

Codeforces Round #648 (Div. 2) 1365G Надежный пароль

G. Надежный пароль

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

Это интерактивная задача.

Ayush придумал еще один способ задать пароль для своего замка. В замке есть n слотов, в каждом слоте может находиться любое неотрицательное целое число. Пароль P это последовательность из n целых чисел, i -й из которых соответствует i -му слоту замка.

Чтобы задать пароль, Ayush придумал последовательность A из n целых чисел из отрезка $[0, 2^{63} - 1]$. Затем, он определил i -й элемент P как **побитовое ИЛИ** всех чисел в массиве кроме A_i .

Вам нужно отгадать пароль. Чтобы задать запрос, вы можете выбрать непустое подмножество индексов массива и спросить **побитовое ИЛИ** всех элементов массива с индексами в этом подмножестве. **Вы можете задать не более 13 запросов.**

Входные данные

В первой строке записано одно целое число n ($2 \leq n \leq 1000$) — количество слотов в замке.

Протокол взаимодействия

Чтобы задать вопрос, в отдельной строке:

- Сначала выведите «? c » (без кавычек), где c ($1 \leq c \leq n$) обозначает размер подмножества запроса, после чего выведите c различных целых чисел из отрезка $[1, n]$, разделенных пробелами.

В ответ на каждый запрос, вы получите число x — побитовое ИЛИ чисел с выбранными индексами. Если вы спросили некорректное множество индексов или вы превысили количество запросов, тогда вы получите $x = -1$. В таком случае вы должны немедленно завершить выполнение программы.

Если вы угадали пароль, в отдельной строке выведите «!» (без кавычек), после чего выведите n целых чисел, разделенных пробелами — последовательность-пароль.

Отгадывание пароля не считается в числе загаданных запросов.

Интерактор не адаптивный. Массив A не меняется с запросами.

После вывода запроса, не забывайте выводить конец строки и сбрасывать поток вывода. Иначе, вы получите верикт **Превышен лимит бездействия**. Чтобы сделать это, используйте:

- `fflush(stdout)` или `cout.flush()` в C++;
- `System.out.flush()` в Java;
- `flush(output)` в Pascal;
- `stdout.flush()` в Python;
- читайте документацию для остальных языков.

Взломы

Чтобы взломать решение, используйте следующий формат:

В первой строке, выведите одно целое число n ($2 \leq n \leq 1000$) — количество слотов в замке. Во второй строке выведите n целых чисел из отрезка $[0, 2^{63} - 1]$, разделенных пробелами — массив A .

Пример

входные данные

3
1
2

4
выходные данные
? 1 1
? 1 2
? 1 3
! 6 5 3

Примечание

Массив A в примере это $\{1, 2, 4\}$. Первый элемент пароля это побитовое ИЛИ элементов A_2 и A_3 , второй элемент это побитовое ИЛИ элементов A_1 и A_3 , а третий элемент это побитовое ИЛИ элементов A_1 и A_2 . Таким образом, пароль равен $\{6, 5, 3\}$.

```

#include
using namespace std;

#define ll long long
#define vint vector< int >

const int Q = 13;

ll query(vint v){
    cout << "? " << v.size() << ' ';
    for(ll i : v)
        cout << i + 1 << ' ';
    cout << endl;
    fflush(stdout);
    ll or_value;
    cin >> or_value;
    return or_value;
}

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

    vector< vint > ask(Q);
    vint assign_mask(n);

    vector< ll > or_value(Q, answer(n));

    int assigned = 0;

    for(int i = 1; i < (1 << Q); i++){
        if(__builtin_popcount(i) != Q / 2)
            continue;
        assign_mask[assigned] = i;
        for(int j = 0; j < Q; j++){
            if((i >> j & 1) == 0)
                ask[j].push_back(assigned);
            assigned++;
        }
        if(assigned == n)
            break;
    }

    for(int i = 0; i < Q; i++)
        if(!ask[i].empty())
            or_value[i] = query(ask[i]);

    for(int i = 0; i < n; i++)
        for(int j = 0; j < Q; j++)
            if(assign_mask[i] >> j & 1)
                answer[i] |= or_value[j];

    cout << "! ";

    for(ll i : answer)
        cout << i << ' ';

    cout << endl;
}

```

Codeforces Round #648 (Div. 2) 1365F Снова обмена

F. Снова обмена

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

Ayush, Ashish и Vivek заняты подготовкой задачи для следующего Codeforces раунда и им нужно помочь проверить тесты на корректность.

Каждый набор входных данных состоит из n и двух массивов a и b , длины n . Если после сколько-то (возможно, нуля) операций описанных ниже, массив a может стать равен массиву b , тест считается корректным. Иначе, он некорректный.

Возможные операции на массиве a следующие:

- выберите целое число k ($1 \leq k \leq \lfloor \frac{n}{2} \rfloor$)
- поменяйте местами префикс длины k с суффиксом длины k

Например, если массив a исходно равен $\{1, 2, 3, 4, 5, 6\}$, после выполнения операции с $k = 2$, он превратится в $\{5, 6, 3, 4, 1, 2\}$.

Вам дано несколько наборов входных данных, помогите определить про каждый из них, корректный он или некорректный.

Входные данные

В первой строке записано одно целое число t ($1 \leq t \leq 500$) — количество наборов входных данных. Далее следуют описания 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_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^9$) — элементы массива a .

В третьей строке записаны n целых чисел b_1, b_2, \dots, b_n ($1 \leq b_i \leq 10^9$) — элементы массива b .

Выходные данные

Для каждого набора входных данных, выведите «Yes», если данный ввод корректный. Иначе, выведите «No».

Вы можете выводить все символы в любом регистре.

Пример

входные данные
5 2 1 2 2 1 3 1 2 3 1 2 3 3 1 2 4 1 3 4 4 1 2 3 2 3 1 2 2 3 1 2 3 1 3 2
выходные данные
yes yes No yes No

Примечание

В первом наборе входных данных можно поменять местами префикс $a[1 : 1]$ с суффиксом $a[2 : 2]$, чтобы получить $a = [2, 1]$.

Во втором наборе входных данных a уже равен b .

В третьем наборе входных данных невозможно получить 3 в a .

В четвертом наборе входных данных сначала можно поменять местами префикс $a[1 : 1]$ с суффиксом $a[4 : 4]$, чтобы получить $a = [2, 2, 3, 1]$. После этого можно поменять местами префикс $a[1 : 2]$ с суффиксом $a[3 : 4]$, чтобы получить $a = [3, 1, 2, 2]$.

В пятом наборе входных данных невозможно превратить a в b .

```
#include
using namespace std;

int main(){
    ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int tc;
    cin >> tc;
    while(tc--){
        int n;
        cin >> n;
        map< pair< int, int >, int > pairs;
        vector< int > a(n), b(n);
        bool possible = 1;

        for(int i = 0; i < n; i++)
            cin >> a[i];

        for(int i = 0; i < n; i++)
            cin >> b[i];

        if(n % 2 == 1 && a[n / 2] != b[n / 2])
            possible = 0;

        for(int i = 0; i < n / 2; i++){
            pair< int, int > p = {min(a[i], a[n - 1 - i]), max(a[i], a[n - 1 - i])};
            pairs[p]++;
        }

        for(int i = 0; i < n / 2; i++){
            pair< int, int > p = {min(b[i], b[n - 1 - i]), max(b[i], b[n - 1 - i])};
            if(pairs[p] <= 0)
                possible = 0;
            pairs[p]--;
        }

        if(possible)
            cout << "Yes" << endl;

        else cout << "No" << endl;
    }
}
```

Codeforces Round #648 (Div. 2) 1365E Максимальное значение подпоследовательности

Е. Максимальное значение подпоследовательности

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

У Ashish есть массив a длины n состоящий из положительных целых чисел.

Определим значение непустой подпоследовательности массива a , состоящей из k чисел, как $\sum 2^i$ по всем целым $i \geq 0$ таким, что хотя бы $\max(1, k - 2)$ чисел в этом подмножестве имеют i -й бит в своей двоичной записи (число x имеет i -й бит в двоичной записи если $\lfloor \frac{x}{2^i} \rfloor \bmod 2$ равно 1).

Напомним, что b является подпоследовательностью a , если b может быть получена удалением нескольких (возможно, нуля) элементов из a .

Помогите ему найти наибольшее значение, которое он может получить, выбрав некоторую подпоследовательность a .

Входные данные

В первой строке записано одно целое число n ($1 \leq n \leq 500$) — размер массива a .

Во второй строке записаны n целых чисел — элементы массива ($1 \leq a_i \leq 10^{18}$).

Выходные данные

Выведите одно целое число — наибольшее значение, которое Ashish может получить, выбрав некоторую подпоследовательность a .

Примеры

входные данные
3 2 1 3
выходные данные
3
входные данные
3 3 1 4
выходные данные
7
входные данные
1 1
выходные данные
1
входные данные
4 7 7 1 1
выходные данные
7

Примечание

В первом примере Ashish может выбрать подпоследовательность $\{2, 3\}$ размера 2. Двоичная запись 2 это 10 а двоичная запись 3 это 11. Так как $\max(k - 2, 1)$ равно 1, значение подпоследовательности равно $2^0 + 2^1$ (и у 2 и у 3 есть 1-й бит в двоичной записи, а у 3 также есть 0-й бит в двоичной записи). Обратите внимание, что он также мог выбрать

подпоследовательность $\{3\}$ или $\{1, 2, 3\}$.

Во втором примере Ashish может выбрать подпоследовательность $\{3, 4\}$ со значением 7.

В третьем примере Ashish может выбрать подпоследовательность $\{1\}$ со значением 1.

В четвертом примере Ashish может выбрать подпоследовательность $\{7, 7\}$ со значением 7.

```
#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 = 505;

int n;
int a[N];

int32_t main()
{
    IOS;
    cin >> n;
    for(int i = 1; i <= n; i++)
        cin >> a[i];
    int ans = 0;
    for(int i = 1; i <= n; i++)
        for(int j = i; j <= n; j++)
            for(int k = j; k <= n; k++)
                ans = max(ans, (a[i] | a[j] | a[k]));
    cout << ans;
    return 0;
}
```

Codeforces Round #648 (Div. 2) 1365D Решить лабиринт

D. Решить лабиринт

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

Vivek столкнулся с интересной задачей. У него есть лабиринт, который можно описать таблицей $n \times m$. Каждая клетка может быть одного из следующих типов:

- Пустая — '.'
- Стена — '#'
- Хороший человек — 'G'
- Плохой человек — 'B'

Единственный выход из лабиринта находится в клетке (n, m) .

Человек может перейти в клетку если она не содержит стену и она имеет общую сторону с его текущей клеткой. Vivek хочет заблокировать (заменить на стены) некоторые пустые клетки, чтобы все хорошие люди могли дойти до выхода из лабиринта, но чтобы все плохие люди не могли. Клетку которая исходно содержит 'G' или 'B' нельзя блокировать, но через нее можно проходить.

Ваша задача — определить, можно ли заменить несколько (ноль или более) пустых клеток на стены, чтобы удовлетворить описанным ограничениям.

Гарантируется, что клетка (n, m) пустая. Vivek **разрешается** ее блокировать.

Входные данные

В первой строке записано одно целое число t ($1 \leq t \leq 100$) — 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 лабиринте.

В каждой из следующих n строк записаны m символов. Они описывают исходный лабиринт. Если символ в строке равен '.', тогда соответствующая клетка пустая, если символ равен '#', тогда клетка содержит стену, а 'G' и 'B' обозначают, что клетка содержит хорошего или плохого человека, соответственно.

Выходные данные

Для каждого набора входных данных, выведите «Yes» или «No», в зависимости от того, можно ли заменить некоторые пустые клетки на стены, чтобы удовлетворить описанным ограничениям.

Вы можете выводить каждую букву в любом регистре (верхнем или нижнем).

Пример

входные данные
6 1 1 . 1 2 G. 2 2 #B G. 2 3 G.# B#. 3 3 #B. #.. GG. 2 2 #B B.
выходные данные

Примечание

В первом и во втором наборах входных данных все условия уже выполнены.

В третьем наборе входных данных есть только одна свободная клетка $(2, 2)$, и если мы заменим ее на стену, то хороший человек в клетке $(1, 2)$ не сможет выйти.

В четвертом наборе входных данных хороший человек в клетке $(1, 1)$ не может сбежать с самого начала, поэтому ответ «N».

В пятом наборе входных данных можно заблокировать клетки $(2, 3)$ и $(2, 2)$.

В последнем наборе входных данных можно заблокировать выход $(2, 2)$.

```
#include
using namespace std;
#define int long long
typedef int ll;
typedef long double ld;
const ll N = 55;
char en = '\n';
ll inf = 1e16;
ll mod = 1e9 + 7;
ll power(ll x, ll n, ll mod) {
    ll res = 1;
    x %= mod;
    while (n) {
        if (n & 1)
            res = (res * x) % mod;
        x = (x * x) % mod;
        n >>= 1;
    }
    return res;
}
ll n, m;
char arr[N][N];
ll dir[4][2] = {{1, 0}, {-1, 0}, {0, 1}, {0, -1}};

bool valid(ll i, ll j) { return i >= 1 && i <= n && j >= 1 && j <= m; }
int32_t main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    ll t;
    cin >> t;
    while (t--) {
        cin >> n >> m;

        for (ll i = 1; i <= n; i++) {
            cin >> (arr[i] + 1);
        }

        for (ll i = 1; i <= n; i++) {
            for (ll j = 1; j <= m; j++) {
```

```

    for (ll k = 0; k < 4; k++) {
        ll ni = i + dir[k][0];
        ll nj = j + dir[k][1];
        if (valid(ni, nj) && arr[ni][nj] == '.')
            arr[ni][nj] = '#';
    }
}
}
}

queue< pair> que;
bool visited[n + 5][m + 5];
memset(visited, false, sizeof(visited));
if (arr[n][m] == '.') {
    que.push({n, m});
    visited[n][m] = true;
}
while (!que.empty()) {
    pair curr = que.front();
    que.pop();
    for (ll k = 0; k < 4; k++) {
        ll ni = curr.first + dir[k][0];
        ll nj = curr.second + dir[k][1];
        if (valid(ni, nj) && !visited[ni][nj] && arr[ni][nj] != '#') {
            que.push({ni, nj});
            visited[ni][nj] = true;
        }
    }
}
bool good = true;
for (ll i = 1; i <= n; i++) {
    for (ll j = 1; j <= m; j++) {
        if ((arr[i][j] == 'G' && !visited[i][j]) or
            (arr[i][j] == 'B' && visited[i][j])) {
            good = false;
            break;
        }
    }
}
cout << (good ? "Yes" : "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, \dots, 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 $c_1 := c_2, c_2 := c_3, \dots, c_n := c_1$ одновременно. Аналогично, циклический сдвиг перестановки c вправо это операция, которая присваивает $c_1 := c_n, c_2 := c_1, \dots, c_n := c_{n-1}$ 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 ($1 \leq n \leq 2 \cdot 10^5$) — размеры массивов.

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

В третьей строке записаны n целых чисел b_1, b_2, \dots, b_n ($1 \leq b_i \leq n$) — 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

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.

```
#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;
}
```

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

В. 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.

```
#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) 1365A Игра с таблицей

А. 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 выигрывает 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 выигрывает. 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 выигрывает.

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

```

#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;
}

```


Educational Codeforces Round 89 (рейтинговый для Див. 2)

1366G Построй строку

G. Построй строку

ограничение по времени на тест: 4 секунды
ограничение по памяти на тест: 512 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Определим функцию $f(s)$, которая принимает строку s , состоящую из строчных латинских букв и точек, и возвращает строку, состоящую из строчных латинских букв следующим образом:

1. пусть r пустая строка;
2. будем обрабатывать символы s слева направо. Для каждого символа c выполним следующее: если c является строчной латинской буквой, то добавим c в конец строки r ; в противном случае удалим последний символ из r (если r пустая — функция аварийно завершает работу);
3. вернуть r как результат функции.

Вам заданы две строки s и t . Вы должны удалить минимально возможное количество символов из s , чтобы $f(s) = t$ (и функция не завершалась аварийно). Обратите внимание, что вам не разрешается вставлять новые символы в s или менять порядок существующих.

Входные данные

Входные данные состоят из двух строк: первая содержит s — строку, состоящую из строчных латинских букв и точек, вторая содержит t — строку, состоящую из строчных латинских букв ($1 \leq |t| \leq |s| \leq 10000$).

Дополнительное ограничение на входные данные: можно удалить некоторое количество символов из s так, чтобы $f(s) = t$.

Выходные данные

Выведите одно целое число — минимально возможное количество символов, которое необходимо удалить из s , чтобы $f(s)$ не завершалась аварийно и вернула t в качестве результата выполнения.

Примеры

входные данные
a.ba.b. abb
выходные данные
2
входные данные
.bbac...a.c.cd bacd
выходные данные
3
входные данные
c..code..c...o.d.de code
выходные данные
3

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

```
using namespace std;
```

```
#define x first  
#define y second  
#define mp make_pair  
#define pb push_back  
#define sz(a) int((a).size())  
#define forn(i, n) for (int i = 0; i < int(n); ++i)  
#define fore(i, l, r) for (int i = int(l); i < int(r); ++i)
```

```
const int INF = 1e9;  
const int N = 10010;
```

```
int n, m;  
string s, t;  
int dp[N][N];  
int nxt[N];
```

```
int main() {  
    cin >> s >> t;  
    n = sz(s), m = sz(t);
```

```
    forn(i, n) if (s[i] != '.') {  
        int bal = 0;  
        nxt[i] = -1;  
        fore(j, i, n) {  
            if (s[j] == '.') --bal;  
            else ++bal;  
            if (bal == 0) {  
                nxt[i] = j;  
                break;  
            }  
        }  
    }  
}
```

```
    forn(i, n + 1) forn(j, m + 1)  
        dp[i][j] = INF;  
    dp[0][0] = 0;
```

```
    forn(i, n) forn(j, m + 1) {  
        dp[i + 1][j] = min(dp[i + 1][j], dp[i][j] + 1);  
        if (j < m && s[i] == t[j])  
            dp[i + 1][j + 1] = min(dp[i + 1][j + 1], dp[i][j]);  
        if (s[i] != '.' && nxt[i] != -1)  
            dp[nxt[i] + 1][j] = min(dp[nxt[i] + 1][j], dp[i][j]);  
    }
```

```
    cout << dp[n][m] << endl;  
}
```

Educational Codeforces Round 89 (рейтинговый для Див. 2)

1366F Прогулка по графу

F. Прогулка по графу

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

Задан простой взвешенный связный неориентированный граф, состоящий из n вершин и m ребер.

Путем длины k в графе назовем последовательность из $k + 1$ вершины v_1, v_2, \dots, v_{k+1} такую, что для каждого i ($1 \leq i \leq k$) ребро (v_i, v_{i+1}) присутствует в графе. У пути из вершины v вершина $v_1 = v$. Обратите внимание, что вершины и ребра могут входить в путь по несколько раз.

Вес пути — это сумма весов ребер в нем.

Для каждого i от 1 до q рассмотрим путь из вершины 1 длины i максимального веса. Чему равна сумма весов этих q путей?

Ответ может быть довольно большим, поэтому выведите его по модулю $10^9 + 7$.

Входные данные

В первой строке записаны три целых числа n, m, q ($2 \leq n \leq 2000$; $n - 1 \leq m \leq 2000$; $m \leq q \leq 10^9$) — количество вершин в графе, количество ребер в графе и количество длин, которые надо учесть в ответе.

В каждой из следующих m строк задано описание ребра: три целых числа v, u, w ($1 \leq v, u \leq n$; $1 \leq w \leq 10^6$) — две вершины v и u соединены неориентированным ребром веса w . Граф не содержит петель и кратных ребер. Гарантируется, что данные ребра задают связный граф.

Выходные данные

Выведите одно целое число — сумма весов путей максимального веса из вершины 1 длин $1, 2, \dots, q$ по модулю $10^9 + 7$.

Примеры

входные данные
7 8 25 1 2 1 2 3 10 3 4 2 1 5 2 5 6 7 6 4 15 5 3 1 1 7 3
выходные данные
4361

входные данные
2 1 5 1 2 4
выходные данные
60

входные данные
15 15 23 13 10 12 11 14 12 2 15 5 4 10 8 10 2 4 10 7 5 3 10 1 5 6 11

```
1 13 8
9 15 4
4 2 9
11 15 1
11 12 14
10 8 12
3 6 11
```

выходные данные

3250

входные данные

```
5 10 10000000
2 4 798
1 5 824
5 2 558
4 1 288
3 4 1890
3 1 134
2 3 1485
4 5 284
3 5 1025
1 2 649
```

выходные данные

768500592

Примечание

Граф из первого примера:

Некоторые максимальные пути:

- длина 1: ребра (1, 7) — вес 3;
- длина 2: ребра (1, 2), (2, 3) — вес $1 + 10 = 11$;
- длина 3: ребра (1, 5), (5, 6), (6, 4) — вес $2 + 7 + 15 = 24$;
- длина 4: ребра (1, 5), (5, 6), (6, 4), (6, 4) — вес $2 + 7 + 15 + 15 = 39$;
- ...

Поэтому ответ — это сумма 25 слагаемых: $3 + 11 + 24 + 39 + \dots$

Во втором примере веса у путей максимального веса равны 4, 8, 12, 16 and 20.

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

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

```
using namespace std;
```

```
const long long INF = 1e18;
const int MOD = 1000'000'007;
const int inv2 = (MOD + 1) / 2;
```

```
struct edge{
    int v, u, w;
};
```

```
struct frac{
    long long x, y;
    frac(long long a, long long b){
        if (b < 0) a = -a, b = -b;
        x = a, y = b;
    }
};
```

```
bool operator <=(const frac &a, const frac &b){
    return a.x * b.y <= a.y * b.x;
}
```

```

}

struct line{
    long long m, c;
    frac intersectX(const line &l) { return frac(c - l.c, l.m - m); }
};

int add(int a, int b){
    a += b;
    if (a >= MOD)
        a -= MOD;
    if (a < 0)
        a += MOD;
    return a;
}

int mul(int a, int b){
    return a * 1ll * b % MOD;
}

int calc(int a1, int d, int n){
    assert(n >= 0);
    return mul(mul(n, inv2), add(mul(2, a1), mul(add(n, -1), d)));
}

int main() {
    int n, m;
    long long q;
    scanf("%d%d%lld", &n, &m, &q);
    vector<edge> e(m);
    vector<int> hv(n);
    for(i, m){
        scanf("%d%d%d", &e[i].v, &e[i].u, &e[i].w);
        --e[i].v, --e[i].u;
        hv[e[i].v] = max(hv[e[i].v], e[i].w);
        hv[e[i].u] = max(hv[e[i].u], e[i].w);
    }

    int ans = 0;
    vector<long long> d(n, -INF), nd(n);
    d[0] = 0;
    for(val, m){
        long long mx = 0;
        for(i, n)
            mx = max(mx, d[i]);
        if (val)
            ans = add(ans, mx % MOD);
        nd = d;
        for(i, m){
            nd[e[i].v] = max(nd[e[i].v], d[e[i].u] + e[i].w);
            nd[e[i].u] = max(nd[e[i].u], d[e[i].v] + e[i].w);
        }
        d = nd;
    }

    vector<line> fin;
    for(i, n) fin.push_back({hv[i], d[i]});
    sort(fin.begin(), fin.end(), [](const line &a, const line &b){
        if (a.m != b.m)
            return a.m < b.m;
        return a.c > b.c;
    });
}

```

```

    },
    fin.resize(unique(fin.begin(), fin.end()), [](const line &a, const line &b){
        return a.m == b.m;
    }) - fin.begin());

vector<line> ch;
for (auto cur : fin){
    while (ch.size() >= 2 && cur.intersectX(ch.back()) <= ch.back().intersectX(ch[int(ch.size()) - 2]))
        ch.pop_back();
    ch.push_back(cur);
}

long long prv = 0;
q -= m;
for(i, int(ch.size()) - 1){
    frac f = ch[i].intersectX(ch[i + 1]);
    if (f.x < 0) continue;
    long long lst = min(q, f.x / f.y);
    if (lst < prv) continue;
    ans = add(ans, calc((ch[i].c + ch[i].m * prv) % MOD, ch[i].m % MOD, lst - prv + 1));
    prv = lst + 1;
}
ans = add(ans, calc((ch.back().c + ch.back().m * prv) % MOD, ch.back().m % MOD, q - prv + 1));

printf("%d\n", ans);
return 0;
}

```

Educational Codeforces Round 89 (рейтинговый для Див. 2)

1366E Два массива

Е. Два массива

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

Вам даны два массива a_1, a_2, \dots, a_n и b_1, b_2, \dots, b_m . Массив b отсортирован в порядке возрастания ($b_i < b_{i+1}$ верно для любого i от 1 до $m - 1$).

Вам нужно разбить массив a на m непрерывных подмассивов так, чтобы для всех i от 1 до m минимум в i -м подмассиве был равен b_i . Обратите внимание, что каждый элемент должен принадлежать ровно одному подмассиву, и они формируются следующим образом: первые несколько элементов массива a принадлежат первому подмассиву, следующие несколько элементов массива a принадлежат второму подмассиву, и так далее.

Например, если $a = [12, 10, 20, 20, 25, 30]$, а $b = [10, 20, 30]$, то существует два подходящих разбиения массива a :

- 1. $[12, 10, 20], [20, 25], [30]$;
- 2. $[12, 10], [20, 20, 25], [30]$.

Вам нужно посчитать количество хороших разбиений массива a . Так как это значение может быть слишком велико — выведите его по модулю 998244353.

Входные данные

Первая строка содержит два числа n и m ($1 \leq n, m \leq 2 \cdot 10^5$) — длины массивов a и b соответственно.

Вторая строка содержит n чисел a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^9$) — массив a .

Третья строка содержит m чисел b_1, b_2, \dots, b_m ($1 \leq b_i \leq 10^9; b_i < b_{i+1}$) — массив b .

Выходные данные

В единственной строке выведите число — количество хороших разбиений массива a по модулю 998244353.

Примеры

входные данные
6 3 12 10 20 20 25 30 10 20 30
выходные данные
2
входные данные
4 2 1 3 3 7 3 7
выходные данные
0
входные данные
8 2 1 2 2 2 2 2 2 1 2
выходные данные
7

#include <bits/stdc++.h>

using namespace std;

```

const int N = 200005;
const int MOD = 998244353;

int mul(int a, int b) {
    return (a * 1LL * b) % MOD;
}

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

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

    reverse(a, a + n);
    reverse(b, b + m);
    a[n] = -1;

    int mn = a[0];
    int pos = 0;
    while (pos < n && mn > b[0]) {
        ++pos;
        mn = min(mn, a[pos]);
    }

    if (pos == n || mn < b[0]) {
        puts("0");
        return 0;
    }

    assert(mn == b[0]);
    int res = 1;
    int ib = 0;
    while (true) {
        assert(mn == b[ib]);
        if (ib == m - 1) {
            if (*min_element(a + pos, a + n) != b[ib]) {
                puts("0");
                return 0;
            }
            break;
        }
    }

    bool f = true;
    int npos = pos;
    while (npos < n && mn != b[ib + 1]) {
        ++npos;
        mn = min(mn, a[npos]);

        if (f && mn < b[ib]) {
            f = false;
            res = mul(res, npos - pos);
        }
    }

    if (npos == n || mn != b[ib + 1]) {
        puts("0");
        return 0;
    }
}

```



```
    ++ib;  
    pos = npos;  
}  
  
printf("%d\n", res);  
return 0;  
}
```

Educational Codeforces Round 89 (рейтинговый для Див. 2)

1366D Два делителя

D. Два делителя

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

Вам заданы n целых чисел a_1, a_2, \dots, a_n .

Для каждого a_i найдите два его делителя $d_1 > 1$ и $d_2 > 1$ такие, что $\gcd(d_1 + d_2, a_i) = 1$ (где $\gcd(a, b)$ — наибольший общий делитель a и b) или скажите, что такой пары нет.

Входные данные

В первой строке задано единственное целое число n ($1 \leq n \leq 5 \cdot 10^5$) — размер массива a .

Во второй строке заданы n целых чисел a_1, a_2, \dots, a_n ($2 \leq a_i \leq 10^7$) — массив a .

Выходные данные

Ради ускорения вывода, выведите ответы в две строки по n чисел в каждой.

В первой и второй строках i -ми по счету числами выведите соответствующие делители $d_1 > 1$ и $d_2 > 1$ такие, что $\gcd(d_1 + d_2, a_i) = 1$ или -1 и -1 , если такой пары делителей нет. Если существует несколько подходящих ответов, выведите любой из них.

Пример

входные данные
10 2 3 4 5 6 7 8 9 10 24
выходные данные
-1 -1 -1 -1 3 -1 -1 -1 2 2 -1 -1 -1 -1 2 -1 -1 -1 5 3

Примечание

Рассмотрим $a_7 = 8$. У него есть 3 делителя больших, чем 1: 2, 4, 8. Не сложно заметить, что сумма любой пары делителей делится на 2, также как и a_7 .

Существуют и другие подходящие пары делителей d_1 и d_2 для $a_{10} = 24$, например, (3, 4) или (8, 3). Вы можете вывести любую из них.

```

fun main() {
    val n = readLine()!!.toInt()
    val a = readLine()!!.split(' ').map { it.toInt() }

    val minDiv = IntArray(1e7.toInt() + 2) { it }
    for (i in 2 until minDiv.size) {
        if (minDiv[i] != i)
            continue
        for (j in i until minDiv.size step i)
            minDiv[j] = minOf(minDiv[j], i)
    }

    fun getPrimeDivisors(v: Int): ArrayList<Int> {
        val ans = ArrayList<Int>()
        var curVal = v
        while (curVal != 1) {
            if (ans.isEmpty() || ans.last() != minDiv[curVal])
                ans.add(minDiv[curVal])
            curVal /= minDiv[curVal]
        }
        return ans
    }

    val d1 = IntArray(n)
    val d2 = IntArray(n)
    for (id in a.indices) {
        val list = getPrimeDivisors(a[id])
        if (list.size < 2) {
            d1[id] = -1
            d2[id] = -1
        } else {
            d1[id] = list[0]
            list.removeAt(0)
            d2[id] = list.reduce { s, t -> s * t }
        }
    }
    println(d1.joinToString(" "))
    println(d2.joinToString(" "))
}

```

Educational Codeforces Round 89 (рейтинговый для Див. 2)

1366C Палиндромные пути

С. 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 ($1 \leq t \leq 200$) — количество наборов входных данных.

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

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

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:

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

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

```
#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();
}
```

Educational Codeforces Round 89 (рейтинговый для Див. 2)

1366B Перемешивание

В. 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, \dots, 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 $l_i \leq c, d \leq r_i$, 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 ($1 \leq t \leq 100$) — 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 ($1 \leq n \leq 10^9; 1 \leq m \leq 100; 1 \leq x \leq n$).

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

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.

```
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)

1366A Лопаты и мечи

А. 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 ($1 \leq t \leq 1000$) — количество наборов входных данных.

Единственная строка каждого набора входных данных содержит два числа a и b ($0 \leq a, b \leq 10^9$) — 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.

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


Codeforces Round #650 (Div. 3) 1367F2 Летающая сортировка (сложная версия)

F2. Летающая сортировка (сложная версия)

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

Это сложная версия задачи. В этой версии в заданном массиве могут встречаться одинаковые числа и ограничения на n больше, чем в простой версии задачи.

Вам дан массив a из n целых чисел (в массиве могут быть одинаковые элементы). Вы можете производить над элементами массива следующие операции:

- выбрать любой индекс i ($1 \leq i \leq n$) и переместить элемент $a[i]$ в **начало** массива;
- выбрать любой индекс i ($1 \leq i \leq n$) и переместить элемент $a[i]$ в **конец** массива.

Например, если $n = 5$, $a = [4, 7, 2, 2, 9]$, то можно применить следующую последовательность операций:

- После применения операции первого типа ко второму элементу массив a станет равным $[7, 4, 2, 2, 9]$;
- После применения операции второго типа ко второму элементу массив a станет равным $[7, 2, 2, 9, 4]$.

Вы можете проводить операции любого типа произвольное количество раз в любом порядке.

Найдите минимальное суммарное количество операций первого и второго типа, которые сделают массив a отсортированным по неубыванию. Иными словами, сколько минимум операций надо применить, чтобы массив удовлетворял неравенствам $a[1] \leq a[2] \leq \dots \leq a[n]$?

Входные данные

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

Каждый набор начинается со строки, в которой записано целое число n ($1 \leq n \leq 2 \cdot 10^5$) — размер массива a .

Далее следуют n целых чисел a_1, a_2, \dots, a_n ($0 \leq a_i \leq 10^9$) — массив, который требуется отсортировать заданными операциями. (В массиве могут быть одинаковые элементы).

Сумма n по всем наборам тестовых данных в одном тесте не превосходит $2 \cdot 10^5$.

Выходные данные

Для каждого набора тестовых данных выведите одно целое число — минимальное суммарное количество операций первого и второго типа, которые сделают массив отсортированным по неубыванию.

Пример

входные данные
9 5 4 7 2 2 9 5 3 5 8 1 7 5 1 2 2 4 5 2 0 1 3 0 1 0 4 0 1 0 0 4 0 1 0 1 4 0 1 0 2 20 16 15 1 10 0 14 0 10 3 9 2 5 4 5 17 9 10 20 0 9
выходные данные

```
2
2
0
0
1
1
1
1
1
16
```

Примечание

В первом тестовом наборе нужно переместить две двойки в начало массива. Следовательно, искомая последовательность операций может иметь вид: $[4, 7, 2, 2, 9] \rightarrow [2, 4, 7, 2, 9] \rightarrow [2, 2, 4, 7, 9]$.

Во втором тестовом наборе нужно переместить единицу в начало массива, а восьмерку — в конец. Искомая последовательность операций имеет вид: $[3, 5, 8, 1, 7] \rightarrow [1, 3, 5, 8, 7] \rightarrow [1, 3, 5, 7, 8]$.

В третьем тестовом наборе массив уже отсортирован.

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

```
using namespace std;
using ld = long double;
using ll = long long;
```

```
void solve() {
    int n;
    cin >> n;
    vector<int> v(n);
    vector<pair<int, int>> a(n);
    for (int i = 0; i < n; i++) {
        cin >> v[i];
        a[i] = {v[i], i};
    }
    sort(a.begin(), a.end());
    vector<int> p(n);
    int j = 0;
    unordered_multiset<int> next;
    for (int i = 0; i < n; i++) {
        if (i > 0 && a[i].first != a[i - 1].first) {
            j++;
        }
        p[a[i].second] = j;
        next.insert(j);
    }
    unordered_map<int, int> d;
    vector<int> dp1(n), dp2(n), dp3(n), cnt(n);
    for (int i = 0; i < n; i++) {
        if (next.count(p[i])) {
            next.erase(next.find(p[i]));
        }
        if (d.count(p[i] - 1)) {
            if (!d.count(p[i])) {
                dp2[i] = max(dp2[i], dp1[p[i] - 1] + 1);
                if (!next.count(p[i] - 1)) {
                    dp2[i] = max(dp2[i], dp2[p[i] - 1] + 1);
                }
            }
        }
        if (!next.count(p[i] - 1)) {
            dp3[i] = max(dp3[i], dp2[p[i] - 1] + 1);
        }
        dp3[i] = max(dp3[i], dp1[p[i] - 1] + 1);
    }
    if (d.count(p[i])) {
```

```

        dp3[i] = max(dp3[i], dp3[d[p[i]]] + 1);
        dp2[i] = max(dp2[i], dp2[d[p[i]]] + 1);
        dp1[i] = dp1[d[p[i]]] + 1;
    } else {
        dp1[i] = 1;
    }
    dp2[i] = max(dp2[i], dp1[i]);
    dp3[i] = max(dp3[i], dp2[i]);
    d[p[i]] = i;
}
cout << n - *max_element(dp3.begin(), dp3.end()) << "\n";
}

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

```

Codeforces Round #650 (Div. 3) 1367F1 Летающая сортировка (простая версия)

F1. Летающая сортировка (простая версия)

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

Это простая версия задачи. В этой версии все числа в заданном массиве различны и ограничения на n меньше, чем в сложной версии задачи.

Вам дан массив a из n целых чисел (в массиве нет одинаковых элементов). Вы можете производить над элементами массива следующие операции:

- выбрать любой индекс i ($1 \leq i \leq n$) и переместить элемент $a[i]$ в **начало** массива;
- выбрать любой индекс i ($1 \leq i \leq n$) и переместить элемент $a[i]$ в **конец** массива.

Например, если $n = 5$, $a = [4, 7, 2, 3, 9]$, то можно применить следующую последовательность операций:

- после применения операции первого типа ко второму элементу массив a станет равным $[7, 4, 2, 3, 9]$;
- после применения операции второго типа ко второму элементу массив a станет равным $[7, 2, 3, 9, 4]$.

Вы можете проводить операции любого типа произвольное количество раз в любом порядке.

Найдите минимальное суммарное количество операций первого и второго типа, которые сделают массив a отсортированным по неубыванию. Иными словами, сколько минимум операций надо применить, чтобы массив удовлетворял неравенствам $a[1] \leq a[2] \leq \dots \leq a[n]$?

Входные данные

В первой строке записано одно целое число t ($1 \leq t \leq 100$) — количество наборов тестовых данных в тесте. Далее следуют t наборов тестовых данных.

Каждый набор начинается со строки, в которой записано целое число n ($1 \leq n \leq 3000$) — размер массива a .

Далее следуют n целых чисел a_1, a_2, \dots, a_n ($0 \leq a_i \leq 10^9$) — массив, который требуется отсортировать заданными операциями. Все числа в заданном массиве различны.

Сумма n по всем наборам тестовых данных в одном тесте не превосходит 3000.

Выходные данные

Для каждого набора тестовых данных выведите одно целое число — минимальное суммарное количество операций первого и второго типа, которые сделают массив отсортированным по неубыванию.

Пример

входные данные
4 5 4 7 2 3 9 5 3 5 8 1 7 5 1 4 5 7 12 4 0 2 1 3
выходные данные
2 2 0 2

Примечание

В первом тестовом наборе нужно переместить сначала тройку, а потом двойку в начало массива. Следовательно, искомая последовательность операций может иметь вид: $[4, 7, 2, 3, 9] \rightarrow [3, 4, 7, 2, 9] \rightarrow [2, 3, 4, 7, 9]$.

Во втором тестовом наборе нужно переместить единицу в начало массива, а восьмерку — в конец. Искомая последовательность операций имеет вид: $[3, 5, 8, 1, 7] \rightarrow [1, 3, 5, 8, 7] \rightarrow [1, 3, 5, 7, 8]$.

В третьем тестовом наборе массив уже отсортирован.

Codeforces Round #650 (Div. 3) 1367E Сборка ожерелья

Е. Сборка ожерелья

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

В магазине продаются n бусинок. Цвет каждой бусинки описывается строчной буквой латинского алфавита («a»–«z»). Вы хотите купить какие-то бусинки, чтобы собрать из них ожерелье.

Ожерелье — набор бусинок, соединенных по кругу.

Например, если в магазине продаются бусинки «a», «b», «c», «a», «c», «c», то вы можете собрать следующие ожерелья (это не все возможные варианты):

А следующие ожерелья нельзя собрать из бусинок, которые продаются в магазине:

Первое ожерелье собрать не получится, потому что в нем есть три бусинки «a» (из двух доступных). Второе ожерелье собрать не получится, потому что в нем есть бусинка «d», которой нет в магазине.

Назовем ожерелье k -красивым, если при его повороте по часовой стрелке на k бусинок ожерелье остается неизменным. Например, вот последовательность из трех поворотов некоторого ожерелья.

Так как после трех поворотов по часовой стрелке ожерелье не изменилось, то оно является 3-красивым. Как можно заметить, это ожерелье также является 6-красивым, 9-красивым и так далее, но не является 1-красивым или 2-красивым.

В частности, ожерелье длины 1 является k -красивым для любого целого k . Ожерелье, которое состоит из бусинок одинакового цвета, тоже является красивым для любого k .

Вам даны числа n и k , а также строка s , содержащая n строчных букв латинского алфавита — каждая буква задает бусинку в магазине. Вы можете купить любое подмножество бусинок и соединить их в произвольном порядке. Найдите максимальную длину k -красивого ожерелья, которое вы можете собрать.

Входные данные

В первой строке записано одно целое число t ($1 \leq t \leq 100$) — количество наборов тестовых данных в тесте. Далее следуют t наборов тестовых данных.

Первая строка каждого набора содержит два целых числа n и k ($1 \leq n, k \leq 2000$).

Вторая строка каждого набора содержит строку s , содержащую n строчных букв латинского алфавита — набор бусинок в магазине.

Гарантируется, что сумма n по всем наборам тестовых данных в тесте не превосходит 2000.

Выходные данные

Выведите t ответов на наборы тестовых данных. Каждый ответ является целым положительным числом — максимальной длиной k -красивого ожерелья, которое вы можете собрать.

Пример

входные данные
6 6 3 abcbac 3 6 aaa 7 1000 abczgyo 5 4 ababa 20 10 aaebdbabdbbdaadaadc 20 5 ecbedecacbcbbcbbdec
выходные данные

```
6
3
5
4
15
10
```

Примечание

Первый набор тестовых данных разобран в условии.

Во втором наборе тестовых данных 6-красивое ожерелье можно собрать из всех букв.

В третьем наборе тестовых данных 1000-красивое ожерельем можно собрать, например, из бусинок «abzyo».

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

```
int main() {
    int test;
    cin >> test;

    while (test--) {
        int n, k;
        cin >> n >> k;
        string s;
        cin >> s;

        vector<int> cnt(26);

        for (char c : s) {
            cnt[c - 'a']++;
        }

        for (int len = n; len >= 1; len--) {
            vector<bool> used(len);
            vector<int> cycles;

            for (int i = 0; i < len; i++) {
                if (used[i]) {
                    continue;
                }

                int j = (i + k) % len;
                used[i] = true;
                cycles.push_back(0);
                cycles.back()++;

                while (!used[j]) {
                    cycles.back()++;
                    used[j] = true;
                    j = (j + k) % len;
                }
            }

            vector<int> cur_cnt(cnt);

            sort(cycles.begin(), cycles.end());
            sort(cur_cnt.begin(), cur_cnt.end());

            bool can_fill = true;

            while (!cycles.empty()) {
                if (cur_cnt.back() < cycles.back()) {
```

```
can_fill = false;
break;
} else {
    cur_cnt.back() -= cycles.back();
    cycles.pop_back();
    sort(cur_cnt.begin(), cur_cnt.end());
}
}

if (can_fill) {
    cout << len << endl;
    break;
}
}
}
}
```


Codeforces Round #650 (Div. 3) 1367D Задача на доске

D. Задача на доске

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

Поликарп написал на доске некоторую строку s из строчных букв латинского алфавита ('a'-'z'). Эта строка вам известна и задана во входных данных.

После этого он стёр какие-то буквы из строки s , а оставшиеся буквы он переписал в **произвольном** порядке. В результате он получил некоторую новую строку t . Её вам и предстоит найти по некоторой дополнительной информации.

Предположим, что длина строки t равна m , а символы пронумерованы слева направо от 1 до m . В таком случае вам задана последовательность из m целых чисел: b_1, b_2, \dots, b_m , где b_i равно сумме расстояний $|i - j|$ от индекса i до всех таких индексов j , что $t_j > t_i$ (считайте, что 'a' < 'b' < ... < 'z'). Иными словами, для вычисления b_i Поликарп находит все такие индексы j , что в индексе j находится буква, которая стоит позже в алфавите чем t_i , и суммирует все значения $|i - j|$.

Например, если $t = \text{«abzb»}$, то:

- так как $t_1 = \text{'a'}$, то все остальные индексы содержат буквы, которые позже в алфавите, то есть:
 $b_1 = |1 - 2| + |1 - 3| + |1 - 4| = 1 + 2 + 3 = 6$;
- так как $t_2 = \text{'b'}$, то только индекс $j = 3$ содержит букву, которая позже в алфавите, то есть: $b_2 = |2 - 3| = 1$;
- так как $t_3 = \text{'z'}$, то индексов j , что $t_j > t_i$ не существует: $b_3 = 0$;
- так как $t_4 = \text{'b'}$, то только индекс $j = 3$ содержит букву, которая позже в алфавите, то есть: $b_4 = |4 - 3| = 1$.

Таким образом, если $t = \text{«abzb»}$, то $b = [6, 1, 0, 1]$.

По заданной строке s и массиву b найдите любую возможную строку t , для которой выполняются следующие два требования одновременно:

- t получается из s путём стирания некоторых букв (возможно, нуля) и потом записи оставшихся в **произвольном** порядке;
- по строке t получается заданный во входных данных массив b , если его построить по правилам, которые описаны выше.

Входные данные

В первой строке записано целое число q ($1 \leq q \leq 100$) — количество наборов входных данных в тесте. Далее следуют q наборов входных данных.

Каждый набор входных данных состоит из трех строк:

- строки s , которая имеет длину от 1 до 50 и состоит из строчных букв латинского алфавита;
- строки, которая содержит целое число m ($1 \leq m \leq |s|$), где $|s|$ — длина строки s , а m — длина массива b ;
- строки, которая содержит целые числа b_1, b_2, \dots, b_m ($0 \leq b_i \leq 1225$).

Гарантируется, что в каждом наборе данных входные данные таковы, что ответ существует.

Выходные данные

Выведите q строк: k -я из них должна содержать ответ (строку t) на k -й набор входных данных. Гарантируется, что ответ на каждый набор входных данных существует. Если ответов несколько, то выведите любой.

Пример

входные данные

```
4
abac
3
2 1 0
abc
1
0
```

```
abba
3
1 0 1
ecoosdcefr
10
38 13 24 14 11 5 3 24 17 0
```

выходные данные

```
aac
b
aba
codeforces
```

Примечание

В первом наборе входных данных подходят такие строки t : «aac», «aab».

Во втором наборе входных данных подходят такие строки t : «a», «b», «c».

В третьем наборе входных данных подходит только строка t равная «aba», но символ 'b' может быть со второй или с третьей позиции.

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

```
using namespace std;
```

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

```
int main() {  
    int q;  
    cin >> q;  
    forn(qq, q) {  
        string s;  
        cin >> s;  
        int n;  
        cin >> n;  
        vector<int> b(n);  
        forn(i, n)  
            cin >> b[i];  
        vector<vector<int>> groups;  
        while (true) {  
            vector<int> pos;  
            forn(i, n)  
                if (b[i] == 0)  
                    pos.push_back(i);  
            if (pos.empty())  
                break;  
            groups.push_back(pos);  
            forn(i, n)  
                if (b[i] == 0)  
                    b[i] = INT_MAX;  
            else  
                for (int pp: pos)  
                    b[i] -= abs(i - pp);  
        }  
        map<char, int> cnts;  
        forn(i, s.size())  
            cnts[s[i]]++;  
        auto j = cnts.rbegin();  
        string t(n, '?');  
        for (auto g: groups) {  
            while (j->second < g.size())  
                j++;  
            for (int pp: g)  
                t[pp] = j->first;  
            j++;  
        }  
        cout << t << endl;  
    }  
}
```

Codeforces Round #650 (Div. 3) 1367C Социальная дистанция

C. Социальная дистанция

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

Поликарп с друзьями хочет сходить в новый ресторан. Ресторан представляет из себя n столиков, расставленных вдоль прямой. За некоторыми столиками уже сидят люди. Столики пронумерованы от 1 до n в порядке слева направо. Состояние ресторана описывается строкой длины n , которая содержит символы '1' (стол occupied) и '0' (стол свободен).

Правила ресторана запрещают людям садиться на расстоянии k или меньше друг от друга. То есть, если человек сидит за столиком номер i , то все столики с номерами от $i - k$ до $i + k$ (кроме i -го) должны быть свободны. Иными словами, разница (то есть модуль разности) номеров между любыми двумя занятыми столиками должна быть строго больше k .

Например, если $n = 8$ и $k = 2$, то:

- строки «10010001», «10000010», «00000000», «00100000» соответствуют правилам ресторана;
- строки «10100100», «10011001», «11111111» не соответствуют правилам ресторана, так как в каждой из них есть пара единиц на расстоянии меньшем или равном $k = 2$.

В частности, если состояние ресторана описывается строкой без единиц или строкой с одной единицей, то требование ресторана выполнено.

Вам задана бинарная строка s , которая описывает текущее состояние ресторана. Гарантируется, что для строки s правила ресторана выполнены.

Найдите максимальное количество свободных столиков, которые можно занять, чтобы не нарушить правила ресторана. Формально, какое максимальное количество нулей можно заменить на единицы так, что требование все еще будет выполняться?

Например, если $n = 6$, $k = 1$, $s = «100010»$, то ответ на задачу будет 1, так как есть только один свободный столик на позиции 3, который можно занять в соответствии с правилами ресторана.

Входные данные

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

Каждый набор начинается со строки, в которой записано два целых числа n и k ($1 \leq k \leq n \leq 2 \cdot 10^5$) — количество столиков в ресторане и минимальное разрешенное расстояние между двумя людьми.

Во второй строке каждого набора записана строка s длины n , состоящая из нулей и единиц — описание свободных и занятых столиков в ресторане. Заданная строка соответствует правилам ресторана — разница индексов между любыми двумя единицами строго больше k .

Сумма n по всем наборам тестовых данных в одном тесте не превосходит $2 \cdot 10^5$.

Выходные данные

Для каждого тестового набора выведите одно целое число — количество столиков, которые можно дополнительно занять, чтобы не нарушить правила ресторана. Если дополнительных столиков занять нельзя, то, очевидно, надо вывести 0.

Пример

входные данные

```
6
6 1
100010
6 2
000000
5 1
10101
3 1
001
2 2
```

```
00
1 1
0
```

выходные данные

```
1
2
0
1
1
1
1
```

Примечание

Первый набор тестовых данных разобран в условии.

Во втором наборе тестовых данных ответ **2**, так как можно выбрать первый и шестой столики.

В третьем наборе тестовых данных нельзя занять никакой свободный столик, не нарушив правила ресторана.

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

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

    for (int test = 1; test <= t; test++) {
        int n, k;
        cin >> n >> k;
        string s;
        cin >> s;

        int res = 0;

        for (int i = 0; i < n; i) {
            int j = i + 1;

            for (; j < n && s[j] != '1'; j++);

            int left = s[i] == '1' ? k : 0;
            int right = j < n && s[j] == '1' ? k : 0;
            int len = j - i;

            if (left == k) {
                len--;
            }

            len -= left + right;

            if (len > 0) {
                res += (len + k) / (k + 1);
            }

            i = j;
        }

        cout << res << endl;
    }

    return 0;
}
```

Codeforces Round #650 (Div. 3) 1367B Четный массив

В. Четный массив

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

Вам дан массив $a[0 \dots n - 1]$ длины n , который состоит из неотрицательных целых чисел. **Обратите внимание:** массив нумеруется с нуля.

Назовём массив *хорошим*, если четность каждой позиции совпадает с четностью элемента в ней. Более формально, массив является хорошим, если для всех i ($0 \leq i \leq n - 1$) выполнено равенство $i \bmod 2 = a[i] \bmod 2$, где $x \bmod 2$ — остаток от деления x на 2.

Например, массивы $[0, 5, 2, 1]$ и $[0, 17, 0, 3]$ — хорошие, а массив $[2, 4, 6, 7]$ — плохой, потому что для $i = 1$ четность i и $a[i]$ различна: $i \bmod 2 = 1 \bmod 2 = 1$, но $a[i] \bmod 2 = 4 \bmod 2 = 0$.

За один ход вы можете взять **любые** два элемента массива и поменять их местами (эти элементы не обязательно соседние).

Найдите минимальное количество ходов, за которое можно сделать массив a хорошим, либо укажите, что это сделать невозможно.

Входные данные

В первой строке записано одно целое число t ($1 \leq t \leq 1000$) — количество наборов тестовых данных в тесте. Далее следуют t наборов тестовых данных.

Каждый набор начинается со строки, в которой записано целое число n ($1 \leq n \leq 40$) — размер массива a .

Далее следует строка, содержащая n целых чисел a_0, a_1, \dots, a_{n-1} ($0 \leq a_i \leq 1000$) — исходный массив.

Выходные данные

Для каждого набора тестовых данных выведите одно целое число — минимальное количество ходов, за которое можно сделать заданный массив a хорошим, или -1 , если это сделать невозможно.

Пример

входные данные
4 4 3 2 7 6 3 3 2 6 1 7 7 4 9 2 1 18 3 0
выходные данные
2 1 -1 0

Примечание

В первом наборе тестовых данных в первый ход можно поменять местами элементы на позициях 0 и 1, а во второй ход поменять местами элементы на позициях 2 и 3.

Во втором наборе тестовых данных в первый ход надо поменять местами элементы на позициях 0 и 1.

В третьем наборе тестовых данных нельзя сделать массив хорошим.

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

```
using namespace std;  
using ld = long double;  
using ll = long long;
```

```
void solve() {  
    int n;  
    cin >> n;  
    int a = 0, b = 0;  
    for (int i = 0; i < n; i++) {  
        int x;  
        cin >> x;  
        if (x % 2 != i % 2) {  
            if (i % 2 == 0) {  
                a++;  
            } else {  
                b++;  
            }  
        }  
    }  
    if (a != b) {  
        cout << -1 << endl;  
    } else {  
        cout << a << endl;  
    }  
}
```

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

Codeforces Round #650 (Div. 3) 1367A Короткие подстроки

А. Короткие подстроки

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

Алиса отгадывает строки, которые загадал ей Боб.

Сначала Боб придумал секретную строку a , состоящую из строчных букв латинского алфавита. Строка a имеет длину 2 или более символов. Затем по строке a он строит новую строку b и даёт Алисе строку b , чтобы она могла угадать строку a .

Боб строит b по a следующим образом: он выписывает все подстроки длины 2 строки a в порядке слева направо, а потом соединяет их в том же порядке в строку b .

Например, если Боб загадал строку $a = \text{«abac»}$, то все подстроки длины 2 строки a таковы: «ab» , «ba» , «ac» . Следовательно, строка $b = \text{«abbaac»}$.

Вам задана строка b . Помогите Алисе определить строку a , которую загадал Боб. Гарантируется, что b была построена по алгоритму, приведенному выше. Можно доказать, что ответ на задачу единственный.

Входные данные

В первой строке находится одно целое положительное число t ($1 \leq t \leq 1000$) — количество наборов тестовых данных в тесте. Далее следуют t наборов тестовых данных.

Каждый набор тестовых данных состоит из одной строки, в которой записана строка b , состоящая из строчных букв латинского алфавита ($2 \leq |b| \leq 100$) — строка, которую загадал Боб, где $|b|$ — длина строки b . Гарантируется, что b была построена по алгоритму, приведенному выше.

Выходные данные

Выведите t ответов на наборы тестовых данных. Каждый ответ — это строка a , состоящая из строчных букв латинского алфавита, которую загадал Боб.

Пример

входные данные
4 abbaac ac bccddaaf zzzzzzzzzz
выходные данные
abac ac bcdaf zzzzzz

Примечание

Первый набор тестовых данных разобран в условии.

Во втором наборе тестовых данных Боб загадал строку $a = \text{«ac»}$, строка a имеет длину 2, поэтому строка b совпадает со строкой a .

В третьем наборе тестовых данных Боб загадал строку $a = \text{«bcdaf»}$, подстроки длины 2 строки a таковы: «bc» , «cd» , «da» , «af» , поэтому строка $b = \text{«bccddaaf»}$.


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

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

```
    for (int test = 1; test <= t; test++) {
        string b;
        cin >> b;
```

```
        string a = b.substr(0, 2);
```

```
        for (int i = 3; i < b.size(); i += 2) {
            a += b[i];
        }
```

```
        cout << a << endl;
    }
```

```
    return 0;
}
```

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

Джонни и древний компьютер

А. 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 ($1 \leq t \leq 1000$) — 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 ($1 \leq a, b \leq 10^{18}$) — 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).

```
#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;  
}
```

