Codeforces Round #641 (Div. 2) 1350A Орак и делители

A. Орак и делители

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Орак изучает теорию чисел, и его очень заинтересовали свойства делимости.

Для двух натуральных чисел a и b, a является делителем b если и только если существует такое натуральное число c, что $a \cdot c = b$.

Для $n \geq 2$, обозначим за $f(n)$ минимальный делитель n, отличный от 1.

Например, $f(7) = 7, f(10) = 2, f(35) = 5$.

Для выбранного числа n Орак решил прибавить $f(n)$ к n.

Например, если у него было число $n = 5$, новое значение n будет равно 10. А если у него было число $n = 6$, n станет равным 8.

Ораку так это понравилось, что он решил проделать подобные операции по несколько раз.

Для двух положительных чисел n и k, Орак попросил вас прибавить $f(n)$ к n ровно k раз (обратите внимание, что n изменяется после каждой операции, соотвественно $f(n)$ тоже может измениться) и сказать ему итоговое значение n.

Например, если Орак скажет вам, что $n = 5$ и $k = 2$, сначала вы должны прибавить $f(5) = 5$ к $n = 5$, и новое значение n станет равным $n = 10$, после этого вы должны прибавить $f(10) = 2$ к 10, и новое (и итоговое!) значение n будет равно 12.

Орак может задавать вам эти вопросы несколько раз.

Входные данные

В первой строке записано одно целое число t $(1 \leq t \leq 100)$: количество запросов Орака.

В каждой из следующий t строк записаны два целых числа n, k $(2 \leq n \leq 10^6, 1 \leq k \leq 10^9)$, описывающие очередной запрос.

Гарантируется, что сумма всех n не превосходит 10^6.

Выходные данные

Выведите t строк, в i-й из которых должно быть записано итоговое значение n для i-го запроса Орака.

Пример

входные данные
3 5 1 8 2 3 4
выходные данные
10 12 12

Примечание

В первом запросе $n = 5$ и $k = 1$. Делители 5 это 1 и 5, и минимальный делитель, отличный от 1 это 5. Соответственно, единственная операция это прибавление $f(5) = 5$ к 5, поэтому результат равен 10.

Во втором запросе $n = 8$ и $k = 2$. Делители 8 это $1, 2, 4, 8$, и минимальный из них кроме 1 равен 2, затем, после одной операции 8 превратится в $8 + (f(8) = 2) = 10$. Делители 10 это $1, 2, 5, 10$, минимальный делитель, отличный от 1 это 2, поэтому ответ равен $10 + (f(10) = 2) = 12$.

В третьем запросе n изменялось следующим образом: $3 \rightarrow 6 \rightarrow 8 \rightarrow 10 \rightarrow 12$.

1350A - Orac and Factors

If we simulate the whole process we will get TLE because k is too large. So we need some trivial observations:

- If n is even, then for each operation n will be added by 2 and keep being even.
- If n is odd, then for the first time n will be added by an odd number and then become even.

So it's easy to see that the answer is

$$\begin{cases} n+2k & n \text{ is even} \\ n+2(k-1)+d(n) & n \text{ is odd} \end{cases}$$

where $d(n)$ is the smallest positive factor of x except 1, which can be calculated in $O(n)$ time.

The overall complexity is $O(n)$.

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <algorithm>

using namespace std;

int main()
{
 int T;
 cin >> T;
 while(T--)
 {
  int n,k;
  cin >> n >> k;
  if(n%2==0)
  {
   cout << n+2*k << endl;
   continue;
  }
  int p = 0;
  for(int i = n; i>=2; i--)
   if(n%i==0)
       p = i;
  cout << n+p+2*(k-1) << endl;
 }
 return 0;
}
```

```cpp
#include <iostream>
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <algorithm>

using namespace std;

int main()
{
 int T;
 cin >> T;
 while(T--)
 {
  int n,k;
  cin >> n >> k;
  if(n%2==0)
  {
   cout << n+2*k << endl;
   continue;
  }
  int p = 0;
  for(int i = n; i>=2; i--)
   if(n%i==0)
       p = i;
  cout << n+p+2*(k-1) << endl;
 }
 return 0;
}
```

Codeforces Round #641 (Div. 2) 1350B Орак и модели

B. Орак и модели

ограничение по времени на тест: 3 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

В магазине есть n моделей, пронумерованных от 1 до n, размеры которых равны $s_1, s_2, \ldots, s_n$.

Орак купит некоторые из этих моделей и упорядочит их по возрастанию номеров (индексов, а не размеров).

Орак считает, что полученная расстановка **красивая**, если для любых двух соседних моделей с номерами i_j и i_{j+1} (обратите внимание, что $i_j < i_{j+1}$, так как Орак упорядочил их правильно), i_{j+1} делится на i_j и $s_{i_j} < s_{i_{j+1}}$.

Например, для 6 моделей с размерами $\{3, 6, 7, 7, 7, 7\}$, он может купить модели с индексами 1, 2, и 6, и полученная расстановка будет красивой. Обратите внимание, что расстановка из одной модели также считается красивой.

Орак хочет знать, какое наибольше число моделей он может купить, и он может задавать вам эти вопросы по несколько раз.

Входные данные

В первой строке записано одно целое число t $(1 \le t \le 100)$: количество запросов.

Каждый запрос состоит из двух строк, в первой из которых записано одно целое число n $(1 \le n \le 100\,000)$: количество моделей в магазине, а во второй записаны n целых чисел $s_1, \ldots, s_n$ $(1 \le s_i \le 10^9)$: размеры моделей.

Гарантируется, что сумма величин n не превосходит $100\,000$.

Выходные данные

Выведите t строк, в i-й из которых должно быть записано максимальное число моделей, которое Орак может купить для i-го запроса.

Пример

входные данные
4 4 5 3 4 6 7 1 4 2 3 6 4 9 5 5 4 3 2 1 1 9

выходные данные
2 3 1 1

Примечание

Для первого запроса, например, Орак может купить модели с индексами 2 и 4, расстановка которых будет красивой так как 4 делится на 2 и 6 больше, чем 3. Рассмотрев остальные варианты, можно легко убедиться, что нет красивой расстановки с более, чем тремя моделями.

Во втором запросе Орак может купить модели с индексами 1, 3, и 6. Рассмотрев остальные варианты, можно легко убедиться, что нет красивой расстановки с более, чем тремя моделями.

В третьем примере не существует красивой расстановки с более, чем одной моделью.

1350B - Orac and Models

Considering DP, we can design DP statuses as follow: f_i stands for the length of the longest beautiful sequence end up with index i.

We can find the transformation easily:

$$f_i = \max_{j|i,s_j<s_i} \{f_j + 1\}$$

Then, the length of answer sequence is the maximum value among $f_1, f_2, \cdots, f_n$.

About the complexity of DP: If you transform by iterating multiples, it will be $O(n \log n)$ (According to properties of Harmonic Series); if you iterate divisors, then it will be $O(n\sqrt{n})$. Fortunately, both of them are acceptable in this problem.

```
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <algorithm>

using namespace std;
const int MAXN = 500005;
inline int readint()
{
 int res = 0;
 char c = 0;
 while(!isdigit(c))
  c = getchar();
 while(isdigit(c))
  res = res*10+c-'0', c = getchar();
 return res;
}
int n,a[MAXN],f[MAXN];

int main()
{
 int T = readint();
 while(T--)
 {
  n = readint();
  for(int i = 1; i<=n; i++)
   a[i] = readint();
  for(int i = 1; i<=n; i++)
   f[i] = 1;
  for(int i = 1; i<=n; i++)
   for(int j = i*2; j<=n; j += i)
    if(a[i]<a[j])
     f[j] = max(f[j],f[i]+1);
  int ans = 0;
  for(int i = 1; i<=n; i++)
   ans = max(ans,f[i]);
  cout << ans << endl;
 }
 return 0;
}
```

```cpp
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <algorithm>

using namespace std;
const int MAXN = 500005;
inline int readint()
{
 int res = 0;
 char c = 0;
 while(!isdigit(c))
  c = getchar();
 while(isdigit(c))
  res = res*10+c-'0', c = getchar();
 return res;
}
int n,a[MAXN],f[MAXN];

int main()
{
 int T = readint();
 while(T--)
 {
  n = readint();
  for(int i = 1; i<=n; i++)
   a[i] = readint();
  for(int i = 1; i<=n; i++)
   f[i] = 1;
  for(int i = 1; i<=n; i++)
   for(int j = i*2; j<=n; j += i)
    if(a[i]<a[j])
     f[j] = max(f[j],f[i]+1);
  int ans = 0;
  for(int i = 1; i<=n; i++)
   ans = max(ans,f[i]);
  cout << ans << endl;
 }
 return 0;
}
```

Educational Codeforces Round 87 (рейтинговый для Див. 2)
1354A Будильник

A. Будильник

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Поликарп провел целый день за подготовкой задач для вас. Теперь ему нужно поспать хотя бы a минут, чтобы отдохнуть.

Поликарп может проснуться только от звука будильника. Он только заснул, а его первый будильник прозвонит ровно через b минут.

Каждый раз когда Поликарп просыпается, он решает, заводить ли будильник еще раз или нет. Если он проспал меньше a минут суммарно, то он заводит будильник так, чтобы тот прозвонил ровно через c минут, и тратит d минут, чтобы снова заснуть. Иначе же он встает с кровати, начиная новый день.

Если будильник звонит, пока Поликарп все еще засыпает, то он снова его заводит на c минут и опять тратит d минут, чтобы заснуть.

Вам же всего лишь предстоит выяснить, когда Поликарп встанет с кровати или сообщить, что это не случится никогда.

Пожалуйста, прочитайте пояснение, чтобы получше понять пример.

Входные данные

В первой строке записано одно целое число t ($1 \le t \le 1000$) — количество наборов входных данных.

В единственной строке каждого набора входных данных записаны четыре целых числа a, b, c, d ($1 \le a, b, c, d \le 10^9$) — время, которое необходимо Поликарпу, чтобы выспаться, время до первого звонка будильника, время до каждого последующего звонка будильника и время, необходимое Поликарпу, чтобы заснуть.

Выходные данные

На каждый набор входных данных выведите одно целое число. Если Поликарп никогда не встанет с кровати, то выведите `-1`. Иначе выведите время, которое Поликарп потратит до того, как встать с кровати.

Пример

входные данные
7 10 3 6 4 11 3 6 4 5 9 4 10 6 5 2 3 1 1 1 1 3947465 47342 338129 123123 234123843 13 361451236 361451000

выходные данные
27 27 9 -1 1 6471793 358578060125049

Примечание

В первом наборе входных данных Поликарп просыпается через 3 минуты. Он отдохнул только 3 минуты из необходимых 10. Поэтому он снова заводит будильник, чтобы тот прозвенел через 6 минут, и тратит 4 минуты, чтобы уснуть. То есть он отдыхает в течение еще 2 минут, что суммируется в $3 + 2 = 5$ минут сна. Он повторяет эту процедуру еще три раза, и получается 11 минут сна. Наконец, он встает с кровати. Он потратил 3 минуты до первого будильника, в потом заново завел будильник четыре раза. Ответ равен $3 + 4 \cdot 6 = 27$.

Второй набор входных данных похож на первый, но Поликарпу надо 11 минут, чтобы выспаться, вместо 10. Однако, это ничего не меняет, потому что Поликарп получает 11 минут сна при таких настройках будильника так и так.

В третьем наборе входных данных Поликарп просыпается достаточно отдохнувшим уже после первого будильника. Поэтому ответ равен $b = 9$.

В четвертом наборе входных данных Поликарп просыпается через 5 минут. К сожалению, дальше он не может отдохнуть ни минуты, заводя будильник все снова и снова бесконечно :(

1354A - Alarm Clock

Let's handle some cases. Firstly, if $b \geq a$ then Polycarp wakes up rested enough immediately, so b is the answer.

Otherwise, what does Polycarp do? He sets alarm to go off in c minutes and falls asleep in d minutes. Thus, he spends $c - d$ minutes sleeping. Notice that if $c - d$ is non-positive, then Polycarp always resets his alarm without sleeping. So for that case the answer is `-1`.

Finally, if Polycarp resets his alarm x times then he ends up with $b + x \cdot (c - d)$ minutes of sleep in total and ends up spending $b + x \cdot c$ minutes of time. We know that $b + x \cdot (c - d)$ should be greater or equal to a and x should be the smallest possible.

$$b + x \cdot (c - d) \geq a \leftrightarrow x \cdot (c - d) \geq a - b \leftrightarrow x \geq \frac{a - b}{c - d}$$

Thus, the smallest possible integer x is equal to $\lceil \frac{a-b}{c-d} \rceil$. And the answer is $b + x \cdot c$.

Overall complexity: $O(1)$ per testcase.

```
t = int(input())
for _ in range(t):
 a, b, c, d = map(int, input().split())
 if b >= a:
  print(b)
  continue
 if c <= d:
  print(-1)
  continue
 a -= b
 dif = c - d
 print(b + (a + dif - 1) // dif * c)
```

Educational Codeforces Round 87 (рейтинговый для Див. 2)
1354B Троичная строка

B. Троичная строка

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Вам дана строка s, каждый символ которой — 1, 2 или 3. Вы должны выбрать кратчайшую непрерывную подстроку s, в которой каждый из трех символов встречается хотя бы один раз.

Непрерывная подстрока s — строка, которую можно получить из s удалением какого-то количества (возможно, ни одного) символов из начала строки s и какого-то количества (возможно, ни одного) символов из конца строки s.

Входные данные

В первой строке задано одно целое число t ($1 \le t \le 20000$) — количество наборов входных данных.

Каждый набор входных данных состоит из одной строки s ($1 \le |s| \le 200000$). Гарантируется, что каждый символ s — 1, 2 или 3.

Сумма длин всех строк не превосходит 200000.

Выходные данные

Для каждого набора тестовых данных выведите одно число — длину кратчайшей подстроки s, содержащей символы всех трех типов. Если такой подстроки нет, выведите 0.

Пример

входные данные
7 123 12222133333332 112233 332211 12121212 333333 31121

выходные данные
3 3 4 4 0 0 4

Примечание

Рассмотрим пример из условия:

В первом наборе входных данных можно использовать подстроку 123.

Во втором наборе входных данных можно использовать подстроку 213.

В третьем наборе входных данных можно использовать подстроку 1223.

В четвертом наборе входных данных можно использовать подстроку 3221.

В пятом наборе входных данных в s нету символа 3.

В шестом наборе входных данных в s нету символа 1.

В седьмом наборе входных данных можно использовать подстроку 3112.

1354B - Ternary String

There are multiple solutions involving advanced methods such as binary search or two pointers, but I'll try to describe a simpler one.

The main idea of my solution is that the answer should look like `abb...bbbbbc`: one character of type a, a block of characters of type b, and one character of type c. If we find all blocks of consecutive equal characters in our string, each candidate for the answer can be obtained by expanding a block to the left and to the right by exactly one character. So the total length of all candidates is $O(n)$, and we can check them all.

Why does the answer look like `abb...bbbbbc`? If the first character of the substring appears somewhere else in it, it can be deleted. The same applies for the last character. So, the first and the last characters should be different, and should not appear anywhere else within the string. Since there are only three types of characters, the answer always looks like `abb...bbbbbc`.

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

using namespace std;

char buf[200043];

int main()
{
 int t;
 scanf("%d", &t);
 for(int i = 0; i < t; i++)
 {
  scanf("%s", buf);
  string s = buf;
  int ans = int(1e9);
  int n = s.size();
  vector<pair<char, int> > c;
  for(auto x : s)
  {
   if(c.empty() || c.back().first != x)
    c.push_back(make_pair(x, 1));
   else
    c.back().second++;
  }
  int m = c.size();
  for(int i = 1; i < m - 1; i++)
   if(c[i - 1].first != c[i + 1].first)
    ans = min(ans, c[i].second + 2);
  if(ans > n)
   ans = 0;
  printf("%d\n", ans);
 }
}
```

Codeforces Round #643 (Div. 2) 1355A Последовательность с цифрами

A. Последовательность с цифрами

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Определим рекуррентную последовательность следующим образом:

$$a_{n+1} = a_n + minDigit(a_n) \cdot maxDigit(a_n).$$

Здесь $minDigit(x)$ и $maxDigit(x)$ — минимальная и максимальная цифры в десятичной записи числа x без ведущих нулей соответственно. Для примеров обратитесь к примечаниям.

Ваша задача — по заданным a_1 и K вычислить a_K.

Входные данные

В первой строке записано единственное число t ($1 \le t \le 1000$) — количество независимых наборов входных данных.

Каждый набор входных данных состоит из двух целых чисел a_1 и K ($1 \le a_1 \le 10^{18}$, $1 \le K \le 10^{16}$), записанных через пробел на отдельной строке.

Выходные данные

Для каждого набора входных данных выведите одно число a_K на отдельной строке.

Пример

входные данные
8 1 4 487 1 487 2 487 3 487 4 487 5 487 6 487 7

выходные данные
42 487 519 528 544 564 588 628

Примечание

$a_1 = 487$

$a_2 = a_1 + minDigit(a_1) \cdot maxDigit(a_1) = 487 + \min(4, 8, 7) \cdot \max(4, 8, 7) = 487 + 4 \cdot 8 = 519$

$a_3 = a_2 + minDigit(a_2) \cdot maxDigit(a_2) = 519 + \min(5, 1, 9) \cdot \max(5, 1, 9) = 519 + 1 \cdot 9 = 528$

$a_4 = a_3 + minDigit(a_3) \cdot maxDigit(a_3) = 528 + \min(5, 2, 8) \cdot \max(5, 2, 8) = 528 + 2 \cdot 8 = 544$

$a_5 = a_4 + minDigit(a_4) \cdot maxDigit(a_4) = 544 + \min(5, 4, 4) \cdot \max(5, 4, 4) = 544 + 4 \cdot 5 = 564$

$a_6 = a_5 + minDigit(a_5) \cdot maxDigit(a_5) = 564 + \min(5, 6, 4) \cdot \max(5, 6, 4) = 564 + 4 \cdot 6 = 588$

$a_7 = a_6 + minDigit(a_6) \cdot maxDigit(a_6) = 588 + \min(5, 8, 8) \cdot \max(5, 8, 8) = 588 + 5 \cdot 8 = 628$

1355A - Sequence with Digits

Let's calculate the sequence for fixed $a_1 = 1$: $1, 2, 6, 42, 50, 50, 50, \ldots$

We got lucky and the minimal digit has become 0, after that the element has stopped changing because we always add 0.

Actually it is not luck and that will always happen. Note that we add no more than $9 \cdot 9 = 81$ every time, so the difference between two consecutive elements of the sequence is bounded by 81. Assume that we will never have minimal digit equal to 0. Then the sequence will go to infinity. Let's take $X = 1000(\lfloor \frac{a_1}{1000} \rfloor + 1)$. All the numbers on segment $[X; X + 99]$ have 0 in hundreds digit, so none of them can be element of our sequence. But our sequence should have numbers greater than X. Let's take the smallest of them, it should be at least $X + 100$. But then the previous number in the sequence is at least $(X + 100) - 81 = X + 19$. It is greater than X but smaller than the minimal of such numbers. Contradiction.

In the previous paragraph we have actually shown that we have no numbers greater than $X + 100$ in our sequence and we will see the number with 0 among first 1001 elements.

That means that we can build the sequence till we find the first number with 0 and then it will repeat forever.

In reality the maximal index of the first elements with 0 is 54 and minimal a_1 for that to happen is 28217.

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <unordered_set>
#include <unordered_map>
#include <queue>
#include <ctime>
#include <cassert>
#include <complex>
#include <string>
#include <cstring>
#include <chrono>
#include <random>
#include <bitset>
using namespace std;

#ifdef LOCAL
 #define eprintf(...) fprintf(stderr, __VA_ARGS__);fflush(stderr);
#else
 #define eprintf(...) 42
#endif

using ll = long long;
using ld = long double;
using uint = unsigned int;
using ull = unsigned long long;
template<typename T>
using pair2 = pair<T, T>;
using pii = pair<int, int>;
using pli = pair<ll, int>;
using pll = pair<ll, ll>;
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());

#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second

double startTime;
double getCurrentTime() {
 return ((double)clock() - startTime) / CLOCKS_PER_SEC;
```

```
return ((double)clock() - startTime) / CLOCKS_PER_SEC;
}

ll getAdd(ll x) {
 ll m1 = 10, m2 = 0;
 while(x > 0) {
  ll y = x % 10;
  x /= 10;
  m1 = min(m1, y);
  m2 = max(m2, y);
 }
 return m1 * m2;
}

int main()
{
 startTime = (double)clock();
// freopen("input.txt", "r", stdin);
// freopen("output.txt", "w", stdout);

 int t;
 scanf("%d", &t);
 while(t--) {
  ll x, k;
  scanf("%lld%lld", &x, &k);
  k--;
  while(k--) {
   ll y = getAdd(x);
   if (y == 0) break;
   x += y;
  }
  printf("%lld\n", x);
 }

 return 0;
}
```

Codeforces Round #643 (Div. 2) 1355B Юные следопыты

B. Юные следопыты

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Отряд юных следопытов отправился в учебную экспедицию навстречу своим первым приключениям. И возглавляет их старший следопыт Рассел. Вот герои зашли в лес, разбили лагерь и дальше решили разделиться на группы, чтобы исследовать как можно больше интересных мест. Рассел должен был выбрать состав групп, но столкнулся с одной проблемой...

Многие юные следопыты неопытны, и отправлять их маленькими группами — не всегда хорошая идея. Даже сам Рассел недавно стал старшим следопытом и нечасто бывал в экспедициях. Каждый следопыт характеризуется своей неопытностью — целым положительным числом e_i. Рассел решил, что юный следопыт с неопытностью e может идти лишь в группе, количество следопытов в которой не меньше e.

Теперь задача Рассела — определить, какое наибольшее число групп следопытов он сможет организовать. При этом может получиться, что некоторые следопыты не войдут в состав ни одной группы, это не страшно, ведь и в лагере для них найдется работа. Рассел очень переживает за успех экспедиции, и потому попросил вас помочь ему.

Входные данные

В первой строке записано число T ($1 \le T \le 2 \cdot 10^5$) — количество независимых тестовых случаев. В следующих $2T$ строках следует описание тестовых случаев.

В первой строке описания каждого теста задано целое число юных следопытов N ($1 \le N \le 2 \cdot 10^5$).

В следующей строке записаны N целых чисел $e_1, e_2, \ldots, e_N$ ($1 \le e_i \le N$), где e_i — неопытность i-го следопыта.

Гарантируется, что сумма N по всем тестовым случаям не превосходит $3 \cdot 10^5$.

Выходные данные

Выведите T чисел, каждое на отдельной строке.

В i-й строке выведите наибольшее число групп, которое можно организовать в i-м тестовом случае.

Пример

входные данные
2 3 1 1 1 5 2 3 1 2 2
выходные данные
3 2

Примечание

В первом примере можно сформировать три группы, в каждой из которых будет один следопыт. Это возможно, так как неопытность всех трех следопытов равна 1, что не меньше, чем размер их групп.

Во втором примере можно сформировать две группы. В первой группе окажутся следопыты с неопытностью 1, 2 и 3, а во второй группе — два следопыта с неопытностью 2.

Этот способ — не единственный возможный. Можно, например, сформировать одну группу из трех следопытов с неопытностью 2, а также еще одну группу, в которой будет всего один следопыт с неопытностью 1. При таком разбиении на группы следопыт с неопытностью 3 не войдет в состав ни одной группы.

1355B - Young Explorers

Let's sort all the explorers by non-decreasing inexperience. Suppose we have formed some group, how can we check is this group is valid? Inexperience of all the explorers in the group should be not greater than the group size. But we have sorted all the explorers, so the last explorer from the group has the largest inexperience. Therefore, to check the group for validity it is

necessary and sufficient to check that inexperience of the last explorer is not greater than the group size.

We can notice that we don't even look at all the explorers except the last one, the only important thing is their number. In fact, we can organize the creation of groups in this way: first choose the explorers that will be the last in their groups, then assign sufficient number of other explorers to corresponding groups. It is not profitable to assign more explorers than needed for this particular last explorer, because we can always leave them at the camp.

So how should we choose the last explorers? We want to make more groups, so the groups themselves should me smaller... It is tempting to use the following greedy algorithm: let's greedily pick the leftmost (which means with the smallest necessary group size) explorer such that they have enough explorers to the left of them to create a valid group. The idea is that we spend the smallest number of explorers and leave the most potential last explorers in the future. Let's strictly prove this greedy:

The solution is defined by positions of the last explorers in their corresponding groups $1 \le p_1 < p_2 < \ldots < p_k \le n$. Notice that the solution is valid if and only if $e_{p_1} + e_{p_2} + \ldots + e_{p_i} \le p_i$ for all $1 \le i \le k$ (we always have enough explorers to form first i groups).

Let $1 \le p_1 < p_2 < \ldots < p_k \le n$ be the greedy solution and $1 \le q_1 < q_2 < \ldots < q_m \le n$ be the optimal solution such that it has the largest common prefix with greedy one among all optimal solutions. Let t be the position of first difference in these solutions. $t \le k$ since otherwise the greedy algorithm couldn't add one more group but it was possible. $p_t < q_t$ since otherwise the greedy algorithm would take q_t instead of p_t. Since the explorers are sorted we have $e_{p_t} \le e_{q_t}$. But then $1 \le q_1 < q_2 < \ldots < q_{t-1} < p_t < q_{t+1} < \ldots < q_m \le n$ is a valid optimal solution and it has strictly larger common prefix with the greedy one which contradicts the choosing of our optimal solution.

To implement this solution it is enough to sort the explorers by the non-decreasing inexperience, then go from left to right and maintain the number of unused explorers. As soon as we encounter the possibility to create a new group, we do it.

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

using namespace std;

int main() {
    int t;
    cin >> t;

    while (t--) {
        int n;
        cin >> n;
        vector <int> a(n);
        for (int i = 0; i < n; i++) {
            cin >> a[i];
        }
        sort(a.begin(), a.end());
        int ans = 0, cur = 0;
        for (int i = 0; i < n; i++) {
            if (++cur == a[i]) {
                ans++;
                cur = 0;
            }
        }
        cout << ans << '\n';
    }
    return 0;
}
```

Codeforces Round #643 (Div. 2) 1355C Сосчитайте треугольники

C. Сосчитайте треугольники

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Как и у любого неизвестного математика, у Юрия есть любимые числа: A, B, C и D, причем $A \le B \le C \le D$. Также Юрий очень любит треугольники, поэтому в один день он задумался: сколько существует невырожденных треугольников с целочисленными длинами сторон x, y и z таких, что $A \le x \le B \le y \le C \le z \le D$?

Сейчас Юрий очень занят подготовкой задач для очередной олимпиады, поэтому он просит вас помочь посчитать количество интересующих его треугольников.

Треугольник называется невырожденным тогда и только тогда, когда его вершины не лежат на одной прямой.

Входные данные

В единственной строке через пробел записаны четыре целых числа A, B, C и D ($1 \le A \le B \le C \le D \le 5 \cdot 10^5$) — любимые числа Юрия.

Выходные данные

Выведите одно число — количество невырожденных треугольников с целочисленными длинами сторон x, y и z, для которых выполнено неравенство: $A \le x \le B \le y \le C \le z \le D$.

Примеры

входные данные
1 2 3 4
выходные данные
4

входные данные
1 2 2 5
выходные данные
3

входные данные
500000 500000 500000 500000
выходные данные
1

Примечание

В первом примере можно составить треугольники со следующими длинами сторон: $(1, 3, 3)$, $(2, 2, 3)$, $(2, 3, 3)$ и $(2, 3, 4)$.

Во втором примере можно составить треугольники: $(1, 2, 2)$, $(2, 2, 2)$ и $(2, 2, 3)$.

В третьем примере можно составить лишь один равносторонний треугольник. Длины всех сторон будут равны $5 \cdot 10^5$.

1355C - Count Triangles

Since $x \le y \le z$ to be a non-degenerate triangle for given triple it is necessary and sufficient to satisfy $z < x + y$. Let's calculate for all $s = x + y$ how many ways there are to choose (x, y). To do that we will try all x and add 1 on segment $[x + B; x + C]$ offline using prefix sums. Let's calculate prefix sums once more, now we can find in $O(1)$ how many ways there are to choose (x, y) such that their sum if greater than z. Try all z, calculate the answer. Total complexity — $O(C)$.

```
// #pragma comment(linker, "/stack:200000000")
// #pragma GCC optimize("Ofast,no-stack-protector")
// #pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,mmx,avx,avx2,tune=native")
```

```
// #pragma GCC optimize("unroll-loops")

#include <bits/stdc++.h>

#ifdef PERVEEVM_LOCAL
    #define debug(x) std::cerr << (#x) << ":\t" << (x) << std::endl
#else
    #define debug(x) 238;
#endif

#define fastIO std::ios_base::sync_with_stdio(false); std::cin.tie(0); std::cout.tie(0)
#define NAME "File"

using ll = long long;
using ld = long double;

#ifdef PERVEEVM_LOCAL
    std::mt19937 rnd(238);
#else
    std::mt19937 rnd(std::chrono::high_resolution_clock::now().time_since_epoch().count());
#endif

const double PI = atan2(0.0, -1.0);
const int INF = 0x3f3f3f3f;
const ll LINF = (ll)2e18;

ll calcProgression(ll a, ll d, ll n) {
    return (2 * a + d * (n - 1)) * n / 2;
}

void run() {
    int a, b, c, d;
    scanf("%d%d%d%d", &a, &b, &c, &d);

    ll ans = 0;
    for (int z = c; z <= d; ++z) {
        int minX = std::max(a, z - c + 1);
        if (minX > b) {
            continue;
        }

        int mid = z - b + 1;
        int start = c - std::max(b, z - minX + 1) + 1;

        if (mid <= minX) {
            ans += 1ll * (c - b + 1) * (b - minX + 1);
        } else if (mid > b) {
            // ans += 1ll * (b - minX + 1) * (b - minX + 2) / 2;
            ans += calcProgression(start, 1, b - minX + 1);
        } else {
            ans += calcProgression(start, 1, mid - minX + 1);
            ans += 1ll * (b - mid) * (c - b + 1);
            // ans += 1ll * (mid - minX + 1) * (mid - minX + 2) / 2 + 1ll * (b - mid) * (c - b + 1);
        }
    }

    printf("%lld\n", ans);
}

int main(void) {
    // freopen(NAME".in", "r", stdin);
    // freopen(NAME".out", "w", stdout);
```

```
    // freopen(NAME".out", "w", stdout);

    auto start = std::chrono::high_resolution_clock::now();
    run();
    auto end = std::chrono::high_resolution_clock::now();

    #ifdef PERVEEVM_LOCAL
        std::cerr << "Execution time: "
                  << std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count()
                  << " ms" << std::endl;
    #endif

    return 0;
}

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <unordered_set>
#include <unordered_map>
#include <queue>
#include <ctime>
#include <cassert>
#include <complex>
#include <string>
#include <cstring>
#include <chrono>
#include <random>
#include <bitset>
using namespace std;

#ifdef LOCAL
 #define eprintf(...) fprintf(stderr, __VA_ARGS__);fflush(stderr);
#else
 #define eprintf(...) 42
#endif

using ll = long long;
using ld = long double;
using uint = unsigned int;
using ull = unsigned long long;
template<typename T>
using pair2 = pair<T, T>;
using pii = pair<int, int>;
using pli = pair<ll, int>;
using pll = pair<ll, ll>;
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());

#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second

double startTime;
double getCurrentTime() {
 return ((double)clock() - startTime) / CLOCKS_PER_SEC;
```

```
}

const int N = (int)1e6 + 77;
int A, B, C, D;
ll a[N];

int main()
{
 startTime = (double)clock();
// freopen("input.txt", "r", stdin);
// freopen("output.txt", "w", stdout);

 scanf("%d%d%d%d", &A, &B, &C, &D);
 for (int i = A; i <= B; i++) {
  a[i + B]++;
  a[i + C + 1]--;
 }
 for (int i = 1; i < N; i++)
  a[i] += a[i - 1];
 for (int i = 1; i < N; i++)
  a[i] += a[i - 1];
 ll ans = 0;
 for (int i = C; i <= D; i++)
  ans += a[N - 1] - a[i];
 printf("%lld\n", ans);

 return 0;
}
```

Codeforces Round #645 (Div. 2) 1358A Освещение парка

A. Освещение парка

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

В связи с эпидемией коронавируса власти города обязали жителей соблюдать социальную дистанцию. Мэр города Семён Сергеевич хочет осветить парк Глухарники, чтобы люди даже ночью могли видеть друг друга и соблюдали дистанцию.

Парк представляет из себя прямоугольную таблицу состоящую из n строк и m столбцов, где клетки таблицы — площади, а границы между клетками — улицы. Также улицами являются внешние границы. Каждая улица имеет длину 1. Например, у парка размера $n = m = 2$ всего 12 улиц.

Вам поручили разработать план освещения парка. Вы можете ставить фонари в серединах улиц. Фонарь освещает две площади, между которыми он стоит (или только одну площадь, если он стоит на границе парка).

Пример парка размеров: $n = 4$, $m = 5$. Освещенные площади отмечены жёлтым цветом. Обратите внимание, что все улицы имеют длины 1. Фонари ставятся в середины улиц. На картинке **не все** площади освещены.

Семён Сергеевич хочет потратить на освещение наименьшее возможное количество денег, но также хочет чтобы люди по всему парку держали социальную дистанцию. Поэтому он просит вас узнать, какое минимальное количество фонарей понадобится, чтобы осветить все площади.

Входные данные

Первая строка содержит одно целое число t ($1 \le t \le 10^4$) — количество наборов входных данных. Далее следуют t наборов входных данных.

Каждый набор входных данных записывается одной строкой, содержащей два натуральных числа n и m ($1 \le n, m \le 10^4$) — размеры парка.

Выходные данные

Выведите t ответов на наборы тестовых данных. Каждый ответ должен содержать одно целое число — минимальное количество фонарей для того, чтобы осветить все площади.

Пример

входные данные
5 1 1 1 3 2 2 3 3 5 3
выходные данные
1 2 2 5 8

Примечание

Возможное оптимальное расположение фонарей для 2-го набора входных данных примера:

Возможное оптимальное расположение фонарей для 3-го набора входных данных примера:

1358A - Park Lighting

Note that if at least one of the sides is even, the square can be divided into pairs of neighbors and the answer is $\frac{nm}{2}$.

If both sides are odd, we can first light up a $(n-1) \times m$ part of the park. Then we'll still have the part $m \times 1$. We can light it up with $\frac{m+1}{2}$ lanterns. Then the total number of the lanterns is $\frac{(n-1) \cdot m}{2} + \frac{m+1}{2} = \frac{nm-m+m+1}{2} = \frac{nm+1}{2}$.

Note that both cases can be combined into one formula: $\lfloor \frac{nm+1}{2} \rfloor$.

The overall compexity is $\mathcal{O}(1)$ per test.

```
#include <iostream>

using namespace std;

int main() {
    int t, n, m;
    cin >> t;
    while (t--) {
        cin >> n >> m;
        cout << (n * m + 1) / 2 << '\n';
    }
}
```

Codeforces Round #645 (Div. 2) 1358B Марья Ивановна нарушает самоизоляцию

B. Марья Ивановна нарушает самоизоляцию

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Марье Ивановне, самой активной бабушке двора, надоело сидеть дома. Она решила организовать обряд против коронавируса.

У неё есть n подруг-бабушек (сама Марья Ивановна в это количество не входит). Бабушка с номером i готова прийти на обряд при условии, если в момент её появления во дворе кроме неё там будет как минимум a_i других бабушек. Обратите внимание, что бабушки могут приходить во двор одновременно. Формально, бабушка i готова прийти, если количество бабушек пришедших ранее или одновременно с ней больше или равно a_i.

Бабушки собираются во дворе так.

- Изначально во дворе находится только Марья Ивановна (то есть количество бабушек во дворе равно 1). Все остальные n бабушек пока сидят по домам.
- На очередном шаге Марья Ивановна выбирает подмножество бабушек, ни одна из которых ещё не вышла во двор. Каждой из них она обещает, что когда бабушка выйдет, во дворе будет не менее a_i других бабушек (включая Марью Ивановну). Марья Ивановна может звонить сразу нескольким соседкам. В таком случае выбранные бабушки выйдут во двор **одновременно**.
- Вы не можете обманывать бабушек, то есть ситуация, когда i-я бабушка, после выхода во двор обнаружит, что сейчас во дворе строго меньше a_i других бабушек (кроме неё самой, но включая Марью Ивановну), запрещена. Обратите внимание, что если несколько бабушек появились во дворе одновременно, то они каждая из них видит в том числе остальных в момент выхода.

Ваша задача — найти, какое максимальное количество бабушек (включая себя) Марья Ивановна может собрать во дворе для проведения обряда «обкуривания от Короны-Вируса». Ведь чем больше людей в одном месте во время карантина, тем эффективнее обряд!

Рассмотрим пример: если $n = 6$ и $a = [1, 5, 4, 5, 1, 9]$, то:

- на первом шаге Марья Ивановна может позвать бабушек с номерами 1 и 5, каждая из них будет видеть двух бабушек в момент выхода во двор (заметим, что $a_1 = 1 \leq 2$ и $a_5 = 1 \leq 2$);
- на втором шаге Марья Ивановна может позвать бабушек с номерами 2, 3 и 4, каждая из них будет видеть пять бабушек в момент выхода во двор (заметим, что $a_2 = 5 \leq 5$, $a_3 = 4 \leq 5$ и $a_4 = 5 \leq 5$);
- бабушку номер 6 позвать во двор не получится — следовательно, ответ на для такого примера равен 6 (сама Марья Ивановна и еще 5 суммарно вышедших бабушек).

Входные данные

В первой строке находится одно целое число t ($1 \leq t \leq 10^4$) — количество наборов входных данных в тесте. Далее следуют описания наборов входных данных.

Первая строка описания набора содержит одно целое число n ($1 \leq n \leq 10^5$) — количество бабушек (Марья Ивановна в это количество не входит).

Вторая строка содержит n целых чисел $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 2 \cdot 10^5$).

Гарантируется, что сумма значений n по всем наборам входных данных не превосходит 10^5.

Выходные данные

Для каждого набора входных данных выведите единственное целое число k ($1 \leq k \leq n + 1$) — максимальное возможное количество бабушек во дворе.

Пример

входные данные
4 5

```
1 1 2 2 1
6
2 3 4 5 6 7
6
1 5 4 5 1 9
5
1 2 3 5 6
```

выходные данные

```
6
1
6
4
```

Примечание

В первом наборе входных данных примера Марья Ивановна на первом шаге может позвать всех бабушек. Тогда каждая из них увидит пять бабушек, когда выйдет. Иными словами, ни одна не уйдёт домой. Поэтому во дворе будет Марья Ивановна и пять других бабушек.

Во втором наборе входных данных примера никто не сможет находиться во дворе, поэтому Марья Ивановна останется там одна.

Третий набор входных данных примера подробно разобран выше.

В четвёртом тестовом случае Марья Ивановна на первом шаге может позвать бабушек с номерами 1, 2 и 3. Если на втором шаге Марья Ивановна позовёт только 4-ю или только 5-ю, то когда эта бабушка выходила бы во двор, то она бы увидела четверых бабушек, то есть Марья Ивановна по отдельности позвать 4-ю и 5-ю не может. Если она позовёт сразу обеих — 4-ю и 5-ю, то когда они будут выходить, то увидят по $4 + 1 = 5$ бабушек. Несмотря на то, что 4-ю бабушку это устраивает, 5-ю — этот вариант не устраивает. Следовательно, позвать одновременно 4-ю и 5-ю Марья Ивановна тоже не может. То есть во дворе будет только Марья Ивановна и три бабушки из первого шага.

1358B - Maria Breaks the Self-isolation

Let x be the maximum number of grannies that can go out to the yard. Then if Maria Ivanovna calls them all at the same time, then everyone will see x grannies. Since x is the maximum answer, then each granny of them satisfy $a_i \le x$ (otherwise there's no way for these grannies to gather in the yard), that is, such call is correct. So it is always enough to call once.

Note that if you order grannies by a_i, Maria Ivanovna will have to call x first grannies from this list. She can take x grannies if $a_x \le x$ (otherwise, after all x grannies arrived, the last one will leave). To find x we can do a linear search.

The overall compexity is $\mathcal{O}(n \log n)$ per test.

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

void solve() {
    int n;
    cin >> n;
    vector<int> arr(n);
    for (int &el : arr)
        cin >> el;
    sort(arr.begin(), arr.end());
    for (int i = n - 1; i >= 0; i--) {
        if (arr[i] <= i + 1) {
            cout << i + 2 << '\n';
            return;
        }
    }
    cout << 1 << '\n';
}

int main() {
    int t;
    cin >> t;
    while (t--)
        solve();
}
```

Codeforces Round #645 (Div. 2) 1358C Обновление Celex

C. Обновление Celex

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

В связи с карантином, у Сайкромофта появилось больше свободного времени для реализации новых функций в «Celex-2021». Разработчики сделали новую функцию GAZ-GIZ, которая от левого верхнего угла бесконечно заполняет бесконечную вправо и вниз таблицу следующим образом:

Клетка с координатами (x, y) находится на пересечение x-й строки и y-го столбца. Левая верхняя клетка $(1, 1)$ содержит число 1.

Разработчики функции SUM тоже не спят. От скуки они сговорились с разработчиками функции RAND, поэтому они добавили возможность посчитать сумму на произвольном пути от одной клетки до другой, передвигаясь вниз или вправо. Формально, из клетки (x, y) за один шаг можно переместиться в клетку $(x + 1, y)$ или $(x, y + 1)$.

После очередного обновления Dinwows, Левиан решил изучать «Celex-2021» (ведь он хочет стать бухгалтером!). После заполнения таблицы функцией GAZ-GIZ он попросил вас посчитать количество возможных различных сумм на пути от заданной клетки (x_1, y_1) до другой заданной клетки (x_2, y_2), если за один ход вы можете ходить только на одну ячейку вниз или вправо.

Формально, рассмотрим все пути из клетки (x_1, y_1) в клетку (x_2, y_2) такие, что каждая следующая клетка в пути располагается либо справа, либо снизу от предыдущей. Посчитайте количество различных сумм значений элементов для всех таких путей.

Входные данные

Первая строка содержит одно целое число t ($1 \le t \le 57179$) — количество наборов входных данных.

Каждая из следующих t строк содержит четыре целых положительных числа x_1, y_1, x_2, y_2 ($1 \le x_1 \le x_2 \le 10^9$, $1 \le y_1 \le y_2 \le 10^9$) — координаты стартовой и конечной клеток.

Выходные данные

На каждый набор входных данных в отдельной строке выведите ответ — количество возможных различных сумм на пути от одной клетки до другой.

Пример

входные данные
4 1 1 2 2 1 2 2 4 179 1 179 100000 5 7 5 7

выходные данные
2 3 1 1

Примечание

В первом наборе входных данных есть две возможных суммы: $1 + 2 + 5 = 8$ и $1 + 3 + 5 = 9$.

1358C - Celex Update

Let's look at the way with the minimum sum (first we go $y_2 - y_1$ steps right, and then $x_2 - x_1$ steps down). Let's look at such a change in the "bends" of the way:

After each step, the sum on the way will increase by 1.

We're going to bend like this until we get to the maximum sum. We're not going to miss any possible sum, because we're

incrementing the sum by 1. We started with the minimum sum and finished with the maximum sum, so we can use these changes to get all possible sums.

In order for us to come from the minimum to the maximum way, we must bend the way exactly 1 time per each cell of table (except for the cells of the minimum way). That is, the number of changes equals the number of cells not belonging to the minimum way — $(x_2 - x_1) \cdot (y_2 - y_1)$. Then the number of different sums will be $(x_2 - x_1) \cdot (y_2 - y_1) + 1$.

The overall compexity is $\mathcal{O}(1)$ per test.

```
#include <iostream>

using namespace std;

int main() {
    int t;
    cin >> t;
    while (q--) {
        long long a, b, c, d;
        cin >> a >> b >> c >> d;
        cout << (c - a) * (d - b) + 1 << '\n';
    }
}
```

Educational Codeforces Round 88 (рейтинговый для Див. 2)
1359A Берляндский покер

A. Берляндский покер

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

В берляндский покер играют с колодой из n карт, m из которых являются джокерами. В игре участвует k игроков (n делится на k).

В начале игры каждый игрок берет $\frac{n}{k}$ карт из колоды (таким образом, каждая карта берется ровно одним игроком). Игрок, у которого максимальное количество джокеров в руке, является победителем, и он получает количество очков, равное $x - y$, где x — количество джокеров в руке победителя, а y — максимальное количество джокеров среди всех других игроков. Если есть два или более игроков с максимальным количеством джокеров, все они являются победителями, и они получают 0 очков.

Вот несколько примеров:

- $n = 8$, $m = 3$, $k = 2$. Если один игрок получает 3 джокера и 1 простую карту, а другой игрок получает 0 джокеров и 4 простые карты, то первый игрок является победителем и получает $3 - 0 = 3$ очка;
- $n = 4$, $m = 2$, $k = 4$. Два игрока получают простые карты, а два других игрока получают джокеры, так что оба они являются победителями и получают 0 очков;
- $n = 9$, $m = 6$, $k = 3$. Если первый игрок получает 3 джокера, второй игрок получает 1 джокера и 2 простые карты, а третий игрок получает 2 джокера и 1 простую карту, то первый игрок является победителем, и он получает $3 - 2 = 1$ очко;
- $n = 42$, $m = 0$, $k = 7$. Поскольку джокеров нет, каждый получает 0 джокеров, каждый является победителем, и каждый получает 0 очков.

Для заданных n, m и k вычислите максимальное количество очков, которое игрок может получить за победу в игре.

Входные данные

Первая строка входных данных содержит одно целое число t ($1 \le t \le 500$) — количество наборов входных данных.

Каждый набор входных данных содержит три целых числа n, m и k ($2 \le n \le 50$, $0 \le m \le n$, $2 \le k \le n$, k делит n).

Выходные данные

Для каждого набора входных данных выведите одно целое число — максимальное количество очков, которое игрок может получить за победу в игре.

Пример

входные данные
4 8 3 2 4 2 4 9 6 3 42 0 7
выходные данные
3 0 1 0

Примечание

Тесты из примера разобраны в условии.

1359A - Berland Poker

There are many different ways to solve this problem. The easiest one, in my opinion, is to iterate on the number of jokers the winner has (let it be a_1) and the number of jokers the runner-up has (let it be a_2). Then the following conditions should be met:

- $a_1 \geq a_2$ (the winner doesn't have less jokers than the runner-up);
- $a_1 \leq \frac{n}{k}$ (the number of jokers in the winner's hand does not exceed the number of cards in his hand);
- $a_1 + a_2 \leq m$ (the number of jokers for these two players does not exceed the total number of jokers);
- $a_1 + (k-1)a_2 \geq m$ (it is possible to redistribute remaining jokers among other players so that they have at most a_2 jokers).

Iterating on a_1 and a_2, then checking these constraints gives us a $O(n^2)$ solution. It is possible to get a constant-time solution using some greedy assumptions and math (the first player should get as many jokers as possible, while the remaining jokers should be evenly distributed among other players).

```
t = int(input())

for i in range(t):
    n, m, k = map(int, input().split())
    d = n // k
    a1 = min(m, d)
    a2 = (m - a1 + k - 2) // (k - 1)
    print(a1 - a2)
```

```
t = int(input())

for i in range(t):
    n, m, k = map(int, input().split())
    ans = 0
    d = n // k
    for a1 in range(m + 1):
        for a2 in range(a1 + 1):
            if(a1 > d):
                continue
            if(a1 + a2 > m):
                continue
            if(a1 + (k - 1) * a2 < m):
                continue
            ans = max(ans, a1 - a2)
    print(ans)
```

Educational Codeforces Round 88 (рейтинговый для Див. 2)
1359B Новая Театральная площадь

B. Новая Театральная площадь

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Возможно, вы помните Театральную площадь из задачи 1A. Сегодня ее покрытие наконец-то заменят.

Площадь все еще является прямоугольником $n \times m$ метров. Однако, рисунок на ней будет более сложным в этот раз. Пусть $a_{i,j}$ будет j-й ячейкой в i-м ряду покрытия плитками.

Вам задан рисунок покрытия:

- если $a_{i,j} =$ «*», то j-я ячейка в i-м ряду должна быть **черной**;
- если $a_{i,j} =$ «.», то j-я ячейка в i-м ряду должна быть **белой**.

Черные ячейки уже покрыты. Вам же необходимо покрыть белые ячейки. Существует две опции плиток:

- 1×1 плитки — каждая плитка стоит x бурлей и покрывает ровно 1 ячейку;
- 1×2 плитки — каждая плитка стоит y бурлей и покрывает ровно 2 соседние ячейки в **одном ряду**. **Обратите внимание, что нельзя вращать эти плитки или резать их на плитки 1×1.**

Вам необходимо покрыть все белые ячейки, никакие две плитки не должны накладываться друг на друга и никакие черные ячейки не должны быть накрыты плитками.

Чему равна наименьшая суммарная цена плиток, которые могут покрыть все белые клетки?

Входные данные

В первой строке записано одно целое число t ($1 \le t \le 500$) — количество наборов входных данных. Затем следует описание t наборов.

В первой строке каждого набора входных данных записаны четыре целых числа n, m, x и y ($1 \le n \le 100$; $1 \le m \le 1000$; $1 \le x, y \le 1000$) — размеры Театральной площади, цена плитки 1×1 и цена плитки 1×2.

В каждой из следующих n строк записаны по m символов. j-й символ в i-й строке — это $a_{i,j}$. Если $a_{i,j} =$ «*», то j-я ячейка в i-м ряду должна быть **черной**, а если $a_{i,j} =$ «.», то j-я ячейка в i-м ряду должна быть **белой**.

Гарантируется, что сумма $n \times m$ по всем наборам входных данных не превышает 10^5.

Выходные данные

Для каждого набора входных данных выведите одно целое число — наименьшая суммарная цена плиток, которые могут покрыть все белые клетки, в бурлях.

Пример

входные данные
4 1 1 10 1 . 1 2 10 1 .. 2 1 10 1 . . 3 3 3 7 ..* *.. .*.

выходные данные
10 1 20 18

Примечание

В первом наборе входных данных необходимо использовать одну плитку 1×1, несмотря на то, что плитка 1×2 дешевле. Поэтому суммарная цена равна 10 бурлей.

Во втором наборе можно использовать либо две плитки 1×1 и потратить 20 бурлей, либо одну плитку 1×2 плитку и потратить 1 бурль. Второй вариант дешевле, поэтому ответ равен 1.

Третий набор показывает, что нельзя поворачивать плитки 1×2. Приходится использовать две плитки 1×1 с суммарной ценой 20.

В четвертом наборе самый дешевый способ — это использовать 1×1 плитки повсюду. Итоговая цена — $6 \cdot 3 = 18$.

1359B - New Theatre Square

Notice that rows can be solved completely separately of each other. Each tile takes either one or two squares but it's always in the same row.

So let's take a look at a single row. There are sequences of dot characters separated by some asterisks. Once again each of these sequences can be solved independently of the others.

Thus, we have these empty strips of empty squares $1 \times k$ which, when solved, can be summed up into the whole answer.

There are two cases, depending on if a 1×2 is cheaper than two 1×1 tiles.

If it is then we want to use of many 1×2 tiles as possible. So given k, we can place $\lfloor \frac{k}{2} \rfloor$ 1×2 tiles and cover the rest $k - 2 \cdot \lfloor \frac{k}{2} \rfloor = k \bmod 2$ squares with 1×1 tiles.

If it isn't cheaper then we want to cover everything with 1×1 tiles and never use 1×2 ones. So all k should be 1×1.

The easier way to implement this might be the following. Let's update the price of the 1×2 tile with the minimum of y and $2 \cdot x$. This way the first algorithm will produce exactly the same result of the second one in the case when a 1×2 tile isn't cheaper than two 1×1 ones.

Overall complexity: $O(nm)$ per testcase.

```
t = int(input())
for _ in range(t):
 n, m, x, y = map(int, input().split())
 ans = 0
 y = min(y, 2 * x)
 for __ in range(n):
  s = input()
  i = 0
  while i < m:
   if s[i] == '*':
    i += 1
    continue
   j = i
   while j + 1 < m and s[j + 1] == '.':
    j += 1
   l = j - i + 1
   ans += l % 2 * x + l // 2 * y
   i = j + 1
 print(ans)
```

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

#define forn(i, n) for (int i = 0; i < int(n); i++)

using namespace std;


int main() {
 int tc;
 scanf("%d", &tc);
 forn(_, tc){
  int h, c, t;
  scanf("%d%d%d", &h, &c, &t);
  if (h + c - 2 * t >= 0)
   puts("2");
  else{
   int a = h - t;
   int b = 2 * t - c - h;
   int k = 2 * (a / b) + 1;
   long long val1 = abs(k / 2 * 1ll * c + (k + 1) / 2 * 1ll * h - t * 1ll * k);
   long long val2 = abs((k + 2) / 2 * 1ll * c + (k + 3) / 2 * 1ll * h - t * 1ll * (k + 2));
   printf("%d\n", val1 * (k + 2) <= val2 * k ? k : k + 2);
  }
 }
 return 0;
}
```

Educational Codeforces Round 88 (рейтинговый для Див. 2)
1359C Смешиваем воду

C. Смешиваем воду

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Есть два бесконечных источника воды:

- горячая вода температуры h;
- холодная вода температуры c ($c < h$).

Вы совершаете следующую чередующуюся последовательность действий:

1. набрать **одну** кружку **горячей** воды и вылить ее в бесконечно глубокую бочку;
2. набрать **одну** кружку **холодной** воды и вылить ее в бесконечно глубокую бочку;
3. набрать **одну** кружку **горячей** воды ...
4. и так далее ...

Обратите внимание, что вы всегда начинаете с кружки горячей воды.

Бочка изначально пустая. Необходимо налить **хотя бы одну кружку** в бочку. Температура воды в бочке равна средней температуре вылитых кружек.

Вы хотите получить воду температуры как можно ближе к t. То есть если температура воды в бочке равна t_b, то **абсолютная разность** t_b и t ($|t_b - t|$) должна быть минимально возможна.

Сколько кружек необходимо налить в бочку, чтобы температура стала как можно ближе к t? Если существует несколько ответов с минимальной абсолютной разностью, то выведите наименьший из них.

Входные данные

В первой строке записано одно целое число T ($1 \le T \le 3 \cdot 10^4$) — количество наборов входных данных.

В каждой из следующих T строк записаны по три целых числа h, c и t ($1 \le c < h \le 10^6$; $c \le t \le h$) — температура горячей воды, температура холодной воды и желаемая температура в бочке.

Выходные данные

На каждый набор входных данных выведите одно положительное целое число — минимальной количество кружек, которое необходимо вылить в бочку, чтобы получить температуру, как можно более близкую к t.

Пример

входные данные
3 30 10 20 41 15 30 18 13 18

выходные данные
2 7 1

Примечание

В первом наборе входных данных температура после 2 налитых кружек: 1 горячей и 1 холодной — равна 20. И это самое близкое возможное значение.

Во втором наборе температура после 7 налитых кружек: 4 горячих и 3 холодных — примерно 29.857. Если наливать больше воды, то температура не станет ближе к t.

В третьем наборе температура после 1 налитой кружки: 1 горячей — равна 18. Это совпадает с t.

1359C - Mixing Water

So there are two kinds of stops to consider: k hot and k cold cup and $(k+1)$ hot and k cold cups.

The first case is trivial: the temperature is always $\frac{h+c}{2}$. In the second case the temperature is always strictly greater than $\frac{h+c}{2}$. Thus, if $t \leq \frac{h+c}{2}$, then the answer is 2.

Let's show that otherwise the answer is always achieved through the second case.

The temperature after $(k+1)$ hot cups and k cold cups is $t_k = \frac{(k+1)\cdot h + k\cdot c}{2k+1}$. The claim is that $t_0 > t_1 > \ldots$. Let's prove that by induction.

$t_0 = h, t_1 = \frac{2\cdot h + c}{3}$. $c < h$, thus $t_0 > t_1$.

Now compare t_k and t_{k+1}.

$$t_k > t_{k+1}$$
$$\frac{(k+1)\cdot h + k\cdot c}{2k+1} > \frac{(k+2)\cdot h + (k+1)\cdot c}{2k+3}$$
$$\frac{k\cdot(h+c) + h}{2k+1} > \frac{(k+1)\cdot(h+c) + h}{2k+3}$$
$$2k\cdot(k\cdot(h+c)+h) + 3k\cdot(h+c) + 3h > 2k\cdot((k+1)\cdot(h+c)+h) + (k+1)\cdot(h+c) + h$$
$$2k\cdot(k\cdot(h+c)+h-(k+1)\cdot(h+c)-h) > (k+1)\cdot(h+c) + h - 3k\cdot(h+c) - 3h$$
$$2k\cdot(-(h+c)) > (-2k+1)\cdot(h+c) - 2h$$
$$2h > (h+c)$$
$$h > c$$

We can also show that this series converges to $\frac{h+c}{2}$:

I'm sorry that I'm not proficient with any calculus but my intuition says that it's enough to show that $\forall k\, t_k > \frac{h+c}{2}$ and $\forall \varepsilon \exists k\, t_k < \frac{h+c}{2}$ with $k \geq 0$.

So the first part is:

$$\frac{(k+1)\cdot h + k\cdot c}{2k+1} > \frac{h+c}{2}$$
$$\frac{k\cdot(h+c)+h}{2k+1} > \frac{h+c}{2}$$
$$2k\cdot(h+c) + 2h > (2k+1)\cdot(h+c)$$
$$2h > h + c$$
$$h > c$$

And the second part is:

$$\frac{(k+1)\cdot h + k\cdot c}{2k+1} < \frac{h+c}{2} + \varepsilon$$
$$\frac{k\cdot(h+c)+h}{2k+1} < \frac{h+c}{2} + \varepsilon$$
$$2k\cdot(h+c) + 2h < (2k+1)\cdot(h+c) + (2k+1)\cdot\varepsilon$$
$$2h < (h+c) + (2k+1)\cdot\varepsilon$$
$$h < c + (2k+1)\cdot\varepsilon$$

$$\frac{h-c}{\varepsilon} < 2k+1$$

So that claim makes us see that for any t greater than $\frac{h+c}{2}$ the answer is always achieved from the second case.

That allows us to find such k, that the value of t_k is exactly t. However, such k might not be integer. $\frac{(k+1)\cdot h+c}{2k+1} = t \leftrightarrow \frac{k\cdot(h+c)+h}{2k+1} = t \leftrightarrow k\cdot(h+c)+h = 2kt+t \leftrightarrow k\cdot(h+c-2t) = t-h \leftrightarrow k = \frac{t-h}{h+c-2t}$.

The only thing left is to compare which side is better to round k to. It seems some implementations with float numbers might fail due to precision errors. However, it's possible to do these calculations completely in integers.

Let's actually rewrite that so that the denominator is always positive $k = \frac{h-t}{2t-h-c}$. Now we can round this value down and compare k and $k+1$.

So the optimal value is k if $|\frac{k\cdot(h+c)+h}{2k+1} - t| \le |\frac{(k+1)\cdot(h+c)+h}{2k+3}| - t$. So $|(k\cdot(h+c)+h) - t\cdot(2k+1)|\cdot(2k+3) \le |((k+1)\cdot(h+c)+h) - t\cdot(2k+3)|\cdot(2k+1)$. Otherwise, the answer is $k+1$.

You can also find the optimal k with binary search but the formulas are exactly the same and you have to rely on monotonosity as well. Also, these formulas can get you the better understanding for the upper bound of the answer.

Overall complexity: $O(1)$ or $O(\log h)$ per testcase.

Codeforces Round #647 (Div. 2) - Thanks, Algo Muse! 1362A Джонни и древний компьютер

A. Джонни и древний компьютер

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Недавно Джонни обнаружил древний сломанный компьютер. У него есть только один регистр, в который можно записать некоторое значение. После чего, за одну операцию вы можете применить к значению битовый сдвиг влево или вправо на не более чем три позиции. Сдвиг вправо запрещен, если в результате **будут потеряны единичные биты**. Так что, на самом деле, за одну операцию вы можете умножить или разделить значение на 2, 4 или 8, и деление разрешено только если значение делится нацело на выбранный делитель.

Формально, если регистр содержит целое положительное число x, за одну операцию оно может быть заменено одним из следующих:

- $x \cdot 2$
- $x \cdot 4$
- $x \cdot 8$
- $x/2$, если x делится на 2
- $x/4$, если x делится на 4
- $x/8$, если x делится на 8

Например, если $x = 6$, за одну операцию оно может быть заменено на 12, 24, 48 или 3. Значение 6 не делится на 4 или 8, поэтому существуют только четыре варианта замены.

Теперь Джонни интересуется, какое минимальное количество операций необходимо, если он запишет в регистр значение a и в конце хочет получить там значение b.

Входные данные

Входные данные состоят из нескольких наборов входных данных. Первая строка содержит целое число t ($1 \le t \le 1000$) — количество наборов входных данных. Следующие t строк содержат описание наборов входных данных.

Первая и единственная строка каждого набора входных данных содержит целые числа a и b ($1 \le a, b \le 10^{18}$) — исходное значение и желаемое итоговое значение, соответственно.

Выходные данные

Выведите t строк, каждая строка должна содержать одно целое число, обозначающее минимальное количество операций, которое Джонни должен выполнить. Если Джонни не сможет получить значение b в конце, выведите -1.

Пример

входные данные
10 10 5 11 44 17 21 1 1 96 3 2 128 1001 1100611139403776 1000000000000000000 1000000000000000000 7 1 10 8
выходные данные
1 1 -1 0 2 2 14 0

```
-1
-1
```

Примечание

В первом наборе входных данных, Джонни может получить 5 из 10 сделав один сдвиг вправо на один (т.е. поделив на 2).

Во втором наборе входных данных, Джонни может получить 44 из 11 сделав один сдвиг влево на два (т.е. умножив на 4).

В третьем наборе входных данных, Джонни не может получить значение 21 из значения 17.

В четвертом наборе входных данных, исходное и желаемое значения совпадают, поэтому Джонни придется сделать 0 операций.

В пятом наборе входных данных, Джонни может получить 3 из 96 сделав два сдвига вправо: один на 2, и другой на 3 (т.е. поделив на 4 и 8).

1362A - Johnny and Ancient Computer

Let us write a as $r_a \cdot 2^x$ and b as $r_b \cdot 2^y$, where r_a and r_b are odd. The only operation we have changes x by $\{-3, -2, -1, 1, 2, 3\}$ so r_a must be equal to r_b, otherwise the answer is -1. It is easy to notice that we can greedily move x toward y so the answer is equal to $\lceil \frac{|x-y|}{3} \rceil$.

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

using namespace std;

typedef long long LL;

LL getR(LL a){
 while(a % 2 == 0)
  a /= 2;
 return a;
}

void solve(){
 LL a, b;
 scanf("%lld %lld", &a, &b);
 if(a > b) swap(a, b);

 LL r = getR(a);
 if(getR(b) != r){
  puts("-1");
  return;
 }

 int ans = 0;
 b /= a;

 while(b >= 8)
  b /= 8, ++ans;
 if(b > 1) ++ans;
 printf("%d\n", ans);
}

int main(){
 int quest;
 scanf("%d", &quest);

 while(quest--)
  solve();
 return 0;
}
```

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

using namespace std;

typedef long long LL;

LL getR(LL a){
 while(a % 2 == 0)
  a /= 2;
 return a;
}

void solve(){
 LL a, b;
 scanf("%lld %lld", &a, &b);
 if(a > b) swap(a, b);

 LL r = getR(a);
 if(getR(b) != r){
  puts("-1");
  return;
 }

 int ans = 0;
 b /= a;

 while(b >= 8)
  b /= 8, ++ans;
 if(b > 1) ++ans;
 printf("%d\n", ans);
}

int main(){
 int quest;
 scanf("%d", &quest);

 while(quest--)
  solve();
 return 0;
}
```

Codeforces Round #647 (Div. 2) - Thanks, Algo Muse! 1362B Джонни и его хобби

B. Джонни и его хобби

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Среди многочисленных хобби Джонни, есть два, казалось бы, безвредных: применение побитовых операций и прокрадывание в офис его отца. Как это обычно и бывает у маленьких детей, Джонни не подозревает, что комбинирование этих двух занятий может доставить ему много неприятностей.

На столе его отца хранится множество S, содержащее очень важные числа. В момент, когда Джонни узнал об этом, он решил, что это хорошая идея выбрать **положительное** целое число k и заменить все элементы s из множества S на $s \oplus k$ ($\oplus$ обозначает операцию исключающего или).

Помогите Джонни выбрать такое k, что его отец не заметит никакой разницы после того, как Джонни закончит играть (т.е. Джонни должен получить такое же множество, какое было изначально). Возможно, что ни одного такого числа не существует. Также, возможно, что существует несколько подходящих чисел. В таком случае, выведите минимальное из них. Обратите внимание, что порядок элементов в множестве не имеет значения, т.е. множество $\{1, 2, 3\}$ равно множеству $\{2, 1, 3\}$.

Формально, требуется найти минимальное положительное целое число k, такое что $\{s \oplus k | s \in S\} = S$ или сообщить, что ни одного подходящего числа не существует.

Например, если $S = \{1, 3, 4\}$ и $k = 2$, новое множество будет равно $\{3, 1, 6\}$. Если $S = \{0, 1, 2, 3\}$ и $k = 1$, после игры множество останется таким же.

Входные данные

В первой строке дано одно целое число t ($1 \le t \le 1024$) — количество наборов входных данных. В следующих строках даны t наборов входных данных, каждый из них занимает две строки.

В первой строке набора входных данных дано одно целое число n ($1 \le n \le 1024$), обозначающее количество элементов в множестве S. Вторая строка содержит n **различных** целых чисел s_i ($0 \le s_i < 1024$) — элементы S.

Гарантируется, что сумма n по всем наборам входных данных не превышает 1024.

Выходные данные

Выведите t строк, i-я из них должна содержать ответ на i-й набор входных данных — минимальное положительное целое число k, удовлетворяющее условиям, или -1, если ни одного подходящего k не существует.

Пример

входные данные
6 4 1 0 2 3 6 10 7 14 8 3 12 2 0 2 3 1 2 3 6 1 4 6 10 11 12 2 0 1023

выходные данные
1 4 2 -1 -1 1023

Примечание

В первом наборе входных данных, ответ 1, потому что это минимальное положительное целое число и оно удовлетворяет всем условиям.

1362B - Johnny and His Hobbies

Consider i-th least significant bit (0 indexed). If it is set in k, but not in s, it will be set in $k \oplus s$. Hence $k \oplus s \geq 2^i$.

Consider such minimal positive integer m, that $2^m > s$ holds for all $s \in S$. k cannot have the i-th bit set for any $i \geq m$. From this follows that $k < 2^m$. So there are only 2^m feasible choices of k. We can verify if a number satisfies the condition from the statement in $\mathcal{O}(n)$ operations. This gives us a solution with complexity $\mathcal{O}(n \cdot 2^m)$. Note that in all tests m is at most 10.

There is also another solution possible. It uses the observation that if k satisfies the required conditions, then for every $s \in S$ there exists such $t \in S$ ($t \neq s$) , that $t \oplus s = k$. This gives us $n - 1$ feasible choices of k and thus the complexity of this solution is $\mathcal{O}(n^2)$.

```cpp
//O(n * maxA) solution

#include <bits/stdc++.h>

using namespace std;

const int N = 1025;

int n;
int in[N];
bool is[N];

bool check(int k){
 for(int i = 1; i <= n; ++i)
  if(!is[in[i] ^ k])
   return false;
 return true;
}

void solve(){
 for(int i = 0; i < N; ++i)
  is[i] = false;

 scanf("%d", &n);
 for(int i = 1; i <= n; ++i){
  scanf("%d", &in[i]);
  is[in[i]] = true;
 }

 for(int k = 1; k < 1024; ++k)
  if(check(k)){
   printf("%d\n", k);
   return;
  }

 puts("-1");
}

int main(){
 int cases;
 scanf("%d", &cases);

 while(cases--)
  solve();
 return 0;
}
```

```cpp
//O(n * maxA) solution

#include <bits/stdc++.h>

using namespace std;

const int N = 1025;

int n;
int in[N];
bool is[N];

bool check(int k){
 for(int i = 1; i <= n; ++i)
  if(!is[in[i] ^ k])
   return false;
 return true;
}

void solve(){
 for(int i = 0; i < N; ++i)
  is[i] = false;

 scanf("%d", &n);
 for(int i = 1; i <= n; ++i){
  scanf("%d", &in[i]);
  is[in[i]] = true;
 }

 for(int k = 1; k < 1024; ++k)
  if(check(k)){
   printf("%d\n", k);
   return;
  }

 puts("-1");
}

int main(){
 int cases;
 scanf("%d", &cases);

 while(cases--)
  solve();
 return 0;
}
```

Codeforces Round #647 (Div. 2) - Thanks, Algo Muse! 1362C Джонни и ещё одно падение рейтинга

C. Джонни и ещё одно падение рейтинга

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Последний контест, проведённый на любимой Джонни платформе по спортивному программированию, был встречен довольно позитивно. Однако, рейтинг Джонни снова упал! Он думает, что хотя задачи были неплохие, они не показывают истинный уровень участников.

Теперь мальчик смотрит на рейтинги соседних участников, записанные в двоичной системе счисления. Он считает, что чем больше эти рейтинги различаются, тем более нечестно, что эти участники расположены на соседних позициях. Он определяет различие между двумя числами как количество позиций битов, на которых одно из чисел содержит ноль, а другое — единицу (мы считаем, что числа дополнены ведущими нулями до одинаковой длины). Например, различие между числами $5 = 101_2$ и $14 = 1110_2$ равно 3, так как 0101 и 1110 отличаются в 3 позициях. Джонни определяет нечестность контеста как сумму таких различий для всех пар соседних участников.

Джонни только что прислал вам последовательность рейтингов и хочет, чтобы вы вычислили нечестность контеста. Вы заметили, что вы получили последовательные целые числа от 0 до n. Это странно, но мальчик упорно твердит, что все правильно. Поэтому, помогите ему и вычислите желаемую нечестность для полученной последовательности чисел.

Входные данные

Входные данные состоят из нескольких наборов входных данных. Первая строка содержит одно целое число t ($1 \le t \le 10\,000$) — количество наборов входных данных. Следующие t строк содержат описание наборов входных данных.

Первая и единственная строка каждого набора входных данных содержит одно целое число n ($1 \le n \le 10^{18}$).

Выходные данные

Выведите t строк. Для каждого набора входных данных, вы должны вывести одну строку, содержащую одно целое число — нечестность контеста, если последовательность рейтингов равна 0, 1, ..., $n - 1$, n.

Пример

входные данные
5 5 7 11 1 2000000000000
выходные данные
8 11 19 1 3999999999987

Примечание

Для $n = 5$, мы вычисляем нечестность следующей последовательности (числа от 0 до 5, записанные в двоичной системе счисления и дополненные ведущими нулями до одинаковой длины):

- 000
- 001
- 010
- 011
- 100
- 101

Различия равны 1, 2, 1, 3, 1, соответственно. Поэтому, нечестность равна $1 + 2 + 1 + 3 + 1 = 8$.

1362C - Johnny and Another Rating Drop

Let us start by calculating the result for $n = 2^k$. It can be quickly done by calculating the results for each bit separately and summing these up. For i-th bit, the result is equal to $\frac{2^k}{2^i}$ as this bit is different in $d - 1$ and d iff d is a multiple of 2^i. Summing these up we get that the result for $n = 2^k$ is equal to $2^{k+1} - 1 = 2n - 1$.

How to compute the answer for arbitrary n? Let us denote $b_1 > b_2 > \ldots > b_k$ as set bits in the binary representation of n. I claim that the answer is equal to the sum of answers for $2^{b_1}, 2^{b_2}, \ldots, 2^{b_k}$. Why?

We can compute results for intervals $[0, 2^{b_1}], [2^{b_1}, 2^{b_1} + 2^{b_2}], \ldots, [n - 2^{b_k}, n]$. We can notice that the result for interval $[s, s + 2^i]$, where s is a multiple of 2^i, is equal to the answer for $[0, 2^i]$ so we can just compute the results for intervals $[0, 2^{b_1}], [0, 2^{b_2}], \ldots, [0, 2^{b_k}]$!

This allows us to compute the answer for arbitrary n in $\mathcal{O}(\log n)$ – just iterate over all bits b and add $2^{b+1} - 1$ if b is set. Equivalently we can just write down $2n - $ **#bits set** as the answer.

Final complexity is $\mathcal{O}(t \log n)$.

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

using namespace std;

typedef long long LL;

void solve(){
 LL a;
 scanf("%lld", &a);

 LL ans = 0;
 for(int i = 0; i < 60; ++i)
  if(a & (1LL << i))
   ans += (1LL << (i + 1)) - 1;
 printf("%lld\n", ans);
}

int main(){
 int quest;
 scanf("%d", &quest);

 while(quest--)
  solve();
 return 0;
}
```

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

using namespace std;

typedef long long LL;

void solve(){
 LL a;
 scanf("%lld", &a);

 LL ans = 0;
 for(int i = 0; i < 60; ++i)
  if(a & (1LL << i))
   ans += (1LL << (i + 1)) - 1;
 printf("%lld\n", ans);
}

int main(){
 int quest;
 scanf("%d", &quest);

 while(quest--)
  solve();
 return 0;
}
```

Codeforces Round #646 (Div. 2) 1363A Выбор нечетных

A. Выбор нечетных

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

У Shubham есть массив a размера n, и он хочет выбрать из него ровно x элементов так, чтобы их сумма была нечетной. Эти элементы не обязаны быть последовательными. Элементы массива не обязательно различны.

Скажите ему, может ли он сделать это.

Входные данные
В первой строке входных данных содержится одно целое число t $(1 \leq t \leq 100)$ — количество наборов входных данных. Далее следуют описания наборов входных данных.

Первая строка каждого набора входных данных содержит два целых числа n и x $(1 \leq x \leq n \leq 1000)$ — длину массива и количество элементов, которые нужно выбрать соответственно.

Следующая строка каждого набора входных данных содержит n целых чисел $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 1000)$ — элементы массива.

Выходные данные
Для каждого набора входных данных выведите «Yes» или «No» в зависимости от того, можно ли выбрать ровно x элементов, чтобы их сумма была нечетной.

Вы можете выводить буквы в любом регистре.

Пример

входные данные
5 1 1 999 1 1 1000 2 1 51 50 2 2 51 50 3 3 101 102 103
выходные данные
Yes No Yes Yes No

Примечание
В 1-м наборе входных данных: мы должны выбрать элемент 999, и сумма будет нечетной.

В 2-м наборе входных данных: мы должны выбрать элемент 1000, поэтому сумма не будет нечетной.

В 3-м наборе входных данных: мы можем выбрать элемент 51.

В 4-м наборе входных данных: мы должны выбрать оба элемента 50 и 51 — так что общая сумма нечетна.

В 5-м наборе входных данных: мы должны выбрать все элементы — но общая сумма не является нечетной.

1363A - Odd Selection

Key Idea: The sum of x numbers can only be odd if we have an odd number of numbers which are odd. (An odd statement, indeed).

Detailed Explanation: We first maintain two variables, `num_odd` and `num_even`, representing the number of odd and even

numbers in the array, respectively. We then iterate over the number of odd numbers we can choose; which are $1, 3, 5, \ldots$ upto `min(num_odd,x)`, and see if `num_even` $\geq x - i$ where i is the number of odd numbers we have chosen.

Time complexity: $O(N)$

```
#include <bits/stdc++.h>
using namespace std;

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define endl "\n"
#define int long long

const int N = 2e5 + 5;

int n, x;
int a[N], f[2];

int32_t main()
{
 IOS;
 int t;
 cin >> t;
 while(t--)
 {
    f[0] = f[1] = 0;
    cin >> n >> x;
    for(int i = 1; i <= n; i++)
    {
     cin >> a[i];
     f[a[i] % 2]++;
    }
    bool flag = 0;
    for(int i = 1; i <= f[1] && i <= x; i += 2) //Fix no of odd
    {
     int have = f[0], need = x - i;
     if(need <= f[0])
          flag = 1;
    }
    if(flag)
       cout << "Yes" << endl;
    else
       cout << "No" << endl;
 }
 return 0;
}
```

Codeforces Round #646 (Div. 2) 1363B Ненависть к подпоследовательностям

B. Ненависть к подпоследовательностям

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

У Shubham есть бинарная строка s. Бинарная строка — это строка, содержащая только символы «0» и «1».

Он может выполнить следующую операцию над строкой любое количество раз:

- Выбрать индекс строки, и поменять символ с этим индексом. Это означает, что если символ был «0», он становится «1», и наоборот.

Строка называется хорошей, если она не содержит строк «010» или «101» в качестве подпоследовательностей — например, «1001» содержит «101» как подпоследовательность, следовательно, это не хорошая строка, а «1000» не содержит ни «010» ни «101» как подпоследовательностей, поэтому это хорошая строка.

Какое минимальное количество операций ему придется выполнить, чтобы строка стала хорошей? Можно показать, что с помощью данных операций можно сделать любую строку хорошей.

Строка a является подпоследовательностью строки b, если a может быть получена из b удалением нескольких (возможно, ни одного или всех) символов.

Входные данные

В первой строке входных данных содержится одно целое число t $(1 \le t \le 100)$ — количество наборов входных данных.

Каждая из следующих t строк содержит бинарную строку s $(1 \le |s| \le 1000)$.

Выходные данные

Для каждой строки выведите минимальное количество операций необходимых для того, чтобы сделать ее хорошей.

Пример

входные данные
7 001 100 101 010 0 1 001100

выходные данные
0 0 1 1 0 0 2

Примечание

В наборах входных данных 1, 2, 5, 6 строки уже являются хорошими — поэтому никаких операций не требуется.

Для набора 3: «001» можно получить, поменяв первый символ, и это один из возможных способов получить хорошую строку.

Для набора 4: «000» можно получить, поменяв второй символ, и это один из возможных способов получить хорошую строку.

Для набора 7: «000000» можно получить, поменяв третий и четвертый символы, и это один из возможных способов получить хорошую строку.

1363B - Subsequence Hate

Key Idea: There are two types of good strings: Those which start with a series of 1's followed by 0's (such as 1111100) and those which start with a series of 0's followed by 1's (such as 00111). Note that there are strings which do belong to both categories (such as 000).

Detailed Explanation: We will use the key idea to compute the minimum change required to achieve every possible string of each of the two types, and then take the minimum across them.

First, let us compute the total number of 1's and 0's in the string, denoted by `num_ones` and `num_zeros`. Now, as we iterate through the string, let us also maintain `done_ones` and `done_zeros`, which denote the number of 1's and 0's encountered so far.

Let us iterate through the string. When we are at position i (indexed from 1), we want to answer two questions: what is the cost for changing the string into $11..000$ (where number of 1's = i) and what is the cost for changing the string into $00..111$ (where number of 0's = i).

Assuming that `done_zeros` and `done_ones` also consider the current index, the answer to the first question is `done_zeros + num_ones - done_ones`. This is because `done_zeros` 0's must be converted to 1's, and `num_ones - done_ones` 1's must be converted to 0's. Similarly, the answer for the second question is `done_ones + num_zeros - done_zeros`.

The answer is the minimum over all such changes possible. Please do not forget to consider the all 1's and all 0's string in the above solution.

Time Complexity: $O(N)$

```cpp
#include <bits/stdc++.h>
using namespace std;

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define endl "\n"
#define int long long

const int N = 1e5 + 5;

int32_t main()
{
 IOS;
 int t;
 cin >> t;
 while(t--)
 {
  string s;
  cin >> s;
  int suf0 = 0, suf1 = 0;
  for(auto &it:s)
  {
   suf0 += (it == '0');
   suf1 += (it == '1');
  }
  int ans = min(suf0, suf1); //Make whole string 0/1
  int pref0 = 0, pref1 = 0;
  for(auto &it:s)
  {
   pref0 += (it == '0'), suf0 -= (it == '0');
   pref1 += (it == '1'), suf1 -= (it == '1');
   //Cost of making string 0*1*
   ans = min(ans, pref1 + suf0);
   //Cost of making string 1*0*
   ans = min(ans, pref0 + suf1);
  }
  cout << ans << endl;
 }
 return 0;
}
```

Codeforces Round #646 (Div. 2) 1363C Игра на листьях

C. Игра на листьях

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Ayush и Ashish играют в игру на некорневом дереве, состоящем из n вершин, пронумерованных от 1 до n. Игроки делают следующий ход по очереди:

- Выберите любой лист в дереве и удалите его вместе со всеми ребрами, для которых этот лист является одним из концов. Лист — это вершина со степенью, не превосходящей 1.

Дерево — это связный неориентированный граф без циклов.

Дана специальная вершина с номером x. Игрок, который удаляет эту вершину, выигрывает игру.

Ayush ходит первым. Определите победителя игры, если каждый игрок играет оптимально.

Входные данные
В первой строке входных данных содержится одно целое число t $(1 \le t \le 10)$ — количество наборов входных данных. Далее следуют описания наборов входных данных.

Первая строка каждого набора входных данных содержит два целых числа n и x $(1 \le n \le 1000, 1 \le x \le n)$ — количество вершин в дереве и специальную вершину, соответственно.

Каждая из следующих $n-1$ строк содержит два целых числа u, v $(1 \le u, v \le n,\ u \ne v)$, что означает, что между вершинами u и v есть ребро.

Выходные данные
Для каждого набора входных данных, если побеждает Ayush, выведите "`Ayush`", иначе выведите "`Ashish`" (без кавычек).

Примеры

входные данные
1 3 1 2 1 3 1
выходные данные
Ashish

входные данные
1 3 2 1 2 1 3
выходные данные
Ayush

Примечание
В первом наборе входных данных Ayush может удалить только вершину 2 или 3, после чего вершина 1 становится листом, и Ashish может удалить ее в свою очередь.

Во втором наборе входных данных Ayush может удалить вершину 2 на самом первом шаге.

1363C - Game On Leaves

Key Idea: The main idea of this problem is to think backwards. Instead of thinking about how the game will proceed, we think about how the penultimate state of the game will look like, etc. Also, we take care of the cases where the game will end

immediately (i.e, when the special node is a leaf node).

Detailed Explanation: First, let us take care of the cases where the game ends immediately. This only occurs when the special node x is a leaf node, so all we must do is check that `deg[x] = 1`. Please note that $n = 1$ must be handled seperately here (just output `Ayush`).

Now, in the case where x is not a leaf node, the answer is as follows: `Ashish` wins if n is odd, and `Ayush` wins if n is even. I will provide a short sketch of the proof below.

With the hint from the key idea, let us analyze this game backwards. (I will assume that $n > 10$ for the sake of a clear explanation). When x is removed from the game, it cannot be the only node remaining (because then the previous player could have also removed x, and thus he did not play optimally). Assume the structure of the game is something like the following WLOG at the last step (The tree attached to x could be any tree):

Consider also that `Ayush` won, and the last move was to remove x. Now, what could have been the state before this move? If `Ashish` had removed a node from the tree, then he did not play optimally - since he could have removed x! Thus, he must have removed something from x, which looks like the following:

Considering this state, `Ashish` should not infact remove 6, and instead remove something from the tree! Hence, the state that we assumed the game should look like at the end is impossible - and indeed, the tree attached to x should only consist of only one node (we already proved that x cannot be the only node remaining).

Thus, all we have to do is find who's turn it will be when the structure of the tree is as follows:

It is `Ashish`'s turn if n is odd, and `Ayush`'s turn if n is even. QED!

Time complexity: $O(N)$

```cpp
#include <bits/stdc++.h>
using namespace std;

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define endl "\n"
#define int long long

const int N = 2e5 + 5;

int n, x;
int deg[N];
vector<int> g[N];

int32_t main()
{
 IOS;
 int t;
 cin >> t;
 while(t--)
 {
  memset(deg, 0, sizeof(deg));
  cin >> n >> x;
  for(int i = 1; i <= n - 1; i++)
  {
   int u, v;
   cin >> u >> v;
   deg[u]++, deg[v]++;
  }
  if(deg[x] <= 1)
   cout << "Ayush" << endl;
  else
  {
   if(n % 2)
    cout << "Ashish" << endl;
   else
    cout << "Ayush" << endl;
  }
 }
 return 0;
}
```

Codeforces Round #648 (Div. 2) 1365A Игра с таблицей

A. Игра с таблицей

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Ashish и Vivek играют в игру на таблице с n строками и m столбцами, захватывая клетки. Незахваченные клетки обозначены 0, а захваченные клетки обозначены 1. Вам дано исходное состояние таблицы.

На каждом ходу, игрок должен захватить одну клетку. Клетку можно захватить, если она еще не захвачена, и она не находится в одной строке или столбце с другой захваченной клеткой. Игра кончается, когда игрок не может сделать ход, в таком случае, он проигрывает.

Если Ashish и Vivek ходят по очереди и Ashish ходит первым, найдите победителя игры если они оба играют оптимально.

Оптимальная игра между двумя игроками означает, что оба игрока выбирают лучшую возможную стратегию, чтобы получить наиболее благоприятный для себя результат игры.

Входные данные

В первой строке записано одно целое число t $(1 \leq t \leq 50)$ — количество наборов входных данных. Далее следуют описания наборов входных данных.

В первой строке каждого набора входных данных записаны два целых числа n, m $(1 \leq n, m \leq 50)$ — количество строк и столбцов в таблице.

В каждой из следующих n строк записаны m целых чисел, j-е число на i-й строке описывает $a_{i,j}$ $(a_{i,j} \in \{0, 1\})$.

Выходные данные

Для каждого набора входных данных, если Ashish выиграет при правильной игре, выведите «Ashish», иначе выведите «Vivek» (без кавычек).

Пример

входные данные
4 2 2 0 0 0 0 2 2 0 0 0 1 2 3 1 0 1 1 1 0 3 3 1 0 0 0 0 0 1 0 0

выходные данные
Vivek Ashish Vivek Ashish

Примечание

В первом наборе входных данных: Один из возможных исходов игры следующий: Ashish захватывает клетку $(1, 1)$, затем Vivek захватывает клетку $(2, 2)$. Ashish не может захватить ни клетку $(1, 2)$, ни клетку $(2, 1)$, так как клетки $(1, 1)$ и $(2, 2)$ уже захвачены. Таким образом, Ashish проигрывает. Можно показать, что вне зависимости от ходов Ashish, Vivek выиграет. Во втором наборе входных данных: Ashish захватывает клетку $(1, 1)$, единственная клетка, которую можно захватить. После этого у Vivek не будет возможных ходов.

В третьем наборе входных данных: Ashish не может сделать ход, поэтому Vivek выиграет.

В четвертом наборе входных данных: Ashish захватывает клетку $(2, 3)$, у Vivek не останется возможных ходов.

1365A - Matrix Game

Key Idea:

Vivek and Ashish can never claim cells in rows and columns which already have at least one cell claimed. So we need to look at the parity of minimum of the number of rows and columns which have no cells claimed initially.

Solution:

Let a be the number of rows which do not have any cell claimed in them initially and similarly b be the number of columns which do not have any cell claimed initially. Each time a player makes a move both a and b decrease by 1, since they only claim cells in rows and columns with no claimed cells.

If either one of a or b becomes 0, the player whose turn comes next loses the game. Since both a and b decrease by 1 after each move, $\min(a, b)$ becomes 0 first. So, if $\min(a, b)$ is odd, Ashish wins the game otherwise Vivek wins.

Time complexity: $O(n \cdot m)$

```
#include
using namespace std;

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define endl "\n"
#define int long long

const int N = 51;

int n, m;
int a[N][N];

int32_t main()
{
 IOS;
 int t;
 cin >> t;
 while(t--)
 {
  cin >> n >> m;
  set< int > r, c;
  for(int i = 1; i <= n; i++)
  {
   for(int j = 1; j <= m; j++)
   {
    cin >> a[i][j];
    if(a[i][j] == 1)
     r.insert(i), c.insert(j);
   }
  }
  int mn = min(n — r.size(), m — c.size());
  if(mn % 2)
   cout << "Ashish" << endl;
  else
   cout << "Vivek" << endl;
 }
 return 0;
}
```

Codeforces Round #648 (Div. 2) 1365B Проблематичная сортировка

B. Проблематичная сортировка

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

У Ashish есть n элементов, расположенных по порядку.

Каждый элемент задается двумя целыми числами a_i — значение элемента и b_i — тип элемента (есть только два возможных типа: 0 и 1). Он хочет отсортировать элементы в порядке неубывания a_i.

Он может совершать следующую операцию произвольное число раз:

- Выбрать любые два таких элемента i и j, что $b_i \neq b_j$ и поменять их местами. Таким образом, он может за ход поменять местами два элемента разных типов.

Скажите ему, может ли он отсортировать массив в порядке неубывания a_i, используя описанные операции.

Входные данные
В первой строке записано одно целое число t $(1 \leq t \leq 100)$ — количество наборов входных данных.

В первой строке каждого набора входных данных записано одно целое число n $(1 \leq n \leq 500)$ — размеры массивов.

Во второй строке записаны n целых чисел a_i $(1 \leq a_i \leq 10^5)$ — значение i-го элемента.

В третьей строке записаны n целых чисел b_i $(b_i \in \{0, 1\})$ — тип i-го элемента.

Выходные данные
Для каждого набора входных данных, выведите «Yes» или «No» (без кавычек) в зависимости от того, возможно ли отсортировать массив в порядке неубывания значений используя описанные операции.

Вы можете выводить каждый символ в любом регистре (верхнем или нижнем).

Пример

входные данные
5 4 10 20 20 30 0 1 0 1 3 3 1 2 0 1 1 4 2 2 4 8 1 1 1 1 3 5 15 4 0 0 0 4 20 10 100 50 1 0 0 1

выходные данные
Yes Yes Yes No Yes

Примечание
В первом наборе входных данных: элементы уже находятся в отсортированном порядке.

Во втором наборе входных данных: Ashish сначала может поменять местами элементы на позициях 1 и 2, затем поменять местами элементы на позициях 2 и 3.

В четвертом наборе входных данных: Нельзя поменять местами никакие два элемента, так как нет пары i и j, что $b_i \neq b_j$. Таким образом, элементы не могут быть отсортированы.

В пятом наборе входных данных: Ashish может поменять местами элементы на позициях 3 и 4, а затем элементы на позициях 1 и 2.

1365B - Trouble Sort

Key Idea:

If there is at least one element of type 0 and at least one element of type 1, we can always sort the array.

Solution:

If all the elements are of the same type, we cannot swap any two elements. So, in this case, we just need to check if given elements are already in sorted order.

Otherwise, there is at least one element of type 0 and at least one element of type 1. In this case, it is possible to swap any two elements! We can swap elements of different types using only one operation. Suppose we want to swap two elements a and b of the same type. We can do it in 3 operations. Let c be an element of the type different from a and b. We can first swap a and c, then swap b and c and then swap a and c again. In doing so, c remains at its initial position and a, b are swapped. This is exactly how we swap two integers using a temporary variable. Since we can swap any two elements, it is always possible to sort the array in this case.

Time complexity: $O(n)$

```cpp
#include
using namespace std;

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define endl "\n"
#define int long long

const int N = 1e3 + 5;

int n;
int a[N], b[N];

int32_t main()
{
 IOS;
 int t;
 cin >> t;
 while(t--)
 {
  cin >> n;
  bool sorted = 1, have0 = 0, have1 = 0;
  for(int i = 1; i <= n; i++)
  {
   cin >> a[i];
   if(i >= 2 && a[i] < a[i - 1])
    sorted = 0;
  }
  for(int i = 1; i <= n; i++)
  {
   cin >> b[i];
   if(!b[i])
    have0 = 1;
   else
    have1 = 1;
  }
  if(have0 && have1)
   cout << "Yes" << endl;
  else if(sorted)
   cout << "Yes" << endl;
  else
   cout << "No" << endl;
 }
 return 0;
}
```

Codeforces Round #648 (Div. 2) 1365C Соответствия поворотом

C. Соответствия поворотом

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

После мистического исчезновения Ashish, каждый из его любимых учеников Ishika и Hriday, получил одну половину секретного сообщения. Эти сообщения могут быть описаны перестановками размера n. Назовем их a и b.

Напомним, что перестановка из n элементов это последовательность чисел $a_1, a_2, \ldots, a_n$, в которой каждое число от 1 до n встречается ровно один раз.

Сообщение может быть расшифровано из конфигурации перестановок a и b, в котором количество совпадающих пар элементов максимально. Пара элементов a_i и b_j называется совпадающей, если:

- $i = j$, таким образом, у них один и тот же индекс.
- $a_i = b_j$

Его ученикам разрешается совершать следующую операцию произвольное число раз:

- выбрать число k и циклически сдвинуть одну из перестановок влево или вправо k раз.

Циклический сдвиг перестановки c влево это операция, которая присваивает $c_1 := c_2, c_2 := c_3, \ldots, c_n := c_1$ одновременно. Аналогично, циклический сдвиг перестановки c вправо это операция, которая присваивает $c_1 := c_n, c_2 := c_1, \ldots, c_n := c_{n-1}$ одновременно.

Помогите Ishika и Hriday найти наибольшее возможное число совпадающих пар в данных перестановках после применения описанных операций несколько (возможно, ноль) раз.

Входные данные

В первой строке записано одно целое число n $(1 \le n \le 2 \cdot 10^5)$ — размеры массивов.

Во второй строке записаны n целых чисел a_1, a_2, ..., a_n $(1 \le a_i \le n)$ — элементы первой перестановки.

В третьей строке записаны n целых чисел b_1, b_2, ..., b_n $(1 \le b_i \le n)$ — элементы второй перестановки.

Выходные данные

Выведите наибольшее возможное число совпадающих пар в данных перестановках после применения описанных операций несколько (возможно, ноль) раз.

Примеры

входные данные
5 1 2 3 4 5 2 3 4 5 1
выходные данные
5

входные данные
5 5 4 3 2 1 1 2 3 4 5
выходные данные
1

входные данные
4 1 3 2 4 4 2 3 1
выходные данные

```
2
```

Примечание

В первом примере можно сдвинуть b направо на $k = 1$. Получившиеся перестановки будут $\{1, 2, 3, 4, 5\}$ и $\{1, 2, 3, 4, 5\}$.

Во втором примере не требуется совершать никаких операций. По всем возможным сдвигам a и b, число совпадающих пар не будет превышать 1.

В третьем примере можно сдвинуть b влево на $k = 1$. Получившиеся перестановки будут $\{1, 3, 2, 4\}$ и $\{2, 3, 1, 4\}$. Позиции 2 и 4 будут являться совпадающей парой. По всем возможным циклическим сдвигам a и b, количество совпадающих пар не будет превышать 2.

1365C - Rotation Matching

Key Idea:

We only need to perform shifts on one of the arrays. Moreover, all the shifts can be of the same type (right or left)!

Solution:

First of all, a left cyclic shift is the same as $n - 1$ right cyclic shifts and vice versa. So we only need to perform shifts of one type, say right.

Moreover, a right cyclic shift of b is the same as performing a left cyclic shift on a and vice versa. So we don't need to perform any shifts on b.

Now the problem reduces to finding the maximum number of matching pairs over all right cyclic shifts of a. Since n right cyclic shifts on a results in a again, there are only $n - 1$ right cyclic shifts possible.

Since both arrays are a permutation, each element in a would match with its corresponding equal element in b only for one of the shifts. For example, if a is $\{2, 3, 1\}$ and b is $\{3, 1, 2\}$, the number 3 in a would match with the number 3 in b only if one right cyclic shift is performed. So for each element in a we can find the number of right cyclic shifts after which it would match with its corresponding equal element in b. If $a_i = b_j$, then a_i would match with b_j after $k = j - i$ right cyclic shifts. If $j - i < 0$, then a_i would with b_j after $n - j + i$ shifts.

Now for each shift, we can find the number of matching pairs and take the maximum.

Time complexity: $O(n)$ or $O(n \cdot log(n))$ if you use a map.

```cpp
#include
using namespace std;

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define endl "\n"
#define int long long

const int N = 2e5 + 5;

int n;
int a[N], b[N], pos[N];
map< int, int > offset;

int32_t main()
{
 IOS;
 cin >> n;
 for(int i = 1; i <= n; i++)
 {
  cin >> a[i];
  pos[a[i]] = i;
 }
 for(int i = 1; i <= n; i++)
  cin >> b[i];
 for(int i = 1; i <= n; i++)
 {
  int cur = pos[b[i]] - i;
  if(cur < 0)
   cur += n;
  offset[cur]++;
 }
 int ans = 0;
 for(auto &it:offset)
  ans = max(ans, it.second);
 cout << ans;
 return 0;
}
```

Educational Codeforces Round 89 (рейтинговый для Див. 2)
1366A Лопаты и мечи

A. Лопаты и мечи

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Поликарп играет в известную компьютерную игру (мы не хотим упоминать ее название). В этой игре он может создавать инструменты двух видов — лопаты и мечи. На создание лопаты Поликарп тратит две палки и один алмаз; на создание меча Поликарп тратит два алмаза и одну палку.

Каждый инструмент может быть продан за один изумруд. Как много изумрудов может заработать Поликарп, если у него есть a палок и b алмазов?

Входные данные
Первая строка содержит число t ($1 \le t \le 1000$) — количество наборов входных данных.

Единственная строка каждого набора входных данных содержит два числа a и b ($0 \le a, b \le 10^9$) — количество палок и алмазов соответственно.

Выходные данные
На каждый набор входных данных выведите число — максимальное количество изумрудов, которое может заработать Поликарп.

Пример

входные данные
4 4 4 1000000000 0 7 15 8 7
выходные данные
2 0 7 5

Примечание
В первом наборе входных данных Поликарп может заработать два изумруда следующим образом: создать один меч и одну лопату.

Во втором наборе входных данных у Поликарпа нет алмазов, а значит, он не сможет ничего создать.

1366A - Shovels and Swords

There are three constraints on the number of emeralds:

1. the number of emeralds can't be greater than a;
2. the number of emeralds can't be greater than b;
3. the number of emeralds can't be greater than $\frac{a+b}{3}$.

So the answer is $\min(a, b, \frac{a+b}{3})$.

```
for _ in range(int(input())):
    l, r = map(int, input().split())
    print(min(l, r, (l + r) // 3))
```

Educational Codeforces Round 89 (рейтинговый для Див. 2)
1366B Перемешивание

B. Перемешивание

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Вам задан массив, состоящий из n чисел a_1, a_2, ..., a_n. Изначально $a_x = 1$, а остальные элементы равны 0.

Вы выполняете m операций. Во время i-й операции вы выбираете два индекса c и d таких, что $l_i \le c, d \le r_i$, и меняете местами a_c и a_d.

Посчитайте количество индексов k таких, что существуют возможность выбрать операции так, что в конце $a_k = 1$.

Входные данные
Первая строка содержит число t ($1 \le t \le 100$) — количество наборов входных данных. Затем следует описание каждого из t наборов входных данных.

Первая строка каждого набора входных данных содержит три целых числа n, x и m ($1 \le n \le 10^9$; $1 \le m \le 100$; $1 \le x \le n$).

Каждая из следующих m строк содержит описание операций; а именно — в i-й строке содержится два целых числа l_i и r_i ($1 \le l_i \le r_i \le n$).

Выходные данные
На каждый набор входных данных выведите одно число — количество индексов k таких, что существуют возможность выбрать операции так, что в конце $a_k = 1$.

Пример

входные данные
3 6 4 3 1 6 2 3 5 5 4 1 2 2 4 1 2 3 3 2 2 3 1 2

выходные данные
6 2 3

Примечание
В первом наборе входных данных условие $a_k = 1$ выполняется для любого k. Для этого, можно выполнить следующие операции:

1. поменять местами a_k и a_4;
2. поменять местами a_2 и a_2;
3. поменять местами a_5 и a_5.

Во втором наборе входных данных подходят только индексы $k = 1$ и $k = 2$. Для выполнения $a_1 = 1$, нужно поменять местами a_1 и a_1 во второй операции. Для выполнения $a_2 = 1$, нужно поменять местами a_1 и a_2 во второй операции.

1366B - Shuffle

Let's consider how the set of possible indices where the 1 can be changes. Initially, only one index is correct — x. After performing an operation l, r such that $x < l$ or $x > r$ this set does not change. But after performing an operation l, r such that $l \le x \le r$ we should insert the elements $\{l, l+1, l+2, \ldots, r-1, r\}$ into this set, if they are not present.

Now consider how the set $\{L, L+1, L+2, \ldots, R-1, R\}$ changes. If segments $[l, r]$ and $[L, R]$ do not share any indices, there are no changes — but if they do, the set turns into $\{\min(l, L), \min(l, L)+1, \min(l, L)+2, \ldots, \max(r, R)-1, \max(r, R)\}$.

So the set of reachable indices is always a segment of numbers, and to process an operation, we should check whether the segment from operation intersects with the segment of indices we have — and if it is true, unite them.

```
for _ in range(int(input())):
    n, x, m = map(int, input().split())
    l, r = x, x
    for _ in range(m):
        L, R = map(int, input().split())
        if max(l, L) <= min(r, R):
            l = min(l, L)
            r = max(r, R)

    print(r - l + 1)
```

Educational Codeforces Round 89 (рейтинговый для Див. 2)
1366C Палиндромные пути

C. Палиндромные пути

ограничение по времени на тест: 1.5 секунд
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Вам задана матрица из n строк (пронумерованных от 1 до n) и m столбцов (пронумерованных от 1 до m). Обозначим за $a_{i,j}$ число в клетке на пересечении i-й строки и j-го столбца, каждое число либо 0, либо 1.

Изначально в ячейке $(1,1)$ находится фишка, которая будет перемещена в ячейку (n,m) при помощи последовательности шагов. На каждом шаге фишка перемещается либо в ячейку справа от текущей, либо в ячейку снизу (если фишка сейчас в ячейке (x,y), ее можно переместить либо в $(x+1,y)$, либо в $(x,y+1)$). Фишка не может покидать матрицу.

Рассмотрим все пути фишки из ячейки $(1,1)$ в ячейку (n,m). Назовем путь *палиндромным*, если число в первой ячейке пути равно числу в последней ячейке пути, число во второй ячейке равно числу в предпоследней ячейке, и так далее.

Ваша цель — заменить минимальное количество элементов матрицы так, чтобы **все** пути стали *палиндромными*.

Входные данные

В первой строке задано одно целое число t ($1 \le t \le 200$) — количество наборов входных данных.

В первой строке каждого набора заданы два целых числа n и m ($2 \le n, m \le 30$) — размеры матрицы.

Затем следуют n строк, i-я из которых содержит m целых чисел $a_{i,1}$, $a_{i,2}$, ..., $a_{i,m}$ ($0 \le a_{i,j} \le 1$).

Выходные данные

Для каждого набора входных данных выведите одно целое число — минимальное количество элементов, которое надо заменить.

Пример

входные данные
4 2 2 1 1 0 1 2 3 1 1 0 1 0 0 3 7 1 0 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 0 1 3 5 1 0 1 0 0 1 1 1 1 0 0 0 1 0 0

выходные данные
0 3 4 4

Примечание

Итоговые матрицы в первых трех примерах:

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

1366C - Palindromic Paths

Let's group the cells by their distance from the starting point: the group 0 consists of a single cell $(1, 1)$; the group 1 consists of the cells $(1, 2)$ and $(2, 1)$, and so on. In total, there are $n + m - 1$ groups.

Let's analyze the groups k and $n + m - 2 - k$. There are two cases:

- if $k = 0$ or $n + m - 2 - k = 0$, then we are looking at the starting cell and the ending cell, and their contents should be equal;
- otherwise, suppose two cells (x, y) and $(x + 1, y - 1)$ belong to the same group. We can easily prove that the contents of these two cells should be equal (for example, by analyzing two paths that go through cell $(x + 1, y)$ and coincide after this cell, but one goes to $(x + 1, y)$ from (x, y), and another — from $(x + 1, y - 1)$) — and, using induction, we can prove that the contents of all cells in a group should be equal. And since the paths should be palindromic, the contents of the group k should be equal to the contents of the group $n + m - 2 - k$.

So, in each pair of groups, we should calculate the number of 1's and 0's, and choose which of them to change. Note that if $n + m$ is even, the central group has no pair, so it should not be modified.

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

using namespace std;

void solve()
{
 int n, m;
 cin >> n >> m;
 vector<vector<int> > a(n, vector<int>(m));
 for(int i = 0; i < n; i++)
  for(int j = 0; j < m; j++)
   cin >> a[i][j];
 vector<vector<int> > cnt(n + m - 1, vector<int>(2));
 for(int i = 0; i < n; i++)
  for(int j = 0; j < m; j++)
   cnt[i + j][a[i][j]]++;
 int ans = 0;
 for(int i = 0; i <= n + m - 2; i++)
 {
  int j = n + m - 2 - i;
  if(i <= j) continue;
  ans += min(cnt[i][0] + cnt[j][0], cnt[i][1] + cnt[j][1]);
 }
 cout << ans << endl;
}

int main() {
 int t;
 cin >> t;
 for(int i = 0; i < t; i++)
  solve();
}
```

Codeforces Round #652 (Div. 2) 1369A ПокупатеЛи

A. ПокупатеЛи

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Ли собирается украсить свой дом к вечеринке с помощью нескольких правильных многоугольников...

Ли считает правильный (выпуклый) n-угольник *красивым* тогда и только тогда, когда он может повернуть многоугольник таким образом, чтобы хотя бы одна из его сторон стала параллельна оси OX и хотя бы одна из его сторон стала параллельна оси OY одновременно.

Напомним, что правильный n-угольник — это выпуклый многоугольник из n вершин такой, что все его стороны и углы равны.

Ли пришел в магазин: в магазине продаются t правильных многоугольников. Для каждого из них выведите `YES`, если многоугольник красивый, или `NO` в противном случае.

Входные данные

В первой строке задано одно целое число t ($1 \le t \le 10^4$) — количество многоугольников в магазине.

В каждой из следующих t строк задано по одному целому числу n_i ($3 \le n_i \le 10^9$): то есть i-й многоугольник является правильным n_i-угольником.

Выходные данные

Для каждого многоугольника в магазине, выведите `YES`, если он красивый, или `NO` в противном случае (регистр букв не важен).

Пример

входные данные
4 3 4 12 1000000000
выходные данные
NO YES YES YES

Примечание

В примере, в магазине продаются 4 многоугольника. Несложно определить, что равносторонний треугольник (правильный 3-угольник) не является красивым, квадрат (правильный 4-угольник) является красивым, и правильный 12-угольник (изображен ниже) также является красивым.

1369A - FashionabLee

$\mathcal{Complete\ Proof}$:

Proof by contradiction :

One can prove that if two edges in a regular polygon make a $x < 180$ degrees angle, then for each edge a there exist two another edges b and c such that a and b make a x degrees angle as well as a and c. (proof is left as an exercise for the reader)

Consider a rotation such that an edge a is parallel to OX-axis and an edge b is parallel to OY-axis, then $a \perp b$ (a and b are perpendicular, i. e. the angle between them is 90 degrees), we can see that there exist a third edge c such that it's also parallel to OX-axis and a forth edge d such that it's also parallel to OY-axis, so $a \perp d$ and $b \perp c$ and $c \perp d$.

Our polygon is regular so all the angles are equal, so that the number of angles between a and b is equal to the number of

angles between b and c and so on, also we know that a regular n-sided convex polygon has n angles, so n is divisible by 4, contradiction!

```
t = int(input())
for testcase in range(t):
    n = int(input())
    if(n%4 == 0) :
        print("Yes")
    else :
        print("No")
```

```
#include <iostream>

using namespace std;

int main(){
    int t;
    cin >> t;
    while(t--){
        int n;
        cin >> n;
        if(n % 4 == 0){
            cout << "YES\n";
        }
        else cout << "NO\n";
    }
}
```

Codeforces Round #652 (Div. 2) 1369B ОчистителЛи

B. ОчистителЛи

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Ли убирался у себя в дома перед вечеринкой, когда нашел под ковром «грязную» строку. Теперь он хочет очистить строку, но сделать это стильно...

Строка s, которую нашел Ли, является двоичной строкой длины n (т. е. строка состоит только из символов `0` и `1`).

За один шаг, он может выбрать два последовательных символа s_i и s_{i+1} и, если символ s_i равен `1` и s_{i+1} равен `0`, он может удалить **ровно один из символов** (Ли может выбрать какой удалить, но не может удалить оба символа одновременно). После удаления строка сжимается.

Ли может сделать произвольное количество шагов (возможно, ни одного шага) и он хочет сделать строку s как можно более *чистой*. Он считает, что из двух различных строк x и y **более короткая** строка чище, а если они равны по длине, то чище та, что *лексикографически меньше*.

Сейчас же вам необходимо ответить на t наборов входных данных: для i-го набора, выведите самую чистую строку, которую может получить Ли за произвольное количество шагов.

Небольшое напоминание: если у нас есть две строки x и y равной длины, то x лексикографически меньше чем y, если существует такая позиция i, что $x_1 = y_1$, $x_2 = y_2$,..., $x_{i-1} = y_{i-1}$ и $x_i < y_i$.

Входные данные

В первой строке задано одно целое число t ($1 \le t \le 10^4$) — количество наборов входных данных.

В следующих $2t$ строках заданы сами наборы входных данных — по одному на две строки.

В первой строке каждого набора задано одно целое число n ($1 \le n \le 10^5$) — длина строки s.

Во второй строке задана сама бинарная строка s. Строка s — это строка длины n, состоящая только из нулей и единиц.

Гарантируется, что сумма n по всем наборам не превосходит 10^5.

Выходные данные

Выведите t ответов — по одному на набор входных данных.

Ответом на i-й набор является самая чистая строка, которую может получить Ли за произвольное (возможно, нулевое) количество шагов.

Пример

входные данные
5 10 0001111111 4 0101 8 11001101 10 1110000000 1 1

выходные данные
0001111111 001 01 0 1

Примечание

В первом наборе входных данных, Ли не может сделать ни одного шага.

Во втором наборе, Ли должен удалить s_2.

В третьем наборе, Ли может, например, выполнить следующие шаги: 11001101 → 1100101 → 110101 → 10101 → 1101 → 101 → 01.

1369B - AccurateLee

Complete Proof :

Realize that the answer is always non-descending, and we can't perform any operations on non-descending strings.

First we know that we can't perform any operations on non-descending strings, so the answer to a non-descending string is itself. From now we consider our string s to not to be non-descending. (i.e. there exist index i such that $1 \leq i \leq n-1$ and $s_i > s_{i+1}$)

Also realize that the remaining string wont be empty, so "0" is the cleanest possible answer, but we can't reach it probable.

Now realize that leading zeroes and trailing ones can't be present in any operation. So they have to be in the answer, erase them from s, and add them to the answer for the modified s. From now we know that the string s has no leading zeroes and/or trailing ones, and is not non-descending, so it starts with 1 and ends with 0. (why?)

With some small paperwork, we will realize that the answer to a string that starts with 1 and ends with 0 is a single 0(proof is bellow). So if the string s is non-descending and it has x leading zeroes and y trailing ones(x and y can be equal to zero), then the answer is $\underbrace{0\,0\ldots0}_{x}\,0\,\underbrace{1\,1\ldots1}_{y}$ (its $x+1$ zeroes and y ones in order)

The Small Paperwork :

We will randomly perform operations until we can't do any more or the string's length is equal to 2, but we wont erase the first 1 and the last 0, we want to prove that the remaining string's length is exactly 2 after the process ends, proof by contradiction :

So it's length is at least 3, so we have at least two 1 or at least two 0. If we had two or more 0 then the string $[s_1\ s_2\ \ldots\ s_{n-1}]$ will not be non-descending(so we can perform more operations as we proved in `STAR`, but the process have ended, contradiction!) and if we had two or more 1 then the string $[s_2\ s_3\ \ldots\ s_n]$ will not be non-descending. So the length of the remaining string is exactly 2, and we haven't erased first '1' and last '0', so the string is equal to "10", now erase '1' to get the cleanest string.

Sorry if the proof seems too long and hard, i wanted to explain it accurately. ^-^

```
t = int(input())
for testcase in range(t):
    n = int(input())
    s = input()
    lef, rig, sw = 1, 1, 0
    for i in range(n-1):
        if(s[i] > s[i+1]):
            sw = 1
            break
    if(sw == 0):
        print(s)
        continue
    for i in range(n):
        if (s[i] == '1'):
            lef = i
            break
    for i in range(n-1, 0, -1):
        if (s[i] == '0'):
            rig = i
            break
    st = s[:lef] + '0' + s[rig+1:]
    print(st)
```

```cpp
#include <iostream>
#include <string>

using namespace std;

int main(){

    int t;
    cin >> t;
    while(t--){
        int n;
        cin >> n;
        string s;
        cin >> s;
        int sw = 1;
        for(int i = 1; i < s.size(); i++){
            if(s[i] < s[i-1])sw = 0;
        }
        if(sw){
            cout << s << '\n';
            continue;
        }
        string ans;
        for(int i = 0; i < s.size(); i++){
            if(s[i] == '1')break;
            ans.push_back('0');
        }
        ans.push_back('0');
        for(int i = s.size()-1; i >= 0; i--){
            if(s[i] == '0')break;
            ans.push_back('1');
        }
        cout << ans << '\n';
    }
}
```

Codeforces Round #652 (Div. 2) 1369C ПриятеЛи

C. ПриятеЛи

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Ли наконец стал мастером на Codeforces, и потому решил сходить за подарками своим друзьям. Он приобрел n целых чисел, и теперь настало время распределить их между друзьями...

У Ли есть n целых чисел $a_1, a_2, \ldots, a_n$ в своем рюкзаке, а также у него k друзей. Ли хочет распределить **все** целые числа из рюкзака между друзьями так, чтобы i-му другу досталось ровно w_i чисел и каждое число досталось ровно одному другу.

Назовем *уровнем счастья* друга сумму максимального и минимального числа, которое он получит.

Ли хочет сделать друзей как можно более счастливыми, другими словами, он хочет максимизировать суммарный уровень счастья друзей. Конечно же, Ли просит вас помочь ему посчитать этот максимальный суммарный уровень счастья.

Входные данные

В первой строке задано единственное число t ($1 \le t \le 10^4$) — количество наборов входных данных.

В следующих $3t$ строках заданы сами наборы — по одному на три строки.

В первой строке каждого набора входных данных заданы два целых числа n и k ($1 \le n \le 2 \cdot 10^5$; $1 \le k \le n$) — количество целых чисел в рюкзаке Ли и количество его друзей.

Во второй строке каждого набора заданы n целых чисел $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$) — сами числа в рюкзаке.

В третьей строке заданы k целых чисел $w_1, w_2, \ldots, w_k$ ($1 \le w_i \le n$; $w_1 + w_2 + \ldots + w_k = n$) — количество чисел, которое Ли собирается дать каждому другу.

Гарантируется, что сумма n по всем наборам не превосходит $2 \cdot 10^5$.

Выходные данные

Для каждого набора входных данных выведите по одному числу — максимальный суммарный уровень счастья, который сможет достигнуть Ли.

Пример

входные данные
3 4 2 1 13 7 17 1 3 6 2 10 10 10 10 11 11 3 3 4 4 1000000000 1000000000 1000000000 1000000000 1 1 1 1
выходные данные
48 42 8000000000

Примечание

В первом наборе входных данных, Ли нужно отдать наибольшее число первому другу (его уровень счастья будет равен $17 + 17$) и остальные числа — второму (его уровень счастья будет равен $13 + 1$).

В втором наборе, Ли нужно отдать $\{10, 10, 11\}$ и первому и второму другу, тогда суммарный уровень счастья будет равен $(11 + 10) + (11 + 10)$

В третьем наборе, у Ли четыре друга и четыре числа. Не важно, как он распределит числа между ними.

1369C - RationalLee

Complete Proof :

First if $w_i = 1$ for some i, then assign the greatest element to i-th friends, it's always better obviously.

Sort the elements in non-descending order and sort the friends in non-ascending order of w_i. Define v_i the set of indices of elements to give to i-th friend. Also define l_i the minimum element to give to i-th friend and r_i the maximum element to give to i-th friend, and define $m = \max\limits_{1 \le i \le k} w_i$.

Now it's easy to see that the first element of a(the smallest element) is always equal to l_i for some i, Indeed it's better to have the rest of v_i equal to a small number except one of them, which should be equal to a very large number. So we can greedily assign a_1, a_2 ... $a_{w_i - 1}$ to v_i, and then assign a_n to it, also it's better to have $w_i = m$. One can prove that there exist an optimal distributing such that the set($\{a_1, a_2 \dots a_{m-1}, a_n\}$) is equal to one of v_i-s(proof is blow). So add $a_1 + a_n$ to the answer for remaining elements of a(excluding the set) and remaining friends(excluding one of the friends with maximum w_i) and so, it will be optimal.

Look at an optimal distributing (which maximizes sum of happiness), first element of a is in v_i for example, we want to prove that in at least one of the optimal distributings $w_i - 1$ smallest elements of a are in v_i (including the first element), proof by contradiction:

If at least one of the smallest $w_i - 1$ elements is not in v_i, then call the smallest of them x, lets say it's in v_j, now add x to v_i(and erase it from v_j), instead add a greater number than x in v_i to v_j (it's at least two of them, and one of them is r_i, so there exist another one, erase it from v_i and add it to v_j), it's easy to see that sum of happiness won't decrease that way, continue the process until all $w_i - 1$ smallest elements are in v_i, so we have an optimal answer which has all $w_i - 1$ smallest elements in v_i, contradiction!

As we proved above, we have an optimal distributing such that all $w_i - 1$ smallest elements are in v_i(for some i), now we want to prove that the greatest element is in v_i in at least one of the optimal distributings, again proof by contradiction.

Lets say it's not that way, so look at an optimal distributing such that first $w_i - 1$ elements are in v_i and r_i is not equal to the greatest element(for some i), if there exist such j that $r_i < l_j$, then swap r_i and l_j, the resulting distributing has the same happiness, continue it until no such j exist, now lets say the greatest element of a is in v_j for some j, also we know that r_j is equal to the greatest element of a and $l_j \le r_i$(if $r_i < l_j$ then the process of swapping is not finished, which is contradiction). So now we can swap r_i and r_j, again the resulting distributing has happiness greater than or equal to the happiness of the optimal distributing(the one we chose in the beginning), and so, its also an optimal distributing, and r_i is equal to the greatest element, we have found an optimal distributing such that first $w_i - 1$ elements of a and a_n are in v_i(for some i), contradiction!

Now we have proved that there exist an optimal distributing such that first $w_i - 1$ elements of a and a_n are in v_i(for some i), call such optimal distributing `STAR`, and now the only remaining part is to prove that there exist an optimal distributing such that first $m - 1$ elements of a and a_n are in v_i(for some i). See the whole algorithm, its like "we choose a permutation of friends then we do that greedy assignment to them one by one from left to right", now we want to prove that there exist an optimal distributing such that it's `STAR` and it's permutation is sorted in non-descending order of w, call them `GOOD` distributings. Again, proof by contradiction :

Choose a distributing such that it's a `STAR`, it's permutation(called p) is not sorted in non-descending order of w(otherwise it's a `GOOD` distributing, contradiction!), so there exist an i such that $w_{p_i} > w_{p_{i+1}}$, now swap them(i. e. swap p_i and p_{i+1} and then do the same greedy assignment using the modified permutation of friends), it's easy to see that happiness of friends after $i + 1$ in permutation p wont change, also happiness of friends before i in the permutation wont change as well.

Now look at the happiness of p_i and p_{i+1}, you can realize that sum of happiness will increase.

You really don't need to prove it like that, it's not time friendly at all. ^-^

```
t = int(input())
for tc in range(t):
    n, k = map(int, input().split())
    a = list(map(int, input().split()))
    w = list(map(int, input().split()))
    a.sort(reverse = True)
    w.sort()
    ii, l, r = k, 0, n-1
    ans = 0
    for i in range(k):
        if(w[i] > 1):
            ii = i
            break
        ans = ans+a[l]*2
        l = l+1
    for u in range(k-1, ii-1, -1):
        i = w[u]
        ans = ans + a[l] + a[r]
        r = r-i+1
        l = l+1
    print(ans)
```

```cpp
#include <bits/stdc++.h>
#define ll long long
#define fr first
#define sc second
#define int ll

using namespace std;
const int MN = 2e5+7;

vector<int> v[MN];

signed main(){
    ios::sync_with_stdio(false);
    cin.tie();
    cout.tie();

    int t;
    cin >> t;
    while(t--){
        int n, k;
        cin >> n >> k;
        for(int i = 0; i <= n; i++)v[i].clear();
        ll a[n], w[k];
        for(int i = 0; i < n; i++){
            cin >> a[i];
        }
        for(int i = 0; i < k; i++){
            cin >> w[i];
        }
        sort(w, w+k);
        sort(a, a+n);
        for(int i = 0; i < k/2; i++)swap(w[i], w[k-i-1]);
        int po = 0;
        for(int i = 0; i < n-k; i++){
            while(w[po] == v[po].size()+1)po++;
            v[po].push_back(a[i]);
        }
        ll ans = 0;
        int qf = 1;
        for(int i = 0; i < k; i++){
            ans += a[n-i-1];
            if(v[i].size())ans += v[i][0];
            else ans += a[n-qf], qf++;
        }

        cout << ans << '\n';
    }
}
```

Codeforces Round #651 (Div. 2) 1370A Максимальный НОД

A. Максимальный НОД

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Рассмотрим все целые числа в промежутке от 1 до n (включительно).

По всем парам **различных** целых чисел из этого промежутка, найдите максимальное возможное значение наибольшего общего делителя чисел в паре. Более формально, найдите максимальное значение $\gcd(a, b)$ по всем $1 \leq a < b \leq n$.

Наибольшим общим делителем $\gcd(a, b)$ пары положительных целых чисел a и b называется наибольшее целое число, являющееся делителем числа a и делителем числа b.

Входные данные
В первой строке находится единственное целое число t ($1 \leq t \leq 100$) — количество наборов входных данных. Описание наборов входных данных следует.

В единственной строке описания каждого набора входных данных находится единственное целое число n ($2 \leq n \leq 10^6$).

Выходные данные
Для каждого набора входных данных, выведите максимальное значение $\gcd(a, b)$ по всем $1 \leq a < b \leq n$.

Пример

входные данные
2 3 5
выходные данные
1 2

Примечание
В первом наборе входных данных $\gcd(1, 2) = \gcd(2, 3) = \gcd(1, 3) = 1$.

Во втором наборе входных данных 2 является максимальным возможным значением, соответствующим $\gcd(2, 4)$.

1370A - Maximum GCD

Key Idea:

Answer for any $n \geq 2$ is equal to $\lfloor \frac{n}{2} \rfloor$.

Solution:

Let the maximum gcd be equal to g. Since the two numbers in a pair are distinct, one of them must be $> g$ and both of them must be divisible by g. The smallest multiple of g, greater than g, is $2 \cdot g$. Since each number in the pair must be $\leq n$, we must have $2 \cdot g \leq n$, or $g \leq \lfloor \frac{n}{2} \rfloor$. We can achieve $g = \lfloor \frac{n}{2} \rfloor$, by choosing $\lfloor \frac{n}{2} \rfloor$ and $2 \cdot \lfloor \frac{n}{2} \rfloor$.

Time Complexity: $O(1)$

```cpp
#include < bits/stdc++.h >
using namespace std;

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define endl "\n"
#define int long long

const int N = 1e5 + 5;

int32_t main()
{
 IOS;
 int t;
 cin >> t;
 while(t--)
 {
  int n;
  cin >> n;
  cout << n / 2 << endl;
 }
 return 0;
}
```

Codeforces Round #651 (Div. 2) 1370B Сжатие массива и НОД

B. Сжатие массива и НОД

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

У Ashish есть массив a, состоящий из $2n$ положительных целых чисел. Он хочет сжать массив a в массив b размера $n-1$. Чтобы это сделать, он сначала выбирает ровно 2 (любые два) элемента массива a и удаляет их из массива. После этого он выполняет следующую операцию, пока массив a не пустой:

- удалить любые два элемента из массива a и добавить их сумму в массив b.

Получившийся массив b должен удовлетворять одному условию. Наибольший общий делитель ($\gcd$) всех элементов массива должен быть больше 1.

Напомним, что наибольший общий делитель ($\gcd$) массива положительных целых чисел равен наибольшему целому числу, которое является делителем всех элементов массива.

Можно доказать, что всегда можно таким образом сжать массив a в массив b размера $n-1$, так что $gcd(b_1, b_2 \ldots, b_{n-1}) > 1$.

Помогите Ashish найти способ это сделать.

Входные данные
В первой строке находится единственное целое число t ($1 \le t \le 10$) — количество наборов входных данных. Описание наборов входных данных следует.

В первой строке описания каждого набора входных данных находится единственное целое число n ($2 \le n \le 1000$).

Во второй строке описания каждого набора входных данных находится $2n$ целых чисел $a_1, a_2, \ldots, a_{2n}$ ($1 \le a_i \le 1000$) — элементы массива a.

Выходные данные
Для каждого набора входных данных, выведите $n-1$ строку — выполненные операции, чтобы сжать массив a в массив b. **Изначальное удаление двух элементов не является операцией и про это действие не нужно ничего выводить.**

В i-й из этих строк должно находиться два целых числа, индексы (нумерация с 1) двух элементов массива a, которые используются в i-й операции. Все $2n-2$ выведенных индекса должны быть различными целыми числами от 1 до $2n$.

Вам не нужно выводить индексы двух изначально удаленных элементов из массива a.

Если есть несколько возможных ответов, вы можете найти любой.

Пример

входные данные
3 3 1 2 3 4 5 6 2 5 7 9 10 5 1 3 3 4 5 90 100 101 2 3

выходные данные
3 6 4 5 3 4 1 9 2 3 4 5 6 10

Примечание
В первом наборе входных данных $b = \{3+6, 4+5\} = \{9, 9\}$ и $\gcd(9, 9) = 9$.

Во втором наборе входных данных $b = \{9 + 10\} = \{19\}$ и $\gcd(19) = 19$.

В третьем наборе входных данных $b = \{1 + 2, 3 + 3, 4 + 5, 90 + 3\} = \{3, 6, 9, 93\}$ и $\gcd(3, 6, 9, 93) = 3$.

1370B - GCD Compression

Key Idea:

It is always possible to form $n - 1$ pairs of elements such that their gcd is divisible by 2.

Solution:

We can pair up the odd numbers and even numbers separately so that the sum of numbers in each pair is divisible by 2. Note that we can always form $n - 1$ pairs in the above manner because in the worst case, we would discard one odd number and one even number from a. If we discarded more than one even or odd numbers, we could instead form another pair with even sum.

Time Complexity: $O(n)$

```
#include < bits/stdc++.h >
using namespace std;

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define endl "\n"
#define int long long

const int N = 2e5 + 5;

int n;
int a[N];

int32_t main()
{
 IOS;
 int t;
 cin >> t;
 while(t--)
 {
  cin >> n;
  vector< int > even, odd;
  for(int i = 1; i <= 2 * n; i++)
  {
   cin >> a[i];
   if(a[i] % 2)
    odd.push_back(i);
   else
    even.push_back(i);
  }
  vector< pair< int, int > > ans;
  for(int i = 0; i + 1 < odd.size(); i += 2)
   ans.push_back({odd[i], odd[i + 1]});
  for(int i = 0; i + 1 < even.size(); i += 2)
   ans.push_back({even[i], even[i + 1]});
  for(int i = 0; i < n - 1; i++)
   cout << ans[i].first << " " << ans[i].second << endl;
 }
 return 0;
}
```

Codeforces Round #651 (Div. 2) 1370C Игра с числом

C. Игра с числом

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Ashishgup и FastestFinger играют в игру.

Они начинают с целого числа n и начинают делать ходы по очереди. На каждом ходу игрок может сделать **любой из** следующих двух ходов:

- Разделить n на один из его нечетных делителей, который больше чем 1.
- Вычесть 1 из n, если n больше чем 1.

Обратите внимание, что множество делителей числа включает само число.

Если игрок **не может сделать ход** он проигрывает игру.

Ashishgup ходит первым. Определите победителя игры, если оба игрока играют оптимально.

Входные данные

В первой строке находится единственное целое число t ($1 \le t \le 100$) — количество наборов входных данных. Описание наборов входных данных следует.

В единственной строке описания каждого набора входных данных находится единственное целое число n ($1 \le n \le 10^9$).

Выходные данные

Для каждого набора входных данных, выведите «`Ashishgup`», если он побеждает в игре и «`FastestFinger`» иначе (без кавычек).

Пример

входные данные
7 1 2 3 4 5 6 12
выходные данные
FastestFinger Ashishgup Ashishgup FastestFinger Ashishgup FastestFinger Ashishgup

Примечание

В первом наборе входных данных $n = 1$ и Ashishgup не может сделать ход. Он проигрывает.

Во втором наборе входных данных $n = 2$ и Ashishgup вычитает 1 на первом ходу. Теперь $n = 1$ и FastestFinger не может сделать ход, поэтому он проигрывает.

В третьем наборе входных данных $n = 3$ и Ashishgup делит на 3 на первом ходу. Теперь $n = 1$ и FastestFinger не может сделать ход, поэтому он проигрывает.

В последнем наборе входных данных $n = 12$ и Ashishgup делит на 3 на первом ходу. Теперь $n = 4$, FastestFinger может только вычесть 1 и Ashishgup получает число 3. Наконец, он побеждает после деления этого числа на 3.

[1370C - Number Game]

Key Idea:

`FastestFinger` wins for $n = 1$, $n = 2^x$ where ($x > 1$) and $n = 2 \cdot p$ where p is a prime ≥ 3 else `Ashishgup` wins.

Solution:

Let's analyse the problem for the following 3 cases:

- Case 1: n is odd
  Here `Ashishgup` can divide n by itself, since it is odd and hence $\frac{n}{n} = 1$, and `FastestFinger` loses. Here $n = 1$ is an exception.

- Case 2: n is even and has no odd divisors greater than 1
  Here n is of the form 2^x. As n has no odd divisors greater than 1, `Ashishgup` is forced to subtract it by 1 making n odd. So if $x > 1$, `FastestFinger` wins. For $x = 1$, $n - 1$ is equal to 1, so `Ashishgup` wins.

- Case 3: n is even and has odd divisors
  If n is divisible by 4 then `Ashishgup` can divide n by its largest odd factor after which n becomes of the form 2^x where $x > 1$, so `Ashishgup` wins.

  Otherwise n must be of the form $2 \cdot p$, where p is odd. If p is prime, `Ashishgup` loses since he can either reduce n by 1 or divide it by p both of which would be losing for him. If p is not prime then p must be of the form $p_1 \cdot p_2$ where p_1 is prime and p_2 is any odd number > 1. `Ashishgup` can win by dividing n by p_2.

```
#include< bits/stdc++.h >
using namespace std;

const int N = 50000;

void player_1(){
 cout << "Ashishgup" << endl;
}

void player_2(){
 cout << "FastestFinger" << endl;
}

bool check_prime(int n){
 for(int i = 2; i < min(N, n); i++)
  if(n % i == 0)
   return 0;
 return 1;
}

int main(){
 int tc;
 cin >> tc;
 while(tc--){
  int n;
  cin >> n;
  bool lose = (n == 1);
  if(n > 2 && n % 2 == 0){
   if((n & (n — 1)) == 0)
    lose = 1;
   else if(n % 4 != 0 && check_prime(n / 2))
    lose = 1;
  }
  if(lose)
   player_2();
  else player_1();
 }
}
```

Codeforces Round #654 (Div. 2) 1371A Волшебные палочки

A. Волшебные палочки

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

У пингвина Rocher есть n палочек — у него ровно одна палочка длины i для всех $1 \le i \le n$.

Он может соединять некоторые палочки. Если он соединяет две палочки, которые имеют длины a и b, он получает одну палочку длины $a + b$. Две палочки, которые были использованы в этой операции, пропадают из его множества. Новая соединенная палочка появляется в его множестве и может быть использована в следующих соединениях.

Он хочет создать максимальное количество палочек, имеющих одинаковую длину. Не обязательно при этом, чтобы в итоге все палочки имели одинаковую длину — некоторые палочки могут иметь другие длины. Какое максимальное количество палочек одинаковой длины он может получить в итоге?

Входные данные

Каждый тест состоит из нескольких наборов входных данных. В первой строке находится единственное целое число t ($1 \le t \le 1000$) — количество наборов входных данных. Следующие t строк содержат описания наборов входных данных.

Для каждого набора входных данных в единственной строке находится единственное целое число n ($1 \le n \le 10^9$).

Выходные данные

Для каждого набора входных данных выведите единственное целое число — ответ на задачу.

Пример

входные данные
4 1 2 3 4
выходные данные
1 1 2 2

Примечание

В третьем наборе входных данных он может соединить две палочки длины 1 и 2 и он получит одну палочку длины 3. Так он получит две палочки одинаковой длины 3.

В четвертом наборе входных данных он может соединить две палочки длины 1 и 3 и он получит одну палочку длины 4. После этого у него будет три палочки, имеющие длины $\{2, 4, 4\}$. Две палочки имеют одинаковую длину и она палочка будет иметь другую длину.

1371A - Magical Sticks

Output $\lceil \frac{n}{2} \rceil$.

- When n is even, we can create $1 + n = 2 + (n - 1) = 3 + (n - 2) = \ldots$
- When n is odd, we can create $n = 1 + (n - 1) = 2 + (n - 2) = \ldots$

Initially, there are only 1 stick which has length $i (1 \le i \le n)$. If we connect 2 sticks s_1 and s_2, after that, there is a stick which has a different length from s_1 and s_2. Then, we can create at most $1 + \lfloor \frac{n-1}{2} \rfloor$ sticks that have the same length. The value is equal to $\lceil \frac{n}{2} \rceil$.

Total complexity: $O(1)$

```c
#include<stdio.h>

int main(){
  long long n,t;
  scanf("%lld",&t);
  while(t>0){
    t--;
    scanf("%lld",&n);
    if(n%2){printf("%lld\n",(n/2)+1);}
    else{printf("%lld\n",n/2);}
  }
  return 0;
}
```

Codeforces Round #654 (Div. 2) 1371B Магический календарь

B. Магический календарь

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Профессиональный едок Алиса ставит в расписание в своем магическом календаре тренировки для подготовки к очередному контесту по поеданию. Календарь необычен тем, что неделя не обязательно может состоять из 7 дней!

Более точно, она может выбрать целое число k, которое удовлетворяет $1 \leq k \leq r$ и выставить, что неделя состоит из k дней.

Алиса собирается закрасить некоторые n последовательных дней в календаре. В этом календаре недели соответсвуют строкам и идут подряд слева направо. Для последнего дня недели первая клетка следующей (снизу) строки соотвествует следующему дню.

Она хочет, чтобы **все закрашенные клетки были связны по сторонам**. Это означает, что для любых двух закрашенных клеток, должна существовать как минимум одна последовательность закрашенных клеток, начинающаяся в одной из этих клеток и заканчивающаяся в другой, что любые две соседние клетки в этой последовательности имеют общую сторону.

Алиса рассматривает форму закрашенных клеток. Две формы являются одинаковыми, если можно совместить их **только с помощью параллельных переносов, паралельных сторонам календаря (то есть перемещая вверх-вниз и вправо-влево)**.

Например, на картинке неделя состоит из 4 дней и Алиса закрашивает 5 последовательных дней. [1] и [2] имеют различную форму, но [1] и [3] имеют одинаковую форму.

Алисе интересно узнать **какое количество различных форм** существует **если она установит из скольки дней состоит неделя и выберет последовательные n дней и закрасит их в календаре, начиная в какой-то день недели**. Как уже было сказано до этого, она рассматривает только формы, где все закрашенные клетки связны по сторонам.

Входные данные

Каждый тест состоит из нескольких наборов входных данных. В первой строке находится единственное целое число t ($1 \leq t \leq 1000$) — количество наборов входных данных. Следующие t строк содержат описания набов входных данных.

Для каждого набора входных данных в единственной строке находится два целых числа n, r ($1 \leq n \leq 10^9, 1 \leq r \leq 10^9$).

Выходные данные

Для каждого набора входных данных выведите единственное целое число — ответ на задачу.

Обратите внимание, что ответ на некоторые наборы входных данных не влезает в 32-битный целочисленный тип, поэтому вы должны использовать как минимум 64-битный целочисленный тип вашего языка программирования.

Пример

входные данные
5 3 4 3 2 3 1 13 7 1010000 9999999
выходные данные
4 3 1 28 510049495001

Примечание

В первом наборе входных данных Алиса может выбрать $1, 2, 3$ или 4 дня, как количество дней, которое будет в неделе.

Всего есть 6 возможных раскрасок, которые изображены на рисунке, но среди них есть только 4 различные формы. Поэтому, ответ равен 4. Обратите внимание, что **последний пример на картинке это некорректная раскраска**, потому что не все клетки связны по сторонам.

В последнем наборе входных данных будьте аккуратны с возможным переполнением, описанном в формате выходных данных.

1371B - Magical Calendar

First, let's consider in case of a week has exactly w days.

- If $w < n$, the length of painted cells is strictly more than one week. So there are w valid shapes. (The first week contains $1, 2, \ldots, w$ days) The shapes have w-day width, then if the value of w are different, the shapes are also different.
- Otherwise $(n \leq w)$, there is only one valid liner pattern. The shape is insensitive to the chosen value of w.

We can sum up this for $1 \leq w \leq r$, by using following well-known formula: $a + (a + 1) + (a + 2) + \ldots + b = \frac{(a+b)*(b-a+1)}{2}$

Total complexity : $O(1)$

```
#include<stdio.h>

int main(){
  long long n,l=1,r,t,res;
  scanf("%lld",&t);
  while(t>0){
    t--;
    res=0;
    scanf("%lld%lld",&n,&r);
    if(n<=l){printf("1\n");continue;}
    if(n<=r){r=n-1;res=1;}
    printf("%lld\n",res+((l+r)*(r-l+1))/2);
  }
  return 0;
}
```

Codeforces Round #654 (Div. 2) 1371C Печенье для тебя

C. Печенье для тебя

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Анна очень смелая девочка, ее любят все в городе. Также все горожане обожают ее печенье. Она планирует провести вечеринку с печеньем. Сейчас у нее есть a единиц ванильного печенья и b единиц шоколадного печенья для вечеринки.

Она пригласит n гостей первого типа и m гостей второго типа на вечеринку. Они придут на вечеринку в некотором порядке. После того, как они придут на вечеринку, каждый гость выберет один из двух типов печенья (ванильное или шоколадное), чтобы съесть. Есть различие в том, как гости выбирают тип печенья:

Если всего v единиц ванильного печенья и c единиц шоколадного печенья в момент, когда гость приходит, тогда

- если гость первого типа: если $v > c$ гость выбирает **ванильное** печенье. Иначе, гость выбирает **шоколадное** печенье.
- если гость второго типа: если $v > c$ гость выбирает **шоколадное** печенье. Иначе, гость выбирает **ванильное** печенье.

После этого:

- Если есть хотя бы одна единица выбранного типа печенья, гость съедает одну.
- Иначе (если не осталось печенья выбранного типа), гость остается голодным и уходит домой.

Анна хочет узнать, существует ли какой-то порядок гостей, такой что **ни один гость не останется голодным**. Ваша задача состоит в том, чтобы ответить на ее вопрос.

Входные данные

Каждый тест состоит из нескольких наборов входных данных. В первой строке находится единственное целое число t ($1 \le t \le 1000$) — количество наборов входных данных. Следующие t строк содержат описания набов входных данных.

Для каждого набора входных данных в единственной строке находится четыре целых числа a, b, n, m ($0 \le a, b, n, m \le 10^{18}, n + m \neq 0$).

Выходные данные

Для каждого набора входных данных, выведите ответ в одной строке. Если есть хотя бы один подходящий порядок гостей, выведите «Yes». Иначе, выведите «No».

Вы можете выводить каждый символ в любом регистре (верхнем или нижнем).

Пример

входные данные
6 2 2 1 2 0 100 0 1 12 13 25 1 27 83 14 25 0 0 1 0 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
выходные данные
Yes No No Yes No Yes

Примечание

В первом наборе входных данных рассмотрим порядок $\{1, 2, 2\}$ типов гостей. Тогда:

- Первый гость съедает шоколадное печенье. После этого остается 2 единицы ванильного печенья и 1 единица шоколадного печенья.
- Второй гость съедает шоколадное печенье. После этого остается 2 единицы ванильного печенья и 0 единиц шоколадного печенья.
- Последний гость выбирает шоколадное печенье, чтобы съесть, но больше не осталось шоколадного печенья. Поэтому гость остается голодным.

Поэтому такой порядок гостей Анна выбрать не может.

Рассмотрим порядок $\{2, 2, 1\}$ типов гостей. Тогда:

- Первый гость съедает ванильное печенье. После этого остается 1 единица ванильного печенья и 2 единицы шоколадного печенья.
- Второй гость съедает ванильное печенье. После этого остается 0 единиц ванильного печенья и 2 единицы шоколадного печенья.
- Последний гость съедает шоколадное печенье. После этого остается 0 единиц ванильного печенья и 1 единица шоколадного печенья.

Поэтому ответ для этого набора входных данных «`Yes`».

В пятом наборе входных данных можно увидеть, что количество единиц печенья ($a + b$) может быть равно нулю, но количество гостей ($n + m$) никогда не равно нулю.

В шестом наборе входных данных будьте осторожны с переполнением 32-битного целочисленного типа.

1371C - A Cookie for You

If $m < \min(a, b), n + m \le a + b$ are satisfied, the answer is "`Yes`". Otherwise, the answer is "`No`". Let's proof it.

Of course, $n + m \le a + b$ must be satisfied, because violating this inequality means lack of cookies.

When a type 2 guest comes, or when $a = b$, the value of $\min(a, b)$ is decremented by 1.

You need to consider only about the case that all type 2 guests come first and after that all type 1 guests come, because if there is a type 1 guest before a type 2 guest, swapping them is better to make no one angry. (Because if there is a type 1 guest before a type 2 guest, the type 1 guest have a possibility to decrease the value of $min(a, b)$ unnecessarily.)

At last, all of type 1 guests eat one cookie when there is at least one cookie(both types are ok).

Total complexity: $O(1)$

```
#include<stdio.h>

int main(){
  long long t,a,b,n,m,k;
  scanf("%lld",&t);
  while(t>0){
    t--;
    scanf("%lld%lld%lld%lld",&a,&b,&n,&m);
    if(a>b){k=a;a=b;b=k;}
    if(a<m){printf("No\n");continue;}
    if(a+b<n+m){printf("No\n");continue;}
    printf("Yes\n");
  }
}
```

Codeforces Round #655 (Div. 2) 1372A Омкар и полнота

A. Омкар и полнота

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Вы были благословлены как дитя Омкара. Чтобы выразить свою благодарность, пожалуйста, решите для Омкара эту задачу!

Массив a длины n называется **полным**, если все его элементы положительны, не превышают 1000, и для любых трех индексов x, y, z ($1 \leq x, y, z \leq n$), $a_x + a_y \neq a_z$ (не обязательно различных).

Вам дано одно целое число n. Найдите **полный** массив длины n. Гарантируется, что при данных ограничениях решение существует.

Входные данные

Каждый тест содержит несколько наборов входных данных. Первая строка содержит t ($1 \leq t \leq 1000$) — количество наборов входных данных. Описание наборов входных данных приведено ниже.

Единственная строка каждого набора входных данных содержит одно целое число n ($1 \leq n \leq 1000$).

Гарантируется, что сумма n по всем наборам входных данных не превышает 1000.

Выходные данные

Для каждого набора входных данных выведите **полный** массив в отдельной строке. Все элементы должны быть целыми числами между 1 и 1000 и для любых трех индексов x, y, z ($1 \leq x, y, z \leq n$) (не обязательно различных), должно выполняться $a_x + a_y \neq a_z$.

Если существует несколько решений, вы можете вывести любое.

Пример

входные данные
2 5 4
выходные данные
1 5 3 77 12 384 384 44 44

Примечание

Можно показать, что массивы с примера являются **полными** массивами. Например, $44 + 44 \neq 384$.

Ниже приведены некоторые примеры массивов, которые НЕ **полные** для 1-го теста:

$[1, 2, 3, 4, 5]$

Обратите внимание, что $a_1 + a_2 = a_3$.

$[1, 3000, 1, 300, 1]$

Обратите внимание, что $a_2 = 3000 > 1000$.

1372A - Omkar and Completion

Notice that since all elements must be positive, $k \neq 2k$. The most simple construction of this problem is to simply make all elements equal to 1.

```
import java.util.*

fun main() {
    val jin = Scanner(System.`in`)
    for (c in 1..jin.nextInt()) {
        val n = jin.nextInt()
        for (j in 1..n) {
            print("1 ")
        }
        println()
    }
}
```

Codeforces Round #655 (Div. 2) 1372B Омкар и последний урок математики

B. Омкар и последний урок математики

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

На последнем уроке математики Омкар он узнал о наименьшем общем кратном, или HOK. $HOK(a, b)$ — это наименьшее положительное целое число x, которое делится и на a и на b.

Омкар, обладающий похвально любопытным умом, сразу же подумал о задаче, связанной с операцией HOK: по целому числу n найдите положительные целые числа a и b такие, что $a + b = n$ и $HOK(a, b)$ принимает минимально возможное значение.

Можете ли вы помочь Омкару решить его смешную математическую задачу?

Входные данные
Каждый тест содержит несколько наборов входных данных. Первая строка содержит количество наборов входных данных t ($1 \leq t \leq 10$). Описание наборов входных данных приведено ниже.

Каждый набор входных данных состоит из одного целого числа n ($2 \leq n \leq 10^9$).

Выходные данные
Для каждого набора входных данных выведите два положительных целых числа a и b такие, что $a + b = n$ и $HOK(a, b)$ минимально возможный.

Пример

входные данные
3 4 6 9
выходные данные
2 2 3 3 3 6

Примечание
Для первого набора входных данных мы можем выбрать числа $1, 3$ или $2, 2$. $HOK(1, 3) = 3$ и $HOK(2, 2) = 2$, поэтому мы выводим $2\ 2$.

Для второго набора входных данных мы можем выбрать числа $1, 5$, $2, 4$ или $3, 3$. $HOK(1, 5) = 5$, $HOK(2, 4) = 4$ и $HOK(3, 3) = 3$, поэтому мы выводим $3\ 3$.

Для третьего набора входных данных $HOK(3, 6) = 6$. Можно показать, что нет других пар чисел с суммой 9, имеющих меньший HOK.

1372B - Omkar and Last Class of Math

Short Solution: The two integers are k and $n - k$, where k is the largest proper factor of n.

Proof: Let the two integers be k and $n - k$. Assume WLOG that $k \leq n - k$. Notice that this implies that $n - k \geq \frac{n}{2}$.

We first claim that $LCM(k, n - k) = n - k < n$ if $k \mid n$, and we prove this as follows: if $k \mid n$, then there exists some integer m such that $m \cdot k = n$. The integer $n - k$ can then be written as $(m - 1) \cdot k$, which is a multiple of k. Thus, $LCM(k, n - k) = n - k$ if $k \mid n$.

We now show that $LCM(k, n - k) \geq n$ if $k \nmid n$. We show this by using the fact that $LCM(a, b) = b$ iff $a \mid b$, so if $k \nmid n$, $k \nmid n - k$, and so $LCM(k, n - k) \neq n - k$. And since $LCM(k, n - k)$ must be a multiple of both k and $n - k$, it follows that $LCM(k, n - k) \geq 2 \cdot (n - k) \geq 2 \cdot \frac{n}{2} = n$.

We have now established that to minimize $LCM(k, n-k)$, k must be a factor of n. And, since $LCM(k, n-k) = n-k$ when k is a factor of n, we need to minimize $n-k$, so we must maximize k by choosing it to be the largest proper factor of n (i. e. the largest factor of n other than n).

We then simply need to find k, the largest proper factor of n. If p is the smallest prime dividing n, then $k = \frac{n}{p}$, so it suffices to find the smallest prime factor of n. We can do this by simply checking all values of p such that $2 \leq p \leq \sqrt{n}$. If n is not prime, then it must have a prime factor not exceeding $\sqrt{n}$. Furthermore, if we do not find a factor of n between 2 and $\sqrt{n}$, then n must be prime so we simply get $p = n$ and $k = \frac{n}{p} = 1$.

We're given that $n \leq 10^9$, so $\sqrt{n} \leq 10^{\frac{9}{2}} < 10^5$. $t \leq 10$, meaning that we will check less than 10^6 numbers, which runs well under the time limit.

```
fun main() {
    for (c in 1..readLine()!!.toInt()) {
        val n = readLine()!!.toInt()
        var p = 0
        for (m in 2..100000) {
            if (n % m == 0) {
                p = m
                break
            }
        }
        if (p == 0) {
            p = n
        }
        println("${n / p} ${n - (n / p)}")
    }
}
```

Codeforces Round #655 (Div. 2) 1372C Омкар и бейсбол

C. Омкар и бейсбол

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Патрик любит играть в бейсбол, но иногда он тратит так много часов на пробежки, что его разум начинает затуманиваться! Патрик уверен, что его набранные очки за n игр соответствуют тождественной перестановке (т.е. в первой игре он набирает 1, во второй игре он набирает 2 и так далее). Однако, когда он посмотрел на свои записи, он увидел, что все значения перепутаны!

Определим специальный обмен следующим образом: выберите любой подмассив очков и переставьте местами его элементы так, чтобы ни один элемент не оказался в той же позиции, где он был до обмена. Например, выполнение специального обмена на $[1, 2, 3]$ может дать $[3, 1, 2]$, но не может дать $[3, 2, 1]$, так как 2 находится в той же позиции.

Вам дана перестановка из n целых чисел. Пожалуйста, помогите Патрику найти минимальное количество специальных обменов, необходимых для того, чтобы сделать ее отсортированной! Можно доказать, что при данных ограничениях это число не превышает 10^{18}.

Массив a является подмассивом массива b, если a можно получить из b, удалив несколько (возможно, ноль или все) элементов из начала и несколько (возможно, ноль или все) элементов с конца.

Входные данные

Каждый тест содержит несколько наборов входных данных. Первая строка содержит количество наборов входных данных t ($1 \le t \le 100$). Описание наборов входных данных приведено ниже.

Первая строка каждого набора входных данных содержит целое число n ($1 \le n \le 2 \cdot 10^5$) — длину данной перестановки.

Вторая строка каждого набора входных данных содержит n целых чисел $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — начальную перестановку.

Гарантируется, что сумма n по всем наборам входных данных не превышает $2 \cdot 10^5$.

Выходные данные

Для каждого набора входных данных выведите одно целое число: минимальное количество специальных обменов, необходимых для сортировки перестановки.

Пример

входные данные
2 5 1 2 3 4 5 7 3 2 4 5 1 6 7
выходные данные
0 2

Примечание

Первая перестановка она уже отсортирована, поэтому обмены не нужны.

Можно показать, что для сортировки второй перестановки нужно как минимум 2 обмена.

$[3, 2, 4, 5, 1, 6, 7]$

Сделаем специальный обмен для диапазона $(1, 5)$

$[4, 1, 2, 3, 5, 6, 7]$

Сделаем специальный обмен для диапазона $(1, 4)$

$[1, 2, 3, 4, 5, 6, 7]$

1372C - Omkar and Baseball

You need at most 2 special exchanges to sort the permutation. Obviously, 0 special exchanges are needed if the array is already sorted. Let's look into cases in which you need 1 special exchange to sort the array.

Refer to i as a matching index if $a_i = i$. If there are no matching indices, then you can just use one special exchange to sort the entire thing. Otherwise, you can use the location of matching indices to determine whether you need more than 1 special exchange. If all matching indices are located in some prefix of the permutation, you can sort the permutation with one special exchange. The same is true for a suffix. In other words, if you can choose a subarray in the permutation such that all elements contained in the subarray are NOT matching and the elements outside of this subarray are matching, then one special exchange is needed to sort the array.

Otherwise, you need 2 special exchanges to sort the permutation. Let's prove why you do not need more than 2 special exchanges. You can quickly check that you need at most 2 special exchanges for all permutations of length ≤ 3. For permutations of length ≥ 4, I claim that we can perform 2 special exchanges on the whole array; to show this it suffices to construct a permutation p that has no matching indices with either the given permutation or the identity permutation $1, 2, \ldots, n$. We can do this as follows:

For simplicity, assume that n is even. We will assign the numbers $\frac{n}{2} + 1, \frac{n}{2} + 2, \ldots, n$ to the first $\frac{n}{2}$ positions of our permutation p and the numbers $1, 2, \ldots, \frac{n}{2}$ to the last $\frac{n}{2}$ positions of p. This ensures that p has no matching indices with the identity permutation. Then, for all integers b such that their position i in a (i. e. the j such that $a_j = b$) is in the appropriate half of p, assign $p_i = b$; assign other b to arbitrary positions in the appropriate half of p. Finally, cyclically rotate each half of p – this ensures that p has no matching indices with a.

As an example, let's take $a = [3, 6, 2, 4, 5, 1]$. You can quickly check that this cannot be done in less than 2 special exchanges. The construction of p goes as follows:

First, we move all numbers to the proper half of p, so that $p = [4, 5, 6, 1, 2, 3]$.

Observing that $a_2 = 6$ and $a_6 = 1$, we set $p_2 = 6$ and $p_6 = 1$ then replace the remaining elements arbitrarily into the correct half, so we can get, for example, $p = [4, 6, 5, 2, 3, 1]$.

Finally, we cyclically rotate each half of p, obtaining $p = [5, 4, 6, 1, 2, 3]$, which has no matching indexes with either $a = [3, 6, 2, 4, 5, 1]$ or $[1, 2, 3, 4, 5, 6]$.

This can be extended to odd n by first choosing some element other than 1 and a_1 to be p_1 (this works for $n \geq 3$ and we must have $n \geq 5$ anyway in this case), and then running the same algorithm on the rest of p.

```
import java.io.BufferedReader
import java.io.InputStreamReader

fun main() {
    val jin = BufferedReader(InputStreamReader(System.`in`))
    for (c in 1..jin.readLine().toInt()) {
        val n = jin.readLine().toInt()
        val scores = jin.readLine().split(" ").map { it.toInt() - 1 }
        if ((1 until n).all { scores[it - 1] < scores[it] }) {
            println(0)
        } else {
            var stage = 0
            var answer = 1
            for (j in 0 until n) {
                if (scores[j] == j) {
                    if (stage == 1) {
                        stage = 2
                    }
                } else {
                    if (stage == 0) {
                        stage = 1
                    } else if (stage == 2) {
                        answer = 2
                        break
                    }
                }
            }
            println(answer)
        }
    }
}
```

Educational Codeforces Round 90 (рейтинговый для Див. 2)
1373A Магазины пончиков

A. Магазины пончиков

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Существует два конкурирующих магазина пончиков.

Первый магазин продает пончики в розницу: каждый пончик стоит a долларов.

Второй магазин продает пончики оптом: коробка из b пончиков стоит c долларов. То есть если вы хотите купить x пончиков в этом магазине, то вам придется купить минимальное количество коробок такое, что суммарное количество пончиков больше или равно x.

Вы хотите определить два **положительных целых** значения:

1. сколько пончиков можно купить, чтобы они стоили дешевле в первом магазине, чем во втором?
2. сколько пончиков можно купить, чтобы они стоили дешевле во втором магазине, чем в первом?

Если какое-то из значений не существует, то оно должно быть равно -1. Если существует несколько решений, выведите любое из них.

Выведенные значения должны быть меньше или равны 10^9. Можно показать, что в данных ограничениях такие значения всегда существуют, если значения существуют вообще.

Входные данные
В первой строке записано одно целое t ($1 \le t \le 1000$) — количество наборов входных данных.

В каждой из следующих t строк записаны по три целых числа a, b и c ($1 \le a \le 10^9$, $2 \le b \le 10^9$, $1 \le c \le 10^9$).

Выходные данные
На каждый набор входных данных выведите два **положительных** целых числа. Для обоих магазинов выведите такой x, что купить x пончиков в этом магазине строго дешевле, чем купить x пончиков в другом магазине. x должно быть больше 0 и меньше или равно, чем 10^9.

Если такого x не существует, то выведите -1. Если существует несколько решений, выведите любое из них.

Пример

входные данные
4 5 10 4 4 5 20 2 2 3 1000000000 1000000000 1000000000
выходные данные
-1 20 8 -1 1 2 -1 1000000000

Примечание
В первом наборе входных данных любое количество пончиков будет дешевле во втором магазине. Например, для 3 или 5 пончиков придется купить коробку из 10 пончиков за 4 доллара. Однако 3 или 5 пончиков будут стоить 15 или 25 долларов, соответственно. Для 20 пончиков придется купить две коробки за 8 долларов суммарно. Обратите внимание, что 3 и 5 также являются правильными ответами для второго магазина вместе со многими другими ответами.

Во втором наборе входных данных любое количество пончиков будет либо дешевле в первом магазине, либо по одинаковой цене. 8 пончиков стоят 32 доллара в первом магазине и 40 долларов во втором магазине (потому что придется купить две коробки). 10 пончиков будут стоить 40 долларов в обоих магазинах, поэтому 10 не будет правильным ответом ни для одного из магазинов.

В третьем наборе входных данных 1 пончик стоит 2 и 3 доллара, соответственно. 2 пончика стоят 4 и 3 доллара.

Поэтому 1 является правильным ответом для первого магазина, а 2 является правильным ответом для второго магазина.

В четвертом наборе входных данных 10^9 пончиков стоят 10^{18} долларов в первом магазине и 10^9 долларов во втором магазине.

1373A - Donut Shops

At first notice that if there exists a value for the second shop, then the value divisible by b also exists. For any x you can round it up to the nearest multiple of b. That won't change the price for the second shop and only increase the price for the first shop.

You can also guess that if there exists a value for the first shop, then the value with 1 modulo b also exists (exactly 1 donut on top of some number of full boxes). Following the same logic — the second shop needs an entire new box and the first shop needs only an extra donut.

So let's take a look at the smallest values of two kinds:

- $x = b$: this value is valid for the second shop if one box is cheaper than b donuts in the first shop. Otherwise, no matter how many boxes will you take, they will never be cheaper than the corresponding number of donuts.
- $x = 1$: this value is valid for the first shop if one donut is cheaper than one box in the second shop. Apply the same idea — otherwise no value for the first shop is valid.

Overall complexity: $O(1)$ per testcase.

```
for tc in range(int(input())):
 a, b, c = map(int, input().split())
 print(1 if a < c else -1, end=" ")
 print(b if c < a * b else -1)
```

Educational Codeforces Round 90 (рейтинговый для Див. 2)
1373B 01-игра

B. 01-игра

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Алиса и Боб играют в игру.

Изначально у них есть строка s, состоящая только из символов `0` и `1`.

Алиса и Боб ходят по очереди: Алиса делает первый ход, второй делает Боб, третий ход делает Алиса, и так далее. Во время своего хода игрок должен выбрать два **соседних различных** символа строки s и удалить их. Например, если $s = 1011001$, тогда возможны следующие ходы:

1. удалить s_1 и s_2: $1011001 \rightarrow 11001$;
2. удалить s_2 и s_3: $1011001 \rightarrow 11001$;
3. удалить s_4 и s_5: $1011001 \rightarrow 10101$;
4. удалить s_6 и s_7: $1011001 \rightarrow 10110$.

Если игрок не может сделать ход — он проигрывает. Оба игрока играют оптимально. Вам нужно определить, сможет ли Алиса выиграть.

Входные данные
Первая строка содержит число t ($1 \le t \le 1000$) — количество наборов входных данных.

Единственная строка каждого набора входных данных содержит строку s ($1 \le |s| \le 100$), состоящую только из символов `0` и `1`.

Выходные данные
На каждый набор входных данных выведите ответ в отдельной строке.

Если Алиса может выиграть, выведите `DA` в любом регистре. Иначе выведите `NET` в любом регистре.

Пример

входные данные
3 01 1111 0011
выходные данные
DA NET NET

Примечание
В первом наборе входных данных после хода Алисы строка s станет пустой и Боб не сможет сделать ход.

Во втором наборе входных данных Алиса не может сделать ход изначально.

В третьем наборе входных данных после хода Алисы строка s превратится в 01. А после хода Боба строка s станет пустой и Алиса не сможет сделать ход.

1373B - 01 Game

If there is at least one character `0` and at least one character `1`, then current player can always make a move. After the move the number of character `0` decreases by one, and the number of character `1` decreases by one too. So the number of moves is always $min(c0, c1)$, where $c0$ is the number of characters `0` in string s, and $c1$ is the number of characters `1` in string s.

So if $min(c0, c1)$ is odd then Alice wins, otherwise Bob wins.

```python
for _ in range(int(input())):
    s = input()
    print('DA' if min(s.count('0'), s.count('1')) % 2 == 1 else 'NET')
```

Educational Codeforces Round 90 (рейтинговый для Див. 2)
1373C Плюсы и минусы

C. Плюсы и минусы

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Вам задана строка s состоящая только из символов + и –. Вы выполняете некоторый процесс с этой строкой. Этот процесс можно описать следующим псевдокодом:

```
res = 0
for init = 0 to inf
    cur = init
    ok = true
    for i = 1 to |s|
        res = res + 1
        if s[i] == '+'
            cur = cur + 1
        else
            cur = cur - 1
        if cur < 0
            ok = false
            break
    if ok
        break
```

Обратите внимание, что inf обозначает бесконечность, а символы строки пронумерованы от 1 до $|s|$.

Вам нужно определить значение переменной res после выполнения процесса.

Входные данные
Первая строка содержит целое число t ($1 \le t \le 1000$) — количество наборов входных данных.

Единственная строка каждого набора входных данных содержит строку s ($1 \le |s| \le 10^6$), состоящую только из символов + и –.

Гарантируется, что сумма $|s|$ по всем наборам входных данных не превосходит 10^6.

Выходные данные
На каждый набор входных данных выведите ответ — значение переменной res после завершения процесса.

Пример

входные данные
3 --+- --- ++--+-
выходные данные
7 9 6

1373C - Pluses and Minuses

Let's replace all + with `1`, and all – with `-1`. After that let's create a preffix-sum array p ($p_i = \sum_{j=1}^{i} s_j$). Also lets create array f such that f_i is equal minimum index j such that $p_j = -i$ (if there is no such index $p_i = -1$).

Let's consider the first iteration of loop $for\ init\ =\ 0\ to\ inf$. If $f_1 = -1$ then process ends and $res = |s|$. Otherwise the condition $if\ cur\ <\ 0$ fulfilled then the value of i will be equal to f_1. So, the value of res is equal to f_1 after first iteration.

Now, let's consider the second iteration of loop $for\ init\ =\ 0\ to\ inf$. If $f_2 = -1$ then process ends and $res = f_1 + |s|$. Otherwise the condition $if\ cur\ <\ 0$ fulfilled then the value of i will be equal to f_2. So, the value of res is equal to $f_1 + f_2$ after second iteration.

In this way we can calculate the value of res after the process ends.

```
for _ in range(int(input())):
    s = input()
    cur, mn, res = 0, 0, len(s)
    for i in range(len(s)):
        cur += 1 if s[i] == '+' else -1
        if cur < mn:
            mn = cur
            res += i + 1

    print(res)
```

Educational Codeforces Round 91 (рейтинговый для Див. 2)
1380A Три индекса

A. Три индекса

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Вам задана перестановка $p_1, p_2, \ldots, p_n$. Напомним, что последовательность из n целых чисел называется *перестановкой*, если она содержит все целые числа от 1 до n ровно один раз.

Вам необходимо найти три индекса i, j и k такие, что:

- $1 \le i < j < k \le n$;
- $p_i < p_j$ и $p_j > p_k$.

Или сообщить, что таких трех индексов нет.

Входные данные
Первая строка содержит одно целое число T ($1 \le T \le 200$) — количество наборов входных данных.

Следующие $2T$ содержат описание наборов входных данных — две строки на каждый набор. Первая строка каждого набора входных данных содержит единственное целое число n ($3 \le n \le 1000$) — длина перестановки p.

Вторая строка содержит n целых чисел $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$; $p_i \ne p_j$ если $i \ne j$) — перестановка p.

Выходные данные
Для каждого набора входных данных:

- если есть такие индексы i, j и k, выведите `YES` (без учета регистра) и сами индексы;
- если таких трех индексов нет, выведите `NO` (без учета регистра).

Если допустимых наборов индексов несколько, выведите любой из них.

Пример

входные данные
3 4 2 1 4 3 6 4 6 1 2 5 3 5 5 3 1 2 4

выходные данные
YES 2 3 4 YES 3 5 6 NO

1380A - Three Indices

A solution in $O(n^2)$: iterate on j, check that there exists an element lower than a_j to the left of it, and check that there exists an element lower than a_j to the right of it. Can be optimized to $O(n)$ with prefix/suffix minima.

A solution in $O(n)$: note that if there is some answer, we can find an index j such that $a_{j-1} < a_j$ and $a_j > a_{j+1}$ (if there is no such triple, the array descends to some point and ascends after that, so there is no answer). So we only have to check $n - 2$ consecutive triples.

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

using namespace std;

const int N = 1000;

int n;
int a[N];

void solve() {
 cin >> n;
 for (int i = 0; i < n; ++i)
  cin >> a[i];
 for (int i = 1; i < n - 1; ++i) {
  if (a[i] > a[i - 1] && a[i] > a[i + 1]) {
   cout << "YES" << endl;
   cout << i << ' ' << i + 1 << ' ' << i + 2 << endl;
   return;
  }
 }
 cout << "NO" << endl;
}

int main() {
 int T;
 cin >> T;
 while (T--)
  solve();
}
```

Educational Codeforces Round 91 (рейтинговый для Див. 2)
1380B Универсальное решение

B. Универсальное решение

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Недавно, вы обнаружили бота, с которым можно сыграть в «Камень, ножницы, бумага». К сожалению, бот использует довольно примитивный алгоритм игры: у него есть строка $s = s_1 s_2 \dots s_n$ длины n, где каждый символ — это `R`, `S` или `P`.

Во время инициализации, бот выбирает стартовую позицию pos ($1 \le pos \le n$), и потом может сыграть любое количество раундов. В первом раунде, он выбирает «Камень», «Ножницы» или «Бумагу» на основании значения s_{pos}:

- если s_{pos} равняется `R`, то бот выбирает «Камень»;
- если s_{pos} равняется `S`, то бот выбирает «Ножницы»;
- если s_{pos} равняется `P`, то бот выбирает «Бумагу»;

Во втором раунде, выбор бота основан на значении s_{pos+1}. В третьем раунде — на s_{pos+2} и так далее. После s_n, бот возвращается к s_1 и продолжает игру.

Вы планируете сыграть n раундов и уже определили строку s, однако не знаете, чему равняется стартовая позиция pos. Но так как тактика бота очень скучная, вы решили найти такие n ходов в раундах, чтобы максимизировать среднее количество побед.

Другими словами, предположим, что ваши ходы — это $c_1 c_2 \dots c_n$ и если бот начнет с позиции pos, то вы выиграете в $win(pos)$ раундах. Найдите $c_1 c_2 \dots c_n$ такие, что $\frac{win(1)+win(2)+\dots+win(n)}{n}$ — максимально возможное.

Входные данные

В первой строке задано единственное число t ($1 \le t \le 1000$) — количество наборов входных данных.

В следующих t строках заданы сами наборы — по одному в строке. В единственной строке каждого набора задана строка $s = s_1 s_2 \dots s_n$ ($1 \le n \le 2 \cdot 10^5$; $s_i \in \{\mathrm{R}, \mathrm{S}, \mathrm{P}\}$) — строка бота.

Гарантируется, что суммарная длина всех строк в одном тесте не превосходит $2 \cdot 10^5$.

Выходные данные

Для каждого набора входных данных, выведите n ходов $c_1 c_2 \dots c_n$ таких, которые максимизируют среднее количество побед. Выведите их в том же формате, в котором задана строка s.

Если существует несколько оптимальных ответов, выведите любой из них.

Пример

входные данные
3 RRRR RSP S
выходные данные
PPPP RSP R

Примечание

В первом наборе входных данных, бот (с какой бы позиции не начал) будет всегда выбирать «Камень», поэтому мы можем всегда выбирать «Бумагу». То есть, в любом случае, мы выиграем все $n = 4$ раунда, и, соответственно, среднее количество побед также равно 4.

Во втором наборе:

- если бот начнет с позиции $pos = 1$, то (s_1, c_1) — ничья, (s_2, c_2) — ничья и (s_3, c_3) — ничья, поэтому $win(1) = 0$;

- если бот начнет с позиции $pos = 2$, то (s_2, c_1) — победа, (s_3, c_2) — победа и (s_1, c_3) — победа, поэтому $win(2) = 3$;
- если бот начнет с позиции $pos = 3$, то (s_3, c_1) — проигрыш, (s_1, c_2) — проигрыш и (s_2, c_3) — проигрыш, поэтому $win(3) = 0$;

Среднее равно $\frac{0+3+0}{3} = 1$, и можно доказать, что это максимально возможное среднее количество побед.
Картинка из Википедии, описывающая игру «Камень, ножницы, бумага»:

1380B - Universal Solution

Let's look at the contribution of each choice c_i to the total number of wins $win(1) + win(2) + \cdots + win(n)$ (we can look at "total" instead of "average", since "average" is equal to "total" divided by n). For example, let's look at the first choice c_1: in $win(1)$ we compare c_1 with s_1, in $win(2)$ — c_1 with s_2, in $win(3)$ — c_1 with s_3 and so on.

In the result, we compare c_1 with all s_i once. So, to maximize the total sum, we need to choose c_1 that beats the maximum number of s_i or, in other words, let's find the most frequent character in s and choose c_1 that beats it.

Okay, we found the optimal c_1. But if we look at the contribution of any other c_i we can note that we compare any c_i with all s_i once. So we can choose all c_i equal to c_1 which is equal to the choice that beats the most frequent choice in s.

```
fun main() {
    val winBy = mapOf('R' to 'P', 'S' to 'R', 'P' to 'S')
    repeat(readLine()!!.toInt()) {
        val s = readLine()!!
        val maxCnt = s.groupingBy { it }.eachCount().maxBy { it.value }!!.key
        println("${winBy[maxCnt]}".repeat(s.length))
    }
}
```

Educational Codeforces Round 91 (рейтинговый для Див. 2)
1380C Собери команды

C. Собери команды

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

У вас есть n программистов, которых вы хотите распределить по командам. Навык i-го программиста равен a_i. Вы хотите собрать из них максимальное количество команд. Для команд есть одно ограничение: количество программистов в команде, умноженное на минимальный навык среди всех программистов этой команды, должно быть как минимум x.

Каждый программист может находиться максимум в одной команде. Некоторые программисты могут остаться без команды.

Посчитайте максимальное количество команд, которое вы можете собрать.

Входные данные
Первая строка содержит одно число t ($1 \le t \le 1000$) — количество наборов входных данных.

Первая строка каждого набора входных данных содержит два числа n и x ($1 \le n \le 10^5; 1 \le x \le 10^9$) — количество программистов и ограничение на навык команды соответственно.

Вторая строка каждого набора входных данных содержит n чисел $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), где a_i равно навыку i-го программиста.

Сумма n по всем наборам не превосходит 10^5.

Выходные данные
На каждый набор входных данных выведите одно число — максимальное количество команд, которое вы можете собрать.

Пример

входные данные
3 5 10 7 11 2 9 5 4 8 2 4 2 3 4 11 1 3 3 7
выходные данные
2 1 0

1380C - Собери команды

At first, notice that if only $k < n$ programmers are taken, then the same or even better answer can be achieved if k strongest programmers are taken.

Now let's sort the programmers in non-decreasing order and choose some assignment into the teams. For each team only the rightmost taken programmer of that team matters (the sorted sequence implies that the rightmost is the weakest).

Take a look at the team with the strongest weakest member. If the number of programmers in it is less than the position of the weakest member, then you can safely rearrange the programmers before him in such a way that none of parameters of later teams change and the weakest member in the first one only becomes stronger. After that you can get rid of the first team (as it takes exactly the prefix of all the programmers) and proceed to fix the later teams.

Thus, we can see that there is an optimal solution such that each team is a segment and all the teams together take some prefix of the programmers. So we can finally run a greedy solution that takes programmers from left to right and increases the answer if the conditions for the latest team hold.

Overall complexity: $O(n \log n)$.

```
for _ in range(int(input())):
    n, x = map(int, input().split())
    a = sorted(list(map(int, input().split())), reverse=True)
    res, cur = 0, 1
    for s in a:
        if s * cur >= x:
            res += 1
            cur = 0
        cur += 1
    print(res)
```