Максимальное количество баллов за олимпиаду — 600

Задание 1. Расстояние между словами

Будем называть словом любую последовательность букв русского алфавита. За одну операцию разрешается сделать одно из трёх действий:

- убрать из слова одну любую букву,
- добавить в любое место слова любую букву,
- заменить любую букву в слове на любую другую букву.

Расстоянием между словами будем называть наименьшее число операций, которое позволяет получить из одного слова другое. Сколько слов находятся на расстоянии 2 и от слова СТУК, и от слова КУСТЫ?

Ответ: 10

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. Если некоторое слово находится на расстоянии 2 от слов КУСТЫ и СТУК, то расстояние между словами КУСТЫ и СТУК не более 4. Изучим кратчайшие «пути» (по количеству операций) между ними.

Покажем, что из слова КУСТЫ можно получить слово СТУК минимум за 4 операции, причём эти 4 операции: убрать К, У, заменить Ы на У или К и добавить К или У. Если это утверждение доказано, то слова на расстоянии 2 от обоих могут быть только после применения двух из указанных операций, а таких слов 10:

СТЫ, УСТУ, УСТК, КСТУ, КСТК, УСТУЫ, УСТЫК, КСТУЫ, КСТЫК, КУСТУК.

Поскольку операций не более 4, то к какой-то букве не применялись операции, назовем её неподвижной. Пусть она ровно одна. Тогда из остальных букв удалили одну, а остальные поменяли. В результате неподвижная буква либо сохранила свой порядковый номер в слове, либо уменьшила его на 1. Но с такими условиями общих букв у слов КУСТЫ и СТУК нет. Следовательно, неподвижных букв как минимум две. Они должны идти в исходном и конечном слове в одинаковом порядке, поэтому это обязательно буквы С и Т. Таким образом, как минимум две операции потребуется, чтобы удалить все буквы до С и Т, ещё как минимум две — чтобы после СТ появились У и К. Итого нужно не менее 4 операций, причём равенство достигается только в случаях, которые мы описали в начале решения.

Задание 2. След степени матрицы

Матрицей $n \times m$ будем называть таблицу из чисел, состоящую из n строк и m столбцов.
Умножение матриц — это применение правила «строка на столбец». Если

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

то

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

Обычно порядок важен: как правило, $MN \neq NM$. Возведением матрицы в степень называют многократное умножение её самой на себя: $A^1 = A$, а далее $A^{k+1} = A^k A$.
Дана матрица:

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix},$$

Найдите сумму левого верхнего и правого нижнего элементов матрицы A^{20}.

Ответ: 15127

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. Рассмотрим умножение произвольной матрицы $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ на матрицу $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$.

Имеем

$$MA = \begin{pmatrix} a + b & a \\ c + d & c \end{pmatrix}.$$

Видно, что при умножении на A правый столбец результата совпадает с левым столбцом исходной матрицы, а новый левый столбец получается как сумма двух столбцов исходной матрицы.

Обозначим числа Фибоначчи: $F_0 = 0$, $F_1 = 1$, $F_{n+1} = F_n + F_{n-1}$. Из описанного свойства умножения следует, что при последовательном умножении на матрицу A столбцы матриц A^n эволюционируют по рекуррентному правилу Фибоначчи.

Докажем по индукции, что для всех $n \geq 1$

$$A^n = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}.$$

База индукции. При $n = 1$

$$A^1 = A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} F_2 & F_1 \\ F_1 & F_0 \end{pmatrix}.$$

Индукционный переход. Пусть формула верна для некоторого n. Тогда

$$A^{n+1} = A^n A = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} F_{n+1} + F_n & F_{n+1} \\ F_n + F_{n-1} & F_n \end{pmatrix}.$$

По определению чисел Фибоначчи это равно

$$\begin{pmatrix} F_{n+2} & F_{n+1} \\ F_{n+1} & F_n \end{pmatrix},$$

что завершает индукцию.

Следовательно,

$$A^{20} = \begin{pmatrix} F_{21} & F_{20} \\ F_{20} & F_{19} \end{pmatrix}$$

и искомая сумма равна $F_{21} + F_{19}$.

Вычислим значения: $F_{19} = 4181$, $F_{21} = 10946$. Тогда $F_{21} + F_{19} = 10946 + 4181 = 15127$.

Задание 3. ReLU

Напомним, что

$$\mathrm{ReLU}(z) = \max(0, z).$$

Рассмотрим функцию действительного аргумента x:

$$F(x) = c + w_1 \cdot \mathrm{ReLU}(x - t_1) + w_2 \cdot \mathrm{ReLU}(x - t_2) + \cdots + w_{2025} \cdot \mathrm{ReLU}(x - t_{2025}),$$

где $t_1, \ldots, t_{2025}$ — попарно различные действительные числа, каждый коэффициент w_i равен либо $+1$, либо -1, а c — действительное число.

Какое наибольшее число различных решений может иметь уравнение $F(x) = 0$, если известно, что у него конечное число решений?

Ответ: 1013

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. Упорядочим точки: $t_1 < t_2 < \cdots < t_{2025}$. На каждом промежутке (t_k, t_{k+1}) функция F является линейной. Её наклон равен

$$a_k = \sum_{i \leq k} w_i, \quad a_0 = 0.$$

При переходе через каждую точку t_i наклон изменяется на ± 1.

Рассмотрим уравнение $F(x) = 0$, то есть точки пересечения графика F с осью x. На каждом промежутке, где наклон a_k имеет фиксированный ненулевой знак, график может пересечь ось x не более одного раза.

Если при переходе через некоторую точку t_i наклон меняется с положительного на отрицательный, то в этой точке наклон обязательно проходит через значение 0, то есть функция достигает горизонтального положения. Если на каком-то промежутке наклон равен нулю, то соответствующий участок графика является горизонтальным; на таком отрезке либо нет решений уравнения $F(x) = 0$, либо решений бесконечно много. Последний случай запрещён по условию, поэтому горизонтальные отрезки не дают вклад в число решений.

Следовательно, число решений уравнения $F(x) = 0$ не превосходит числа групп подряд идущих промежутков, на которых наклоны a_k имеют одинаковый ненулевой знак.

Предположим, что таких групп не меньше 1014. Тогда между ними должно быть как минимум 1013 переходов, в которых наклон обращается в ноль. Это требует наличия как минимум 2026 различных точек t_i, что противоречит условию $t_1, \ldots, t_{2025}$. Следовательно, число решений не превосходит 1013.

Покажем, что это значение достижимо. Положим $c = -1$ и зададим веса периодическим блоком

$$(w_1, w_2, w_3, w_4) = (+1, -1, -1, +1),$$

повторяя его до w_{2025}. Тогда частичные суммы наклонов имеют вид

$$(a_k) = (0, 1, 0, -1, 0, 1, 0, -1, \dots),$$

то есть $a_k \neq 0$ ровно при нечётных k.

Положим $t_1 = 0$, а далее зададим

$$t_{k+1} - t_k = \begin{cases} 3, & k \text{ нечётно}, \\ 1, & k \text{ чётно}. \end{cases}$$

При таком выборе на каждом нечётном промежутке (t_k, t_{k+1}) функция F меняет знак и имеет ровно одно решение уравнения $F(x) = 0$, а на чётных промежутках решений нет.

Таким образом, число решений равно числу нечётных промежутков, то есть 1013. Это и есть наибольшее возможное число решений.

Задание 4. Чистка журнала заказов маркетплейса

Дан файл с журналом заказов маркетплейса, который вы можете скачать в форматах XLSX, ODS или CSV.
Каждая строка файла содержит три поля в порядке `delivery_hours` (вещественное число), `label` (строка) и `rating` (вещественное число).
Сначала оставьте только строки, где время доставки попадает в диапазон: $1 \leqslant$ `delivery_hours` $\leqslant 96$. Затем в столбце `label` замените все пустые значения модой — самым частым значением этого столбца после фильтрации.
Далее нормируйте `rating`: пусть среди оставшихся строк $\texttt{rating}_{\min}$ — минимальный рейтинг, а $\texttt{rating}_{\max}$ — максимальный. Для любой строки её нормированный рейтинг вычисляется по формуле

$$\frac{\texttt{rating} - \texttt{rating}_{\min}}{\texttt{rating}_{\max} - \texttt{rating}_{\min}}.$$

Сколько строк после всех преобразований одновременно удовлетворяют следующим условиям:

- `label` равен моде;
- нормированный рейтинг не меньше 0.8;
- `delivery_hours` $\leqslant 48$?

Ответ: 711

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. Решение выполняется последовательно по условиям задачи. Сначала из таблицы удаляются строки с нереалистичным временем доставки, остаются только значения `delivery_hours` $\in [1, 96]$. Затем в столбце `label` все пропуски (включая пустые строки) заменяются наиболее часто встречающимся после фильтрации значением. После этого столбец `rating` приводится к шкале $[0, 1]$ с помощью min–max-нормализации, результат сохраняется в новом столбце `rating_norm`. На последнем шаге подсчитывается количество строк, в которых одновременно выполняются условия: значение `label` совпадает с модой, `rating_norm` ≥ 0.8 и `delivery_hours` ≤ 48.

Решение на языке Python:

```python
import pandas as pd
import numpy as np

df = pd.read_csv("orders6_raw.csv")

df = df[(df["delivery_hours"] >= 1) & (df["delivery_hours"] <= 96)].copy()

lbl = df["label"].replace("", np.nan)
mode_label = lbl.mode(dropna=True)
mode_label = mode_label.iloc[0] if not mode_label.empty else "unknown"
df["label"] = lbl.fillna(mode_label)

r_min = df["rating"].min()
r_max = df["rating"].max()
den = (r_max - r_min)
```

```
df["rating_norm"] = (df["rating"] - r_min) / den if den != 0 else 0.0

answer = (
    (df["label"] == mode_label)
    & (df["rating_norm"] >= 0.8)
    & (df["delivery_hours"] <= 48)
).sum()

print(answer)
```

Задание 5. Сжатие размерности

Ограничение по времени: 1 секунда

Ограничение по памяти: 256 мегабайт

В машинном обучении часто возникает проблема: данных слишком много, а ресурсов слишком мало. Например, вы обучаете модель для распознавания изображений, и каждое изображение описывается миллионом признаков (по числу пикселей). Если хранить и обрабатывать все признаки, модель получится очень тяжёлой и будет работать медленно. Чтобы справиться с этим, используют приём сжатия размерности. Его смысл — это удалить или преобразовать те признаки, которые не дают новой информации. Например:

- если у всех объектов один и тот же цвет фона, то этот признак можно убрать;
- если признак не меняется, он не помогает отличать объекты друг от друга.

Рассмотрим упрощённый вариант этой идеи.

Дана выборка объектов a, состоящая из n элементов a_i. Каждый объект a_i представляется двоичной строкой длины 10. Получается таблица признаков: строки — объекты, столбцы — признаки (первый бит — первый признак, второй бит — второй признак и так далее). Все признаки бинарные: каждый бит равен либо 0, либо 1.

Если какой–то признак одинаков у всех чисел, он не несёт информации и должен быть удалён.

Сколько признаков останется после удаления одинаковых?

Формат входных данных

В первой строке входных данных даётся одно число n ($1 \leqslant n \leqslant 1000$) — количество объектов в выборке a.

В следующих n строках даётся по одному целому числу a_i ($0 \leqslant a_i < 1024$) — битовые представления объектов.

Формат выходных данных

Выведите одно целое число — количество оставшихся после удаления признаков.

Примеры

стандартный ввод	стандартный вывод
3 741 121 164	7

Замечание

В первом тестовом примере в массиве a было всего 3 числа: 741, 121, 164. В двоичном виде они выглядят как 1011100101, 0001111001, 0010100100. Удалены будут 2–й, 5–й и 9–й признаки. Останется 7 признаков.

Решение

В этой задаче нет идейной сложности, есть реализационная: нужно уметь получать 10 бит числа. Далее для каждого бита убедиться в том, правда ли что его значения во всех числах совпадают. Для этого достаточно сравнить значение этого бита у всех чисел со значением у первого числа.

Вспомогательный код выделения 10 младших битов числа x:

```
vector<int> bits;
int x;
for (int i = 0; i < 10; ++i) {
    bits.push_back(x % 2);
    x /= 2;
}
```

На практике можно сделать ещё проще: для каждого бита достаточно понять, встречается ли он среди чисел как 0 и как 1. Это удобно проверяется через побитовые операции: `v_or` содержит единицы в битах, где хотя бы у одного числа стоит 1, а `v_and` содержит единицы там, где у всех чисел стоит 1. Тогда в битах, где значения у чисел не совпадают, `v_or` и `v_and` различаются, то есть их XOR равен 1. Остаётся посчитать количество единиц в a = `v_or` $\oplus$ `v_and`.

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

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

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

    int v_or = v[0];
    int v_and = v[0];
    for (int i = 0; i < n; ++i) {
        v_or |= v[i];
        v_and &= v[i];
    }

    int a = v_or ^ v_and;

    int cnt = 0;
    while (a > 0) {
        cnt += a % 2;
        a /= 2;
    }

    cout << cnt << '\n';
    return 0;
}
```

Максимальный балл за задание — 100

Задание 6. Разнообразие разбиения

Решающие деревья — один из базовых методов машинного обучения. Они применяются для самых разных задач: от рекомендаций фильмов до диагностики по медицинским признакам. По сути, дерево — это последовательность вопросов о признаках, которые делят выборку объектов на группы. Классический алгоритм строится так, чтобы после каждого вопроса данные в ветвях становились как можно более однородными по классам — ведь чем однороднее ветка, тем проще в ней принять окончательное решение.

Однако у такого подхода есть недостаток: дерево может переобучаться, слишком подстраиваясь под конкретные данные. Чтобы изучить альтернативные принципы разбиения, исследователи решили рассмотреть противоположную идею — иногда выбирать разбиение, при котором обе ветви оказываются максимально разнообразными по составу классов. В этой задаче вам предстоит реализовать модуль, находящий порог признака, который даёт наибольшее разнообразие.

Дана обучающая выборка из n объектов. Каждый объект описывается одним числовым признаком x_i и меткой класса $y_i \in \{0, 1\}$. Рассмотрим целочисленный порог t и определим разбиение выборки на два блока:

- левый блок $L(t)$ — все объекты с номерами i, для которых $x_i \leqslant t$;
- правый блок $R(t)$ — все объекты с номерами i, для которых $x_i > t$.

Обозначим:

- $n_l = |L(t)|$ и $n_r = |R(t)|$ — количество объектов в левом и правом блоках;
- p_l — доля объектов класса 1 в левом блоке;
- p_r — доля объектов класса 1 в правом блоке;

Мера разнообразия одного блока определяется как

$$d = p(1 - p),$$

где p — доля объектов класса 1 в этом блоке. Общая оценка разбиения порогом t — это взвешенная по размерам блоков сумма мер разнообразия левого и правого блоков:

$$d_{\text{split}}(t) = \frac{n_l}{n} p_l(1 - p_l) + \frac{n_r}{n} p_r(1 - p_r),$$

где $n = n_l + n_r$.

Найдите такое целое значение порога t, при котором величина $d_{\text{split}}(t)$ максимальна. Если несколько порогов дают одинаковое значение разнообразия, выберите наименьший из них.

Формат входных данных

В первой строке входных данных дано единственное целое число n — количество объектов ($2 \leqslant n \leqslant 10^4$). В следующих n строках даны по два целых числа x_i и y_i — признак и класс i–го объекта ($|x_i| < 10^{12}$, $y_i \in 0{,}1$. Обратите внимание, что в данных могут встречаться повторяющиеся признаки и не гарантируется, что y_i будут совпадать при совпадающих x_i.

Формат выходных данных

Выведите одно целое число — оптимальное значение порога t, при котором величина $d_{\text{split}}(t)$ максимальна. Если несколько значений t дают одинаковое значение разнообразия, выведите наименьший из таких порогов.

Система оценки

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи успешно пройдены.

Подзадача	Баллы	Дополнительные ограничения
1	20	Все метки классов одинаковы ($y_i = 0$ или $y_i = 1$)
2	80	$2 \leqslant n \leqslant 10^4$

Примеры

стандартный ввод	стандартный вывод
5 1 0 2 1 3 0 5 0	2

Замечание

Рассмотрим все возможные пороги $t \in 1, 2, 3, 4$. Для каждого порога вычислим меру разнообразия

$$d_{\text{split}} = \frac{n_l}{n} p_l(1 - p_l) + \frac{n_r}{n} p_r(1 - p_r),$$

где p_l и p_r — доли единиц в левой и правой группах соответственно.

t	Левый блок ($x_i \leqslant t$)	Правый блок ($x_i > t$)	p_l	p_r	d_{split}
1	[0]	[1, 0, 1, 0]	0	0.5	0.2
2	[0, 1]	[0, 1, 0]	0.5	$\frac{1}{3}$	0.2(3)
3	[0, 1, 0]	[1, 0]	$\frac{1}{3}$	0.5	0.2(3)
4	[0, 1, 0, 1]	[0]	0.5	0	0.2

Максимальное значение $d_{\text{split}} = 0.2333$ достигается при $t = 2$ и $t = 3$, но по условию при равенстве выбирается наименьший порог: $t = 2$.

Решение

Дана выборка из n объектов с признаком x_i и меткой $y_i \in \{0, 1\}$. Нужно найти целочисленный порог t, при котором величина разнообразия разбиения $d_{\text{split}}(t) = \frac{n_l}{n} p_l(1 - p_l) + \frac{n_r}{n} p_r(1 - p_r)$ максимальна, где левый блок задаётся условием $x_i \leq t$, правый — $x_i > t$. Рассматриваем только пороги, при которых оба блока непустые, а при равенстве значений берём минимальный t.

Перебирать все целые пороги нельзя (диапазон может быть огромным). Ключевое наблюдение: значение разбиения меняется только при переходе через какое-то значение x, поэтому достаточно рассматривать пороги вида $t = x$ для уникальных значений признака.

Отсортируем пары (x_i, y_i) по x_i и будем идти слева направо группами одинакового x. Поддерживаем количество единиц в левом блоке `l1` и общее количество элементов слева `l`. Справа аналогично: `r` $= n -$ `l` и `r1` $=$ `total_ones` $-$ `l1`. Переходя к очередному значению t, мы переносим всю группу с $x = t$ влево (так как условие $x \leq t$), после чего, если справа что-то осталось, считаем $d_{\text{split}}(t)$ и обновляем ответ. Такой проход после сортировки работает за $O(n)$, итого сложность $O(n \log n)$.

```
import sys

def solve() -> None:
    input = sys.stdin.readline
    n = int(input())
    data = [tuple(map(int, input().split())) for _ in range(n)]
    data.sort()

    total_ones = sum(y for _, y in data)

    best_t = None
    best_val = -1.0

    l = 0
    l1 = 0
    i = 0
    while i < n:
```

```
        t = data[i][0]

        grp_ones = 0
        j = i
        while j < n and data[j][0] == t:
            grp_ones += data[j][1]
            j += 1

        l += (j - i)
        l1 += grp_ones
        r = n - l
        r1 = total_ones - l1
        if l > 0 and r > 0:
            # p(1-p) = (ones/size) * (1 - ones/size)
            p_l = l1 / l
            p_r = r1 / r
            val = (l / n) * (p_l * (1 - p_l)) + (r / n) * (p_r * (1 - p_r))

            if val > best_val:
                best_val = val
                best_t = t

        i = j

    print(best_t)

if __name__ == "__main__":
    solve()
```

Максимальный балл за задание — 100

Максимальное количество баллов за олимпиаду — 600

Задание 1. Картинки в папке

В папке на компьютере размещено 3000 изображений трёх типов: с кошкой, с собакой, с кошкой и собакой одновременно. Изображений всех типов одинаковое количество. Модель ИИ ответила на два вопроса к каждой картинке: есть ли на ней кошка и есть ли на ней собака. На первый вопрос было получено 2790 ответов «да», а на второй — 2861. Удалили все изображения, для которых хотя бы один ответ ИИ был неверным. Какое наименьшее число изображений могло остаться в папке?

Ответ: 651

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. Оценка. Заметим, что на первый вопрос ответов НЕТ ровно 210, а на второй — 139. Значит, общее число ответов НЕТ 349. Таким образом, из 1000 изображений с кошкой и собакой хотя бы для $1000 - 349 = 651$ было два ответа ДА, эти изображения точно не будут удалены. Пример. Если модель ИИ ответила НЕТ на первый вопрос ровно для 139 изображений с кошкой и собакой и ответила НЕТ на второй вопрос ровно для 210 других изображений с кошкой и собакой, то верно на оба вопроса она ответила в точности для оставшихся 651 изображений с кошкой и собакой.

Задание 2. Сколько расстановок дают положительный определитель

Матрицей $n \times m$ будем называть таблицу из чисел, состоящую из n строк и m столбцов. Определитель матрицы 2×2 — это число, вычисляемое по формуле. Для $\begin{pmatrix} p & q \\ r & s \end{pmatrix}$ его значение равно $ps - qr$.

Сколько существует способов расставить числа 1, 2, 3, 6 по клеткам матрицы 2×2 (каждое число используется ровно один раз) так, чтобы $ps - qr > 0$?

Ответ: 8

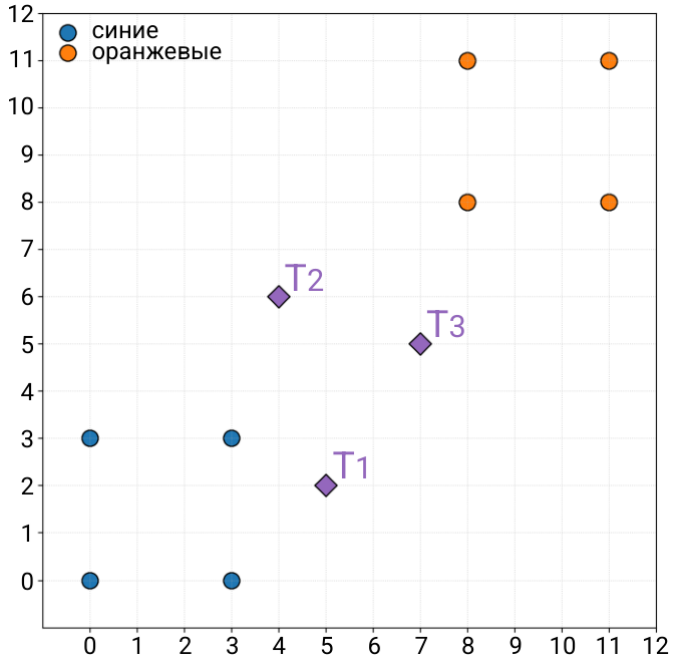
Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. Пусть матрица имеет вид $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ и состоит из чисел 1, 2, 3, 6 без повторений. Знак определителя $det = ad - bc$ зависит только от того, какие два числа стоят на главной диагонали (a, d) и какие два — на побочной (b, c): порядок внутри пары не меняет произведения ad и bc. Разбиения множества $\{1, 2, 3, 6\}$ на две пары всего три: $\{1, 6\}\&\{2, 3\}$(6 и 6), $\{1, 2\}\&\{3, 6\}$(2 и 18), $\{1, 3\}\&\{2, 6\}$(3 и 12). В первом случае произведения равны, детерминант равен нулю и не подходит. В двух остальных случаях он положителен тогда и только тогда, когда на главной диагонали стоит пара с большим произведением. Для фиксированного выбора пар по диагоналям порядок внутри каждой пары можно переставлять двумя способами, итого $2! \cdot 2! = 4$ расстановки. Следовательно, положительных расстановок $4 + 4 = 8$.

Задание 3. Свой среди чужих

На плоскости даны восемь обучающих точек — четыре синие и четыре оранжевых, а также три тестовые точки T_1, T_2, T_3, обозначенные фиолетовыми ромбиками.



Цвет тестовой точки будем предсказывать по цветам её соседей.

Иногда разные координаты имеют очень разный масштаб: одна может меняться от 0 до 10000, а другая — от 0 до 0.00001. Тогда при вычислении расстояний первая координата начинает полностью «перебивать» вторую. Чтобы обе координаты влияли честно, первую иногда растягивают или сжимают.

В этой задаче первая координата умножается на положительное число $\lambda > 0$:

$(x,y) \mapsto (\lambda\, x,y)$ Расстояние между точками считаем евклидовым в новых координатах $(\lambda x,\, y)$. Цвет тестовой точки предсказываем так:

- находим 3 ближайших к ней обучающих точки;
- если среди них больше синих, считаем тестовую точку синей, если больше оранжевых — оранжевой;
- если расстояния совпадают, то более близкой считается точка с меньшим λx, а при равных λx — с меньшим y.

Зафиксируем $\lambda = 1$ и посмотрим, какого цвета при этом каждая тестовая точка T_i.

Назовём значение $\lambda > 0$ *интересным* для точки T_i, если при таком λ предсказанный цвет T_i отличается от её цвета при $\lambda = 1$.

Рассмотрим для каждой точки все её интересные значения λ.

Оказывается, что для каждой T_i возможны только два случая:

- интересных значений нет вообще — тогда цвет точки не меняется ни при каком $\lambda > 0$;
- интересные значения существуют и устроены так: есть число $L_i > 0$, что при всех $0 < \lambda < L_i$ цвет точки T_i другой, чем при $\lambda = 1$, а при всех $\lambda > L_i$ цвет уже такой же, как при $\lambda = 1$. При $\lambda = L_i$ цвет может совпадать или не совпадать с цветом при $\lambda = 1$ — это надо аккуратно проверить.

Число L_i будем называть *границей* интересных значений для точки T_i. Для каждой тестовой точки T_i:

- выведите -1, если интересных значений λ нет;
- найдите её границу L_i и выведите число $(L_i)^2$ в ином случае.

Ответ: -1, $\frac{5}{33}$, $\frac{5}{33}$

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. Зададим координаты обучающих точек (как на рисунке):

$$B_1 = (0,0), \quad B_2 = (0,3), \quad B_3 = (3,0), \quad B_4 = (3,3),$$
$$R_1 = (8,8), \quad R_2 = (8,11), \quad R_3 = (11,8), \quad R_4 = (11,11).$$

Обозначим $t = \lambda 2 > 0$. Тогда $d^2\lambda(T,P) = t(x - x_T)^2 + (y - y_T)^2$, поэтому расстояния — линейные функции от t. Критические точки возможны только при совпадении таких выражений.

1. Точка $T_1 = (5,2)$. Из сравнения расстояний видно, что точки B_3 и B_4 всегда ближе к T_1, чем любая оранжевая точка. Значит, среди трёх ближайших всегда есть как ми м две синие точки. Класс не меняется при любом $\lambda > 0$, поэтому не существует.

2. Точка $T_2 = (4, 6)$. Отбрасываем точки, которые всегда дальше трёх других. Остаются только B_4, B_2, R_1, R_3.
Единственное равенство расстояний:

$$d^2(B_2) = d^2(R_3) \iff 49t + 4 = 16t + 9 \iff t_c = \frac{5}{33}, \quad \lambda_c = \sqrt{\frac{5}{33}}$$

Для $0 < t < t_c$ тройка ближайших — R_1,R_3,B_4 (две оранжевых) → класс оранжевый. Для $t \geqslant t_c$ тройка — R_1,B_4,B_2 (две синих) → класс синий.

При $\lambda = 1$ точка синяя, значит, $\lambda_3^* = \sqrt{\frac{5}{33}}$

3. Точка $T_3 = (7, 5)$.

Та же четвёрка кандидатов B_4,B_2,R_1,R_3 и то же единственное критическое значение $t_c = \frac{5}{33}$. Для $0 < t \geqslant t_c$ тройка — R_1,B_4,B_2 (две синих) → класс синий. Для $t > t_c$ тройка — R_1,B_4,R_3 (две оранжевых) → класс оранжевый. При $\lambda = 1$ точка оранжевая, значит, ответ:

$$\lambda_3^* = \sqrt{\frac{5}{33}}.$$

$$T_1 : -1,\ T_2 : \frac{5}{33},\ T_3 : \frac{5}{33}.$$

Задание 4. Мониторинг энергопотребления магазина

В течение нескольких дней подряд в магазине измеряли энергопотребление каждые 15 минут. Таким образом, за одни сутки получается 96 измерений. Результаты вы можете скачать в форматах XLSX, ODS или CSV.

В таблице содержится непрерывный ряд измерений без пропусков; в каждой строке указаны t (номер измерения, начиная с 0) и power (потребление в ваттах). Обозначим $x_t = \mathsf{power}[t]$.

Чтобы проанализировать динамику потребления, рассмотрим несколько производных величин.

Во-первых, будем отслеживать изменение мощности от шага к шагу. Для каждого $t \geqslant 1$ определим разность: $d_t = x_t - x_{t-1}$. Во-вторых, нас интересуют сглаженные характеристики. Скользящие средние вычисляются по «хвостовым» окнам, то есть берут текущий момент и несколько предыдущих:

- среднее потребление за последние 8 шагов: $\mathsf{ma\_8}[t] = \frac{1}{8}\sum_{j=0}^{7} x_{t-j} \quad (t \geqslant 7)$,
- среднее изменение потребления за последние 4 шага: $\mathsf{ma\_diff\_4}[t] = \frac{1}{4}\sum_{j=0}^{3} d_{t-j} \quad (t \geqslant 4)$.

Кроме того, по всему ряду значений x_t необходимо найти медиану med. Медиана — это число, которое стоит посередине после сортировки всех значений (или среднее двух центральных при чётном числе точек).

Поскольку измерения выполнялись много дней подряд, выражение t `mod` 96 указывает, на каком 15-минутном интервале суток находится момент t. Значения от 36 до 84 соответствуют промежутку с 09:00 до 21:00 включительно.

Сколько моментов времени t одновременно удовлетворяют трём условиям:
$\mathsf{ma\_diff\_4}[t] \geqslant 3.0$, $\mathsf{ma\_8}[t] \geqslant \mathsf{med}$, $36 \leqslant (t \bmod 96) \leqslant 84$?

Ответ: 193

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение.

Сначала по столбцу power строим разностный ряд `diff_1`: это изменение мощности за один шаг (15 минут). Далее считаем два скользящих средних по хвостовым окнам, то есть окно всегда заканчивается в текущем моменте `t`: `ma_8` усредняет последние 8 значений power, а `ma_diff_4` — последние 4 значения `diff_1`. Оба скользящих средних считаем только при полном окне, поэтому первые несколько позиций дают пропуски и в подсчёт не попадают. После этого находим медиану исходного ряда power и отбираем те моменты времени, которые одновременно удовлетворяют трём условиям: средний прирост за последнее окно не меньше 3, сглаженное значение мощности не ниже медианы и шаг относится к «рабочему времени», то есть $36 \leqslant (t \bmod 96) \leqslant 84$. Количество таких моментов и является ответом.

```python
import pandas as pd

df = pd.read_csv("ts6_raw.csv")

s = df["power"]

diff_1 = s.diff()

ma_8      = s.rolling(window=8,  min_periods=8).mean()
ma_diff_4 = diff_1.rolling(window=4, min_periods=4).mean()

med = float(s.median())

t = df["t"].to_numpy()
in_hours = ((t % 96) >= 36) & ((t % 96) <= 84)

answer = ((ma_diff_4 >= 3.0) & (ma_8 >= med) & in_hours).sum()
print(answer)
```

—

Задание 5. Оптимальная конфигурация

Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт
Обучение больших нейронных сетей происходит на кластере — вычислительной инфраструктуре, которая позволяет использовать одновременно большое количество видеокарт. Видеокарты на кластере объединяются в ноды (блоки по 8 карт).

Для обучения больших языковых моделей методом обучения с подкреплением видеокарты делятся на два типа: одни генерируют данные, другие обновляют веса модели на основе генерированных данных. Вторая задача гораздо сложнее, поэтому при наилучшем распределении ресурсов на неё должно выделяться в k раз больше видеокарт. Обучение с оптимальным распределением ресурсов мы будем называть оптимальным.

Вы работаете в очень большой компании, поэтому ваши ресурсы не ограничены. Вы хотите выделить минимальное количество нод так, чтобы их видеокарты можно было разбить на два вышеописанных типа так, чтобы обучение при этом было оптимальным.

Формат входных данных

В первой и единственной строке входных данных вводится одно целое число k ($1 \leqslant k \leqslant 1000000$).

Формат выходных данных

Выведите одно число: минимальное число нод, которые нужно выделить так, чтобы обучение было оптимальным.

Примеры

стандартный ввод	стандартный вывод
5	3

Максимальный балл за задание — 100

Решение

От нас требуется найти минимальное такое число N, которое будет делиться на 8 и на k. Это ровно $N = \mathrm{lcm}(8, k)$.

Так как $8 = 2^3$, можно обойтись без алгоритма Евклида: достаточно домножать k на 2, пока получившееся число не начнёт делиться на 8. Это даст минимальное общее кратное с 8, потому что мы добавляем ровно недостающую степень двойки.

В ответе нужно вывести $\mathrm{lcm}(k, 8)/8$.

```
from math import lcm

k = int(input())
print(lcm(k, 8) // 8)
```

Задание 6. Распознавание объектов

Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт
Одна из задач машинного обучения — распознавание объектов. Она заключается в том, чтобы выделить на изображении зоны, в которых находятся объекты, и сказать, что это за объекты.

Гоша запустил умную нейронную сеть для распознавания объектов. Она вернула таблицу $n \times m$, где в каждой клетке указан номер класса, к которому принадлежит находящийся в клетке объект по мнению модели, или 0, если модель считает, что в этой клетке нет никаких интересующих её объектов.

К сожалению, этого недостаточно: Гоша хочет выделить в таблице прямоугольники-рамки со сторонами, параллельными её краям, так, чтобы внутри этих рамок были распознанные объекты. Объектами Гоша считает все клетки, которые модель распознала как клетки одного конкретного класса.

Помогите Гоше: выделите в таблице все распознанные умной нейронной сетью объекты. Так как объектов в таблице может быть очень много, посчитайте и выведите всего одно число: сумму площадей всех прямоугольников-рамок объектов.

Формат входных данных

В первой строке вводятся два целых числа n, m ($1 \leqslant n, m \leqslant 1000$).
В следующих n строках вводится по m чисел a_{ij} — номера классов объектов, которые модель нашла в конкретной клетке, или 0, если в этой клетке не был распознан ни один объект.

Формат выходных данных

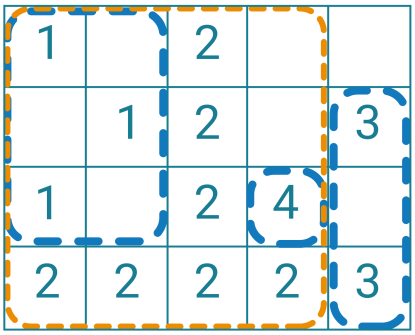
Выведите одно целое число: сумму площадей прямоугольников-рамок для всех распознанных объектов.

Примеры

стандартный ввод	стандартный вывод
1 0 2 0 0 0 1 2 0 3 1 0 2 4 0 2 2 2 2 3	26

Замечание

Рассмотрим первый пример из условия.



Решение

Для каждого цвета клетки требуется найти минимальную рамку, которая покрывает все клетки этого цвета. Будем для каждого цвета хранить минимальные и максимальные координаты по строке и столбцу: `min_x`, `max_x`, `min_y`, `max_y`. Тогда площадь рамки для цвета c равна $(\texttt{max\_x}[c] - \texttt{min\_x}[c] + 1) \cdot (\texttt{max\_y}[c] - \texttt{min\_y}[c] + 1)$, а ответ — сумма этих площадей по всем цветам (цвет 0 игнорируем).

Чтобы не хранить огромные `map` по ключу-цвету, будем динамически присваивать каждому встреченному цвету компактный индекс $0..q-1$ через `unordered_map` и обновлять границы по этому индексу за $O(1)$ на клетку.

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

using namespace std;

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

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

    vector<int> mxi(n * m + 1, -1);
    vector<int> mni(n * m + 1, n);
    vector<int> mxj(n * m + 1, -1);
    vector<int> mnj(n * m + 1, m);

    unordered_map<int, int> id;
    id.reserve(n * m + 1);

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            int x;
            cin >> x;

            int k;
            auto it = id.find(x);
            if (it == id.end()) {
                k = (int)id.size();
                id[x] = k;
            } else {
                k = it->second;
            }

            mxi[k] = max(mxi[k], i);
            mni[k] = min(mni[k], i);
            mxj[k] = max(mxj[k], j);
            mnj[k] = min(mnj[k], j);
        }
    }

    long long sum = 0;
    for (auto &[color, idx] : id) {
        if (color == 0) continue;
        sum += 1LL * (mxi[idx] - mni[idx] + 1) * (mxj[idx] - mnj[idx] + 1);
    }

    cout << sum << '\n';
    return 0;
}
```

Максимальный балл за задание — 100

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

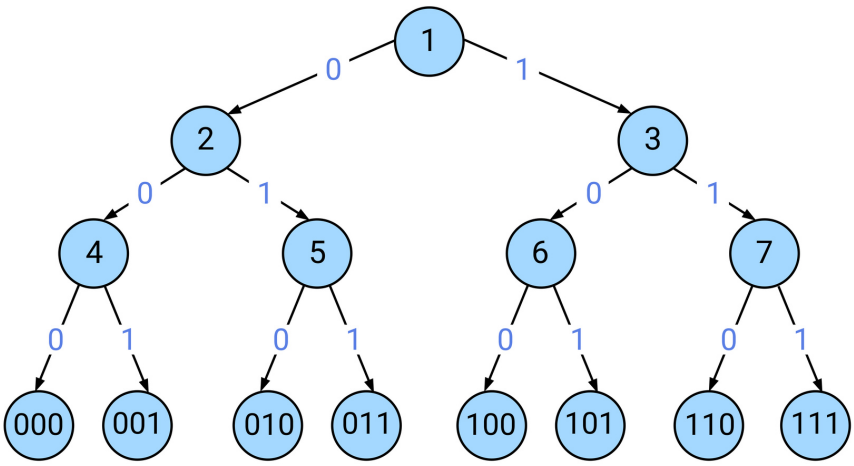
Ограничение по памяти: 256 мегабайт

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.

```cpp
#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;
```

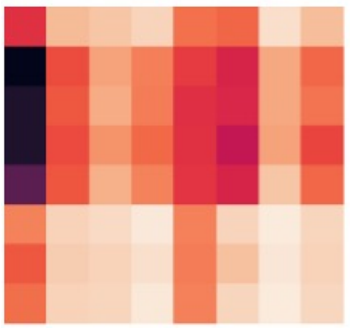
```
    int p101 = (100 - p1) * p3 * (100 - p6);
    int p110 = (100 - p1) * (100 - p3) * p7;
    int p111 = (100 - p1) * (100 - p3) * (100 - p7);

    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} ($1 \leqslant c_{ij} \leqslant k$).

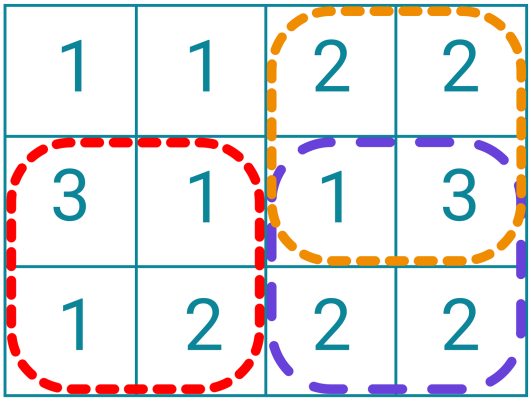
Формат выходных данных

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