Департамент образования администрации города Нижнего Новгорода
Городской ресурсный центр физико-математического образования
Нижегородский государственный университет им. Н. И. Лобачевского

Двенадцатая нижегородская городская олимпиада школьников по информатике имени В. Д. Лелюха

13 февраля 2016 г.

Нижний Новгород
2016

Результаты, архивы и другие материалы олимпиады можно найти на сайте `http://olympiads.nnov.ru`

Оригинал-макет подготовлен в системе LaTeX 2ε с использованием набора шрифтов LH.

Двенадцатая городская олимпиада по информатике

13 февраля 2014 г. Городской ресурсный центр физико-математического образования в лице учредителей: Департамента образования администрации города Нижнего Новгорода, Нижегородского государственного университета им. Н.И. Лобачевского и МБОУ Лицей № 40 проводит двенадцатую городскую олимпиаду по информатике среди учащихся 6–11 классов образовательных учреждений города Нижнего Новгорода.

Целью проведения олимпиады является поиск талантливой молодёжи, привлечение её в науку, повышения уровня преподавания предметов физико-математического цикла.

Спонсорами городской олимпиады школьников по информатике являются НПП «Прима», компания «Яндекс», ННГУ им. Н.И. Лобачевского.

Двенадцатая городская олимпиада проводится в ННГУ им. Лобачевского на базе Института информационных технологий, математики и механики (корпус 2).

Олимпиада проводится в соответствии с Положением о городской олимпиаде по информатике (Приказ № 1453 от 07.11.2008 Департамента образования и СПЗД администрации г. Нижнего Новгорода).

Для участия в городской олимпиаде приглашаются талантливые школьники из Нижегородской области.

Официальные разрешённые среды программирования — Free Pascal, GNU C/C++ (MinGW), C#, Python 3.

Регламент проведения олимпиады

9:30—10:00	Регистрация участников
10:00—10:15	Открытие олимпиады, информация от жюри
10:15—15:15	Решение задач олимпиады
15:15—16:00	Обед
16:00—17:00	Работа жюри
17:00—18:00	Приветствие участников олимпиады, выступления учредителей, спонсоров. Подведение итогов олимпиады. Поздравление победителей и призёров.

Состав оргкомитета олимпиады

Сидоркина С. Л., заместитель директора департамента образования администрации города Нижнего Новгорода;
Цветков М. И., начальник отдела общего среднего образования департамента образования администрации города Нижнего Новгорода;
Тинькова Е. В., консультант отдела общего среднего образования департамента образования администрации города Нижнего Новгорода;
Авралев Н. В., проректор по связям с общественностью ННГУ им. Н.И. Лобачевского;
Борисов Н. А., доцент кафедры МО ЭВМ ННГУ им. Н.И. Лобачевского;
Сысоев А. В., ассистент кафедры МО ЭВМ ННГУ им. Н.И. Лобачевского;
Умнова Н. С., директор муниципального бюджетного образовательного учреждения лицей № 40;
Евстратова Л. П., заместитель директора по учебно-воспитательной работе МБОУ лицей № 40;
Гашпар И. Л., куратор классов НОЦ ИПФ РАН;

Состав предметной комиссии (жюри)

Председатель — Калинин П. А., старший разработчик, ООО «Яндекс», к.ф.-м.н.;

Члены комиссии:
Евстратова Л. П., заместитель директора МОУ лицей № 40, ответственный секретарь комиссии;
Демидов А. Н., инженер, НПП «ПРИМА»;
Жидков Н. В., студент 1 курса СПб АУ;
Калинин Н. А., студент 2 курса ВШОПФ ННГУ;
Кузьмичев Д. А., студент 3 курса МФТИ;
Лазарев Е. А., старший инженер по разработке программного обеспечения, АО «Интел А/О», к.т.н.;
Лопаткин М. А., программист, ООО «Яндекс»;
Матросов М. В., инженер-программист, НПП «АВИАКОМ»;
Шмелёв А. С., инженер-программист, НПП «ПРИМА».

Ниже приведены примеры текстов программ, написанных на разрешённых языках программирования. Программы считывают данные (два числа в одной строке) с клавиатуры и выводят на экран их сумму:

Pascal

```
var a,b:integer;
begin
  read(a,b);
  writeln(a+b);
end.
```

C/C++

```
#include <iostream>
using namespace std;
int main() {
    int a,b;
    cin >> a >> b;
    cout << a+b << endl;
    return 0;
}
```

C#

```
using System;
class Program {
static void Main(string[] args) {
    string[] s = Console.ReadLine()
        .Split();
    int n = Int32.Parse(s[0]);
    int m = Int32.Parse(s[1]);
    Console.Write(n + m);
}
}
```

Python 3

```
a, b = map(int, input().split())
print(a + b)
```


XII Городская олимпиада по информатике им. В. Д. Лелюха 13 февраля 2016 г.

Задача 1. Имена собственные

Входной файл	`proper.in`
Выходной файл	`proper.out`
Ограничение по времени	1 секунда
Ограничение по памяти	256 мегабайт

Вася очень любит читать. При этом он старается каждое произведение читать на языке оригинала. Но у него часто возникает проблема: во многих произведениях встречается очень много имен собственных — имен героев, названий мест и т.д. — и их все очень сложно удержать в голове. Поэтому Вася очень хочет иметь программу, которая из текста книги выделит все имена собственные.

Очередная книга, которую собирается читать Вася, написана на английском языке. Вася знает, что в английском языке (в отличие от русского) в именах собственных *все* слова пишутся с большой буквы. Поэтому, для простоты, Вася решил считать именем собственным любую последовательность подряд идущих слов одного предложения, каждое из которых начинается с большой буквы; при этом такая последовательность не должна включать первое слово соответствующего предложения (потому что оно и так всегда пишется с большой буквы). Словом считается любая последовательность подряд идущих английских букв. Предложением считается последовательность слов, заканчивающаяся на точку, восклицательный или вопросительный знак.

Более формально: вам дан текст, содержащий только английские маленькие и большие буквы, пробелы, переводы строк и знаки препинания. Символы «точка», «восклицательный знак» и «вопросительный знак» считаются *символами конца предложения*, они разбивают текст на предложения. А именно, первым предложением считается последовательность символов от начала текста до первого символа конца предложения, вторым предложением — последовательность символов между первым и вторым символами конца предложения, и т.д.

Все остальные символы, кроме букв, считаются *пробельными символами*, они разбивают предложения на слова. А именно, первым словом предложения считается последовательность букв начиная с первой буквы, присутствующей в предложении, до первого пробельного символа, идущего после этой буквы. Вторым словом считается последовательность букв начиная со следующей буквы и до ближайшего после нее пробельного символа, и т.д.

Слово считается *частью имени собственного*, если оно начинается с большой буквы и при этом не является первым словом в предложении. *Именем собственным* считается каждая последовательность слов-частей имени собственного, принадлежащих одному предложению и ограниченных с обеих сторон или концами предложения, или словами, не являющимися частями имени собственного.

Напишите программу, которая выведет все имена собственные, встречающиеся в данном тексте.

Формат входных данных

Выходной файл содержит заданный текст. В тексте встречаются только маленькие и большие латинские буквы, пробелы, переводы строки и следующие знаки препинания: `!?.;:,"()-`. Длина текста не превосходит 100 000 символов. Гарантируется, что текст заканчивается на символ конца предложения, после которого могут идти несколько пробелов и/или переводов строки.

Формат выходных данных

Выведите в выходной файл все имена собственные, встречающиеся в заданном тексте, по одному имени собственному на строке, в том порядке, в котором они встречаются в тексте. Если одно и то же имя собственное встречается несколько раз в разных местах текста, его надо выводить каждый раз.

Слова в имени собственном разделяйте пробелами; знаки препинания должны отсутствовать в выходном файле.

Пример

Входной файл	*Выходной файл*
This is the living-room of the house occupied by the eminent Professor Michael Verres-Evans, and his wife, Mrs Petunia Evans-Verres, and their adopted son, Harry James Potter-Evans-Verres. There is a letter lying on the living-room table, and an unstamped envelope of yellowish parchment, addressed to Mr H Potter in emerald-green ink. The Professor and his wife are speaking sharply at each other, but they are not shouting. The Professor considers shouting to be uncivilised.	Professor Michael Verres Evans Mrs Petunia Evans Verres Harry James Potter Evans Verres Mr H Potter Professor Professor
The week after Taffimai Metallumai (we will still call her Taffy, Best Beloved) made that little mistake about her Daddy spear and the Stranger-man and the picture-letter and all, she went carp-fishing again with her Daddy. Her Mummy wanted her to stay at home and help hang up hides to dry on the big drying-poles outside their Neolithic Cave, but Taffy slipped away down to her Daddy quite early, and they fished.	Taffimai Metallumai Taffy Best Beloved Daddy Stranger Daddy Mummy Neolithic Cave Taffy Daddy

Решение

Эта задача решалась не очень сложно, требовалась просто аккуратная реализация. Проще всего было решать задачу последовательно: сначала разбить текст на предложения, потом каждое предложение — на отдельные слова, потом пробежаться по списку слов каждого предложения и выделить имена собственные. При

этом при выделении предложений можно было их складывать в отдельный массив, а можно было каждое очередное выделенное предложение сразу передавать в процедуру его обработки. Последний вариант и реализован в примере решения.

Пример правильной программы

```
{$mode delphi}
{$r+,q+,s+,i+}
const sentEnd: set of char = ['.','!','?'];
    letters: set of char = ['a'..'z','A'..'Z'];
    caps:set of char=['A'..'Z'];
var s,s1:string;
    sent:string;
    f:text;
    i:integer;
    ns:integer;
    word:array[1..100000] of string;

procedure process(var s:string);
var nw:integer;
    name:string;
    i:integer;
begin
    inc(ns);
    s:=s+' a a ';
    nw:=1;
    word[1]:='';
    for i:=1 to length(s) do begin
        if s[i] in letters then
            word[nw]:=word[nw] + s[i]
        else if word[nw]<>'' then begin
            inc(nw);
            word[nw]:='';
        end;
    end;
    name:='';
    for i:=2 to nw-1 do begin
        if word[i][1] in caps then begin
            name:=name + word[i] + ' ';
        end else if name<>'' then begin
            writeln(f,name);
            name:='';
        end;
    end;
end;


begin
    assign(f,'proper.in');reset(f);
    s:='';
    while not eof(f) do begin
        readln(f,s1);
        s:=s+' '+s1;
    end;
    close(f);
    s:=s+'.';
    sent:='';
    assign(f,'proper.out');rewrite(f);
    ns:=0;
    for i:=1 to length(s) do begin
        if (s[i] in sentEnd) then begin
            process(sent);
            sent:='';
        end else sent:=sent+s[i];
    end;
    close(f);
end.
```

Задача 2. Железная дорога

Входной файл	bzd.in
Выходной файл	bzd.out
Ограничение по времени	1 секунда
Ограничение по памяти	256 мегабайт

На сайте Берляндских Железных Дорог (БЖД) любой может купить билет. Для этого ему лишь нужно заполнить несложную форму, а затем пройти еще несколько простых шагов. Руководство БЖД недавно задумалось о том, насколько легко людям использовать эту форму, и поручило вам провести измерения, чтобы оптимизировать количество действий, нужных для заполнения формы.

Форма состоит из двух полей: станция отправления и станция назначения. Когда человек заходит на сайт, в этих полях уже введены станции — те, которые сайт считает наиболее вероятными. Также под каждым полем есть список названий некоторых, самых популярных, станций — клик по элементу списка подставляет название станции в соответствующее поле. Наконец, есть кнопка «Поменять местами станцию отправления и станцию назначения». Кроме того, конечно, пользователь также

волен ввести названия нужных ему станций вручную. К сожалению, из-за ошибки в коде сайта, редактировать введенное или выбранное название нельзя, можно только ввести его заново.

Для целей эксперимента клик по кнопке смены станций отправления и назначения или по станции в списке популярных станций оценивается в одно очко действия. Ввод названия станции вручную стоит столько очков действия, сколько символов (включая пробелы и дефисы) содержит это название.

Вам нужно написать программу, которая по желаемым станциям отправления и назначения и начальному виду страницы (введенные станции, список популярных станций) определит минимальное количество очков действия, необходимое для того, чтобы ввести желаемые станции в соответствующие поля ввода.

Формат входных данных

В первой строке входного файла записано количество тестов в этом файле — единственное целое число $K(1 \leqslant K \leqslant 10)$. В последующих строках записаны K тестов.

Каждый отдельный тест состоит из нескольких строк. В первой строке теста записано название желаемой станции отправления. Во второй строке — название желаемой станции назначения. Гарантируется, что желаемая станция отправления не совпадает с желаемой станцией назначения.

В третьей строке — текущее введенное название станции отправления. В четвертой строке — текущее введенное название станции назначения.

В пятой строке — одно целое число $N(0 \leqslant N \leqslant 100)$ — количество популярных вариантов станции отправления. В следующих N строках записаны эти варианты, по одному на строке. Все варианты различны.

После этого записано количество популярных вариантов станции назначения — одно целое число $M(0 \leqslant M \leqslant 100)$. В следующих M строках записаны эти варианты, по одному на строке. Все варианты различны.

Названия станций могут состоять из больших и маленьких латинских букв, пробелов и дефисов. Длина каждого названия не превышает 100 символов. Названия станций всегда начинаются и заканчиваются буквой. Любой пробел или дефис обязательно окружен буквами.

Формат выходных данных

Для каждого теста выведите единственное целое число — количество очков действия, необходимое, чтобы ввести желаемые станции отправления и назначения в соответствующие поля. Выводите по одному ответу на строку. Порядок ответов должен соответствовать порядку тестов во входном файле.

Пример

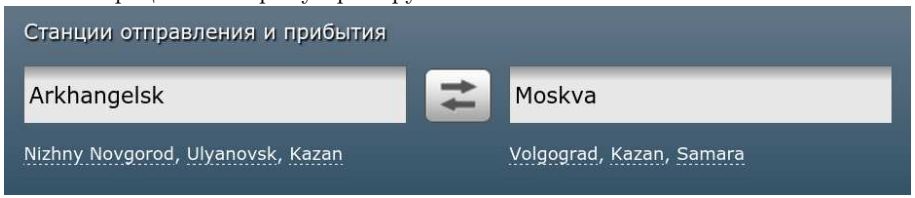
Входной файл	*Выходной файл*
2 Nizhny Novgorod Moskva Samara Petrozavodsk 2 Arkhangelsk Nizhny Novgorod 2 Moskva Samara Moskva Nizhny Novgorod Moskv Novgorod 2 Arkhangelsk Ulyanovsk 2 Volgograd Samara	2 21
1 Moskva Nizhny Novgorod Arkhangelsk Moskva 3 Nizhny Novgorod Ulyanovsk Kazan 3 Volgograd Kazan Samara	2

Примечание: Первый пример состоит из двух тестов. В первом тесте пользователю достаточно кликнуть по `Nizhny Novgorod` в списке популярных станций отправления, чтобы заменить `Samara` на желаемую станцию отправления, а затем кликнуть по `Moskva` в списке популярных станций назначения, чтобы заменить `Petrozavodsk` на желаемую станцию. Два клика требуют двух очков действия.

Во втором тесте у пользователя нет выбора, кроме как ввести названия станций вручную. Ввод `Moskva` потребует 6 очков действий, а ввод `Nizhny Novgorod` — 15 очков. В сумме требуется 21 очко.

Второй пример стоит из единственного теста. В этом тесте пользователю достаточно сначала кликнуть по `Nizhny Novgorod` в списке популярных станций отправления, а затем нажать на кнопку «Поменять местами станцию отправления и станцию назначения». На это потребуется два очка.

Иллюстрация ко второму примеру:



Решение

Несмотря на то, что задача кажется простой, здесь есть где ошибиться. Довольно очевидно, что нам нет смысла вводить или выбирать из списка город, который не совпадает с желаемой станцией отправления или назначения. Поэтому мы можем сразу отбросить из данных нам списков популярных станций все остальные города.

Займемся оценкой того, сколькими способами можно получить желаемый результат. Заметим, что каждое поле можно заполнить не более чем пятью способами: набрав название города вручную, взяв уже введенное содержимое станции назначения (кнопка «поменять местами» позволяет нам таким образом ввести и содержимое станции отправления), взяв уже введенное содержимое станции отправления, выбрав станцию из списка популярных станций отправления и, наконец, выбрав станцию из списка популярных станций назначения. Поскольку полей всего два, то всего получается не более $5 \cdot 5 = 25$ способов, каждый со своей ценой. Разумеется, не все эти способы применимы: нельзя ввести оба поля, использовав изначальное содержимое станции отправления, поскольку у нас нет операции «дублировать содержимое поля». Аналогично, невозможно дважды использовать содержимое станции назначения. Остальные 23 способа применимы, если исходные данные подходят. Например, если нам нужно добраться из города A в город B, и оба этих города есть в списке популярных станций назначения, то мы можем подставить в поле «станция назначения» город A, поменять поля местами, затем подставить в то же поле город B — потратив три очка действия. Оценку оставшихся 22 способов вы можете сделать самостоятельно. Решение, основанное на вышеизложенном разборе случаев, просто последовательно проверяет применимость каждого из них и выбирает способ с наименьшей стоимостью. Подобное решние можно скачать в архиве решений жюри с сайта `http://olympiads.nnov.ru` (решение `mlopatkin_ac.cpp`).

Однако при разборе случаев несложно ошибиться, поэтому изложим и более безопасный способ. Представим себе ориентированный граф, вершинами которого будут все возможные состояния пары полей — пары городов. Если с помощью допустимой операции (подстановка значения поля из списка популярных, ввод значения поля вручную, смена значений полей) мы можем перейти из одного состояния в другое, то в этом графе существует ребро, связывающее вершины, соответствующие этим состояниям. Весом этого ребра будет стоимость операции. Наша задача сводится к поиску кратчайшего пути из состояния (*введенная станция отправления, введенная станция назначения*) в состояние (*желаемая станция отправления, желаемая станция назначения*). Мы можем воспользоваться любым алгоритмом поиска кратчайшего пути во взвешенном графе, например, алгоритмом Дейкстры, как в приведенном ниже решении.

Отметим, что полученный граф будет совсем небольшим (если мы отбросим «неинтересные» популярные станции). В каждом поле может быть либо то, что уже введено в одно из полей, либо одна из желаемых станций, всего полей два — получаем не более 16 состояний.

Пример правильной программы

```
#!/usr/bin/env python
import sys
from collections import defaultdict, deque
INF = 9001

def readline():
    return sys.stdin.readline().strip()

def read_suggests(*interesting_cities):
    n = int(readline())
    for i in xrange(n):
        city = readline()
        if city in interesting_cities:
            yield city

def solve():
    source = readline()
    destination = readline()

    source_content = readline()
    destination_content = readline()

    source_suggests = set(
        read_suggests(source, destination))
    destination_suggests = set(
        read_suggests(source, destination))

    distances = defaultdict(lambda: INF)
    visited = set()

    def get_neighbours(v):
        v_s, v_d = v
        yield ((v_d, v_s), 1)
        for s in source_suggests:
            yield ((s, v_d), 1)
        for d in destination_suggests:
            yield ((v_s, d), 1)
        yield ((source, v_d), len(source))
        yield ((v_s, destination),
               len(destination))

    def get_nearest():
        return min((v for v in distances
                    if v not in visited),
                   key=lambda x: distances[x])

    distances[(source_content,
               destination_content)] = 0

    while ((source, destination)
           not in visited):
        v = get_nearest()
        visited.add(v)
        for n, d in get_neighbours(v):
            if distances[n]>distances[v]+d:
                distances[n]=distances[v]+d;

    return distances[(source, destination)]

n = int(readline())
for i in xrange(n):
    print solve()
```

Задача 3. Последнее число

Входной файл	`last.in`
Выходной файл	`last.out`
Ограничение по времени	1 секунда
Ограничение по памяти	256 мегабайт

На доске записано n целых чисел a_1, a_2, …, a_n. Миша может стереть любые два числа a_i и a_j и вместо них записать их сумму $a_i + a_j$. При этом Миша выплачивает своему другу Жене количество монет, равное минимуму из этих двух стертых чисел.

Миша хочет оставить на доске ровно одно число, отдав Жене как можно меньше монет. Помогите ему в этом.

Формат входных данных

В первой строке входных данных находится число n ($1 \leqslant n \leqslant 10^3$). Во второй строке находится последовательность чисел a_1, a_2, ..., a_n, записанная через пробелы ($1 \leqslant a_i \leqslant 10^3$).

Формат выходных данных

Выведите единственное число — минимальное количество монет, которое необходимо Мише заплатить Жене, чтобы на доске осталось ровно одно число.

Пример

Входной файл	*Выходной файл*
2 7 3	3
3 10 20 30	30

Примечание: Во втором примере Миша может сначала стереть числа 20 и 30 и записать на доске число 50. Затем стереть числа 10 и 50 и записать 60. Итого Миша должен отдать Жене $20 + 10 = 30$ монет.

Решение

После каждого действия, описанного в условии, количество чисел на доске уменьшается на 1. Значит, всего необходимо произвести ровно $n - 1$ действие.

Будем следовать следующей стратегии. Отсортируем все числа по возрастанию, и будем всегда стирать самое маленькое записанное на доске число в паре с самым большим. Следуя этой стратегии, надо будет отдать количество монет, равное сумме всех исходных чисел за исключением наибольшего. Докажем, что эта величина является наименьшей возможной.

Заметим, что в описанной выше стратегии все числа входят в величину выплаты только один раз. После того как число стирается, оно становится частью последнего числа, за которое Миша уже не платит Жене. Предположим, что надо было стереть два числа b_1 и b_2 ($b_1 < b_2$), которые уже были получены путем стирания двух пар исходных чисел, например a_i, a_j и a_k, a_l соответственно. Но тогда число a_i делало вклад в выплату дважды: первый раз когда было получено число b_1, а второй раз когда было получено число $b_1 + b_2$. Получаемое решение является менее выгодным. Значит, подобных стираний делать не надо.

Пример правильной программы

```
{$mode delphi}
var n:integer;
    s, max, x:integer;
    i:integer;
    f:text;

begin
    assign(f,'last.in');reset(f);
    read(f,n);
    max:=0;
    s:=0;
    for i:=1 to n do begin
        read(f,x);
        s:=s+x;
        if x>max then
            max:=x;
    end;
    close(f);
    assign(f,'last.out');rewrite(f);
    writeln(f,s-max);
    close(f);
end.
```


Задача 4. Взять все остатки

Входной файл	rem.in
Выходной файл	rem.out
Ограничение по времени	1 секунда
Ограничение по памяти	256 мегабайт

Недавно Петин учитель рассказал ему об операции деления с остатком. Число p называется *частным*, а число q — *остатком* от деления x на y, если $x = y \cdot p + q$, а также $0 \leqslant q < y$, и p и q целые. С вычислением *частного* у Пети проблем не возникает, а вот получение *остатка* дается ему с трудом. Но так как Петя — отличник, он решил поупражнятся в вычислении остатка.

Дома Петя нашел его любимую последовательность чисел $a_1, a_2, \ldots, a_n$. Теперь Петя будет выбирать числа $b_1, b_2, \ldots, b_m$ и с каждым b_i из них выполнять следующие операции:

1. Делить с остатком b_i на a_1,
2. Получившийся остаток делить с остатком на a_2,
3. Получившийся после второй операции остаток делить на a_3, и так далее,

   ...

n. Остаток от $(n-1)$-ого деления делить с остатком на a_n.

Для того, чтобы проверять себя, Петя попросил вас написать программу, которая по заданным последовательностям $a_1, a_2, \ldots, a_n$ и $b_1, b_2, \ldots, b_m$ найдет для каждого b_i остаток от последнего (n-ого) деления. Помогите Пете!

Формат входных данных

В первой строке входных данных находится число n ($1 \leqslant n \leqslant 10^5$). Во второй строке находится последовательность $a_1, a_2, \ldots, a_n$, записанная через пробелы ($1 \leqslant a_i \leqslant 10^9$). Далее следует число m ($1 \leqslant m \leqslant 10^5$), а после него на четвертой строке — числа $b_1, b_2, \ldots, b_m$, разделенные пробелами ($0 \leqslant b_i \leqslant 10^9$). Все числа во входных данных целые.

Формат выходных данных

Выведите m чисел — остатки от последней операции для каждого b_i.

Пример

Входной файл	*Выходной файл*
4 10 9 5 7 5 14 8 27 11 25	4 3 2 1 0

Примечание: В примере $b_3 = 27$. Последовательность действий, выполняемая Петей, будет следующая. Сначала разделим 27 на $a_1 = 10$, получим в остатке 7. Далее разделим 7 на $a_2 = 9$, получим в остатке 7. Разделим 7 на $a_3 = 5$, получим в остатке 2. Разделим 2 на $a_4 = 7$, получим в остатке 2, это и будет число, которое необходимо вывести.

Решение

Если для каждого числа выполнять все операции, то такое решение будет работать за $O(nm)$, что не уложится в ограничения по времени. Однако, можно заметить, что при выполнении операции деления с остатком x на y, то есть нахождении *частного* p и *остатка* q ($x = y \cdot p + q$, а также $0 \leqslant q < y$), обязательно либо $x < y$, тогда $p = 0$ и $q = x$, или $x \geq y$, тогда $p > 0$ и $q < x/2$. То есть при каждой операции, описанной в условии (взятие остатка от деления) число либо не изменяется, либо уменьшается хотя бы в два раза. Назовем операцию, в которой реализуется второй случай, *интересной*.

Нетрудно заметить, что число *интересных* операций для каждого b_i не превышает $\log_2(10^9) + 1 < 31$. Очевидно, что для каждого b_i можно пропустить *неинтересные* операции, поскольку они не меняют результата. Также, очевидно, что для любых i и j таких, что $j > i$, но $a_i \leqslant a_j$, операция деления с остатком на a_j будет *неинтересной*.

С учетом вышесказанного, будем строить решение таким образом:

1. Сначала удалим все a_j такие, что существует $i < j$ такое, что $a_i \leqslant a_j$. Это можно сделать одним проходом по массиву a. Теперь все элементы этого массива упорядочены по убыванию.

2. Для каждого b_i выполняем все *интересные* операции. Пусть после какой-то операции остаток был равен q. Тогда следующая *интересная* операция будет с числом a_j таким, что $a_j \leqslant q$ и j минимально. Число j можно найти бинарным поиском по массиву a.

Таком образом, на обработку каждой *интересной* операции такое решение выполняет $O(\log(n))$ действий, и итоговая асимптотика равна $O(m \log(n) \log(b_{max}))$, где b_{max} — максимальное из b.

Пример правильной программы

```
const maxn = 100005;

var a:array[1..maxn] of longint;
    n, m: longint;
    i, last, left, right, middle: longint;
    q: longint;
    fin, fout: text;

begin
    assign(fin, 'rem.in');
    reset(fin);
    assign(fout, 'rem.out');
    rewrite(fout);
    read(fin, n);
    for i := 1 to n do read(fin, a[i]);
    last := 1;
    for i := 2 to n do begin
        if a[i] < a[last] then begin
            inc(last);
            a[last] := a[i];
        end;
    end;
    n := last;
    read(fin, m);
    for i := 1 to m do begin
        read(fin, q);
        while true do begin
            left := 0;
            right := n + 1;
            while right - left > 1 do begin
                middle := (right + left) div 2;
                if a[middle] <= q then
                    right := middle
                else left := middle;
            end;
            if right = n + 1 then break;
            q := q mod a[right];
        end;
        write(fout, q, ' ');
    end;
    writeln(fout);
    close(fout);
end.
```


Задача 5. Передача опыта

Входной файл	`travel-cards.in`
Выходной файл	`travel-cards.out`
Ограничение по времени	3 секунды
Ограничение по памяти	256 мегабайт

Как известно, Берляндия — хорошо развитая страна, а поэтому в каждом городе есть школа. На недавнем съезде учителей стало ясно, что методики преподавания различных предметов во всех школах разные. Чтобы исправить это недоразумение, а также для улучшения качества образования было решено, что школам необходимо обмениваться опытом. Так как школы действительно разные, и у каждой имеется уникальный опыт, поступило предложение из каждой школы в каждую другую отправить делегата для перенимания опыта.

К сожалению, города в Берляндии находятся далеко друг от друга, поэтому их связывает минимальная возможная сеть дорог: из каждого города можно добраться по дорогам в любой другой не посещая ни один город дважды, но при этом лишь единственным способом. Также, еще большую проблему для учителей создает сложная система оплаты проезда по дорогам Берляндии.

Каждой дорогой владеет одна из K компаний, и для проезда по дорогам любой компании человек должен один раз купить проездной у этой компании. После этого он может проезжать по любым дорогам этой компании без дополнительной платы. Стоимость проездного i-ой компании ($1 \leqslant i \leqslant K$) равна c_i бурлей.

Каждому из делегатов нужно купить необходимые ему проездные. Помогите учителям посчитать, сколько же всего бурлей будут стоить все проездные для всех делегатов. Так как это число может быть большим, найдите остаток от деления этого числа на $1\,000\,000\,007$ ($10^9 + 7$).

Формат входных данных

В первой строке входных данных находятся два натуральных числа N и K ($1 \leqslant N \leqslant 1\,500\,000$, $1 \leqslant K \leqslant 1\,500\,000$) — число городов в Берляндии и число компаний-владельцев дорог соответственно. На следующих $(N-1)$ строках дано описание дорог Берляндии, на i-ой из этих строк (эти строки нумеруются с единицы, города тоже нумеруются с единицы) находятся два числа b_i и t_i ($1 \leqslant b_i \leqslant N$, $1 \leqslant t_i \leqslant K$), означающие, что существует дорога из города с номером $i+1$ в город с номером b_i, и владеет ей компания с номером t_i. По всем дорогам можно передвигаться в обе стороны; также, гарантируется, что из любого города можно добраться в любой другой по существующим дорогам.

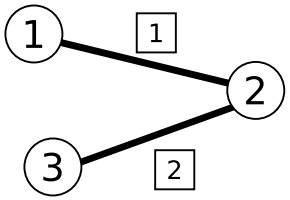
На последней строке входных данных находятся 4 числа A, B, C, c_0, задающие стоимости проездных ($1 \leqslant C \leqslant 1\,000\,000\,001$, $0 \leqslant A, B, c_0 < C$). А именно, стоимости проездных вычисляются по формуле $c_i = (A \cdot c_{i-1} + B) \bmod C$, где c_i — стоимость проездного i-ой компании, а «$X \bmod Y$» — остаток от деления X на Y.

Формат выходных данных

Выведите единственное число — остаток от деления суммарной стоимости необходимых проездных на $1\,000\,000\,007$.

Пример

Входной файл	*Выходной файл*
3 2 1 1 2 2 1 2 100 1	32
3 1 1 1 2 1 1 2 100 1	18



Примечание: В первом тесте из примера карта дорог выглядит так, как показано на рисунке. Кружками обозначены города, линиями — дороги, а числа в квадратах около дорог обозначают номера компаний, обслуживающих эти дороги. Цены проездных в этом примере равны $c_1 = 3$ бурля, $c_2 = 5$ бурлей. Таким образом, делегаты, едущие из первого города в третий, или обратно, должны купить оба проездных, остальным делегатам нужно проехать только одну дорогу, поэтому им нужно купить только один проездной. Итоговая стоимость проездных равна $2 \cdot (3 + 5) + 2 \cdot 3 + 2 \cdot 5 = 32$ бурлей.

Во втором тесте схема дорог выглядит так же, а компания-владелец только одна. Стоимость проездного равна 3 бурля. Поэтому все 6 делегатов должны купить единственный проездной, и суммарная стоимость $6 \cdot 3 = 18$ бурлей.

Тесты, в которых выполняются дополнительные ограничения $1 \leqslant N, K \leqslant 100$, имеют суммарную стоимость не менее 30 баллов.

Тесты, в которых выполняются дополнительные ограничения $1 \leqslant N, K \leqslant 1000$, имеют суммарную стоимость не менее 50 баллов.

Тесты, в которых выполняются дополнительные ограничения $1 \leqslant N, K \leqslant 100\,000$, имеют суммарную стоимость не менее 75 баллов.

Решение

Будем подходить к решению последовательно. Для начала заметим, что можно немного переформулировать условие и подсчитывать не стоимость проездных, которые нужно купить каждому делегату, а количество делегатов, которые купят проездной каждой компании. Обозначим число делегатов, которые должны купить проездной у компании номер i, за D_i. Тогда, посчитав $D_1, \ldots, D_K$, несложно посчитать ответ — он равен $\sum_1^K D_i c_i$. Зафиксируем одну компанию номер i и посчитаем D_i.

Заметим, что сеть дорог в задаче представляет собой *дерево*. Тогда дороги, принадлежащие этой компании, разделяют дерево на несколько *компонент*; нетрудно увидеть, что проездной купят те и только те делегаты, чьи места отправления и места назначения лежат в разных компонентах. Пусть всего дерево разделилось на m компонент, и количества городов в компонентах равно $s_1, \ldots, s_m$, тогда

$D_i = N^2 - \sum_1^m s_i^2$. Действительно, всего делегатов N^2, а не купят проездные только те из них, чей путь лежит в пределах одной компоненты. *Компоненты* можно найти, запустив поиск в глубину и не переходя по дорогам, принадлежащим данной компании. Такое решение будет иметь сложность $O(NK)$, так как для каждой из K компаний мы будем запускать поиск в глубину, имеющий сложность $O(N)$.

Посчитать D_i можно и быстрее, воспользовавшись некоторой структурой данных, например, деревом отрезков или деревом Фенвика, и подвесив дерево за некоторую вершину. Тогда D_i можно найти за временную сложность $O(E_i \log(N))$, где E_i — число дорог, принадлежащее данной компании; таким образом, сложность всего решения будет $O(N \log(N))$. Оставляем читателю как упражнение продумать детали данного решения самостоятельно.

Однако, можно вспомнить, что нам нужно посчитать не одно D_i, а все. Подвесим дерево за любую вершину, и посмотрим на какую-нибудь *компоненту* компании номер i. Заметим, что если некоторая вершина принадлежит компоненте, а ее предок — нет, то дорога из этой вершины в предка должно принадлежать компании номер i; единственное исключение — корень дерева, тогда у нее нет предка. В обоих случаях, очевидно, такая вершина ровно одна, назовем эту вершину *корнем* компоненты. Видно, что все вершины, за исключением корня дерева, являются *корнем* ровно одной компоненты ровно одной компании. Заметим, что размер компоненты s равен разности размера поддерева *корня* компоненты и размеров поддеревьев *корней* компонент той же компании, непосредственно примыкающих к данной компоненте снизу. Тогда построим такое решение задачи: запустим поиск в глубину из корня, и будем для каждой компании i хранить суммарный размер O_i *компонент* данной компании, из которых поиск в глубину уже **вышел**. Тогда размер компоненты можно определить по формуле $s = T - (O_{i,out} - O_{i,in})$, где T — размер поддерева *корня* компоненты, $O_{i,in}$ — значение O_i на момент входа поиска в глубину в *корень* компоненты, а $O_{i,out}$ — значение O_i на момент выхода из нее. Действительно, это так, ведь $(O_{i,out} - O_{i,in})$ — суммарный размер компонент, находящихся ниже данной. Тогда, запустив один поиск в глубину, можно сразу найти все D_i, а по ним сосчитать ответ. Данное решение позволяет решить задачу за линейное время $O(N + K)$ и является полным решением.

Пример правильной программы

```
{$mode delphi}
{$r-,q-,s-,i-}
const maxn=1500000;
      mdl=1000000007;
var gr:array[1..maxn] of array of record
           v:integer; b:integer;
         end;
    f:text;
    n,k:integer;
    c:array[0..maxn] of int64;
    u,v,a,b:integer;
    cmod:integer;
    cursize:array[0..maxn] of integer;
    sumsq:array[0..maxn] of int64;
    ans:int64;
    i:integer;
    nn:int64;

function find(i, x, p:integer):integer;
var j:integer;
    size0:int64;
begin
    size0:=cursize[x];
    result:=1;
    for j:=0 to length(gr[i])-1 do
        if p<>gr[i][j].v then
            result:=result+find(gr[i][j].v,
                                gr[i][j].b, i);
    sumsq[x]:=(sumsq[x] +
        sqr(result-cursize[x]+size0))mod mdl;
    cursize[x]:=size0+result;
end;
```

```
procedure add(u,v,b:integer);
var l:integer;
begin
    l:=length(gr[u]);
    setlength(gr[u], l+1);
    gr[u][l].v:=v;
    gr[u][l].b:=b;
end;

begin
    assign(f,'travel-cards.in');reset(f);
    read(f,n,k);
    for i:=1 to n-1 do begin
        u:=i+1;
        read(f,v,b);
        add(u,v,b);
        add(v,u,b);
    end;
    read(f,a,b,cmod,c[0]);
    close(f);
    for i:=1 to k do
        c[i]:=(a*c[i-1]+b) mod cmod;
    fillchar(cursize, sizeof(cursize), 0);
    fillchar(sumsq, sizeof(sumsq), 0);
    find(1, 0, 0);
    nn:=int64(n)*n mod mdl;
    for i:=1 to k do begin
        sumsq[i]:=(sumsq[i] +
            sqr(int64(n)-cursize[i])) mod mdl;
        ans:=(ans+(nn-sumsq[i])*c[i]) mod mdl;
    end;
    if ans<0 then ans:=ans+mdl;
    assign(f,'travel-cards.out');rewrite(f);
    writeln(f,ans);
    close(f);
end.
```

Задача 6. Светофоры

Входной файл	`traffic.in`
Выходной файл	`traffic.out`
Ограничение по времени	1 секунда
Ограничение по памяти	256 мегабайт

Вы — министр транспорта в Тридевятом королевстве. Дороги, как известно, являются одной из основных проблем королевства, тем не менее, недавно была закончена постройка Великого Королевского Тракта — самой длинной дороги в Тридевятом царстве. Тракт представляет из себя прямую дорогу, поэтому далее будем называть координатой точки на тракте расстояние от начала тракта до нее.

Вам стало известно, что король скоро собирается проехать по этому тракту от начала до конца, причем поедет он со скоростью 1. Когда именно он собирается начать, вам неизвестно, но зато вы уверены, что это прозойдет в какой-то случайный целочисленный момент времени между t_1 и t_2 включительно. В рамках модернизационной программы вы отдали поручение построить на Тракте n светофоров, которые должны будут впечатлить проезжающего короля. Светофоры в некотором смысле одноразовые, а именно они поначалу горят зеленым, затем красным, а потом снова зеленым, не переключаясь больше, пока не сломаются (но можно считать, что сломаются они не скоро). Про каждый светофор вам известно три величины: его положение на Тракте, время, когда он загорится красным, и время, когда он вновь загорится зеленым. Так получилось, что последний светофор стоит прямо на конце Тракта. Понятно, что король не станет подавать плохой пример подданным: никакой светофор он не будет проезжать на красный свет (в том числе это относится и к последнему светофору). В момент, когда красный свет только загорается, ехать уже становится нельзя, а когда загорается зеленый — сразу же можно.

Ваша премия как министра напрямую зависит от того, насколько долго король будет ехать по тракту, поэтому вас интересует вопрос, сколько в среднем времени ему потребуется на то, чтоб проехать Тракт от начала до конца (включая время ожидания на последнем светофоре, если это потребуется). Напишите программу, которая посчитает эту величину.

Формат входных данных

В первой строке входного файла вводится число n — количество светофоров ($1 \leqslant n \leqslant 100\,000$). Затем в каждой из следующих n строк — описания светофоров. В каждой из них содержатся 3 целых числа x_i, s_i, f_i ($0 \leqslant x_i, s_i, f_i \leqslant 10^9$), описывающих очередной светофор: x_i — это положение светофора на Тракте, s_i — момент времени, когда светофор загорится красным, f_i — момент времени, когда он вновь загорится зеленым ($0 \leqslant s_i < f_i$). Светофоры упорядочены по величине x, т.е. для любых номеров i и j таких, что $1 \leqslant i < j \leqslant n$, верно, что $x_i < x_j$. Кроме того, гарантируется, что длина Тракта больше нуля.

В последней строке задаются целые числа t_1 и t_2 — начальный и конечный момент времени, когда король может начать свой путь ($0 \leqslant t_1 \leqslant t_2 \leqslant 10^9$).

Формат выходных данных

Выведите одно число — время в пути, усредненное по всем возможным временам начала (т.е. среднее арифметические времен в пути, вычисленных для всех возможных целочисленных времен старта от t_1 до t_2 включительно). Ваш ответ будет считаться верным, если его относительная погрешность будет не более 10^{-9}, т.е., если ваш ответ равен a, а правильный — b, то должно выполняться $|a - b| \leqslant 10^{-9}b$.

Пример

Входной файл	*Выходной файл*
2 4 11 12 8 14 15 6 9	8.5000000000

Примечание: В примере король может выехать в любой момент между 6 и 9 включительно.

Если король выезжает в момент времени 6, то к первому светофору он подъезжает в момент времени 10, этот светофор еще горит зеленым, поэтому король едет дальше без остановки. Ко второму светофору король подъезжает в момент времени 14, этот светофор в этот момент загорается красным, поэтому король ждет до момента времени 15, когда светофор загорается опять зеленым. Итого получается время в пути равно 9.

Если король выезжает в момент времени 7, то к первому светофору он подъезжает в момент времени 11, когда этот светофор только загорелся красным — король ждет до момента времени 12. Ко второму светофору король подъезжает в момент времени 16, когда этот светофор уже горит зеленым. Итого получается время в пути равно 9.

Если король выезжает в момент времени 8 или 9, то он не ждет ни на одном светофоре, и время в пути получается 8.

Итого среднее время в пути есть $(9 + 9 + 8 + 8)/4 = 8.5$.

Решение

Решать эту задачу можно было следующим образом. Для каждого x и t посчитаем, в какой момент времени король доедет до конца тракта, если в точку с

координатой x он приехал в момент времени t (причем, если в этой точке находится светофор, то имеется в виду, что в момент времени t король только подъехал к светофору). Обозначим такой искомый момент времени как $ans[x,t]$.

Будем идти от бóльших x к меньшим. Во-первых, для $x = x_n$, т.е. для последнего светофора на тракте, ответ очевиден: $ans[x,t] = t$, если в момент времени t последний светофор горит зеленым (т.е. $t < s_n$ или $t > f_n$), иначе $ans[x,t] = f_n$.

Теперь вычислим ответы для всех t при некотором x, считая, что ответы для $x+1$ мы уже знаем.

Если в точке x светофора нет, то все очевидно: $ans[x,t] = ans[x+1,t+1]$. Иначе надо посмотреть, горит ли светофор в точке x красным в момент времени t. Если светофор в этот момент горит зеленым (т.е. $t < s_i$ или $t > f_i$, где i — номер соответствующего светофора), то $ans[x,t] = ans[x+1,t+1]$. Иначе $ans[x,t] = ans[x+1, f_i+1]$.

Таком образом можно вычислить $ans[0,t]$ для всех t — это и будет время прибытия на конец тракта для всех возможных моментов старта. Осталось посчитать сумму $ans[0,t]$ для всех t от t_1 до t_2 включительно, вычесть из нее сумму всех чисел от t_1 до t_2 включительно, и разделить результат на $t_2 - t_1 + 1$.

Непосредственная реализация этого решения, конечно, не укладывается в ограничение по времени. Для ускорения следует использовать следующие четыре соображения.

Во-первых, можно хранить не матрицу ответов, а только текущую ее строку, т.е. все значения $ans[x,t]$ только при текущем x. При этом изложенное выше решение в общем случае (без учета светофоров) при переходе от $x+1$ к x «сдвигает» всю эту строку на единицу (т.е. ответ для данного t при некотором x равен ответу для $t+1$ при $x+1$). Вместо этого можно сохранять ответы на тех же местах, просто сдвигая «начало отсчета» массива. Или, что то же самое, хранить не массив $ans[x,t]$ для каждого t при фиксированном x, а $ans'[x,d] = ans[x,x+d]$ для каждого d при фиксированном x. Тогда в общем случае (без учета светофором) имеем $ans'[x,d] = ans[x,x+d] = ans[x+1,x+d+1] = ans'[x+1,d]$, т.е. все элементы массива остаются на своих местах.

Во-вторых, чтобы учесть очередной светофор, надо просто нескольким подряд идущим элементам массива присвоить одно и то же значение. Это можно делать достаточно быстро с использованием специальных структур данных, например, с помощью дерева отрезков с массовыми операциями.

В-третьих, изложенное выше решение на каждом шагу уменьшает x всего на единицу. На самом деле несложно переходить от каждого светофора сразу к предыдущему, таким образом вычисляя ответы не для всех x, а только для тех x, где находятся светофоры.

Наконец, можно заметить, что решение можно вычислять не для всех d, а только для некоторых «интересных» — равных $s_i - x_i$ или $f_i - x_i$ для некоторого i. Между этими «интересными» точками ответы всегда равны между собой.

Решение с учетом всех этих трех соображений работает за время $O(n \log T)$, где T — максимальное интересующее нас время (например, его можно взять равным $\max(t_2, f_i) + n$), и укладывается в ограничения по времени.

Пример правильной программы

```
#include <cstdio>
#include <vector>
#include <set>
#include <algorithm>

using namespace std;

const int maxn = 3e5+5;
const int inf = 1e9 + 5;
const int START = 1;
const int FINISH = -1;

vector<int> uniq;
int a[maxn];
int b[maxn];
int n, t1, t2, delta;
multiset <int> S;
long long total;
vector <pair<int,int>> list[maxn];
int tree[maxn * 4];
int seg_ans[maxn];

inline void add_with_stops(int first,
                           int last, int T)
{
    first = max(first, t1);
    last = min(last, t2);
    if (first > last)
        return;
    int len = last - first + 1;
    if (len <= 0)
        return;
    long long to_add =
        (long long)len * (T + delta) -
        (long long)(first + last) * len / 2;
    total += to_add;
}

inline void add_no_stops(int first, int last)
{
    first = max(first, t1);
    last = min(last, t2);
    if (first > last)
        return;
    total +=
        (long long)(last - first + 1) * delta;
}

int get(int i, int tl, int tr, int pos)
{
    if (tl + 1 == tr)
        return tree[i];
    int m = (tl + tr) / 2;
    if (pos < m)
        return min(tree[i],
                   get(i * 2, tl, m, pos));
    else return min(tree[i],
                    get(i * 2 + 1, m, tr, pos));
}

int get(int pos)
{
    return get(1, 0, uniq.size(), pos);
}

void upd(int i, int L, int R, int A,
         int B, int val)
{
    if (L >= B || A >= R)
        return;
    if (L == A && R == B)
    {
        tree[i] = val;
        return;
    }
    int M = (L + R) / 2;
    upd(i * 2, L, M, A, min(M, B), val);
    upd(i * 2 + 1, M, R, max(M, A), B, val);
}

void build(int i, int tl, int tr)
{
    tree[i] = inf;
    if (tl + 1 == tr)
        return;
    int m = (tl + tr) / 2;
    build(i * 2, tl, m);
    build(i * 2 + 1, m, tr);
}

int main() {
    freopen("traffic.in", "r", stdin);
    freopen("traffic.out", "w", stdout);
    scanf("%d", &n);

    for(int j=0; j<n; j++)
    {
        int x;
        scanf("%d%d%d", &x, &a[j], &b[j]);

        if (j == n - 1)
            delta = x;
        a[j] -= x;
        b[j] -= x;
        uniq.push_back(a[j]);
        uniq.push_back(b[j]);
        uniq.push_back(b[j] + 1);
    }
    sort(uniq.begin(), uniq.end());
    uniq.resize(
        unique(uniq.begin(),uniq.end())
        - uniq.begin());
    for(int i=0; i<n; i++)
    {
        a[i] = lower_bound(
               uniq.begin(), uniq.end(),a[i])
                    -uniq.begin();
        b[i] = lower_bound(
                uniq.begin(), uniq.end(),b[i])
                    - uniq.begin();
```

```
    }
    build(1, 0, uniq.size());
    for (int i = n - 1; i >= 0; i--)
    {
        int val = get(b[i]);
        if (val == inf)
            seg_ans[i] = uniq[b[i]];
        else seg_ans[i] = seg_ans[val];
        upd(1, 0, uniq.size(),
            a[i], b[i] + 1, i);
    }
    for(int i=0; i<n; i++)
    {
        list[a[i]].push_back(
            make_pair(i, START));
        int pos = lower_bound(
            uniq.begin(), uniq.end(),
            uniq[b[i]] + 1) - uniq.begin();
        list[pos].push_back(
            make_pair(i, FINISH));
    }
    scanf("%d%d", &t1, &t2);
    add_no_stops(t1, uniq[0] - 1);
    for(int j=0; j<uniq.size(); j++)
    {
        for (auto p : list[j])
        {
            if (p.second == START)
                S.insert(seg_ans[p.first]);
            else S.erase(S.find(
                    seg_ans[p.first]));
        }
        int first = uniq[j];
        int last = (j+1 == (int)uniq.size()?
                        t2 : uniq[j + 1] - 1);
        if (S.empty())
            add_no_stops(first, last);
        else add_with_stops(first, last,
                    *S.rbegin());
    }
    long double answer = (long double)total
                            / (t2 - t1 + 1);
    printf("%.10lf\n", (double)answer);
}
```