Московская олимпиада школьников. Информатика. 8 класс. Отборочный этап, 2023/24

15 дек 2023 г., 23:00 — 17 янв 2024 г., 23:59

№ 1

100 баллов

Пробка

Глеб оказался в автомобильной пробке. Применив зоркий глаз, Глеб насчитал, что перед ним стоят n машин. Также он знает, что зеленый свет горит ровно a секунд и за каждую секунду зеленого света с пробки успевают уехать ровно b машин. Красный свет горит ровно c секунд.

Глеб хочет узнать, сколько еще секунд он будет стоять в пробке, если прямо сейчас загорится зеленый свет.

Формат входных данных

Первая строка входных данных содержит одно целое число n ($1 \le n \le 10^9$) — количество машин, стоящих перед Глебом.

Вторая строка входных данных содержит три целых числа a, b и c ($1 \le a, b, c \le 10^9$).

Формат выходных данных

Выведите единственное число — сколько секунд придется стоять Глебу в пробке.

Обратите внимание, что ответ в этой задаче может быть довольно большим и не помещаться в 32-битные типы данных. Рекомендуется использовать 64-битный тип данных, например long long в C++, long в Java или int64 в Pascal.

Критерии оценивания

В этой задаче каждый тест, кроме тестов из условия, оценивается независимо.

Примеры

```
5
4 2 3
```

```
3
```

```
11
4 1 3
```

```
18
```

```
2
2 1 5
```

```
8
```

```
2
1 2 5
```

```
7
```

Ограничения

Время выполнения: 1 секунда

Память: 256 MB

Код C++

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

using namespace std;
#define int long long

signed main() {
    int n, a, b, c;
    cin >> n >> a >> b >> c;
    int n1 = (n + 1 + b - 1) / b * b;
    int n2 = n1 / b;
    int k1 = (n2 - 1) / a * (a + c);
    cout << k1 + ((n2 - 1) % a + 1);
}

```

№ 2

100 баллов

Пёстрая дата

По текущей дате определите ближайшую следующую дату, запись которой в виде день, месяц и год состоит из различных цифр.

Учтите, что если день или номер месяца меньше 10, то они записываются без лишних нулей. Например, дата «8 марта 2019 года» запишется так: «8.3.2019».

Гарантируется, что изначальная дата корректна.

Формат входных данных

В первой строке задано одно целое число d ($1 \le d \le 31$) — день текущей даты.

Во второй строке задано одно целое число m ($1 \le m \le 12$) — месяц текущей даты.

В третьей строке задано одно целое число y ($1000 \le y \le 10^6$) — год текущей даты.

Формат выходных данных

Выведите одну строку — ближайшую следующую дату, в записи которой все цифры различны.

Критерии оценивания

Все тесты в этой задаче оцениваются независимо.

Примечание

Високосный год — такой год, что в нём 366 дней, а именно добавляется 29 февраля. Год является високосным, если он делится на 400 или делится на 4, но не делится на 100.

Примеры

```
18
12
2018
```

```
4.3.2019
```

```
1
2
3456
```

```
7.2.3456
```

Ограничения

Время выполнения: 1 секунда

Память: 256 MB

Код C++

```cpp
#include <fstream>
#include <iostream>
using namespace std;

bool leap_year(int y) {
	return (y % 4 == 0 && y % 100 != 0 || y % 400 == 0);
}

int day_in_month(int m, int y) {
	if (m == 2)
		if (leap_year(y))
			return 29;
		else
			return 28;
	else if (m == 4 || m == 6 || m == 9 || m == 11)
		return 30;
	else
		return 31;
}

void inc(int & d, int & m, int & y) {
	d += 1;
	int dm = day_in_month(m, y);
	if (d > dm) {
		d = 1;
		m += 1;
		if (m > 12) {
			m = 1;
			y += 1;
		}
	}
}

bool good_data(int d, int m, int y) {
	int a[10];
	for (int i = 0; i < 10; i++) a[i] = 0;
	a[d % 10] += 1;
	if (d > 9)	a[d / 10] += 1;
	a[m % 10] += 1;
	if (m > 9) a[m / 10] += 1;
	while (y > 0) {
		a[y % 10] += 1;
		y /= 10;
	}
	bool good = true;
	for (int elem : a)
		if (elem > 1) {
			good = false;
			break;
		}
	return good;
}
int main() {
	int day, month, year;

	cin >> day >> month >> year;
	inc(day, month, year);
	while (!good_data(day, month, year))
		inc(day, month, year);
	cout << day << '.' << month << '.' << year;

	return 0;
```

```
63    }
64
```

№ 3

100 баллов

Очевидная задача про подотрезок

Дан массив a длины n, а также число k. Требуется найти подотрезок этого массива минимальной длины, который содержит ровно k различных значений. Среди всех таких подотрезков выбрать такой, что его левая граница минимальна.

Формат входных данных

Первая строка содержит два целых числа n и k $(1 \le k \le n \le 2 \cdot 10^5)$ — количество элементов массива и число k.

Вторая строка содержит n целых чисел $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 10^6)$ — элементы массива a.

Формат выходных данных

Выведите два числа — левую и правую границу искомого подотрезка, или число -1, если такого отрезка не существует.

Критерии оценивания

Каждый тест в этой задаче оценивается независимо.

Примеры

```
4 2
1 1 2 3
```

```
2 3
```

```
4 3
1 2 2 1
```

```
-1
```

```
6 3
1 1 2 2 3 3
```

```
2 5
```

Ограничения

Время выполнения: 2 секунды

Память: 256 MB

Код C++

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

using namespace std;

#define ll long long
#define ld long double

ld INF = (ld) 1e20;
int iINF = (int) 1e20;
ll mod = (ll) 1e9 + 7;

int main() {
    ll n, k;
    cin >> n >> k;
    vector<ll> num(1000001, 0), a(n);
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
    }
    int ansl = -2, ansr = -2;
    int now = -1, cnt = 0;
    for (int i = 0; i < n; ++i) {
        while (now + 1 < n && cnt < k) {
            ++now;
            if (num[a[now]] == 0) {
                ++cnt;
            }
            ++num[a[now]];
        }
        if ((now - i < ansr - ansl || ansl == -2) && cnt == k) {
            ansl = i, ansr = now;
        }
        num[a[i]]--;
        cnt -= (num[a[i]] == 0);
    }
    if (ansl == -2) {
        cout << -1;
        return 0;
    }
    cout << ansl + 1 << ' ' << ansr + 1;
}

```

№ 4

100 баллов

Интересная лекция

Сегодня Вася побывал на лекции по массивам от одного известного профессора. Лекция была очень интересная, и домашнее задание было не менее занимательным. Звучало оно так:

Дан массив длины n. Требуется сделать отрезок максимальной длины из одинаковых элементов, используя ***максимум один*** *разворот подотрезка массива.*

Васе понравилась эта задача, но сегодня вечером у него работа, поэтому он не успеет сделать домашнее задание. Но вы, как самый лучший друг Васи, согласились помочь ему решить эту задачу.

Напомним, что разворот подотрезка от l до r массива $a_1, \ldots, a_{l-1}, a_l, \ldots, a_r, a_{r+1}, \ldots, a_n$ делает из него массив $a_1, \ldots, a_{l-1}, a_r, a_{r-1}, \ldots, a_{l+1}, a_l, a_{r+1}, \ldots, a_n$

Формат входных данных

В первой строке дано число n $(1 \le n \le 2 \cdot 10^5)$ — количество элементов массива.

Во второй строке даны n чисел a_i $(1 \le a_i \le 10^6)$ — элементы массива.

Формат выходных данных

Требуется вывести длину самого длинного подотрезка из одинаковых элементов, который получается путем использования максимум одного раза разворота подотрезка массива.

Критерии оценивания

Все тесты в этой задаче оцениваются независимо.

Примеры

```
5
1 3 1 3 1
```

```
2
```

```
6
1 1 3 2 2 2
```

```
3
```

```
6
1 1 3 4 1 1
```

```
4
```

```
5
1 2 3 4 5
```

```
1
```

Ограничения

Время выполнения: 2 секунды

Память: 256 MB

Код C++

```cpp

#include <bits/stdc++.h>

using namespace std;

int main() {
    int n;
    cin >> n;
    vector<int> a (n);
    vector<int> first_max (1000001);
    vector<int> second_max (1000001);
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
    }
    int counter = 1;
    for (int i = 1; i < n; ++i) {
        if (a[i] == a[i - 1]) ++counter;
        else {
            if (counter > first_max[a[i - 1]]) {
                second_max[a[i - 1]] = first_max[a[i - 1]];
                first_max[a[i - 1]] = counter;
            } else if (counter > second_max[a[i - 1]]) {
                second_max[a[i - 1]] = counter;
            }
            counter = 1;
        }
    }
    if (counter > first_max[a[n - 1]]) {
        second_max[a[n - 1]] = first_max[a[n - 1]];
        first_max[a[n - 1]] = counter;
    } else if (counter > second_max[a[n - 1]]) {
        second_max[a[n - 1]] = counter;
    }

    int maximum = 0;
    for (int i = 0; i < 1000001; ++i) {
        maximum = max(maximum, first_max[i] + second_max[i]);
    }

    cout << maximum;
}

```

№ 5

100 баллов

Закрась по правилам

Дан клетчатый прямоугольник из N строк по M клеток, каждая из которых или уже закрашена, или еще не закрашена. Если в каком-либо квадрате размером 2×2 три клетки уже закрашены, то можно закрасить и четвёртую клетку.

Оцените, сколько клеток могут в итоге оказаться закрашенными.

Формат входных данных

Первая строка входных данных содержит два целых числа N и M ($1 \leqslant N, M \leqslant 500$) — количество строк и столбцов в прямоугольнике. Следующие N строк по M символов описывают клетки прямоугольника. Символ «.» соответствует незакрашенной клетке, а «#» — закрашенной клетке. Строки нумеруются от 1 до N, столбцы — от 1 до M.

Формат выходных данных

Выведите одно число — максимальное количество клеток, которые могут оказаться закрашенными.

Критерии оценивания

Каждый тест оценивается независимо. Решения, правильно работающие для ограничений $1 \leqslant N, M \leqslant 50$, будут набирать не менее 60 баллов.

Примеры

```
2 2
##
#.
```

```
4
```

```
3 4
#...
#...
###.
```

```
9
```

```
3 5
...##
#....
#.#..
```

```
5
```

Ограничения

Время выполнения: 2 секунды

Память: 256 MB

Код C++

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

using namespace std;

int table[1000][1000], n, m;
vector < pair < int, int> > st;

int sumSquare(int i, int j)
{
    return table[i][j] + table[i + 1][j] + table[i][j + 1] +
table[i + 1][j + 1];
}
void pushSquare(int i, int j)
{
    if ((0 <= i) && (i < n - 1) && (0 <= j) && (j < m - 1))
        if (sumSquare(i, j) == 3)
            st.push_back(make_pair(i, j));
}

void pushNeighbors(int i, int j)
{
    pushSquare(i - 1, j - 1);
    pushSquare(i, j - 1);
    pushSquare(i - 1, j);
    pushSquare(i, j);
}
void updTile(int i, int j)
{
    if ((0 <= i) && (i < n) && (0 <= j) && (j < m))
        if (!table[i][j])
            table[i][j] = 1;
            pushNeighbors(i, j);
}

void updSquare(int i, int j)
{
    updTile(i, j);
    updTile(i + 1, j);
    updTile(i, j + 1);
    updTile(i + 1, j + 1);
}

int main()
{
   // freopen("f.in", "r", stdin);
  //    freopen("f.out", "w", stdout);
    cin >> n >> m;
    char c;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            cin >> c;
            if (c == '#')
                table[i][j] = 1;
        }
    }
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < m - 1; j++)
            if (sumSquare(i, j) == 3)
```

```
                    st.push_back(make_pair(i, j));
        while (!st.empty())
        {
            pair < int, int > p = st.back();
            st.pop_back();
            if (sumSquare(p.first, p.second))
                updSquare(p.first, p.second);
        }
        int answer = 0;
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
                answer += table[i][j];
        cout << answer << endl;
        return 0;
    }

```

№ 6

100 баллов

Ёжик в матрице

Биологи провели следующий эксперимент.

Поле представляет собой матрицу из r строк и c столбцов. Изначально ёж находится на поле в левом верхнем углу, имеющем координаты $(1, 1)$. Ёж перемещается из текущего поля или в поле, которое находится справа от него, или в поле, которое находится снизу от него, и финиширует в правом нижнем углу.

В некоторых полях расположено по одной ягоде, на остальных полях ягод нет. Проходя через соответствующее поле, ёж съедает ягоду.

Биологи выяснили, что ёж всегда находит такой путь, который позволяет ему съесть наибольшее количество ягод. Назовём ягоду *ключевой*, если в случае, если эту ягоду (и только её!) убрать с поля, наибольшее количество ягод, которые может съесть ёж, уменьшится.

Ваша задача — по заданному расположению ягод посчитать количество ключевых ягод на поле.

Формат входных данных

Первая строка входных данных содержит одно целое число t — количество тестовых примеров ($1 \le t \le 10$).

Каждый тестовый пример имеет следующий формат:

В первой строке тестового примера находятся три натуральных числа r, c и b — количество строк в матрице, количество столбцов в матрице и количество ягод в матрице, соответственно ($1 \le r, c \le 10^9$, $1 \le b \le 2 \cdot 10^5$, $1 \le b \le r \cdot c$).

Каждая из последующих b строк содержит два целых числа rb_i и cb_i ($1 \le rb_i \le r$, $1 \le cb_i \le c$) — координаты очередного поля, содержащего ягоду. Гарантируется, что все эти поля попарно различны.

Формат выходных данных

Для каждого тестового примера в отдельной строке выведите целое число — количество ключевых ягод на поле.

Критерии оценивания

Решения верно работающие для поля, каждая сторона которого не превосходит 100, будут набираать не менее 40 баллов. Тесты оцениваются независимо.

Примеры

```
1
6 5 5
5 5
2 2
3 3
4 4
1 1
```

```
5
```

```
1
4 4 5
2 1
3 1
4 1
1 2
1 3
```

```
3
```

Ограничения

Время выполнения: 3 секунды

Память: 256 MB

Код C++

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

using namespace std;

vector<int> lis(vector<int>& a) {
        int n = a.size();
    vector<int> d(n+1, 2e9);

        vector<int> sol;
        for (int i = 0; i < n; i++) {
                int ind = upper_bound(d.begin(), d.end(), a[i]) - d.begin();
                if (d[ind] != 2e9){
                        sol.push_back(ind+1);
                        d[ind] = a[i];
                } else {
                        d[ind] = a[i];
                        sol.push_back(lower_bound(d.begin(), d.end(), 2e9) - d.begin());
                }
        }

        return sol;
}

int main()
{
        ios_base::sync_with_stdio(false);
        cin.tie(nullptr);
        cout.tie(nullptr);
        cerr.tie(nullptr);

        int t;
        cin >> t;

        for (int e = 0; e < t; ++e){
                int n, m, p;
                cin >> n >> m >> p;

                vector<pair<int, int>> a;
                for (int i = 0; i < p; ++i){
                        int x, y;
                        cin >> x >> y;

                        a.push_back({x, y});
                }

                sort(a.begin(), a.end());

                vector<int> col;
                for (int i = 0; i < p; ++i){
                        col.push_back(a[i].second);
                }

                vector<int> pref = lis(col);

                vector<int> rev;
                for (int i = 0; i < p; ++i){
                        rev.push_back(-1 * a[p-1-i].second);
                }

                vector<int> suf = lis(rev);
```

```
                reverse(suf.begin(), suf.end());

                vector<int> candidates;
                vector<int> cnt(p+1,0);

                int max_lis = *max_element(pref.begin(),
    pref.end());

                for (int i = 0; i < p; ++i){
                        if (pref[i] + suf[i] - 1 == max_lis){
                                candidates.push_back(i);
                                cnt[pref[i]]++;
                        }
                }

                int sol = 0;
                for (int c : candidates){
                        if (cnt[pref[c]] == 1){
                                ++sol;
                        }
                }

                cout << sol << '\n';
        }

}

```

№ 7

100 баллов

Гиревой спорт

В школьном тренажерном зале оказалось n различных гирь, разложенных в некотором порядке так, что они образовали массив a, где значение a_i равно удовольствию, которое можно получить при поднятии гири с номером i (удовольствие также может быть отрицательным, например, если гиря задела самолюбие).

Спортсмен хочет выбрать некоторый подотрезок гирь и поднять их. Удовольствие, которое он получит, будет равно сумме удовольствий гирь на этом подотрезке. Все было бы просто, но при поднятии каждой k-й гири из выбранного диапазона спортсмен теряет x единиц удовольствия.

Например, если $k = 2$, а спортсмен поднял 5 гирь, то он получит удовольствие равное сумме удовольствий от поднятия выбранных гирь минус $2 \cdot x$.

Найдите такой подотрезок, чтобы удовольствие, которое получит спортсмен, будет максимально, и выведите величину этого удовольствия.

Можно не поднимать ни одной гири, если это будет оптимально.

Формат входных данных

Первая строка содержит три целых числа n, k и x ($1 \le n \le 2 \cdot 10^5, 1 \le k \le 5, 1 \le x \le 10^4$) — количество элементов массива, а также числа k и x.

Вторая строка содержит n целых чисел $a_1, a_2, \ldots, a_n$ ($-10^4 \le a_i \le 10^4$) "--- элементы массива a.

Формат выходных данных

Выведите одно число — максимальное удовольствие, которое можно получить при описанных правилах.

Пояснения к примерам

В первом примере оптимально взять отрезок с третьего по шестой элемент, тогда спортсмен получит $14 - 2 + 9 + 4 = 25$ единиц удовольствия от гирь и потеряет 8 единиц удовольствия от третей гири.

Во втором примере отпимально не брать ни одну гирю.

В третьем примере можно взять либо отрезок с первого по второй элемент, либо взять только четрвертый элемент.

Критерии оценивания

Все тесты в этой задаче оцениваются независимо.

Примеры

```
6 3 8
1 -3 14 -2 9 4
```

```
17
```

```
5 5 5
-1 -2 -3 -4 -10
```

```
0
```

```
4 2 5
6 6 -9 7
```

```
7
```

Ограничения

Время выполнения: 2 секунды

Память: 256 MB

Код C++

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

using namespace std;

#define ll long long
#define ld long double

ld INF = (ld) 1e20;
int iINF = (int) 1e20;
ll mod = (ll) 1e9 + 7;

int main() {
    ll n, y, x;
    cin >> n >> y >> x;
    vector<ll> a(n);
    ll maxim[n], sum[n], dp[n];
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
        maxim[i] = 0;
        sum[i] = 0;
        dp[i] = 0;
    }
    for (int i = 0; i < n; ++i) {
        ll now = 0;
        for (int j = 0; j < y - 1; ++j) {
            if (i + j >= n)
                break;
            now += a[i + j];
            if (now > maxim[i]) {
                maxim[i] = now;
            }
        }
        if (i + y - 1 < n) {
            now += a[i + y - 1];
        }
        sum[i] = now;
    }
    ll ans = 0;
    for (int i = n - 1; i >= 0; --i) {
        dp[i] = max(dp[i], maxim[i]);
        if (i + y < n) {
            dp[i] = max(dp[i], sum[i] + dp[i + y] - x);
        }
        ans = max(ans, dp[i]);
    }
    cout << max(ans, (ll) 0);
}

```

№ 8

100 баллов

Каждой твари по паре

Коллекционировать животных можно не только на Ноевом ковчеге, но и на дереве. А дерево, как известно из теории графов, является связным ациклическим графом. Иными словами, дерево состоит из n вершин и $n - 1$ ребра между ними. Каждое ребро соединяет некоторые две вершины дерева. Из любой вершины дерева можно добраться по ребрам до любой другой. Также дерево не содержит циклов, т. е. нельзя построить путь по ребрам из некоторой вершины и вернуться в нее, не пройдя по одному и тому же ребру дважды.

Для расположения животных на нашем дереве сделано q гнезд, где q — четное число. Животным каждого вида предназначены два гнезда.

После расселения по дереву каждая пара одного вида может перемещаться по ребрам дерева между своими гнездами. Они не должны мешать другим видам, т. е. чтобы никакие два пути не имели общего ребра. При этом допускается пересечение путей в вершине. Вам необходимо разместить $\frac{q}{2}$ видов животных по гнездам таким образом, чтобы пути животных одного вида удовлетворяли этому условию.

Формат входных данных

В первой строке входных данных находится два целых числа n и q ($2 \le n \le 2 \cdot 10^5$, $2 \le q \le n$, q четно) — общее количество вершин в дереве и количество вершин, в которых расположены гнезда.

Во второй строке входных данных находятся q различных целых чисел m_i ($1 \le m_i \le n$) — номера вершин с гнездами.

В каждой из следующих $n - 1$ строк входных данных находится два целых u_i и v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$). Это означает, что между вершинами u_i и v_i в дереве проведено ребро.

Гарантируется, что ребра во входных данных образуют дерево.

Формат выходных данных

Выходные данные должны содержать $\frac{q}{2}$ строк.

В каждой строке должно содержаться два целых числа x_i и y_i ($1 \le x_i, y_i \le n$) — номера вершин, в которых будут располагаться гнезда i-го вида животных.

Все числа в выходных данных должны быть различны и должны представлять собой номера вершин с гнёздами.

Если разместить животных требуемым образом невозможно, выведите число -1.

Критерии оценивания

Баллы начисляются за каждый тест в отдельности, но ряд тестов имеют специфическую структуру:

1. $u_i = i$, $v_i = i + 1$
2. $u_i = 1$, $v_i = i + 1$
3. $n \le 3000$, $q \le 6$
4. Нет дополнительных ограничений.

Примеры

```
9 6
1 3 2 5 8 6
3 1
3 2
3 4
3 5
3 6
4 7
```

```
7 8
1 9
```

```
3 2
8 5
1 6
```

Ограничения

Время выполнения: 1 секунда

Память: 256 MB

Код C++

```cpp
#include <vector>
#include <array>
#include <iostream>
using namespace std;
const int N = 400100;
vector<int> g[N];
vector<array<int,2>> pairs;
int n, q, paired[N];

void make_paired(int v, int p) {
  for (int u : g[v]) {
    if (u == p) continue;

    make_paired(u, v);

    if (paired[u] >= 0) {
      if (paired[v] >= 0) {
        pairs.push_back({paired[v], paired[u]});
        paired[v] = -1;
        paired[u] = -1;
      } else {
        paired[v] = paired[u];
        paired[u] = -1;
      }
    }
  }
}

int main() {
  ios_base::sync_with_stdio(0); cin.tie(0);

  cin >> n >> q;
  fill(paired, paired + n, -1);
  for (int i = 0; i < q; i++) {
    int v; cin >> v;
    v--;
    paired[v] = v;
  }
  for (int i = 1; i < n; i++) {
    int x, y; cin >> x >> y;
    x--; y--;
    g[x].push_back(y);
    g[y].push_back(x);
  }

  make_paired(0, -1);

  for (auto pair : pairs) {
    cout << pair[0] + 1 << " " << pair[1] + 1 << '\n';
  }

  return 0;
}

```