QUIZ

1) В data.csv файле лежат данные, необходимо построить график по этим данным
2) В data2.csv файле лежат данные, необходимо построить график по этим данным. Также подберите загаданную функцию

ВСЕ ФАЙЛЫ ВЫ МОЖЕТЕ НАЙТИ В ГИТЕ

A - Музыкальный паззл

Влад решил записать мелодию на своей гитаре. Представим мелодию как последовательность нот, которым соответствуют символы 'a', 'b', 'c', 'd', 'e', 'f' и 'g'.

Однако, он не очень опытен в игре на гитаре и может записать **ровно две** ноты за раз. Влад хочет получить мелодию s и для этого он может сводить записанные мелодии вместе. При этом, последний звук в первой мелодии должен совпадать с первым звуком второй мелодии.

Например, если Влад записал мелодии «`ab`» и «`ba`», он может свести их вместе и получить мелодию «`aba`», а потом свести результат с «`ab`» и получить «`abab`».

Помогите Владу определить, какое **минимальное** количество мелодий из двух нот ему нужно записать, чтобы получить мелодию s.

Входные данные

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

Далее следуют описания наборов.

Первая строка набора содержит целое число n ($2 \le n \le 50$) — длина мелодии s.

Вторая строка набора содержит строку s из n символов, каждый из которых 'a', 'b', 'c', 'd', 'e', 'f' или 'g'.

Выходные данные

Выведите t целых чисел, каждое из которых является ответом на соответствующий набор входных данных. В качестве ответа выведите **минимальное** количество мелодий из двух нот, которое нужно записать Владу.

Пример

входные данные
5 4 abab 7 abacaba 6 aaaaaa 7 abcdefg 5 babdd

выходные данные
2 4 1 6 4

Разбор

Построим мелодию последовательно. На первом шаге мы сможем записать ноты s_1 и s_2. На следующем шаге нам необходимо записать s_2 и s_3, из-за того что при склейке должен быть общий символ и так далее.

То есть нам нужно иметь записанными мелодии $s_i + s_{i+1}$ для всех $1 \le i < n$. Нам необходимо только посчитать сколько среди них различных, поскольку нет смысла записывать одну мелодию дважды.

Решение

```
def solve():
    n = int(input())
    s = input()
    cnt = set()
    for i in range(1, n):
        cnt.add(s[i - 1] + s[i])
    print(len(cnt))


t = int(input())
for _ in range(t):
    solve()
```

B - Восстанови погоду

Вам дан массив a, содержащий прогноз погоды в Берляндии за последние n дней. То есть, a_i — это предполагаемая температура воздуха в день i ($1 \le i \le n$).

Также вам дан массив b —температура воздуха, которая была в каждый из дней на самом деле. Однако, все значения в массиве b перемешались.

Определите, в какой день была какая температура, если известно, что погода никогда не отличается от прогноза более чем на k градусов. Другими словами, если в день i настоящая температура воздуха равнялась c, то всегда верно равенство $|a_i - c| \le k$.

Например, пусть задан массив $a = [1, 3, 5, 3, 9]$ длины $n = 5$ и $k = 2$ и массив $b = [2, 5, 11, 2, 4]$. Тогда, чтобы значение b_i соответствовало температуре воздуха в день i, можно переставить элементы массива b так: $[2, 2, 5, 4, 11]$. Действительно:

- В 1-й день $|a_1 - b_1| = |1 - 2| = 1$, выполняется $1 \le 2 = k$
- Во 2-й день $|a_2 - b_2| = |3 - 2| = 1$, выполняется $1 \le 2 = k$
- В 3-й день $|a_3 - b_3| = |5 - 5| = 0$, выполняется $0 \le 2 = k$
- В 4-й день $|a_4 - b_4| = |3 - 4| = 1$, выполняется $1 \le 2 = k$
- В 5-й день $|a_5 - b_5| = |9 - 11| = 2$, выполняется $2 \le 2 = k$

Входные данные

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

Далее следуют описания наборов входных данных.

В первой строке каждого набора входных данных записано два целых числа n ($1 \le n \le 10^5$) и k ($0 \le k \le 10^9$) — количество дней и максимальная разница между предполагаемой и реальной температурой воздуха в каждый из дней.

Во второй строке каждого набора входных данных записано ровно n целых чисел — элементы массива a ($-10^9 \le a_i \le 10^9$).

Во третьей строке каждого набора входных данных записано ровно n целых чисел — элементы массива b ($-10^9 \le b_i \le 10^9$).

Гарантируется, что сумма n по всем наборам не превышает 10^5, и что элементы массива b всегда можно переставить таким образом, чтобы равенство $|a_i - b_i| \le k$ было верно для всех i.

Выходные данные

В отдельной строке для каждого набора входных данных выведите ровно n чисел — значения температуры воздуха в каждый из дней в правильном порядке.

Если существует несколько вариантов ответа — выведите любой из них.

Пример

входные данные
3 5 2 1 3 5 3 9 2 5 11 2 4 6 1 -1 3 -2 0 -5 -1 -4 0 -1 4 0 0 3 3 7 7 7 9 4 8

выходные данные
2 2 5 4 11 0 4 -1 0 -4 0 8 4 9

Разбор

Будем решать задачу при помощи жадного алгоритма.

На основе массива a сформируем массив пар {температура, номер дня} и отсортируем его по возрастанию температуры. Также отсортируем массив b по возрастанию. Теперь значения $a[i].first$ и $b[i]$ являются прогнозируемой и реальной температурой в день $a[i].second$.

Действительно, рассмотрим минимальные температуры $b[1]$ и $a[1].first$. Разница между ними это $t = |b[1] - a[1].first|$. Если мы рассмотрим величину $|b[i] - a[1].first|$ либо $|b[1] - a[i].first|$ при $i > 1$, то она будет не меньше t, так как $a[1] \le a[i]$ и $b[1] \le b[i]$.

Так как гарантируется, что переставить элементы в массиве b возможно, а элементы $b[1]$ и $a[1].first$ имеют наименьшую разницу, то она точно не превосходит k.

Решение

```cpp
#include<bits/stdc++.h>
using namespace std;
void  solve(){
    int n, k;
    cin >> n >> k;
    vector<pair<int, int>>a(n);
    vector<int>b(n), ans(n);
    for(int i = 0; i < n; i++){
        cin >> a[i].first;
        a[i].second = i;
    }
    for(auto &i : b) cin >> i;
    sort(b.begin(), b.end());
    sort(a.begin(), a.end());

    for(int i = 0; i < n; i++){
        ans[a[i].second] = b[i];
    }
    for(auto &i : ans) cout << i << ' ';
    cout << endl;
}
int main(){
    int t;
    cin >> t;
    while(t--) solve();
}
```

D - перевертыш

Вам дана перестановка p длины n.

Перестановкой называется массив, состоящий из n различных целых чисел от 1 до n в произвольном порядке. Например, $\{2,3,1,5,4\}$ является перестановкой, а $\{1,2,2\}$ не является (2 встречается дважды), и $\{1,3,4\}$ тоже не является перестановкой ($n = 3$, но в массиве есть 4).

К перестановке p нужно **ровно один** раз применить следующую операцию:

- Сначала вы выбираете отрезок $[l,r]$ ($1 \le l \le r \le n$, отрезок — непрерывная последовательность чисел $\{p_l, p_{l+1}, \dots, p_{r-1}, p_r\}$) и переворачиваете его. Переворот отрезка означает, что меняются местами пары чисел (p_l, p_r), (p_{l+1}, p_{r-1}), ..., (p_{l+i}, p_{r-i}) (где $l + i \le r - i$).
- Затем вы меняете местами префикс и суффикс: $[r+1, n]$ и $[1, l-1]$ (обратите внимание, что эти отрезки могут быть пустыми).

Например, $n = 5, p = \{2,3,1,5,4\}$ и был выбран отрезок $[l = 2, r = 3]$, тогда после переворота отрезка $p = \{2,1,3,5,4\}$, затем поменяются местами отрезки $[4,5]$ и $[1,1]$. Тогда $p = \{5,4,1,3,2\}$. Можно показать, что это максимальный возможный ответ для данной перестановки.

Требуется вывести лексикографически **максимальную** перестановку, которую можно получить после применения **ровно одной** такой операции.

Перестановка a лексикографически больше перестановки b, если существует i ($1 \le i \le n$) такое, что $a_j = b_j$ для $1 \le j < i$ и $a_i > b_i$.

Входные данные

В первой строке входных данных задано единственное целое число t ($1 \le t \le 1000$) — количество наборов входных данных в тесте.

Далее следуют описания наборов входных данных.

В первой строке набора задано одно целое число n ($1 \le n \le 2000$) — размер перестановки.

Во второй строке задано n целых чисел: $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$) — сама перестановка p.

Гарантируется, что сумма значений n по всем тестам не превышает 2000.

Выходные данные

Для каждого набора входных данных в отдельной строке выведите лексикографически **максимальную** перестановку длины n, которую можно получить из p, применив операцию, описанную в условии **ровно один** раз.

Пример

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

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

Примечание

Первый пример разобран в условии.

Во втором примере следует выбрать отрезок $[l = 9, r = 9]$.

В третьем примере следует выбрать отрезок $[l = 1, r = 1]$.

В четвертом примере следует выбрать отрезок $[l = 1, r = 2]$.

В пятом примере следует выбрать отрезок $[l = 5, r = 6]$.

В шестом примере следует выбрать отрезок $[l = 4, r = 4]$.

В седьмом примере следует выбрать отрезок $[l = 5, r = 5]$.

Разбор

В данных ограничениях можно было решать задачу за $O(n^2)$. Давайте заметим, что может быть не больше двух кандидатов на значение r. Так как первым числом в перестановке будет либо p_{r+1}, если $r < n$, либо p_r, если $r = n$. Тогда давайте перебирать значение r и выберем то, в котором первое число в получившейся перестановке будет как можно больше. Дальше если $p_n = n$, то у нас может быть два кандидата на r это $n, n-1$, но заметим, что всегда выгоднее в таком случае положить $r = n$, так как это не испортит ответ. Далее можно перебрать значение l и получим решение за квадрат, но можно поступить умнее.

Заметим теперь, что в ответ уже записаны все числа p_i где $i > r$. А далее записывается $p_r, p_{r-1}, \dots, p_l$ где l пока что неизвестно, а затем $p_1, p_2, \dots, p_{l-1}$. В таком случае давайте запишем p_r так как $l \le r$, а затем будем записывать $p_{r-1}, p_{r-2}, \dots$ до тех пор, пока они больше чем p_1. А иначе сразу определим значение l и допишем ответ до конца. Таким образом, конструктивно построили максимальную перестановку.

Решение

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

using namespace std;

#define forn(i, n) for (int i = 0; i < int(n); i++)
#define sz(v) (int)v.size()
#define all(v) v.begin(),v.end()
#define eb emplace_back
```

```
void solve() {
    int n; cin >> n;
    vector<int> p(n);
    for (auto &e : p) cin >> e;

    int r = 0;
    for (int i = 0; i < n; ++i) {
        if (p[min(n-1, r+1)] <= p[min(n-1, i+1)]) {
            r = i;
        }
    }
    vector<int> ans;
    for (int i = r + 1; i < n; ++i) ans.eb(p[i]);
    ans.eb(p[r]);
    for (int i = r-1; i >= 0; --i) {
        if (p[i] > p[0]) {
            ans.eb(p[i]);
        } else {
            for (int j = 0; j <= i; ++j) {
                ans.eb(p[j]);
            }
            break;
        }
    }
    for (auto e : ans) cout << e << ' ';
    cout << endl;
}
```

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

    forn(tt, t) {
        solve();
    }
}
```

E - хороводы

n людей пришли на праздник и решили станцевать несколько хороводов. В хороводе не менее 2-х людей и у каждого человека есть ровно два соседа, если в хороводе 2 человека, с обоих сторон один и тот же сосед.

Вы решили узнать, сколько именно было хороводов. Но каждый участник праздника запомнил ровно **одного** соседа. Ваша задача — определить, какое минимальное и максимальное число хороводов могло быть.

Например, если на празднике было 6 человек, и номера соседей, которых они запомнили, равны $[2, 1, 4, 3, 6, 5]$ соответственно, то хороводов могло быть **минимум** 1:

- $1 – 2 – 3 – 4 – 5 – 6 – 1$

и **максимум** 3:

- $1 – 2 – 1$
- $3 – 4 – 3$
- $5 – 6 – 5$

Входные данные

Первая строка содержит положительное число t ($1 \le t \le 10^4$) — количество наборов входных данных. Далее следует описание наборов входных данных.

Первая строка описания каждого набора входных данных содержит положительное число n ($2 \le n \le 2 \cdot 10^5$) — количество людей на празднике.

Вторая строка описания каждого набора входных данных содержит n целых чисел a_i ($1 \le a_i \le n, a_i \ne i$) — номер соседа, которого запомнил i-й человек.

Гарантируется, что входные данные корректны и соответствуют хотя бы одному разбиению людей на хороводы.

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

Выходные данные

Для каждого набора входных данных выведите два целых числа — минимальное и максимальное количество хороводов, которое могло быть.

Пример

входные данные
10 6 2 1 4 3 6 5 6 2 3 1 5 6 4 9 2 3 2 5 6 5 8 9 8 2 2 1 4 4 3 2 1 5 2 3 4 5 1 6 5 3 4 1 1 2 5 3 5 4 1 2 6 6 3 2 5 4 3 6 5 1 4 3 4 2

выходные данные
1 3 2 2 1 3 1 1 1 2 1 1 1 1 2 2 1 2 1 1

Разбор

Построим неориентированный граф, проведём рёбра $i \to a_i$. Разобьём этот граф на компоненты связности, пусть их получилось k. Хороводов не могло быть больше k.

Так как степень каждой вершины не больше двух, компоненты связности это простые циклы и бамбуки. Если соединить вершины степени один в каждом бамбуке, получим разбиение на k хороводов.

Теперь попробуем получить минимальное число хороводов. С циклами ничего нельзя сделать, а все бамбуки можно объединить в один. Если получилось b бамбуков и c циклов, то ответом будет $\langle c + \min(b, 1), c + b \rangle$.

Решение работает за $O(n)$.

Решение

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

using namespace std;

typedef long long ll;

void solve() {
    int n;
    cin >> n;
    vector<int> a(n);
    vector<set<int>> g(n);
    vector<set<int>> neighbours(n);
    vector<int> d(n);
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
        a[i]--;
        g[i].insert(a[i]);
        g[a[i]].insert(i);
    }
```

```cpp
for (int i = 0; i < n; ++i) {
    d[i] = g[i].size();
}
int bamboos = 0, cycles = 0;
vector<bool> vis(n);
```

```cpp
for (int i = 0; i < n; ++i) {
    if (!vis[i]) {
        queue<int> q;
        q.push(i);
        vis[i] = true;
        vector<int> component = {i};
        while (!q.empty()) {
            int u = q.front();
            q.pop();
            for (int v: g[u]) {
                if (!vis[v]) {
                    vis[v] = true;
                    q.push(v);
                    component.push_back(v);
                }
            }
        }
        bool bamboo = false;
        for (int j: component) {
            if (d[j] == 1) {
                bamboo = true;
                break;
            }
        }
        if (bamboo) {
            bamboos++;
        } else {
            cycles++;
        }
    }
}
cout << cycles + min(bamboos, 1) << ' ' << cycles + bamboos << '\n';
```

Домашнее задание

Ссылка на MVP бота