Максимальное количество баллов за олимпиаду — 72
Максимальный балл за задание — 12

Задание 1.

Рассмотрим следующий классификатор точек на плоскости, зависящий от параметра t: точка $(x,\ y)$ относится к первому классу при $t \cdot x + (t-2) \cdot y \geqslant 0$ и ко второму — в противном случае.

При каком наименьшем значении t обе точки $(0,\ 2)$ и $(3,\ 4)$ относятся к первому классу?

Ответ. 2.

Критерии оценивания. Верный ответ — 12 баллов; неверный или отсутствующий ответ — 0 баллов.

Решение.

Для точки $(0, 2)$ условие принимает вид $2(t-2) \geqslant 0$, то есть $t \geqslant 2$. Значит, меньшие t не подходят.

Обратно, при $t \geqslant 2$ оба множителя t и $t-2$ неотрицательны, а координаты обеих точек неотрицательны, поэтому и $t \cdot x + (t-2) \cdot y \geqslant 0$ — обе точки попадают в первый класс.

Наименьшее подходящее значение — $t = 2$.

Задание 2.

В узле решающего дерева 20 объектов трёх классов, причём каждый объект относится ровно к одному классу; доли классов равны p_1, p_2 и p_3 (доля класса — это отношение числа объектов этого класса к общему числу объектов в узле). Индекс Джини узла ($G = 1 - p_1^2 - p_2^2 - p_3^2$) равен 0.56, а размеры первых двух из классов совпадают.

Найдите численность наибольшего класса.

Ответ. 12.

Критерии оценивания. Верный ответ — 12 баллов; неверный или отсутствующий ответ — 0 баллов.

Решение.

Пусть $p_1 = p_2 = p$, тогда $p_3 = 1 - 2p$. Из условия $G = 0.56$ получаем

$$2p^2 + (1-2p)^2 = 1 - G = 0.44.$$

Откуда $6p^2 - 4p + 0.56 = 0$, то есть $75p^2 - 50p + 7 = 0$. Дискриминант

$$D = 50^2 - 4 \cdot 75 \cdot 7 = 2500 - 2100 = 400 = 20^2,$$

поэтому

$$p = \frac{50-20}{2 \cdot 75} = \frac{1}{5} \qquad \text{или} \qquad p = \frac{50+20}{2 \cdot 75} = \frac{7}{15}.$$

При $p = \frac{7}{15}$ получаем $p_3 = \frac{1}{15}$, и число объектов в классах $20 \cdot \frac{7}{15} = \frac{28}{3}$ не целое — противоречие с условием.

Остаётся $p = \frac{1}{5}$: классы содержат 4, 4 и $20 - 8 = 12$ объектов.

Задание 3.

Три модели решают задачу бинарной классификации и ошибаются независимо друг от друга; точность каждой модели, то есть вероятность правильного ответа, равна x, причём $x < 1$. Ансамбль из этих трёх моделей отвечает по правилу большинства голосов, то есть ответ ансамбля совпадает с ответом, который дали хотя бы две из трёх моделей.

Известно, что вероятность ошибки ансамбля составляет 52 % от вероятности ошибки одной отдельной модели. Найдите x.

Ответ. 0,8.

Критерии оценивания. Верный ответ — 12 баллов; неверный или отсутствующий ответ — 0 баллов.

Решение.

Пусть $q = 1 - x$ — вероятность ошибки одной модели. Ансамбль ошибается, когда ошиблись хотя бы две модели: ровно две — это $3q^2(1-q)$, все три — это q^3. Вероятность ошибки ансамбля равна

$$3q^2(1-q) + q^3 = 3q^2 - 2q^3.$$

По условию $3q^2 - 2q^3 = 0.52\,q$. Так как $q > 0$, поделим на q:

$$3q - 2q^2 = 0.52, \qquad \text{то есть} \qquad 2q^2 - 3q + 0.52 = 0.$$

Умножим на 25, чтобы избавиться от дробей: $50q^2 - 75q + 13 = 0$. Дискриминант

$$D = 75^2 - 4 \cdot 50 \cdot 13 = 5625 - 2600 = 3025 = 55^2,$$

поэтому

$$q = \frac{75 - 55}{2 \cdot 50} = 0{,}2 \qquad \text{или} \qquad q = \frac{75 + 55}{2 \cdot 50} = 1{,}3.$$

Вероятность не бывает больше 1, поэтому $q = 0{,}2$ и $x = 1 - q = 0{,}8$.

Задание 4.

В зоопарке на ленивцев установили датчики, которые каждый день оценивают продолжительность их сна. Результаты находятся в файле sloths.csv. Файл содержит четыре столбца:

sloth_id — идентификатор ленивца;

enclosure — номер вольера;

day — номер дня наблюдения;

sleep_hours — продолжительность сна в этот день в часах.

Таблица содержит не более одной строки для одного ленивца и одного дня. Каждый ленивец в течение всего периода наблюдения находится в одном и том же вольере. Из-за сбоев датчиков некоторые значения sleep_hours могут быть пропущены или оказаться невозможными.

Выполните следующие действия:

1. Оставьте только строки, в которых значение sleep_hours не пропущено и удовлетворяет условию $0 \leqslant$ sleep_hours $\leqslant 24$.

2. Для каждого ленивца подсчитайте число дней с корректными измерениями. Оставьте только ленивцев, для которых имеется не менее 10 таких дней.

3. Для каждого оставшегося ленивца найдите среднее значение sleep_hours по всем его корректным измерениям.

4. Для каждого вольера найдите среднее арифметическое значений, полученных на предыдущем шаге для всех оставшихся ленивцев этого вольера. Таким образом, каждый ленивец влияет на результаты по своему вольеру одинаково независимо от числа дней наблюдений.

5. Не учитывайте вольеры, в которых после предыдущих действий осталось менее пяти ленивцев.

6. Найдите вольер с наибольшим средним временем сна. Если таких вольеров несколько, выберите вольер с наименьшим номером.

а) Запишите число строк, оставшихся после выполнения шага 1.

б) Запишите номер вольера с наибольшим средним временем сна.

в) Чему равно среднее время сна в этом вольере? Ответ выразите в часах, округлите до тысячных.

После первого шага в таблице осталось 2344 строки.

Ответ: 2344; вольер 7; 19,560 часа (программа выводит `19.560`).

Критерии оценивания. Баллы за правильные ответы по порядку вопросов: 2; 3; 7 соответственно. Баллы за пункты суммируются, максимум — 12. За неверный или отсутствующий ответ на отдельно оцениваемый пункт — 0 баллов, если частичное оценивание не указано отдельно.

Решение.

Ключевой момент задачи — двухступенчатое усреднение. Сначала нужно найти среднее для каждого отдельного ленивца, а уже затем усреднить эти значения внутри вольера. Если сразу вычислить среднее по всем строкам вольера, ленивцы с большим числом дней наблюдения получат больший вес, что противоречит условию.

Решение на языке Python:

```
import pandas as pd

df = pd.read_csv("data/sloths.csv")
valid = df[df["sleep_hours"].between(0, 24)].copy()

by_sloth = valid.groupby(
    ["enclosure", "sloth_id"]
)["sleep_hours"].agg(mean_sleep="mean", valid_days="count")
```

```
by_sloth = by_sloth[by_sloth["valid_days"] >= 10]

by_enclosure = by_sloth.groupby(
    "enclosure"
)["mean_sleep"].agg(mean_sleep="mean", sloth_count="count")

by_enclosure = by_enclosure[by_enclosure["sloth_count"] >= 5]

best = (by_enclosure.reset_index()
        .sort_values(["mean_sleep", "enclosure"],
                     ascending=[False, True])
        .iloc[0])

print(len(valid))
print(int(best["enclosure"]))
print(f'{best["mean_sleep"]:.3f}')
```

Ответ. По порядку вопросов: 2344; 7; 19,560.

Задание 5.

Из симуляции в реальность

Ограничение по времени: 2 секунды

Ограничение по памяти: 256 мегабайт

Инженеры обучают робот-пылесос с подкреплением; робот раз за разом проходит эпизоды — последовательности действий, получает за них оценку и понемногу улучшает свою стратегию. Сначала эпизоды прогоняют в симуляторе, а лучшие из них затем проверяют в реальной комнате.

Очередной эпизод записан строкой s длины n из символов U, D, L и R. Робот выполняет команды по очереди слева направо, и каждая сдвигает его ровно на одну клетку: U — вверх, D — вниз, L — влево, R — вправо.

Для проверки нужно испытательное поле — прямоугольник из h строк и w столбцов клеток. Инженеры ставят робота в любую клетку поля по своему выбору, и он выполняет весь эпизод. Если очередная команда уводит робота за край поля, то он падает и проверка срывается.

Скажем, что поле подходит, если на нём есть хотя бы одна стартовая клетка, начав с которой, робот выполнит эпизод целиком и ни разу не покинет поле.

Чем больше в поле клеток, тем оно дороже. Помогите инженерам выбрать подходящее поле с наименьшим количеством клеток $h \cdot w$.

Формат входных данных

В первой строке ввода дано одно целое число n — длина эпизода ($1 \leqslant n \leqslant 2 \cdot 10^5$).

Во второй строке дана последовательность s из n символов U, D, L и R — команд управления роботом.

Формат выходных данных

Выведите два целых числа h и w — количество строк и количество столбцов подходящего поля с наименьшим количеством клеток. Если таких полей несколько, выведите любое из них.

Система оценивания

Тесты из условия не оцениваются. За каждый пройденный тест остальных групп начисляется 1 балл независимо от других тестов.

Группа	Баллы	Дополнительные ограничения
1	5	$1 \leqslant n \leqslant 1000$
2	7	Без дополнительных ограничений

Замечание

В первом примере робот, поставленный в левую нижнюю клетку поля 2×3, сдвигается на две клетки вправо и на одну вверх и остаётся на поле.

В третьем примере робот опускается на две клетки, сдвигается на две вправо и поднимается на четыре, то есть

побывает в пяти разных строках и в трёх разных столбцах. Стартовать здесь нужно из третьей сверху клетки первого столбца.

Ответ. $h = y_{\max} - y_{\min} + 1$, $w = x_{\max} - x_{\min} + 1$; крайние координаты считаются по всем позициям робота, включая исходную.

Критерии оценивания. Выведены размеры, позволяющие выполнить траекторию, с минимальным произведением. Группа 1: 5 баллов, $1 \leqslant n \leqslant 1000$; группа 2: 7 баллов без дополнительных ограничений. Тесты из условия не оцениваются. Каждый из остальных тестов оценивается независимо в 1 балл; непройденный тест — 0 баллов. Максимум — 12 баллов.

Решение.

Проследим за координатами робота, считая старт точкой $(0, 0)$. Пусть среди всех посещённых позиций минимальная и максимальная вертикальные координаты равны $y_{\min}$ и $y_{\max}$. Тогда поле обязано иметь хотя бы

$$y_{\max} - y_{\min} + 1$$

строк: робот реально посещает клетки на всём этом диапазоне координат. Аналогично минимальная необходимая ширина равна

$$x_{\max} - x_{\min} + 1.$$

Такого поля достаточно: старт можно сдвинуть так, чтобы весь найденный прямоугольник траектории оказался внутри поля. Поэтому эти размеры одновременно минимальны и дают минимальную площадь.

Сложность: $O(n)$, память $O(n)$ на хранение строки (не считая её — $O(1)$).

```
n = int(input())
s = input()

x = 0
y = 0

min_x = 0
max_x = 0
min_y = 0
max_y = 0

for command in s:
    if command == "L":
        x -= 1
    if command == "R":
        x += 1
    if command == "U":
        y += 1
    if command == "D":
        y -= 1

    min_x = min(min_x, x)
    max_x = max(max_x, x)

    min_y = min(min_y, y)
    max_y = max(max_y, y)

width = max_x - min_x + 1
height = max_y - min_y + 1

print(height, width)
```

Задание 6.

Фишки для Реверси. Коррекция бинарных меток

При подготовке данных для обучения модели машинного обучения был обнаружен набор объектов с бинарными метками. Каждая метка может принимать одно из двух значений: 0 или 1.

В результате ошибки системы разметки некоторые метки оказались неправильными. Для исправления данных используется специальная процедура: за одну операцию можно выбрать две различные записи и одновременно изменить обе

их метки на противоположные.

Даны две бинарные строки длины n: исходные и требуемые значения меток для n объектов. Символ 1 означает метку одного класса, а символ 0 — метку другого класса.

Требуется определить, можно ли с помощью описанных операций получить требуемую последовательность меток из исходной. Если это возможно, выведите любую последовательность операций, которая переводит исходную последовательность в требуемую.

Формат входных данных

В первой строке вводится натуральное число n ($1 \leqslant n \leqslant 200000$).

Во второй строке вводится строка длины n из нулей и единиц — исходные значения меток.

В третьей строке вводится строка длины n из нулей и единиц — требуемые значения меток.

Формат выходных данных

Если преобразование невозможно, выведите NO SOLUTION.

В противном случае в первой строке выведите количество операций t, не превышающее 200000. В следующих t строках выведите по два неравных номера позиций, метки в которых нужно изменить на противоположные. Позиции нумеруются с 1.

Если существует несколько верных ответов, выведите любой из них.

Ответ. Если число различающихся позиций нечётно, выводится `NO SOLUTION`. Иначе можно разбить эти позиции на пары и выполнить по одной операции для каждой пары.

Критерии оценивания. `NO SOLUTION` допускается только при невозможности преобразования. Иначе операций не больше 200000; в каждой указаны две разные допустимые позиции; после всех операций получается нужная строка. Принимается любая такая последовательность; минимальность не требуется.

Решение.

Выпишем позиции, в которых строки различаются. Операция меняет метки в двух позициях, и каждая из них либо перестаёт отличаться от требуемой, либо начинает отличаться. Поэтому число различающихся позиций меняется на -2, 0 или $+2$, и его чётность сохраняется. В требуемом состоянии различий нет, то есть их число чётно, значит, при нечётном количестве различий решение невозможно. Если различий чётное число, разобьём их на пары и для каждой пары выполним одну операцию. После этого все требуемые позиции будут исправлены.

Сложность: $O(n)$, память $O(n)$.

```
n = int(input())
a = input()
b = input()
diff = [i + 1 for i in range(n) if a[i] != b[i]]
if len(diff) % 2:
    print("NO SOLUTION")
else:
    print(len(diff) // 2)
    for i in range(0, len(diff), 2):
        print(diff[i], diff[i + 1])
```

