Максимальное количество баллов за олимпиаду — 600

Задание 1. Математика в чат-боте

Дима выбрал два натуральных числа a и b. Затем он отправил модели ИИ запрос — вычислить a^b. Он записал это выражение на бумажке, сфотографировал и загрузил фотографию. Из–за неаккуратного почерка модель распознала выражение как $a \cdot b$ и посчитала именно его. В итоге её ответ оказался меньше правильного на 110.

Какой ответ выдала модель?

Ответ: 15

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. Из условия мы получаем, что $a^b - ab = 110$. Значит, 110 делится на a. Это оставляет варианты $a = 1$, 2, 5, 10, 11, 22, 55, 110.

$a = 1$: $1 - b = 110$ — нет решений.

$a = 2$: $2^{b-1} - b = 55$. Левая часть меньше 55 при $b \le 6$ и больше 55 при $b \ge 7$.

$a = 5$: $5^{b-1} - b = 22$. Левая часть меньше 22 при $b \le 2$, равна 22 при $b = 3$, больше 22 при $b \ge 4$.

$a = 10$: $10^{b-1} - b = 11$. Левая часть больше 11 при $b \ge 3$, $b = 1$, 2 не подходят. $a = 11$, 22, 55, 110 разбираются аналогично предыдущему случаю.

Задание 2. Максимальный след по всем перестановкам

Матрицей $n \times m$ будем называть таблицу из чисел, состоящую из n строк и m столбцов. Умножение матриц выполняют по правилу «строка на столбец». Если

$$M = \begin{pmatrix} p & q \\ r & s \end{pmatrix}, \qquad N = \begin{pmatrix} u & v \\ w & x \end{pmatrix},$$

то

$$MN = \begin{pmatrix} pu + qw & pv + qx \\ ru + sw & rv + sx \end{pmatrix}.$$

Суммой диагональных элементов (следом) матрицы называют число $tr\begin{pmatrix} p & q \\ r & s \end{pmatrix} = p + s$.

Дана матрица

$$A = \begin{pmatrix} -1 & 4 \\ 5 & 10 \end{pmatrix}.$$

Также даны числа -4, -5, 20, 25. Рассматриваются все 24 матрицы B вида

$$B = \begin{pmatrix} x & y \\ z & w \end{pmatrix},$$

в которых x, y, z, w — некоторая перестановка чисел -4, -5, 20, 25. Для каждой такой B рассмотрите произведения AB и BA.

а) Найдите наибольшее возможное значение $tr(AB)$.

Ответ: 339

Критерий оценивания: точное совпадение ответа — 50 баллов

б) Найдите наибольшее возможное значение $tr(BA)$.

Ответ: 339

Критерий оценивания: точное совпадение ответа — 50 баллов

Максимальный балл за задание — 100

Решение. Полезное свойство: для любых квадратных матриц одинакового размера $\mathrm{tr}(AB) = \mathrm{tr}(BA)$. Поэтому достаточно максимизировать $\mathrm{tr}(AB)$.

Пусть $B = \begin{pmatrix} x & y \\ z & w \end{pmatrix}$. Тогда

$$AB = \begin{pmatrix} -1 & 4 \\ 5 & 10 \end{pmatrix}\begin{pmatrix} x & y \\ z & w \end{pmatrix} = \begin{pmatrix} -x + 4z & -y + 4w \\ 5x + 10z & 5y + 10w \end{pmatrix},$$

и потому

$$\mathrm{tr}(AB) = (-x + 4z) + (5y + 10w) = -x + 4z + 5y + 10w.$$

Нужно распределить -5, -4, 20, 25 по x, y, z, w, чтобы максимизировать линейную форму с коэффициентами -1, 5, 4, 10 соответственно. По неравенству о перестановках максимум достигается, когда наибольшему коэффициенту сопоставлено наибольшее число, и так далее по убыванию:

$$10 \leftrightarrow 25, \quad 5 \leftrightarrow 20, \quad 4 \leftrightarrow (-4), \quad (-1) \leftrightarrow (-5).$$

То есть $w = 25$, $y = 20$, $z = -4$, $x = -5$. Тогда

$$\mathrm{tr}(AB) = -(-5) + 4 \cdot (-4) + 5 \cdot 20 + 10 \cdot 25 = 5 - 16 + 100 + 250 = 339.$$

Из свойства $\mathrm{tr}(AB) = \mathrm{tr}(BA)$ такое же значение получается и для BA.

Итак, наибольшая возможная сумма диагональных элементов равна 339.

Задание 3. Минимизация L_1 и L_2

Вы настраиваете простейшую регрессионную модель, которая всегда предсказывает одно и то же число c (константная модель). Дан набор истинных значений:

$$y = \{1, 2, 3, 9, 10, 10\}.$$

Рассмотрим две функции качества: $L_1(c) = \sum_i |y_i - c|$ и $L_2(c) = \sum_i (y_i - c)^2$.

а) Найдите значение c_1, минимизирующее $L_1(c)$. Если оптимальных значений несколько, в ответ запишите наименьшее.

Ответ: 3

Критерий оценивания: точное совпадение ответа — 50 баллов

б) Найдите значение c_2, минимизирующее $L_2(c)$. Если оптимальных значений несколько, в ответ запишите наименьшее.

Ответ: 35/6

Критерий оценивания: точное совпадение ответа — 50 баллов

Максимальный балл за задание — 100

Решение. Отсортируем значения: 1, 2, 3, 9, 10, 10.

а) *Минимум L_1.*

$L_1(c)$ минимизируется при *медиане* выборки. При чётном числе элементов множество оптимумов — это весь отрезок между двумя серединными значениями. Здесь серединные значения 3 и 9, значит оптимумы $c \in [3, 9]$. По правилу задачи берём наименьшее:

$$c_1 = 3.$$

б) *Минимум L_2.*

$L_2(c)$ — квадратичная парабола; минимум достигается в *среднем*:

$$c_2 = \frac{1 + 2 + 3 + 9 + 10 + 10}{6} = \frac{35}{6}.$$

(Оптимум единственный, так что указание на «наименьший» здесь ничего не меняет.)

Задание 4. Group By

Система электронного тестирования фиксирует результаты проверочных работ школьников, которые вы можете скачать в форматах XLSX, ODS или CSV. Файл содержит пять столбцов:

- `student_id` — идентификатор ученика (целое число);
- `subject` — предмет, по которому выполнен тест (строка);
- `score` — полученный балл (вещественное число);
- `cheat_flag` — подозрение на списывание (True/False);
- `attempt_no` — номер попытки (целое число, 1 означает первую попытку).

Выполните следующие действия с данными:

1. Очистите столбец score: пустые значения замените на 0, значения меньше 0 также замените на 0, значения больше 100 замените на 100.
2. Удалите строки, где `cheat_flag` = True.
3. Оставьте только строки, соответствующие первой попытке, то есть те, где `attempt_no` = 1.
4. Для каждого ученика вычислите его средний балл по предметам — это среднее всех значений `score`, которые остались после действий выше.

Сколько учеников имеют средний балл в диапазоне $60 \leqslant$ `avg_score` < 80?

Ответ: 363

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. Решение задачи на языке Python:

```python
import pandas as pd

df = pd.read_csv("tests.csv")
df["score"] = df["score"].fillna(0)

mask_neg = df["score"] < 0
df.loc[mask_neg, "score"] = 0

mask_high = df["score"] > 100
df.loc[mask_high, "score"] = 100

df = df[df["cheat_flag"] == False]

df = df[df["attempt_no"] == 1]

mean_by_student = df.groupby("student_id")["score"].mean()

cond = (mean_by_student >= 60) & (mean_by_student < 80)
answer = cond.sum()
print(answer)
```

Задание 5. Дерево непринятия решений

Ограничение по времени: 1 секунда

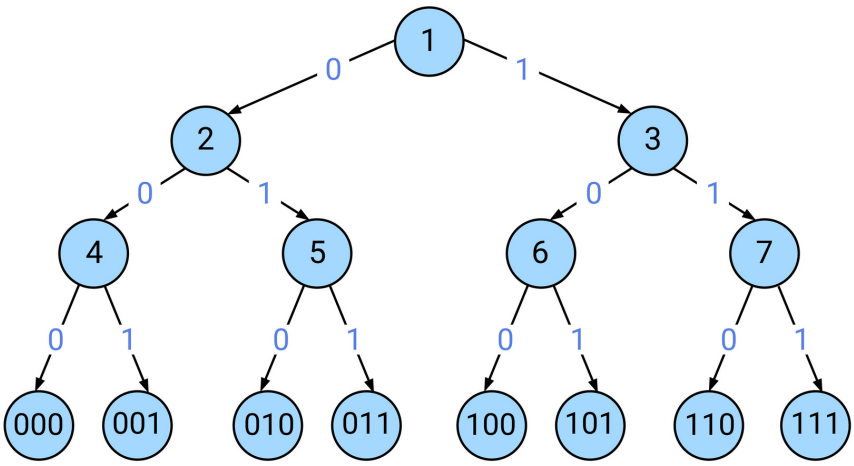
Ограничение по памяти: 256 мегабайт

В машинном обучении часто используют деревья решений. Каждая внутренняя вершина такого дерева соответствует некоторому вопросу, а каждое ребро — выбору ответа (да/нет). Таким образом, за несколько вопросов исходные данные могут быть разбиты на достаточно большое количество классов. Очередным проектом для Димы стало дерево непринятия решений. Его структура похожа на структуру решающего дерева. Это полное бинарное дерево глубины n. В каждой вершине, кроме вершин последнего уровня, хранится число p ($0 \leqslant p \leqslant 100$). Это число обозначает вероятность выбора: с вероятностью p процентов алгоритм выберет пойти влево и, соответственно, с вероятностью $100 - p$ процентов — вправо.

Договоримся: движение влево обозначим цифрой 0, а движение вправо — цифрой 1. Таким образом, каждая вершина нижнего уровня соответствует двоичной строке длины n (последовательности решений от корня до листа).

Вероятность получения этой строки равна произведению вероятностей всех выборов, сделанных на пути от корня до листа.

Дима уже написал структуру для такого дерева и хочет протестировать её. Для этого он создал дерево глубины 3. Значит, всего в нём 7 внутренних вершин. Каждая вершина имеет своё число p. Ниже показана схема расположения этих вершин:



Найдите вероятности всех двоичных строк длины 3 и выведите их в порядке неубывания вероятности. Если вероятности совпадают, строки должны выводиться в лексикографическом порядке.

Формат входных данных

В первой строке заданы 7 целых чисел p ($0 \leqslant p \leqslant 100$) — вероятности для вершин, как показано на рисунке.

Формат выходных данных

Выведите 8 строк. Каждая строка должна содержать двоичную строку длины 3. Строки должны идти в порядке неубывания вероятности. При равенстве вероятностей строки сравниваются лексикографически.

Примеры

стандартный ввод	стандартный вывод
40 90 20 90 100 70 0	011 110 001 101 010 100 000 111

Замечание

В первом тестовом примере двоичные строки имеют следующие вероятности:

- 011 – 0.0
- 110 – 0.0
- 001 – 0.036
- 101 – 0.036
- 010 – 0.04
- 100 – 0.084
- 000 – 0.324
- 111 – 0.48

Решение

В задаче нужно посчитать вероятность для каждого листа. Так как каждая такая вероятность является произведением трёх чисел (вероятностей выбора в вершинах), для сравнения достаточно сравнивать произведения этих чисел, умноженных на 100 (то есть вероятности в процентах). Давайте явно выпишем все 8 таких произведений и для каждого запомним соответствующую строку из трёх бит. После этого отсортируем пары (вероятность, строка) по вероятности и выведем строки в получившемся порядке. Это и будет ответом.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int p1, p2, p3, p4, p5, p6, p7;
    cin >> p1 >> p2 >> p3 >> p4 >> p5 >> p6 >> p7;

    int p000 = p1 * p2 * p4;
    int p001 = p1 * p2 * (100 - p4);
    int p010 = p1 * (100 - p2) * p5;
    int p011 = p1 * (100 - p2) * (100 - p5);
    int p100 = (100 - p1) * p3 * p6;
```

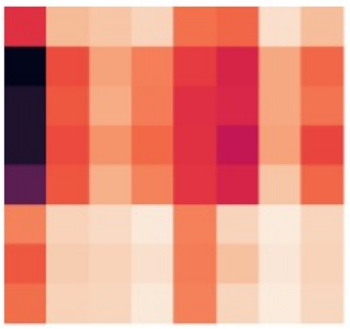
```
    int p101 = (100 - p1) * p3 * (100 - p6);
    int p110 = (100 - p1) * (100 - p3) * p7;
    int p111 = (100 - p1) * (100 - p3) * (100 - p7);

    vector<pair<int, string>> v = {
        {p000, "000"},
        {p001, "001"},
        {p010, "010"},
        {p011, "011"},
        {p100, "100"},
        {p101, "101"},
        {p110, "110"},
        {p111, "111"}
    };

    sort(v.begin(), v.end());
    for (auto &[p, s] : v) {
        cout << s << '\n';
    }
    return 0;
}
```

Максимальный балл за задание — 100

Задание 6. Тепловая карта

Ограничение по времени: 1 секунда

Ограничение по памяти: 256 мегабайт



Слава готовит постер на конференцию. К сожалению, сейчас его тепловая карта не влезает на постер: она слишком большая. Поэтому Слава решил выделить на текущей карте некоторый прямоугольный фрагмент и использовать его для презентации. Славина тепловая карта выглядит как таблица из n строк и m столбцов. Клетка на пересечении i-й строки и j-го столбца имеет цвет c_{ij}. Используются k цветов, которые пронумерованы от 1 до k. Пример аналогичной карты приведён справа. Слава хочет продемонстрировать весь размах значений, поэтому на выбранном фрагменте должна быть хотя бы одна клетка каждого цвета. При этом юный докладчик хочет минимизировать площадь карты, ведь ему нужно уместить её на постер.

Помогите ему: найдите прямоугольник, который можно будет вырезать из его карты так, чтобы на нём были клетки всех k цветов. Гарантируется, что на исходной тепловой карте присутствуют клетки всех k цветов.

Формат входных данных

В первой строке вводятся три натуральных числа n, m, k ($1 \leqslant n$, $m \leqslant 250$, $1 \leqslant k \leqslant 20$).

В следующих n строках задаётся по m натуральных чисел c_{ij} ($1 \leqslant c_{ij} \leqslant k$).

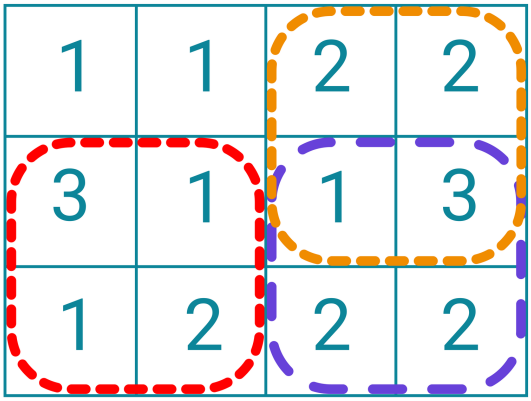
Формат выходных данных

Выведите 4 числа x_1, y_1, x_2, y_2, задающие прямоугольник, который нужно вырезать. Прямоугольник задаётся своими верхней и нижней строками (x_1,x_2) и левым и правым столбцами (y_1,y_2). Если есть несколько способов вырезать график наименьшей площади, выведите любой.

Примеры

стандартный ввод	стандартный вывод
3 4 3 1 1 2 2 3 1 1 3 1 2 2 2	1 3 2 4

Замечание



Все возможные варианты вырезать график наименьшей площади для первого примера.

Решение

Необходимо найти прямоугольную подматрицу минимальной площади, содержащую все k цветов.

Зафиксируем верхнюю и нижнюю строки l и r ($1 \le l \le r \le n$). Тогда задача сводится к поиску минимального по ширине диапазона столбцов, который вместе со строками $[l..r]$ покрывает все цвета.

Для удобства предварительно посчитаем 2D-префиксные суммы для каждого цвета c: `pref`$[c][i][j]$ — сколько клеток цвета c находится в прямоугольнике $[1..i] \times [1..j]$. Тогда количество клеток цвета c в прямоугольнике строк $[l..r]$ и столбцов $[x..y]$ можно получить за $O(1)$: `cnt = pref`$[c][r][y]$−`pref`$[c][l-1][y]$−`pref`$[c][r][x-1]$+`pref`$[c][l-1][x-1]$.

Далее перебираем все пары строк (l,r) и применяем по столбцам двухуказательный проход. Левый указатель x фиксируем, а правый y двигаем вправо, пока прямоугольник $[l..r] \times [x..y]$ не начнёт содержать все цвета. Как только это произойдёт, можно будет обновить ответ площадью $(r-l+1)(y-x+1)$, а затем сдвинуть x дальше. Так как указатель y для фиксированных (l,r) только растёт, проход по столбцам работает за $O(m)$.

Итого: префиксы считаются за $O(knm)$, перебор пар строк даёт $O(n^2)$ запусков двух указателей, каждый за $O(m \cdot k)$ на проверку наличия всех цветов (в данной реализации проверка идёт перебором всех k цветов через префиксы). Общая сложность данной версии: $O(knm + n^2mk)$.

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

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

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

    vector<vector<int>> v(n, vector<int>(m));
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            cin >> v[i][j];
            --v[i][j];
        }
    }

    vector<vector<vector<int>>> pref(
        k, vector<vector<int>>(n + 1, vector<int>(m + 1, 0))
    );

    for (int c = 0; c < k; ++c) {
        for (int i = 1; i <= n; ++i) {
```

```
            for (int j = 1; j <= m; ++j) {
                pref[c][i][j] = pref[c][i - 1][j] + pref[c][i][j - 1]
                              - pref[c][i - 1][j - 1]
                              + (v[i - 1][j - 1] == c);
            }
        }
    }

    int min_area = n * m;
    vector<int> ans(4, -1);

    for (int l = 0; l < n; ++l) {
        for (int r = l; r < n; ++r) {
            int y = 0;
            for (int x = 0; x < m; ++x) {
                if (x > y) y = x;

                while (y < m) {
                    bool ok = true;
                    for (int c = 0; c < k; ++c) {
                        int cnt = pref[c][r + 1][y + 1]
                                - pref[c][l][y + 1]
                                - pref[c][r + 1][x]
                                + pref[c][l][x];
                        if (cnt == 0) {
                            ok = false;
                            break;
                        }
                    }
                    if (ok) break;
                    ++y;
                }

                if (y == m) {
                    x = y;
                    continue;
                }

                int area = (r - l + 1) * (y - x + 1);
                if (area < min_area) {
                    min_area = area;
                    ans = {l + 1, x + 1, r + 1, y + 1};
                }
            }
        }
    }

    cout << ans[0] << ' ' << ans[1] << ' ' << ans[2] << ' ' << ans[3] << '\n';
    return 0;
}
```

Максимальный балл за задание — 100