Московская олимпиада школьников по информатике 2025–2026 8 класс (второй отбор)

4 янв 2026 г., 12:00 — 15 янв 2026 г., 23:59

№ 1

100 баллов

Карусель

Не начинайте решение задач, пока вы не включили запись экрана с видео участника, см. регламент олимпиады, без корректной записи работа не будет засчитана

В парке аттракционов есть карусель, которая вмещает в себя k человек. Одно катание на карусели длится t минут. Сейчас h часов m минут, Петя n-й в очереди на аттракцион. Ему интересно, успеет ли он прокатиться до того, как парк закроется в h_2 часов m_2 минут. А если успеет, то сколько времени ему придется ждать.

Карусель работает с текущего момента до времени закрытия включительно.

Входные данные

В первой строке находятся три числа k, t, n, $(1 \le k, t, n \le 10^9)$ — вместимость карусели, продолжительность катания и номер Пети в очереди соответственно.

Во второй строке находятся два числа h, m — текущее время.

В третьей строке находятся два числа h_2, m_2 количество часов и минут время закрытия парка $(0 \le h, h_2 < 24)$, $(0 \le m, m_2 < 60)$.

Числа в строках разделены одним пробелом.

Гарантируется, что текущий момент времени строго меньше времени закрытия.

Выходные данные

Если Петя не дождется своей очереди, выведите -1.

Иначе выведите сколько времени в минутах Пете придется ждать своей очереди.

Примеры

```
4 2 5
10 5
10 10
```

```
2
```

```
4 4 5
10 5
10 10
```

```
-1
```

Ограничения

Время выполнения: 1 секунда

Память: 256 MB

Код C++

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

using namespace std;

int main()
{
    int k, t, n, h1, m1, h2, m2;
    cin >> k >> t >> n;
    cin >> h1 >> m1;
    cin >> h2 >> m2;
    int m = (h2 * 60 + m2) - (h1 * 60 + m1) + 1;
    if (m / t >= n / k + (n % k != 0))
        cout << n / k * t - (n % k == 0) * t;
    else
        cout << -1;
    return 0;
}

```

№ 2

100 баллов

Будни авиадиспетчера

Не начинайте решение задач, пока вы не включили запись экрана с видео участника, см. регламент олимпиады, без корректной записи работа не будет засчитана

Максим работает авиадиспетчером в аэропорту с единственной взлетно-посадочной полосой (ВПП). Сегодня ему предстоит посадить n самолетов, и ваша задача — рассчитать время завершения этого процесса.

Правила посадки:

1. Продолжительность: Процесс посадки одного самолета занимает ровно b минут. В это время ВПП считается занятой.
2. Очередь и круги: Если самолет прибывает к аэропорту, а ВПП занята, он отправляется на дополнительный круг. Один круг занимает f минут. После завершения круга самолет снова оказывается у ВПП и проверяет её готовность.
3. Свободная полоса: Если в момент прибытия (или возвращения с круга) ВПП свободна, самолет немедленно приступает к посадке.
4. Конфликты: Если несколько самолетов оказываются у свободной ВПП одновременно, один из них (любой) начинает посадку, а остальные немедленно отправляются на дополнительный круг длительностью f минут.

Задание: Определите момент времени, когда последний самолет завершит свою посадку.

Входные данные

Первая строка входных данных содержит три целых числа n, b и f ($1 \le n \le 1000, 1 \le b, f \le 10^9$).

Вторая строка входных данных содержит n целых чисел t_i ($0 \le ti \le 10^9$) — времена первоначального прибытия самолетов (в произвольном порядке).

Выходные данные

Выведите одно число — время, в которое последний самолет закончит посадку.

Примеры

```
10 10 12
13 0 1 10 20 20 2 1 10 20
```

```
120
```

Ограничения

Время выполнения: 1 секунда
Память: 256 MB

Код C++

```cpp
#include <iostream>
#include <map>
#include <set>
using namespace std;
#define ll long long
int main() {
    ll n, b, f; cin >> n >> b >> f;
    multiset<ll> s;
    for (int i = 0; i < n; i++) {
        ll x; cin >> x;
        s.insert(x);
    }
    ll cur = 0;
    while (s.size() != 0) {
        ll x = *(s.begin());
        s.erase(s.begin());
        if (x >= cur) cur = x + b;
        else {
            ll t = (cur - x) / f;
            if ((cur - x) % f != 0) t++;
            s.insert(t * f + x);
        }
    }
    cout << cur;
}
```

№ 3

100 баллов

Готовим к ЕГЭ по информатике старших братьев

Не начинайте решение задач, пока вы не включили запись экрана с видео участника, см. регламент олимпиады, без корректной записи работа не будет засчитана

В ЕГЭ по информатике может встретиться следующая задача. Дана последовательность прописных латинских букв и цифр. Определить максимальную длину подряд идущих цифр, сумма которых делится на k.

Входные данные

Первая строка входных данных содержит одно целое положительное число k, на которое должна делиться сумма последовательности из цифр. ($1 \leq k \leq 10^3$).

Вторая строка содержит непустую строку s. Строка состоит из прописных латинских букв и цифр. Длина строки s не превышает 10^6.

Выходные данные

Выведите одно целое число — длину искомой последовательности.

Система оценивания

В этой задаче три группы тестов. В превой группе длина строки не превосходят 100, во второй группе — 1000, в третьей группе нет дополнительных ограничений. Баллы за группу можно получить только при прохождении всех тестов группы и всех тестов предыдущих групп.

Примеры

```
3
ab99c112111d
```

```
5
```

```
3
ab7c11d
```

```
0
```

```
1
x999yyy0123456789
```

```
10
```

Ограничения

Время выполнения: 3 секунды

Память: 256 MB

Код C++

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include <map>
#include <set>
#include <unordered_map>
#include <queue>

using namespace std;

#define int long long

bool isdigit(char c){
    return c >= '0' && c <= '9';
}

signed main()
{
    ios::sync_with_stdio(0);
    cout.tie(0);
    cin.tie(0);

    int k;
    cin >> k;

    string s;cin >> s;
    int n = s.size();

    int ans = 0;
    for(int i = 0; i < n; i ++){
        if(!isdigit(s[i]))continue;

        int j = i;
        int sum = s[i] - '0';
        unordered_map<int, int> mp;
        mp[0] = i - 1;

        if(sum % k == 0){
            ans = max(ans, 1ll);
        }
        else{
            mp[sum%k] = i;
        }

        while(j + 1 < n && isdigit(s[j+1])){
            j++;
            sum += s[j] - '0';
            if(mp.count(sum % k))ans = max(ans, j - mp[sum%k]);
            else mp[sum%k] = j;
        }

        i = j;
    }

    cout << ans;
    return 0;
}
```

№ 4

100 баллов

Восстановление исходного массива

Не начинайте решение задач, пока вы не включили запись экрана с видео участника, см. регламент олимпиады, без корректной записи работа не будет засчитана

Представьте, что у вас был массив из N натуральных чисел $a_1, a_2, \ldots, a_N$. На его основе вы построили таблицу $N \times N$ по следующим правилам:

1. В ячейках на главной диагонали (где номер строки i совпадает с номером столбца j) всегда записывается 0.
2. В остальных ячейках (где $i \neq j$) записывается максимальное из двух чисел: a_i и a_j.

Вам дана уже готовая таблица. Требуется «расшифровать» её и найти исходный массив $a_1, \ldots, a_N$. Если такая таблица не могла быть получена по этим правилам, выведите -1.

Входные данные

В первой строке находится число N ($1 \leq N \leq 500$) — размер таблицы. В последующих N строках находятся по N целых чисел (от 0 до 1000) — сама таблица.

Входные данные

Если решение существует, выведите N чисел через пробел. Если решений несколько, выведите любое. Если решения нет, выведите -1.

Примеры

```
3
0 4 6
4 0 6
6 6 0
```

```
4 4 6
```

Ограничения

Время выполнения: 1 секунда

Память: 256 MB

Код C++

```cpp
#include <iostream>
#include <algorithm>
#include <cmath>
#include <string>
#include <vector>
using namespace std;
typedef long long ll;
const int MAX_N = 2e3+6;
const double INF = 1e9+1;
int main() {
    int n; cin >> n;
    vector<vector<int>> matrix(n, vector<int>(n));
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> matrix[i][j];
        }
    }
    vector<int> a(n);
    for (int i = 0; i < n; i++) {
        int mn = 100001;
        for (int j = 0; j < n; j++) {
            if (j == i) continue;
            mn = min(mn, matrix[i][j]);
            if (matrix[i][j] != matrix[j][i]) {
                cout << -1 << endl;
                return 0;
            }
        }
        if (mn == 100001 || mn == 0) mn = 1;
        a[i] = mn;
        if (matrix[i][i] != 0) {
            cout << -1 << endl;
            return 0;
        }
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (i == j) continue;
            if (matrix[i][j] != max(a[i], a[j])) {
                cout << -1 << endl;
                return 0;
            }
        }
    }
    for (int x : a) {
        cout << x << ' ';
    }
}
```

№ 5

100 баллов

Реверс и неподвижные точки

Не начинайте решение задач, пока вы не включили запись экрана с видео участника, см. регламент олимпиады, без корректной записи работа не будет засчитана

В руке у Мате находится N карт, на которых написаны числа — перестановка чисел от 1 до N. Карты лежат в фиксированном порядке. Неподвижной точкой называется ситуация, когда число на карте совпадает с её порядковым номером в руке (считая слева направо с 1).

Домагой может один раз выбрать любой непрерывный отрезок карт и развернуть его (первая карта отрезка меняется местами с последней, вторая – с предпоследней и так далее).

Задание: Помогите Домагою выбрать такой отрезок для разворота, чтобы общее количество неподвижных точек во всей руке стало максимально возможным.

Решения, верно работающие при $N \leq 500$ наберут не менее 30 баллов.

Решения, верно работающие при $N \leq 5000$ наберут не менее 60 баллов.

Входные данные

Число N ($1 \leq N \leq 5 \cdot 10^5$). Перестановка из N различных целых чисел от 1 до N.

Выходные данные

Выведите через пробел два числа: значение первой и значение последней карты в выбранном оптимальном отрезке.

Если решений несколько, выведите любое.

Если разворот не может увеличить число точек, можно вывести значения, соответствующие отрезку из одной любой карты.

Примеры

```
4
3 2 1 4
```

```
3 1
```

```
2
1 2
```

```
1 1
```

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

```
6 1
```

Ограничения

Время выполнения: 1 секунда

Память: 256 MB

Код C++

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

#define FOR(i, a, b) for (int (i) = (a); (i) < (b); i++)
#define REP(i, n) FOR(i, 0, n)
#define TRACE(x) cerr << #x << " " << x << endl
#define eb emplace_back
using namespace std;

const int MaxN = 1000100;

//everything is 1-indexed

int N;
int perm[MaxN];
vector <int> radii[2 * MaxN]; // center and radius are doubled
int prefFixedPoints[MaxN];


void findSegmentBounds(const int center, const int radius, int& left, int& right){
        left = (center - radius) / 2;
        right = (center + radius) / 2;
}

int fixedPointsInSegment(int a, int b){ // [a, b]
        return prefFixedPoints[b] - prefFixedPoints[a - 1];
}

void precomputePrefFixedPoints(){
        for (int i = 1; i <= N; i++)
                prefFixedPoints[i] = prefFixedPoints[i - 1] + ( perm[i] == i );
}

void findCenters(){
        for (int i = 1; i <= N; i++)
                        radii[perm[i] + i].emplace_back(abs(perm[i] - i));

        for (int center = 2; center <= N + N; center++){ // all possible centers
                sort(radii[center].begin(), radii[center].end()); // the log in the complexity may be evaded by smart insertion
        }
}

void findBestFlip(){
        int targetMax = -MaxN, targetLeft = -1, targetRight = -1;
        for (int center = 2; center <= N + N; center++){
                int numOfCreatedFixedPoints = 0;
                for (auto radius: radii[center]){
                        numOfCreatedFixedPoints++;

                        int left, right;
                        findSegmentBounds(center, radius, left, right);

                        int numOfLostFixedPoints = fixedPointsInSegment(left, right);      // the center as the fixed point is counted both as a lost and created
                        int res = numOfCreatedFixedPoints - numOfLostFixedPoints; // so the net gain from it is zero
```

```
54
55                          if (targetMax < res){
56                                  targetMax = res;
57                                  targetLeft = left;
58                                  targetRight = right;
59                          }
60                  }
61          }
62
63          printf("%d %d\n", perm[targetLeft], perm[targetRight]);
64  }
65
66  void load(){
67          scanf("%d", &N);
68          for (int i = 1; i <= N; i++)
69                  scanf("%d", perm + i);
70  }
71
72  int main(){
73          load();
74          precomputePrefFixedPoints();
75          findCenters();
76          findBestFlip();
77          return 0;
78  }
```

№ 6

100 баллов

Пересчитай лестницы

Не начинайте решение задач, пока вы не включили запись экрана с видео участника, см. регламент олимпиады, без корректной записи работа не будет засчитана

Мемориальную лестницу было решено изготовить в виде возрастающей по высоте последовательности каменных ступенек. То есть каждая следующая ступенька строго выше предыдущей. Пусть у нас есть n каменоломен, в каждой из которых изготовлено ровно k каменных ступенек. Из каждой каменоломни было решено взять ровно одну ступеньку, причём ступенька, выбранная из i-й каменоломни, должна быть ниже, чем ступенька, выбранная из $(i+1)$-й. Определите количество способов выбрать ступеньки в каменоломнях таким образом.

Ответ может быть очень большим, посчитайте его по модулю 998244353.

Входные данные

В первой строке входных данных расположены два целых числа n и k — число каменоломен и количество ступенек в каждой из них, $1 \le n \le 100$ и $1 \le k \le 10\,000$.

В следующих n строках находится описание ступенек в каменоломнях. Каждая каменоломня описывается k целыми числами в одной строке $h_1, \dots h_k$ — высоты ступенек в ней, $1 \le h_1 < h_2 < \dots < h_k \le 10^9$.

Выходные данные

Выведите единственное целое число — количество возрастающих лестниц по модулю 998244353.

Система оценивания

В этой задаче три группы тестов. В превой группе n и k не превосходят 10, во второй группе — 100, в третьей группе нет дополнительных ограничений. Баллы за группу можно получить только при прохождении всех тестов группы и всех тестов предыдущих групп.

Примеры

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

```
2
```

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

```
0
```

Ограничения

Время выполнения: 2 секунды

Память: 256 MB

Код C++

```cpp
#define _CRT_SECURE_NO_WARNINGS

#pragma comment(linker, "/stack:200000000")
#pragma GCC optimize("Ofast")
#pragma GCC
target_tbl("sse,sse2,sse3,ssse3,sse4,popcnt,abm,mmx,avx,tune=native
")


//#include "stdafx.h"
#include "iostream"
#include "fstream"
#include "algorithm"
#include "iomanip"
#include "stack"
#include "queue"
#include "string"
#include "vector"
#include "map"
#include "set"
#include "list"
#include "deque"
#include "complex"
#include "bitset"
#include "cmath"
#include "unordered_set"
#include "unordered_map"
#include "iterator"
#include <ctime>
#include <cassert>
#include "numeric"
#include <cstdio>
#include "random"
#include "chrono"
#include "cstring"


using namespace std;

#define maxi(a,b) a = max(a, b);
#define mini(a,b) a = min(a, b);

////////////////
#define endl '\n'
////////////////

#define all(a) a.begin(), a.end()
#define rall(a) a.rbegin(), a.rend()
#define sqr(x) ((x) * (x))
#define SZ(a) ((int)(a.size()))
#define watch(x) cout << (#x) << " = " << x << endl;
//typedef long long ll;
typedef long double ld;

//#define int unsigned int
#define int long long
#define double ld
typedef pair<int, int> pii;
typedef pair<double, double> pdd;
typedef vector<int> vi;
typedef vector<double> vd;
typedef vector<pii> vpii;
```

```cpp
typedef vector<vi> vvi;

template<class T>
void show(const vector<T> &a)
{
	for (T x : a)
		cout << x << " ";
	cout << endl;
}
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
mt19937_64 rng_64(chrono::steady_clock::now().time_since_epoch().count());

int xx[8] = { 1, -1, 0, 0, 1, -1, 1, -1 };
int yy[8] = { 0, 0, 1, -1, -1, -1, 1, 1 };
string dir = "RDLU";
string travel = "ENWS";

const int N = 4e5 + 50, oo = 1e18 + 500;
const int mod = 998244353;
//const int mod = 1e9 + 7;
//const int mod = 1e9 + 9;
const int M2 = 1000000093, x2 = 27162;
const int M1 = 1000000087, x1 = 241;
const double eps = 1e-18, PI = 2 * acos(0.0);


signed main()
{
	ios::sync_with_stdio(0);
	cout.tie(0); cin.tie(0);
	//freopen("input.txt", "r", stdin);
	//freopen("output.txt", "w", stdout);
	//freopen("paintbarn.in", "r", stdin);
	//freopen("paintbarn.out", "w", stdout);

	int n, k;
	cin >> n >> k;

	vvi a(n, vi(k));
	for (int i = 0; i < n; i++)
	{
		for (int j = 0; j < k; j++)
		{
			cin >> a[i][j];
		}
	}

	vvi dp(n, vi(k, 0));
	for (int j = 0; j < k; j++)
	{
		dp[0][j] = j + 1;
	}

	for (int i = 1; i < n; i++)
	{
		for (int j = 0; j < k; j++)
		{
			int jj = lower_bound(all(a[i - 1]), a[i][j]) - a[i - 1].begin();
			jj--;

```

```
121                         if (jj >= 0)
122                         {
123                                 dp[i][j] += dp[i - 1][jj];
124                                 if (dp[i][j] >= mod)
125                                         dp[i][j] -= mod;
126                         }
127                 }
128
129                 for (int j = 1; j < k; j++)
130                 {
131                         dp[i][j] += dp[i][j-1];
132                         if (dp[i][j] >= mod)
133                                 dp[i][j] -= mod;
134                 }
135         }
136
137         /*int ans = 0;
138         for (int j = 0; j < k; j++)
139         {
140                 ans = (ans + dp[n - 1][j]) % mod;
141         }
142         cout << ans;*/
143
144         cout << dp[n - 1][k - 1];
145 }
```