв с конца строки *S*;
- первые (*i* - 1) символов строк *T* и *S* не различаются, а буква в *i*-й позиции строки *T* следует в алфавите раньше буквы в *i*-й позиции строки *S*.

Требуется написать программу, которая по именам отца и матери находит лексикографически наибольшее имя для их ребенка.

### Формат входного файла

Первая строка входного файла содержит *X* — имя отца. Вторая строка входного файла содержит *Y* — имя матери. Каждое имя состоит из строчных букв латинского алфавита, включает хотя бы одну букву и имеет длину не более  $10^5$  букв.

### Формат выходного файла

Выходной файл должен содержать искомое лексикографически наибольшее из возможных имен ребенка. В случае, если подходящего имени для ребенка не существует, выходной файл должен быть пустым.

### Примеры входных и выходных файлов

<code>name.in</code>	<code>name.out</code>
<i>abcabca</i> <i>abcda</i>	<i>ca</i>
<i>ccba</i> <i>accbbaa</i>	<i>ccba</i>

### Пояснения к примеру

В первом примере имя ребенка не может начинаться с буквы большей *c*, так как имя отца не содержит таких букв. Буква *c* содержится в обоих именах, следовательно, имя ребенка может начинаться с этой буквы. Единственная буква, которая может идти следом за буквой *c* в имени ребенка — это буква *a*.

### **Система оценивания**

Правильные решения для тестов, в которых имена содержат только буквы «а» и «b» и имеют длину не более 1000, будут оцениваться из 20 баллов.

Правильные решения для тестов, в которых имена содержат только буквы «а» и «b» и имеют длину не более  $10^5$ , будут оцениваться из 40 баллов.

Правильные решения для тестов, в которых имена имеют длину не более 1000, будут оцениваться из 40 баллов.

Несмотря на выделение отдельных групп тестов, на окончательную проверку будут приниматься только решения, правильно работающие для всех тестов, приведенных в условии задачи.

## Задача 7. Две окружности

Имя входного файла:	<code>circles.in</code>
Имя выходного файла:	<code>circles.out</code>
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Юный футболист Митя обнаружил на школьном футбольном поле две различные окружности, нарисованные едва заметной белой краской. Вспомнив истории о загадочных кругах на полях, он отметил эти окружности с помощью небольших камушков. Митя разложил на поле  $n$  камушков так, чтобы каждый из них находился на одной из окружностей или даже на их пересечении, если эти окружности пересекаются. Получилось так, что на каждой окружности размещался хотя бы один камушек. Обладая великолепным глазомером, Митя расположил камушки на окружностях абсолютно точно, без какой-либо погрешности.

На следующий день пошел дождь, краска стерлась, и нарисованные окружности исчезли, но все камушки остались на своих местах. Теперь Мите очень нужно найти доказательство необычного явления, свидетелем которого он был, то есть, восстановить окружности.

Требуется написать программу, которая по координатам камушков на поле находит вариант размещения их на двух несовпадающих окружностях.

### Формат входного файла

Первая строка входного файла содержит целое число  $n$  — количество размещенных Митей камушков на поле ( $2 \leq n \leq 2000$ ). Последующие  $n$  строк содержат целочисленные координаты  $(x_i, y_i)$  камушков — по одной паре координат, разделенных пробелом, в каждой строке ( $-10^6 \leq x_i, y_i \leq 10^6$ ). Никакие два камушка не размещаются в одной точке.

Гарантируется, что ответ для заданного набора камушков существует.

### Формат выходного файла

Выходной файл должен содержать две строки. Первая строка должна содержать последовательность номеров **всех** камушков, которые принадлежат первой окружности, вторая строка — последовательность номеров **всех** камушков, которые принадлежат второй окружности.

Каждый камушек должен встречаться хотя бы в одной из двух последовательностей. Если камушек встречается в обеих последовательностях, то это обозначает, что он находится на пересечении окружностей. Считается, что камушки пронумерованы от 1 до  $n$  в порядке их следования во входных данных.

Нумерация окружностей не имеет значения, то есть выводить две последовательности можно в любом порядке. Числа в последовательностях можно также выводить в произвольном порядке. Каждая из последовательностей должна содержать не менее одного числа. Все числа в строках должны быть разделены пробелами.

Если вариантов расположения окружностей несколько, можно выбрать любой из них.

### **Примеры входных и выходных файлов**

<b>circles.in</b>	<b>circles.out</b>
7 1 -1 0 0 1 1 3 1 3 -1 2 0 4 0	6 1 2 3 4 7 5 6
5 -1000000 0 0 1000000 1000000 0 0 -1000000 0 0	1 2 3 4 1 5

### **Пояснения к примерам**

В первом примере одна из искомых окружностей имеет центр в точке с координатами (1, 0), а вторая — в точке с координатами (3, 0). Обе окружности имеют радиус равный 1.

Во втором примере центр первой окружности совпадает с началом координат, а радиус равен  $10^6$ . Вторая окружность — любая, проходящая через точки  $(-10^6, 0)$  и  $(0, 0)$ .

В обоих примерах возможны и другие правильные ответы.

### **Система оценивания**

Правильные решения для тестов, в которых  $2 \leq n \leq 50$ , будут оцениваться из 50 баллов.

## Задача 8. Столицы

Имя входного файла: capitals.in  
Имя выходного файла: capitals.out  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 512 мегабайт

В стране Триландии близятся выборы новых столиц. Столицы в Триландии необычные, поскольку ими являются одновременно сразу три различных города. Такая идея размещения столиц основана на исследованиях эффективности управления страной, выполненных ведущими экономистами Триландии.

Всего в Триландии  $n$  городов, из которых некоторые пары городов соединены дорогами, и по каждой из них можно проехать в обе стороны. Время проезда по каждой дороге в одну сторону равно одному часу. При этом все города соединены дорогами таким образом, что из каждого города можно добраться в любой другой, причем это можно сделать единственным способом, если по каждой дороге проезжать не более одного раза и только в одну сторону.

Как показали результаты проведенных триландскими экономистами исследований, управление страной будет наиболее эффективным, если три столицы будут выбраны так, что время кратчайшего пути между каждой парой столиц составит ровно  $d$  часов. Перед проведением выборов необходимо знать, сколько существует различных троек городов, удовлетворяющих описанным выше свойствам. Две тройки городов считаются различными, если в первой тройке есть хотя бы один город, которого нет во второй тройке, и наоборот.

Требуется написать программу, которая по количеству городов в Триландии и описанию дорог находит количество троек городов, которые могут быть столицами.

### Формат входного файла

Первая строка входного файла содержит два разделенных пробелом целых числа: количество городов в Триландии  $n$  и требуемое время в пути между столицами  $d$  ( $3 \leq n \leq 10^5$ ,  $1 \leq d < n$ ). Каждая из последующих  $(n - 1)$  строк содержит описание одной дороги: пару разделенных пробелом различных целых чисел  $a_i$  и  $b_i$  — номера городов, которые соединены двусторонней дорогой ( $1 \leq a_i \leq n$ ,  $1 \leq b_i \leq n$ ,  $a_i \neq b_i$ ). Каждая пара городов соединена не более чем одной дорогой.

### Формат выходного файла

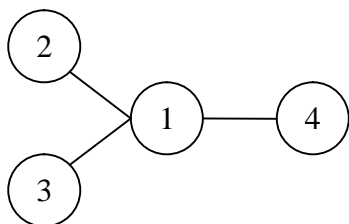
Выходной файл должен содержать одно целое число — количество подходящих троек городов, которые могут быть выбраны столицами. В случае, если нужных троек городов не окажется, выходной файл должен содержать ноль.

### Примеры входных и выходных файлов

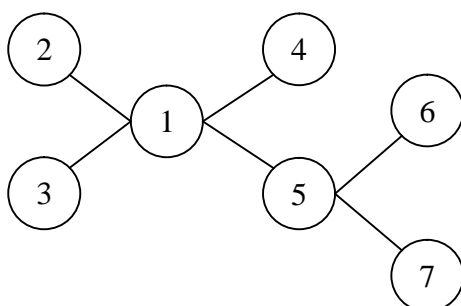
capitals.in	capitals.out
4 2 1 2 1 3 1 4	1
7 2 1 2 1 3 1 4 5 1 5 6 5 7	5

### **Пояснения к примерам**

В первом примере существует единственный способ выбрать три столицы: города под номерами 2, 3 и 4. Рисунок, соответствующий первому примеру, приведен ниже.



Во втором примере существует четыре варианта выбора трёх столиц из четверки городов: 2, 3, 4 и 5. Можно также выбрать столицами города с номерами 1, 6 и 7. Рисунок, соответствующий второму примеру, приведен ниже.



### **Система оценивания**

Правильные решения для тестов, в которых  $3 \leq n \leq 50$ , будут оцениваться из 20 баллов.

Правильные решения для тестов, в которых  $3 \leq n \leq 500$ , будут оцениваться из 40 баллов.

Правильные решения для тестов, в которых  $3 \leq n \leq 5000$ , будут оцениваться из 60 баллов.