Разбор задач

32 Районная олимпиада школьников Красноярского края по информатике, 7-8 классы

(Муниципальный этап ВОШ по информатике)

6 декабря 2018 г.

ID	Задача	Тема	%
1603	A. Принцип Дирихле	Задачи для начинающих	9
1612	B. Красивая дата	Задачи для начинающих	12
1608	C. Максимальный прямоугольник	Задачи для начинающих	20
1595	D. Лексикографически минимальное число	Задачи для начинающих	23
1609	E. Полка с книгами	Сортировка и последовательности	30
1582	F Миссия невыполнима	Двумерные массивы	45

Автор разбора: Беляев Сергей Николаевич

Средняя сложность задач: 23%

Задача A. Принцип Дирихле

(Время: 1 сек. Память: 16 Мб Баллы: 100)

Напомним, что *принцип Дирихле* говорит о следующем: если некоторое количество кроликов рассадить по клеткам и при этом окажется, что число клеток меньше, чем число кроликов, то обязательно найдется хотя бы одна клетка, в которой больше одного кролика.

Нас же будет интересовать более общий случай, когда у нас N клеток и M кроликов. Вам требуется вычислить максимальное количество кроликов, которое гарантированно окажется в одной из клеток.

Входные данные

Входной файл INPUT.TXT содержит целые числа N и M ($1 \le N, M \le 10^{19}$).

Выходные данные

В выходной файл OUTPUT.TXT выведите ответ на задачу.

Пример

№	INPUT.TXT	OUTPUT.TXT
1	2 3	2

Система оценки

Решения, работающие только для N, $M \le 10^9$, будут оцениваться в 80 баллов.

Задача A. Принцип Дирихле

N – количество клеток

M – количество кроликов

K = **M** **div** **N**

Q = **M** **mod** **N**

M = **K**×**N** + Q

Очевидно, что ответ не может быть меньше, чем K, так как (K-1)×N < M. С другой стороны, поскольку M = K×N + Q, то всегда допустима такая рассадка кроликов, что в Q клетках будет (K+1) кроликов, а в остальных (N-Q) клетках – ровно K кроликов:

$$M = (K+1)\times Q + K\times(N-Q)$$

Поэтому ответ не может быть больше, чем K+1. Причем при Q=0 ответ будет равен K.

Получаем простую реализацию:

```
read(N, M)
if(M mod N = 0) write(M div N)
           else write(M div N + 1)
```

Ограничения на N и M в данной задаче требуют правильного выбора целого типа при реализации на разных языках:

```
//C++
#include <iostream>
using namespace std;
int main(){
  unsigned long long n,m;
  cin >> n >> m;
  if(m%n == 0) cout << m/n;
          else cout << m/n + 1;
  return 0;
}
```

```
//PascalABC.NET 3.4
begin
  var a := ReadString.Split;
  var n := a[0].ToBigInteger;
  var m := a[1].ToBigInteger;
  write((m+n-1) div n)
end.
```

```
#Python 3
n, m = map(int, input().split())
print((m+n-1)//n)
```

Задача B. Красивая дата

(Время: 1 сек. Память: 16 Мб Баллы: 100)

Вася считает, что дата, записанная в формате ДДММГГГГ *красивая*, если в ней день, месяц и год – нечётные числа, то есть числа, которые не делятся на 2.

Например, дата «первое сентября 2019 года» (01092019) является красивой, так как числа 1, 9 и 2019 – нечётные. А вот дата «13 июня 1997 года» (13061997) не является красивой, так как в ней номер месяца 6 – чётное число.

Вася устал каждый раз проверять красоту дат и просит вас написать программу, которая ему в этом поможет.

Входные данные

В первой строке входного файла INPUT.TXT содержится корректная дата в виде последовательности десятичных цифр формата ДДММГГГГ. Такое представление даты можно рассматривать как восьмизначное целое число с одним возможным лидирующим нулём. Дата может быть любой из промежутка 10 000 лет: от 1 января 1 года (01010001), до 31 декабря 9999 года (31129999).

Выходные данные

В выходной файл OUTPUT.TXT выведите «Yes» (без кавычек), если дата является красивой, и «No» (без кавычек) – в противном случае.

Примеры

№	INPUT.TXT	OUTPUT.TXT
1	01092019	Yes
2	13061997	No

Задача B. Красивая дата

Здесь достаточно проверить, что 2-я, 4-я и 8-я цифры – нечётные. Для этого можно считать дату в строковую переменную (примеры на C++ и Python) и рассмотреть соответствующие символы. Также можно осуществить чтение в переменную целого типа (пример на Pascal), затем извлечь и проверить данные цифры:

```
//C++
#include <bits/stdc++.h>
using namespace std;
int main(){
  string s;
  cin >> s;
  if(s[1]%2 && s[3]%2 && s[7]%2) cout << "Yes";
                          else cout << "No";
  return 0;
}
```

```
//Pascal
var d : integer;
begin
  read(d);
  if odd(d div 1000000) and odd(d div 10000) and odd(d) then write('Yes')
                                                    else write('No')
end.
```

```
#Python
s = input()
print(["No", "Yes"][int(s[1])%2 and int(s[3])%2 and int(s[7])%2])
```

Задача C. Максимальный прямоугольник

(Время: 1 сек. Память: 16 Мб Баллы: 100)

На клетчатом поле размером N×M разместили две фишки.

Требуется определить максимально возможную площадь прямоугольника, которому принадлежит только одна фишка. При этом границы прямоугольника должны совпадать с границами клеток поля.

Входные данные

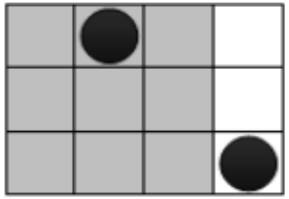
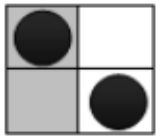
Первая строка входного файла INPUT.TXT содержит два целых числа N и M – количество строк и количество столбцов клетчатого поля ($2 \le N, M \le 1000$).

В следующих двух строках заданы координаты фишек: в (i+1)-й строке записана координата i-й фишки (x_i,y_i). Гарантируется, что фишки имеют различные целочисленные координаты, которые принадлежат клетчатому полю ($1 \le x_i \le M$, $1 \le y_i \le N$, i=1..2).

Выходные данные

В выходной файл OUTPUT.TXT выведите одно целое число – площадь искомого прямоугольника.

Примеры

№	INPUT.TXT	OUTPUT.TXT	Пояснение
1	3 4 2 1 4 3	9	
2	2 2 1 1 2 2	2	

Система оценки

Решения, работающие для $N, M \le 10$, будут оцениваться в 30 баллов.

Решения, работающие для $N, M \le 70$, будут оцениваться в 60 баллов.

Задача C. Максимальный прямоугольник

Частичное решение $O(N^2 \times M^2)$ – 60 баллов

Можно осуществить перебор всех возможных прямоугольников, используя 4 вложенных цикла. Для каждого рассматриваемого прямоугольника определим количество фишек внутри его (это можно сделать за O(1)) и среди тех из них, внутри которых ровно одна фишка, найдем прямоугольник максимальной площади. Асимптотика – $O(N^2 \times M^2)$.

В случае неэффективного подсчета фишек внутри прямоугольника (например, с помощью перебора всех его клеток) мы получим частичное решение на 30 баллов, работающее за $O(N^3 \times M^3)$.

Полное решение O(1) – 100 баллов

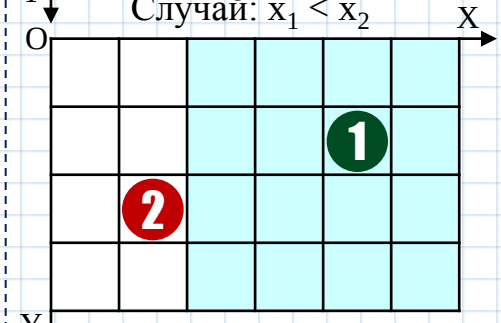
Для определенности будем считать, что зеленая фишка №1 с координатами (x_1,y_1) должна принадлежать искомому прямоугольнику, а красная фишка №2 – не должна. Тогда очевидно, что фишка №2 с координатами (x_2,y_2) должна касаться границы искомого прямоугольника (в противном случае его площадь не будет максимальной).

Так, искомый прямоугольник может иметь следующие 4 варианта координат (прямоугольник определяем через его левый верхний и правый нижний угол):

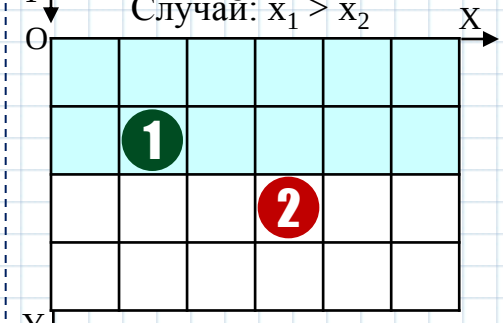
$(1,1) – (x_2-1, N)$, слева: такой прямоугольник возможен, когда $x_1<x_2$;
$(x_2+1,1) – (M, N)$, справа: такой прямоугольник возможен, когда $x_1>x_2$;
$(1,1) – (M, y_2-1)$, сверху: такой прямоугольник возможен, когда $y_1<y_2$;
$(1,y_2+1) – (M, N)$, снизу : такой прямоугольник возможен, когда $y_1>y_2$;

Останется лишь выбрать прямоугольник максимальной площади из всех возможных. Также следует помнить, что как одна, так и другая фишка могут быть внутри искомого прямоугольника. Поэтому вышеупомянутые действия нужно выполнить дважды.

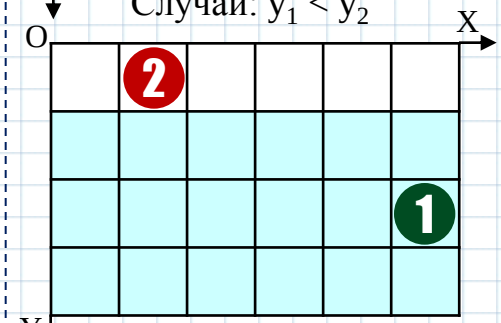
Случай: $x_1 < x_2$



Случай: $x_1 > x_2$



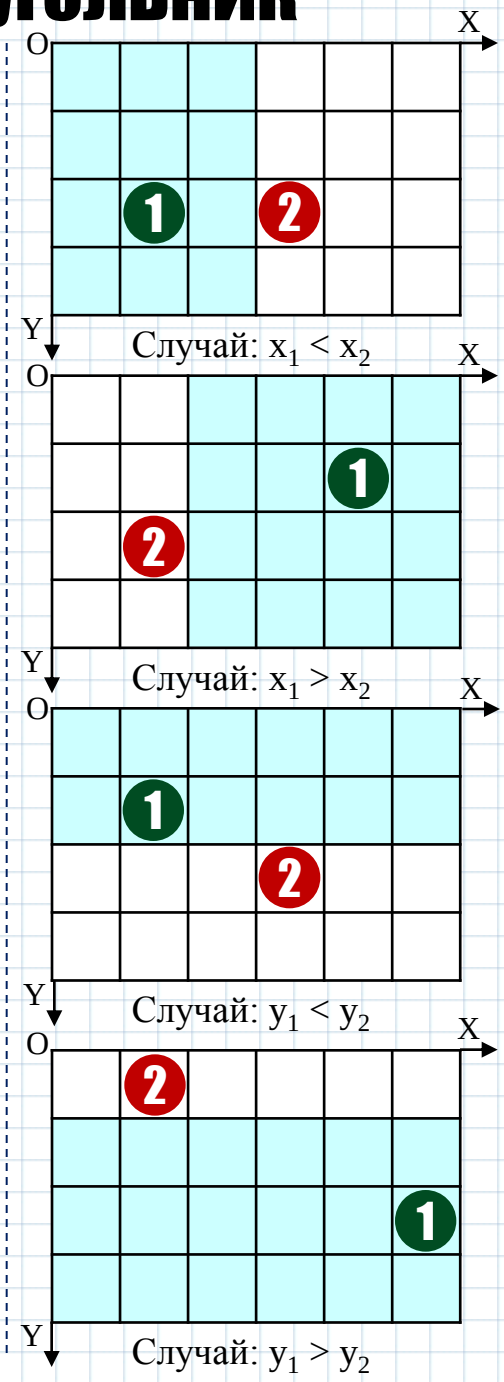
Случай: $y_1 < y_2$



Случай: $y_1 > y_2$

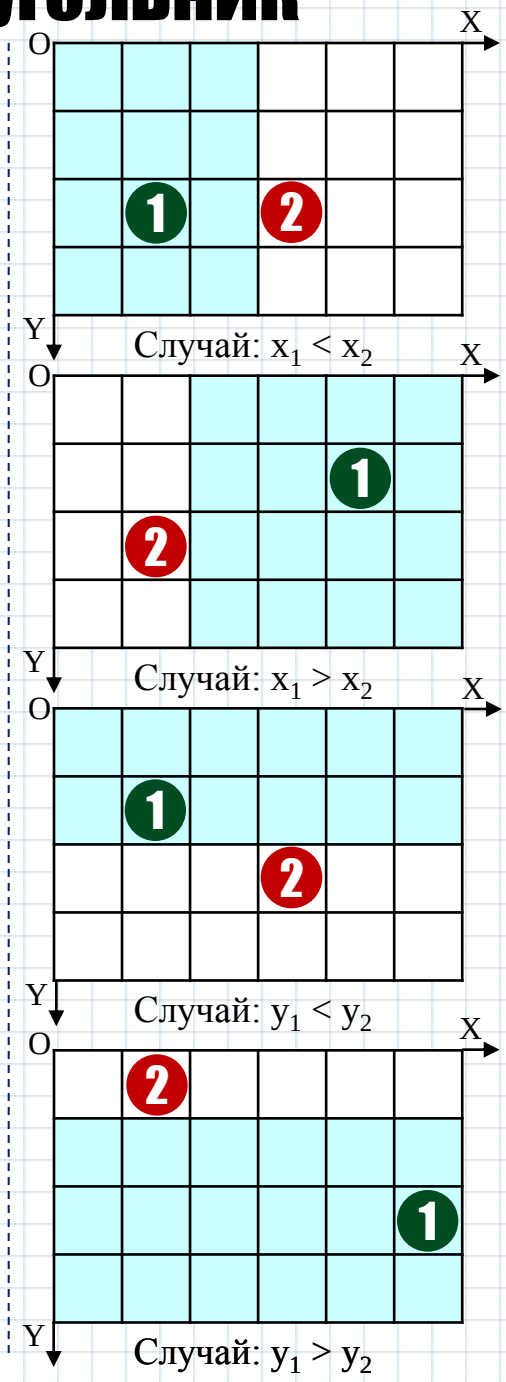
Задача C. Максимальный прямоугольник

```
//C++
#include <bits/stdc++.h>

using namespace std;

int n,m,x,y,x2,y2,res=0;

void find(int x, int y, int x2, int y2){
  if(x<x2) res = max(res, (x2-1)*n);
  if(x>x2) res = max(res, (m-x2)*n);
  if(y<y2) res = max(res, (y2-1)*m);
  if(y>y2) res = max(res, (n-y2)*m);
}

int main(){
  cin >> m >> n >> x >> y >> x2 >> y2;

  find(x,y,x2,y2);
  find(x2,y2,x,y);

  cout << res;
  return 0;
}
```



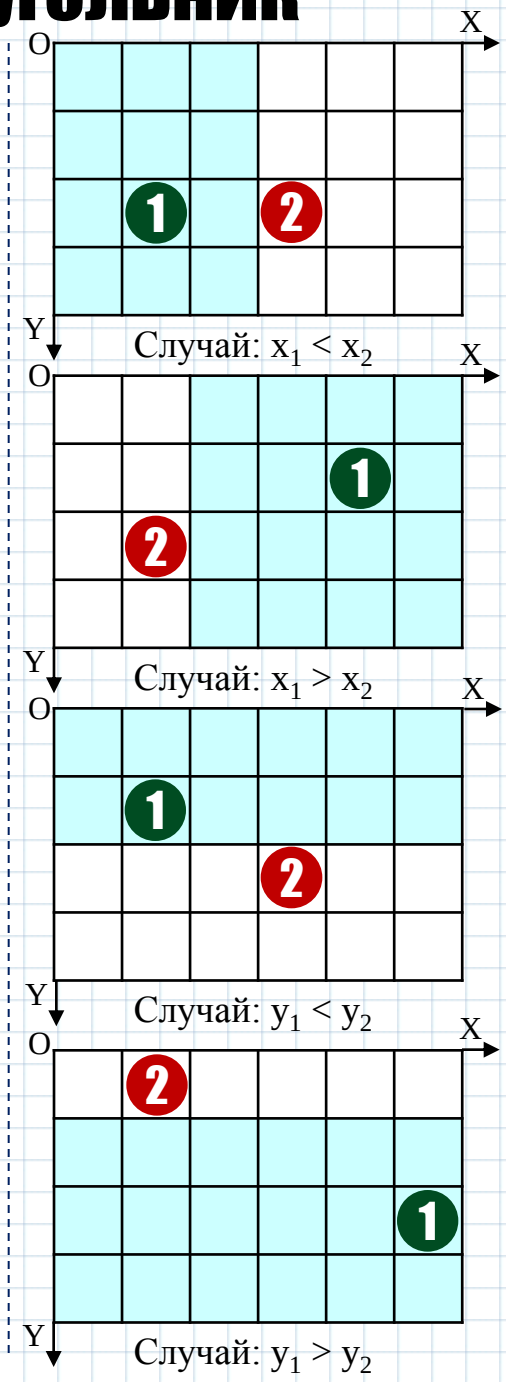
Задача C. Максимальный прямоугольник

```
//Free Pascal
uses Math;

var n,m,x,y,x2,y2,res : integer;

procedure find(x,y,x2,y2 : integer);
begin
  if x<x2 then res := max(res, (x2-1)*n);
  if x>x2 then res := max(res, (m-x2)*n);
  if y<y2 then res := max(res, (y2-1)*m);
  if y>y2 then res := max(res, (n-y2)*m);
end;

begin
  read(m,n,x,y,x2,y2);

  res := 0;
  find(x,y,x2,y2);
  find(x2,y2,x,y);

  write(res)
end.
```



Задача C. Максимальный прямоугольник

```python
#Python
m, n = map(int, input().split())
x, y = map(int, input().split())
x2, y2 = map(int, input().split())

def find(x,y,x2,y2):
    global res
    if x<x2:
        res = max(res, (x2-1)*n)
    if x>x2:
        res = max(res, (m-x2)*n)
    if y<y2:
        res = max(res, (y2-1)*m)
    if y>y2:
        res = max(res, (n-y2)*m)

res = 0
find(x,y,x2,y2)
find(x2,y2,x,y)

print(res)
```



Задача D. Лексикографически минимальное число

(Время: 1 сек. Память: 16 Мб Баллы: 100)

Найдите число, десятичная запись которого является лексикографически минимальной среди десятичных записей всех целых чисел от A до B.

Входные данные

Входной файл INPUT.TXT содержит два целых числа: A и B ($1 \le A \le B \le 10^{18}$).

Выходные данные

В выходной файл OUTPUT.TXT выведите ответ на задачу.

Примеры

№	INPUT.TXT	OUTPUT.TXT
1	4 7	4
2	5 13	10

Пояснение

Чтобы лексикографически сравнить два числа, нужно найти в них первое несовпадение цифр при просмотре слева направо. Меньшим будет то число, у которого несовпадающая цифра меньше, либо то, которое является началом другого, но не совпадает с ним.

Система оценки

Решения, работающие только для $B - A \le 10^6$, будут оцениваться в 40 баллов.

Решения, работающие только для $A, B \le 10^9$, будут оцениваться в 60 баллов.

Задача D. Лексикографически минимальное число

Частичное решение: 40 баллов

Самый простой поиск искомого числа сводится к перебору всех значений X отрезка [A, B], перевода каждого значения в строковый эквивалент и выбор наименьшего среди них. Псевдокод такого решения может выглядеть так:

```
read(A,B)
M = Str(A)
for X = A..B
  M = min(M, Str(X))
write(M)
```

Очевидно, что такое решение работает недостаточно быстро при больших значениях B-A.

Полное решение:

Заметим, что если A и B состоят из одинакового количества цифр, то лексикографическое сравнение чисел всего диапазона эквивалентно обычному сравнению. Поэтому в данном случае ответом будет служить число A.

Когда в числе B цифр больше, чем в числе A, то в большинстве случаев ответом будет число вида «1000…000», которое начинается с единицы, а далее идут нули, количество которых соответствует количеству цифр в числе A. Это всегда так, кроме случая, когда само число A такого же вида, то есть начинается с единицы, а далее идут одни нули. Очевидно, в этом случае ответом будет служить само число A.

Алгоритмическая реализация вышеописанного:

```
string a,b,res
read(a,b)
res = '1'+'0'*len(a)
if(len(a)==len(b) or a<res) res = a
print(res)
```

Задача D. Лексикографически минимальное число

```cpp
//GNU C++11
#include <bits/stdc++.h>

using namespace std;

int main(){
  string a,b;
  cin >> a >> b;
  if(a.length() == b.length()) cout << a;
                          else cout << min(a, "1"+string(a.size(), '0'));
  return 0;
}
```

```
//PascalABC.NET 3.4
var x,y : int64;
    a,b,c : string;
begin
  read(x,y);
  a := x.ToString;
  b := y.ToString;
  c := '1'+'0'*length(a);
  if length(a)=length(b) then write(a)
                         else write(a<c?a:c)
end.
```

```
#Python 3
a,b = input().split()
res = '1'+'0'*len(a)
if len(a)==len(b) or a<res:
    res = a
print(res)
```

Задача E. Полка с книгами

(Время: 2 сек. Память: 32 Мб Баллы: 100)

Мальчик Вася очень любит читать, и у него дома много книг, которые он хранит на большой книжной полке. Вася любит порядок, поэтому он решил расположить свои книги по возрастанию количества страниц в них слева направо. В процессе такой сортировки он заметил, что все книги имеют разное число страниц, по которому любая Васина книга определяется однозначно.

Однажды летом Вася уехал в деревню к бабушке, а его родители заинтересовались увлечением сына и решили познакомиться с его коллекцией. Сначала папа взял несколько подряд стоящих книг слева, а затем мама забрала все оставшиеся книги, которые изначально стояли справа.

Перед возвращением Васи домой родители вернули книги назад на полку: папа поставил все книги, которые брал, слева, а мама аналогичным образом поставила остальные книги справа. Однако родители не заметили того, что книги были упорядочены, и поставили их произвольным образом.

Вернувшись домой, Вася заметил, что порядок книг нарушен. Допросив маму, он узнал то, каким образом случилось такое перемешивание, и его заинтересовал вопрос: сколько книг мог брать папа, и сколько книг могла брать мама?

По итоговому расположению книг помогите Васе ответить на его вопрос. При этом достаточно сказать, сколько книг мог взять папа (этого достаточно, т.к. мама взяла все остальные – и это число легко вычислить). При этом известно, что каждый из родителей взял хотя бы одну книгу.

Входные данные

Первая строка входного файла INPUT.TXT содержит единственное число N – количество книг на полке у Васи. Во второй строке записаны N целых чисел A_i – информация о количестве страниц в каждой книге слева направо на тот момент, когда Вася вернулся домой. Гарантируется, что все значения A_i различны.

Ограничения: $2 \le N \le 3 \times 10^5$, $1 \le A_i \le 10^9$.

Выходные данные

В первой строке выходного файла OUTPUT.TXT выведите целое число K – количество возможных вариантов набора книг, которые мог взять папа. Во второй строке выведите в возрастающем порядке количество книг в каждом из этих наборов.

Примеры

№	INPUT.TXT	OUTPUT.TXT
1	3 1 2 3	2 1 2
2	5 2 1 5 9 6	2 2 3

Система оценки

Решения, работающие только при $N \le 1000$, будут оцениваться в 40 баллов.

Задача E. Полка с книгами

N – количество книг

A[1..N] – информация о книгах на тот момент, когда Вася вернулся домой

Отсортировав заданный массив мы сможем получить некоторый массив B[1..N] – начальную последовательность книг на полке (до отъезда Васи в деревню). Логично, что папа мог взять X книг только тогда, когда наборы (множества) A[1..X] и B[1..X] совпадают. Неэффективным решением такой проверки может быть сортировка A[1..X] и проверка на равенство с B[1..X], что потребует как минимум X операций, в итоге асимптотика решения задачи будет квадратичной – $O(N^2)$, что приведет лишь к частичному решению на 40 баллов. Перебирая и проверяя все значения X от 1 до N-1 можно заметить, что на X-м шаге мы имеем упорядоченный массив A[1..X-1] (за счет прошлого шага) и чтобы упорядочить A[1..X] достаточно элемент A[X] поставить в нужную позицию, сдвинув остальные вправо. Несмотря на то, что общая асимптотика решения не меняется, это позволяет получить 75 баллов.

Более эффективным будет следующее решение на 100 баллов с асимптотикой O(N×Log N). Здесь используется следующая идея: множества книг A[1..X] и B[1..X] совпадают, если max(A[1..X]) = B[X]. Действительно, это означает, что все элементы A[1..X] меньше либо равны B[X], а поскольку всего таких элементов в исходном массиве ровно X и они все входят в множество B[1..X] (т.к. массив B отсортирован и все элементы различны), то A[1..X] = B[1..X]. Элемент max(A[1..X]) на X-м шаге мы можем находить за O(1), так как max(A[1..X]) = max(A[1..X-1], A[X]).

Алгоритмическая реализация:

```
read(n,A)                          //читаем входные данные
B = sorted(A)                      //получаем исходный список книг

for i=2..n                         //записываем в новый A[i] значение max(A[1..i])
  A[i] = max(A[i-1],A[i])

res = []                           //вычисляем список возможных количеств взятых книг
for x=1..n-1
  if(A[x] = B[x]) res.add(x)

writeln(len(res))                  //выводим ответ
for i=1..len(res)
  write(res[i], ' ')
```

Задача E. Полка с книгами – O(N×Log N)

```cpp
//GNU C++11
#include <stdio.h>
#include <bits/stdc++.h>
#define mn 300005

using namespace std;

int i,n,a[mn],b[mn];
vector<int> res;

int main(){
  scanf("%d", &n);
  for(i=0; i<n; i++){
    scanf("%d", &a[i]);
    b[i] = a[i];
    if(i) a[i] = max(a[i-1], a[i]);
  }

  sort(b, b+n);

  for(i=0; i<n-1; i++)
    if(a[i]==b[i])
      res.push_back(i+1);

  printf("%d\n", res.size());
  for(int x : res)
    printf("%d ", x);

  return 0;
}
```

```pascal
//PascalABC.NET 3.4
begin
  var n := ReadInteger;
  var a := ReadArrInteger(n);
  var b := Arr(a);
  for var i:=1 to n-1 do
    a[i] := max(a[i-1], a[i]);
  sort(b);
  var c := Range(1,n-1).Where(x -> a[x-1]=b[x-1]);
  Println(c.Count);
  c.Print(' ')
end.
```

```python
#PyPy 3
n = int(input())
a = list(map(int, input().split()))
b = sorted(a)
for i in range(1, n):
    a[i] = max(a[i-1], a[i])
c = [i for i in range(1, n) if a[i-1]==b[i-1]]
print(len(c))
print(*c)
```

Задача E. Полка с книгами

N – количество книг
A[1..N] – информация о книгах на тот момент, когда Вася вернулся домой

Рассмотрим оптимальное решение этой задачи, работающее за O(N). Используем дополнительные массивы, вычисляемые за линейное время:

B[1..N] – массив, в котором в B[X] будем хранить max(A[1..X]), для X = 1..N
C[1..N] – массив, в котором в C[X] будем хранить min(A[X..N]), для X = 1..N

Понятно, что папа мог взять первые X книг, если все элементы префикса A[1..X] строго меньше всех элементов суффикса A[X+1..N]. Проверить данное утверждение можно условием B[X] < C[X+1].

Алгоритмическая реализация:

```
read(n,A)                          //читаем входные данные

B[1] = A[1]                        //вычисляем B
for i=2..n                         //записываем в B[i] значение max(A[1..i])
  B[i] = max(B[i-1],A[i])

C[n] = A[n]                        //вычисляем C
for i=n-1..1                       //записываем в C[i] значение min(A[i..n])
  C[i] = min(A[i], C[i+1])

res = []                           //вычисляем список возможных количеств взятых книг
for i=1..n-1
  if(B[i] < C[i+1]) res.add(i)

writeln(len(res))                  //выводим ответ
for i=1..len(res)
  write(res[i], ' ')
```

Задача E. Полка с книгами – O(N)

```cpp
//GNU C++11
#include <stdio.h>
#include <bits/stdc++.h>
#define mn 300005

using namespace std;

int i,n,a[mn],b[mn],c[mn];
vector<int> res;

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

  b[0] = a[0];
  for(i=1; i<n; i++)
    b[i] = max(b[i-1],a[i]);
  c[n-1] = a[n-1];
  for(i=n-2; i>=0; i--)
    c[i] = min(a[i],c[i+1]);

  for(i=0; i<n-1; i++)
    if(b[i]<c[i+1])
      res.push_back(i+1);
  printf("%d\n", res.size());
  for(int x : res)
    printf("%d ", x);

  return 0;
}
```

```pascal
//PascalABC.NET 3.4
begin
  var n := ReadInteger;
  var a := ReadArrInteger(n);
  var b := Arr(a);
  var c := Arr(a);
  for var i:=1 to n-1 do
    b[i] := max(b[i-1], a[i]);
  for var i:=n-2 downto 0 do
    c[i] := min(a[i], c[i+1]);
  var res := Range(1,n-1).Where(x -> b[x-1]<c[x]);
  Println(res.Count);
  res.Print(' ')
end.
```

```python
#PyPy 3
n = int(input())
a = list(map(int, input().split()))
b = a[:1] + [0]*(n-1)
for i in range(1,n):
    b[i] = max(b[i-1], a[i])
c = [0]*(n-1) + a[-1:]
for i in range(n-2,0,-1):
    c[i] = min(a[i], c[i+1])
res = [i for i in range(1, n) if b[i-1] < c[i]]
print(len(res))
print(*res)
```

Задача F. Миссия невыполнима

(Время: 1 сек. Память: 16 Мб Баллы: 100)

Вы играете в новую компьютерную игру «Миссия невыполнима 0xFF» и почти завершили ее прохождение. На последнем уровне вам осталось взломать сейф с электронным кодовым замком, представляющим собой сенсорную прямоугольную панель из N строк и M столбцов, пронумерованных по порядку числами от 1 до N и от 1 до M соответственно. Первоначально все N×M ячеек замка имеют определенный цвет. Известно, что различных цветов на панели не может быть более, чем min(N, M)-1, цвета нумеруются целыми числами от 1 до min(N, M)-1. Вам доступна функция перекрашивания ячеек панели: если выбрать в некоторой строке (некотором столбце) две клетки одного цвета, то вся эта строка (весь этот столбец) перекрасится в цвет выбранных клеток. Для открытия сейфа необходимо не более чем за 2×(N+M) операций перекрасить всю панель в какой-нибудь один цвет.

Напишите программу, которая по заданной раскраске панели будет выводить последовательность команд длиной не более 2×(N+M) операций, приводящих к открытию замка сейфа.

Входные данные

В первой строке входного файла INPUT.TXT записаны целые числа N и M ($3 \le N, M \le 100$). Затем в последующих N строках задается начальное состояние панели замка. В (i+1)-й строке записаны M целых чисел от 1 до min(N, M)-1 – номера цветов ячеек панели замка в i-й строке панели.

Выходные данные

В первой строке выходного файла OUTPUT.TXT выведите целое число K – количество операций перекрашивания. В следующих K строках опишите предлагаемые операции перекрашивания. Каждая строка описания должна содержать в начале символ «V», если соответствующая операция перекрашивает столбец, и «H», если строку. Затем в строке через пробел идут четыре целых числа x_1, y_1, x_2, y_2 – координаты двух выбранных клеток (здесь x_1 и x_2 – номера столбцов, а y_1 и y_2 – номера строк). Цвета в этих клетках на начало операции перекрашивания должны быть одинаковыми.

Примеры

№	INPUT.TXT	OUTPUT.TXT
1	3 3 1 1 2 2 2 1 2 2 2	3 V 1 2 1 3 V 2 2 2 3 V 3 1 3 3
2	4 5 1 2 3 2 1 3 2 1 3 2 1 1 1 1 1 2 2 2 2 2	6 H 1 1 5 1 V 1 1 1 3 V 2 1 2 3 V 3 1 3 3 V 4 1 4 3 V 5 1 5 3

Задача F. Миссия невыполнима

Покажем, что любой сейф можно взломать за M+2 действия: всегда можно найти две строки, которые возможно перекрасить в один и тот же цвет, а затем выполнить перекраску во всех столбцах, используя соответствующие ячейки перекрашенных строк.

Действительно, поскольку в каждой строке не более M-1 различных цветов, то всегда можно найти в ней две ячейки одного и того же цвета и перекрасить эту строку в данный цвет. Так мы могли бы перекрасить каждую из N строк в определенный цвет, но нам достаточно двух, цвета которых совпадают. Таковые найдутся из тех же соображений: различных цветов не более, чем N-1, и поэтому все не могут иметь различный цвет после такой перекраски.

Для поиска в i-й строке матрицы A[1..N][1..M] двух ячеек одного цвета можно использовать следующий алгоритм:

```
B[1..M-1] = {0…0}          //B[t] – номер столбца, где цвет ячейки = t
for j=1..M                 //перебор всех ячеек i-й строки (j-столбец)
  C = A[i][j]              //цвет ячейки в i-й строке и j-м столбце
  if(B[C]>0)               //если такой цвет в i-й строке уже был…
    (x1,x2) = (B[C], j)    //запоминаем номера столбцов одного цвета C
    exit                   //завершаем алгоритм
  B[C] = j                 //запоминаем номер столбца с цветом C
```

Полученную в i-й строке информацию удобно сохранить в словарь (ассоциативный массив) D, в котором ключом будет номер цвета C, а значением – кортеж или список (Y, X1, X2), где Y – номер строки, а X1, X2 – номера столбцов. Эта информация потребуется, когда на i-м шаге мы получим возможность окрасить i-ю строку в цвет C, который уже присутствует в словаре. После чего сразу же можно выполнить взлом сейфа: окрасить первую строку на основании значения D[C], а вторую – по текущей информации. Далее, останется перекрасить все столбцы, используя ячейки данных двух строк (с номерами D[C].Y и i).

Задача F. Миссия невыполнима

```cpp
//C++
#include <bits/stdc++.h>

using namespace std;

int main(){
  int n, m, i, j, k, x, c;
  map<int, tuple<int, int, int>> a;

  cin >> n >> m;

  for(i=1; i<=n; i++){
    vector<int> b(m);
    for(j=1; j<=m; j++){
      cin >> k;
      if(b[k]) c = k, x = j;
          else b[k] = j;
    }
    if(a.count(c)){
      cout << m+2 << endl;
      cout << "H " << get<1>(a[c]) << " " << get<0>(a[c])
                   << " " << get<2>(a[c]) << " " << get<0>(a[c]) << endl;
      cout << "H " << b[c] << " " << i << " " << x << " " << i << endl;
      for(k=1; k<=m; k++)
        cout << "V " << k << " " << get<0>(a[c]) << " " << k << " " << i << endl;
      return 0;
    }
    a[c] = make_tuple(i, b[c], x);
  }
}
```

Задача F. Миссия невыполнима

```
//PascalABC.NET 3.4
type int = integer;

var n, m, i, j, k, x, c : int;

begin
  read(n,m);

  var a := new Dictionary<int, (int, int, int)>;

  for i:=1 to n do begin
    var b := new Dictionary<int, int>;
    for j:=1 to m do begin
      read(k);
      if k in b then (c,x) := (k,j)
                else b[k] := j;
    end;
    if c in a then begin
      println(m+2);
      println('H', a[c][1], a[c][0], a[c][2], a[c][0]);
      println('H', b[c], i, x, i);
      for k:=1 to m do
        println('V', k, a[c][0], k, i);
      halt
    end;
    a[c] := (i, b[c], x)
  end
end.
```

Задача F. Миссия невыполнима

```python
#Python 3
n, m = map(int, input().split())
a = dict()
for i in range(1,n+1):
    b = [0]*m
    j = 0
    for k in map(int, input().split()):
        j += 1
        if b[k]:
            c, x = k, j
            break
        b[k] = j
    if c in a:
        print(m+2)
        print('H', a[c][1], a[c][0], a[c][2], a[c][0])
        print('H', b[c], i, x, i)
        for k in range(1, m+1):
            print('V', k, a[c][0], k, i);
        exit(0)
    a[c] = [i, b[c], x]
```