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

Задание 1.

Решающий пень задаёт об объекте ровно один вопрос: «$x \geqslant t$?» — где t — некоторое вещественное число. Для каждого из двух возможных ответов заранее задаётся предсказываемый класс: белый или чёрный. Эти классы могут совпадать. Какое наибольшее число правильных предсказаний может дать решающий пень на следующей выборке?



Ответ. 5.

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

Решение.

Оценка. Посмотрим на три объекта: 1 (белый), 4 (чёрный) и 10 (белый).

Граница делит выборку на две части: значения меньше t и значения не меньше t. Внутри каждой части пень выдаёт один и тот же цвет. Объект 4 лежит в одной части с объектом 1 или в одной части с объектом 10.

В обоих случаях в одной части оказываются чёрный объект и белый, а ответ на них один и тот же, значит, хотя бы один предсказан неверно. Поэтому правильных предсказаний не больше пяти.

Пример на 5. Возьмём $t = 4$ и будем на ответ «да» выдавать чёрный цвет, на «нет» — белый. Тогда 1 и 3 объявлены белыми, а 4, 5, 7 и 10 — чёрными. Неверно предсказан только объект 10.

Задание 2.

Классификатор относит объекты к положительному или отрицательному классу.

Обозначим:

TP — число объектов положительного класса, верно отнесённых к положительному классу;

FN — число объектов положительного класса, ошибочно отнесённых к отрицательному классу;

TN — число объектов отрицательного класса, верно отнесённых к отрицательному классу;

FP — число объектов отрицательного класса, ошибочно отнесённых к положительному классу. Используются следующие метрики:

$$\text{полнота (recall)} = \frac{TP}{TP + FN},$$

$$\text{специфичность (specificity)} = \frac{TN}{TN + FP},$$

$$\text{точность (precision)} = \frac{TP}{TP + FP}.$$

Модель имеет полноту 0.9 и специфичность 0.9. Долей положительного класса назовём отношение числа объектов положительного класса (то есть $TP + FN$) ко всем объектам выборки.

Определите долю положительного класса при точности 0.5.

Определите долю положительного класса при точности 0.9.

Ответ. По порядку вопросов: 0,1; 0,5.

Критерии оценивания. По 6 баллов за каждый правильный ответ. Баллы за пункты суммируются, максимум — 12. За неверный или отсутствующий ответ на отдельно оцениваемый пункт — 0 баллов, если частичное оценивание не указано отдельно.

Решение.

Объекты положительного класса — это те, что классификатор угадал (TP) и не угадал (FN), поэтому их всего $P = TP + FN$. Аналогично объектов отрицательного класса $N = TN + FP$.

Полнота равна $\dfrac{TP}{P} = 0{,}9$, откуда $TP = 0.9P$ и $FN = 0.1P$. Специфичность равна $\dfrac{TN}{N} = 0{,}9$, откуда $TN = 0.9N$ и $FP = 0.1N$.

Подставим это в точность:

$$\text{точность} = \frac{TP}{TP + FP} = \frac{0{,}9P}{0{,}9P + 0{,}1N} = \frac{9P}{9P + N},$$

а доля положительного класса равна

$$\frac{TP + FN}{TP + FN + TN + FP} = \frac{P}{P + N}.$$

Если точность равна 0.5, то $9P + N = 18P$, то есть $N = 9P$. Тогда доля положительного класса равна $\dfrac{P}{P + 9P} = \dfrac{1}{10} = 0.1$.

Если точность равна 0.9, то $9P = 0.9\,(9P + N)$, то есть $0.9P = 0.9N$ и $N = P$. Тогда доля положительного класса равна $\dfrac{P}{P + P} = \dfrac{1}{2} = 0{,}5$.

Задание 3.

Дана выборка из 7 объектов. Значения признака у объектов равны 0, 1, 4, 10, 14, 20, 22. Каждому объекту соответствует метка класса, равная 0 или 1; метки нам неизвестны.

Для каждого объекта алгоритм находит ближайший (по расстоянию между значениями признака) к нему среди остальных объектов и предсказывает, что метка у них одинаковая.

Во всех семи случаях алгоритм сделал верное предсказание. Сколько объектов могло иметь метку 0? Укажите все подходящие варианты. Каждый ответ записывайте в отдельное поле. добавляя их при необходимости.

Ответ. Все подходящие варианты: $0, 2, 3, 4, 5, 7$.

Критерии оценивания. Полный набор шести правильных значений без лишних — 12 баллов; неполный, неверный или отсутствующий ответ — 0 баллов.

Решение.

Найдём для каждого объекта ближайший к нему другой объект.

Все предсказания верны тогда и только тогда, когда внутри каждой из следующих групп метки одинаковы:

$$\{0,\ 1,\ 4\}, \qquad \{10,\ 14\}, \qquad \{20,\ 22\}.$$

Значит, подходит любая расстановка меток, в которой метки внутри каждой из этих групп совпадают, и никакая другая. В первой группе либо ни одного, либо три объекта имеют метку 0, а в каждой из двух остальных групп либо ни одного, либо два объекта имеют метку 0. Поэтому количество объектов с меткой 0 может быть равно 0, 2, 3, 4, 5 или 7, и каждое из этих шести значений действительно достигается.

Задание 4.

Файл: matches.csv

В файле содержатся результаты всех матчей футбольного чемпионата. В чемпионате участвуют 16 команд. Каждая команда играет с каждой другой два раза: один раз дома и один раз в гостях. Таким образом, в файле находятся результаты 240 матчей.

Файл содержит четыре столбца:

home_team — название команды хозяев;

away_team — название команды гостей;

home_goals — число голов хозяев;

away_goals — число голов гостей.

За победу команда получает 3 очка, за ничью — 1 очко, за поражение — 0 очков.

Часть 1. Исходная турнирная таблица

Выполните следующие действия:

1. Для каждого матча определите, сколько очков получили хозяева и сколько — гости.

2. Для каждой команды сложите очки, набранные во всех её матчах. Расположите команды по убыванию суммы очков и получите исходную турнирную таблицу.

Определите победителя исходной таблицы. Запишите название команды.

Поле ввода

Определите отрыв по очкам победителя от команды на втором месте.

Поле ввода

Определите команду, занявшую последнее место.

Поле ввода

После проведения всех матчей команда, занявшая последнее место, снялась с чемпионата, выразив несогласие с рядом судейских решений. По регламенту результаты всех матчей с её участием аннулировали.

Выполните следующие действия:

1. Удалите из данных все матчи, в которых снявшаяся команда была хозяином или гостем.

2. Для оставшихся матчей повторите начисление и суммирование очков. Составьте новую турнирную таблицу.

Определите количество очков команды, занявшей первое место в новой таблице.

Гарантируется, что в новой таблице победитель набрал строго больше очков, чем команда на втором месте.

Поле ввода

.

Ответ. По порядку вопросов: «Зенит»; 8; «Спартак»; 56.

Критерии оценивания. Баллы за правильные ответы по порядку вопросов: 2; 3; 3; 4 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.

Напишем функцию, которая по таблице матчей начисляет очки хозяевам и гостям, объединяет результаты и возвращает команды в порядке убывания набранных очков. Сначала применим её ко всем матчам. Последняя строка полученной таблицы укажет команду, которая снялась с чемпионата. Удалим все матчи с её участием и вызовем ту же функцию ещё раз.

Решение на языке Python:

```
import pandas as pd
import numpy as np

def make_table(matches):
    home_win = matches["home_goals"] > matches["away_goals"]
    draw = matches["home_goals"] == matches["away_goals"]
    away_win = matches["home_goals"] < matches["away_goals"]

    home_points = np.select([home_win, draw], [3, 1], default=0)
    away_points = np.select([away_win, draw], [3, 1], default=0)

    home = pd.DataFrame({
        "team": matches["home_team"], "points": home_points
    })
    away = pd.DataFrame({
        "team": matches["away_team"], "points": away_points
    })

    return (pd.concat([home, away], ignore_index=True)
            .groupby("team")["points"].sum()
            .sort_values(ascending=False))

df = pd.read_csv("data/matches.csv")
table = make_table(df)
```

```
print(table.index[0])
print(int(table.iloc[0] - table.iloc[1]))

withdrawn = table.index[-1]
remaining = df[
    (df["home_team"] != withdrawn)
    & (df["away_team"] != withdrawn)
]
new_table = make_table(remaining)

print(withdrawn)
print(int(new_table.iloc[0]))
```

В исходной таблице «Зенит» набрал 62 очка, а «Крылья Советов» — 54 очка. Поэтому отрыв победителя равен 8 очкам. Последнее место занял «Спартак».

После аннулирования матчей с участием «Спартака» первое место сохранил «Зенит», набравший 56 очков.

Ответ: «Зенит»; 8; «Спартак»; 56.

Задание 5.

Лучевой поиск

Ограничение по времени: 2 секунды

Илья решил своими руками написать лучевой поиск (beam search). Этот алгоритм хранит несколько кандидатов, у каждого из которых есть числовая оценка — скор. На каждом шаге кандидатов продолжают, скоры пересчитывают, а худших кандидатов удаляют.

В код Ильи закралась грубая ошибка: обновление просто прибавляет ко скору каждого кандидата одно и то же число x. Удаление при этом работает так, как задумано: код удаляет кандидата с наименьшим скором и записывает этот скор в лог. Если кандидатов с наименьшим скором несколько, удаляется любой из них — на записанное значение это не влияет. Удалённый кандидат в поиске больше не участвует, и последующие обновления его не меняют.

Изначально у Ильи n кандидатов, скор i-го из них равен a_i. Затем код выполняет q действий — обновлений и удалений — в известном порядке. Помогите Илье узнать, какие числа окажутся в логе.

Формат входных данных

В первой строке ввода даны два целых числа n и q — количество кандидатов и количество действий ($1 \le n, q \le 2 \cdot 10^5$).

Во второй строке через пробел перечислены n целых чисел a_i — начальные скоры кандидатов ($1 \leqslant a_i \leqslant 10^6$).

В следующих q строках описаны действия в порядке выполнения. Каждое имеет один из двух видов:

1 x — обновление: скор каждого оставшегося кандидата увеличивается на x ($1 \leqslant x \leqslant 1000$);

2 — удаление кандидата с наименьшим скором среди оставшихся.

Гарантируется, что среди действий есть хотя бы одно удаление и всего удалений не больше n.

Формат выходных данных

Для каждого удаления в порядке выполнения действий выведите в отдельной строке скор удалённого кандидата.

Система оценивания

Тесты из условия не оцениваются. Остальные тесты оцениваются независимо друг от друга, каждый — в 1 балл.

Группа	Тестов	Баллы	Дополнительные ограничения
1	5	5	n, $q \leqslant 2000$
2	7	7	без дополнительных ограничений

Замечание

В первом примере три кандидата со скорами 5, 1 и 3.

Действие 1: скоры увеличиваются на 2 и становятся 7, 3, 5.

Действие 2: удаляется кандидат со скором 3, остаются 7 и 5.

Действие 3: удаляется кандидат со скором 5, остаётся 7.

Действие 4: скор увеличивается на 10 и становится 17.

Действие 5: удаляется кандидат со скором 17.

Ответ. При j-м удалении выводится $a_{(j)} + \Delta_j$, где $a_{(j)}$ — j-й элемент исходного массива после сортировки по неубыванию, а Δ_j — сумма прибавлений до этого удаления.

Критерии оценивания. Должны быть верны все выведенные значения скора удалённых кандидатов в порядке удалений. Группа 1: 5 тестов, $n, q \leqslant 2000$; группа 2: 7 тестов 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.

Каждое обновление прибавляет одно и то же число ко всем ещё не удалённым кандидатам. Поэтому отсортированный по скору порядок кандидатов никогда не меняется: кандидат с минимальным текущим скором — это кандидат с минимальным исходным скором среди оставшихся.

Отсортируем исходные a_i. Отдельно будем хранить общую прибавку *add*. При удалении берём очередной элемент отсортированного массива и выводим его плюс *add*.

Сложность: $O(n \log n + q)$, память $O(n)$.

```
n, q = map(int, input().split())
a = sorted(map(int, input().split()))

add = 0
ptr = 0

for _ in range(q):
    op = list(map(int, input().split()))
    if op[0] == 1:
        add += op[1]
    else:
        print(a[ptr] + add)
        ptr += 1
```

Задание 6.

Самая необычная строка

Ограничение по времени: 1 секунда

Для эксперимента в области машинного обучения собран набор из n объектов. Каждый объект описывается m бинарными признаками: значение каждого признака равно 0 или 1.

Объекты пронумерованы от 1 до n в том порядке, в котором они записаны в наборе данных.

Для оценки того, насколько объект отличается от остальных объектов, используется расстояние Хэмминга. Расстоянием Хэмминга между двумя объектами называется количество признаков, значения которых у этих объектов различаются.

Необычностью объекта назовём сумму расстояний Хэмминга от него до всех объектов набора.

Требуется найти объект с максимальной необычностью и значение его необычности. Если максимальная необычность достигается у нескольких объектов, следует выбрать объект с наименьшим номером.

Формат входных данных

В первой строке вводятся через пробел два натуральных числа n и m ($1 \leqslant n \cdot m \leqslant 200000$).

В следующих n строках вводится по m цифр (0 или 1) в каждой.

Формат выходных данных

Выведите два числа через пробел: номер (наименьший) самой необычной строки и её необычность.

Ответ. Пусть c_j — число единиц в столбце j, а $D_i = \sum_{j:a_{ij}=1}(n - c_j) + \sum_{j:a_{ij}=0} c_j$. Выводятся наименьший индекс i с максимальным D_i и само D_i.

Критерии оценивания. Оба выведенных числа должны быть верными; при равенстве необычности выбирается наименьший индекс.

Решение.

Для каждого столбца посчитаем число единиц c_j. Если в строке в столбце j стоит 1, она отличается в этом столбце от $n - c_j$ строк; если стоит 0 — от c_j строк. Сумма этих величин по всем столбцам и есть необычность строки. Переберём строки по порядку и обновим ответ только при строгом улучшении, чтобы при равенстве сохранить меньший номер.

Сложность: $O(nm)$, память $O(nm + m)$.

```
import sys


def main():
    n, m = map(int, sys.stdin.readline().split())
    rows = [sys.stdin.readline().strip() for _ in range(n)]
    ones = [0] * m
    for row in rows:
        for j, ch in enumerate(row):
            ones[j] += ch == "1"
    best_idx = 1
    best_score = -1
    for i, row in enumerate(rows, 1):
        score = 0
        for j, ch in enumerate(row):
            score += n - ones[j] if ch == "1" else ones[j]
        if score > best_score:
            best_score = score
            best_idx = i
    print(best_idx, best_score)


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

