Максимальное количество баллов за олимпиаду — 600

Задание 1. Конференция по ИИ

На конференцию по применению ИИ съехались 15 учёных из 15 разных стран, один из них — из России. У каждого иностранного учёного есть совместные статьи ровно с 13-ю гостями конференции. Сколько участников конференции могут являться соавторами нашего соотечественника? Укажите все подходящие варианты. Каждый ответ записывайте в отдельное поле, добавляя их при необходимости.

Ответ: 0, 2, 4, 6, 8, 10, 12, 14

Критерий оценивания: точное совпадение ответа — 100 баллов. За ответ {2, 4, 6, 8, 10, 12, 14} — 60 баллов

Максимальный балл за задание — 100

Решение. Пусть каждый учёный пожмёт руку тому, с кем у него нет совместной статьи. Тогда каждый иностранец пожмёт руку одному другому учёному. Если n пар иностранцев пожали друг другу руки, то наш соотечественник пожал руку $15 - 2n$ учёным, а с оставшимися $2n$ у него есть совместные статьи. При $n = 0, 1, 2, \ldots 7$ получаются соответствующие примеры.

Задание 2. Ошибающийся чат-бот

Модель ИИ по запросу «Какое наибольшее число прямоугольников из 4 клеток можно вырезать из квадрата 25×25?» выдала ответ на запрос «Какое наибольшее число квадратиков из 4 клеток можно вырезать из прямоугольника 25×25?». На сколько ответ модели отличается от верного?

Ответ: 12

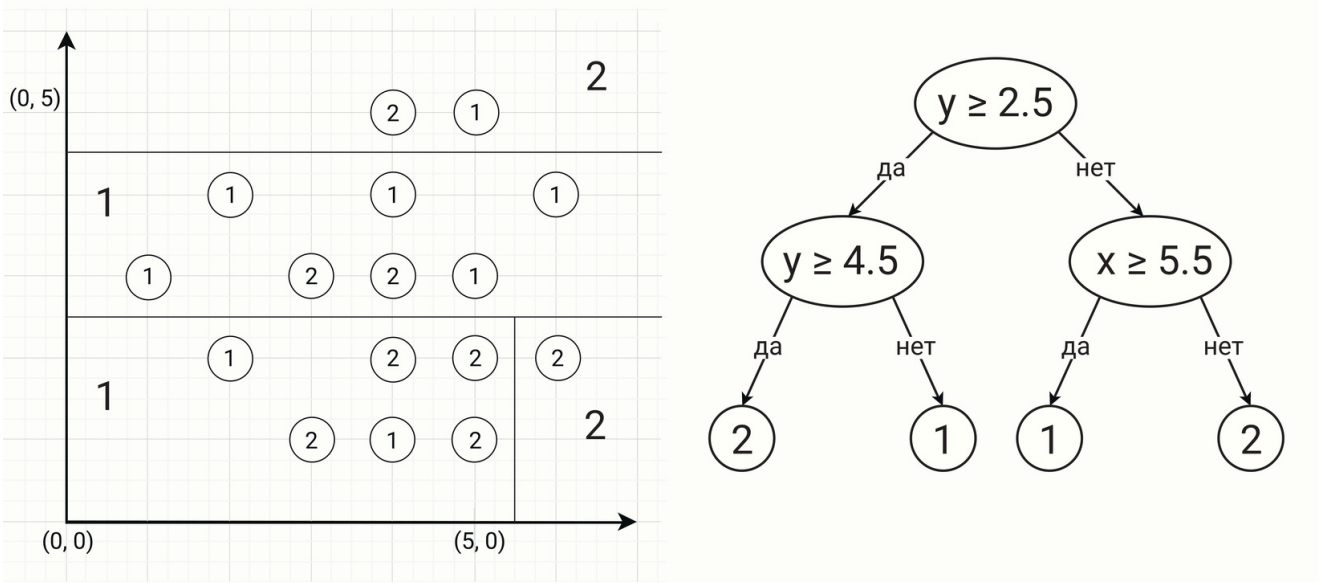
Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

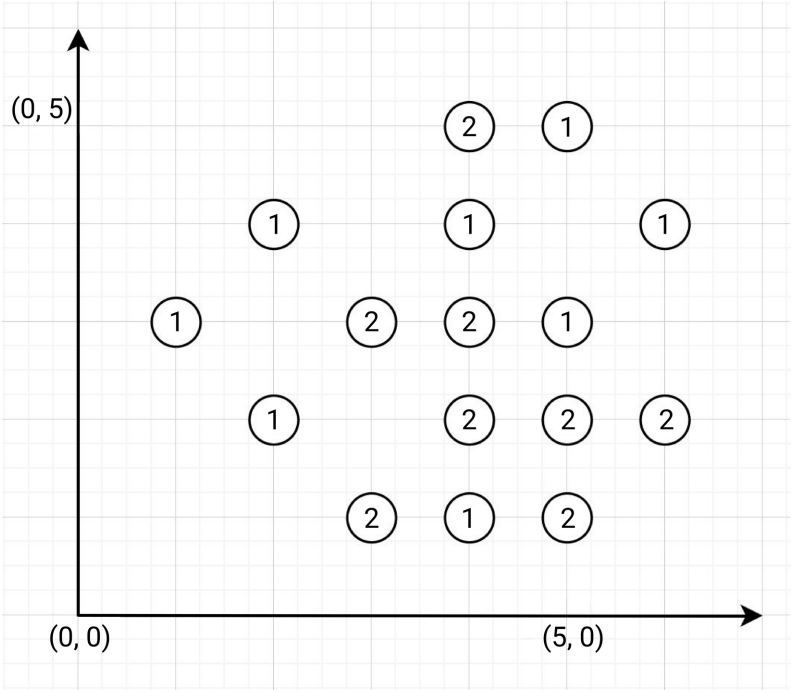
Решение. Ответим на оба вопроса. Легко видеть, что квадрат 24×24 разбивается на 144 квадратика 2×2, а из оставшейся каёмки можно вырезать ещё 12 прямоугольников из 4 клеток (отметим, что квадрат — тоже прямоугольник), итого 156 прямоугольников. Больше 156 прямоугольников вырезать нельзя, поскольку $157 \cdot 4 = 628$, а в квадрате 25×25 всего 625 клеточек. А больше 144 квадратов 2×2 не вырезать, потому что каждый такой квадрат должен содержать клетку на пересечении строки и столбца с чётными номерами (мы считаем, что строки и столбцы последовательно нумеруются 1,2,..., 25), а таких клеток 144. Итого: верный ответ — 144, ответ модели — 156, отличие равно 12.

Задание 3. Решающее дерево глубины 2

Решающее дерево глубины 2 — это полное бинарное дерево с 7 вершинами: в 4 листьях стоят предсказанные классы, в 3 внутренних вершинах — пороговые условия на одну переменную, например, $y \geqslant 2.5$ (то есть проверки вида $x \geqslant t$ или $y \geqslant t$). Пример такого решающего дерева приведён ниже.

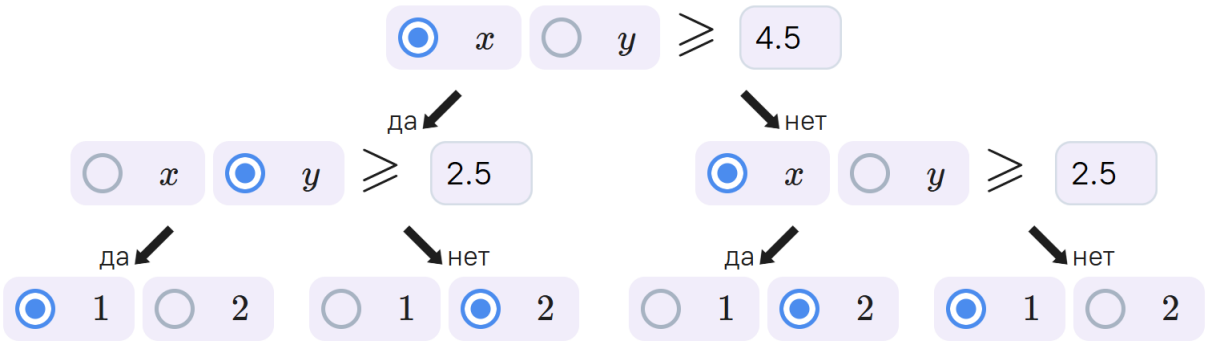


Даны 16 точек на плоскости, каждая принадлежит одному из классов 1, 2.



Заполните пропуски в решающем дереве глубины 2 так, чтобы доля правильно предсказанных ответов оказалась наибольшей.

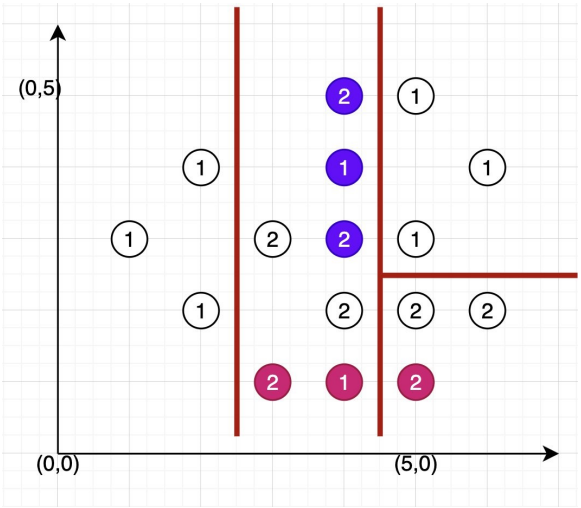
Ответ:



Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение.



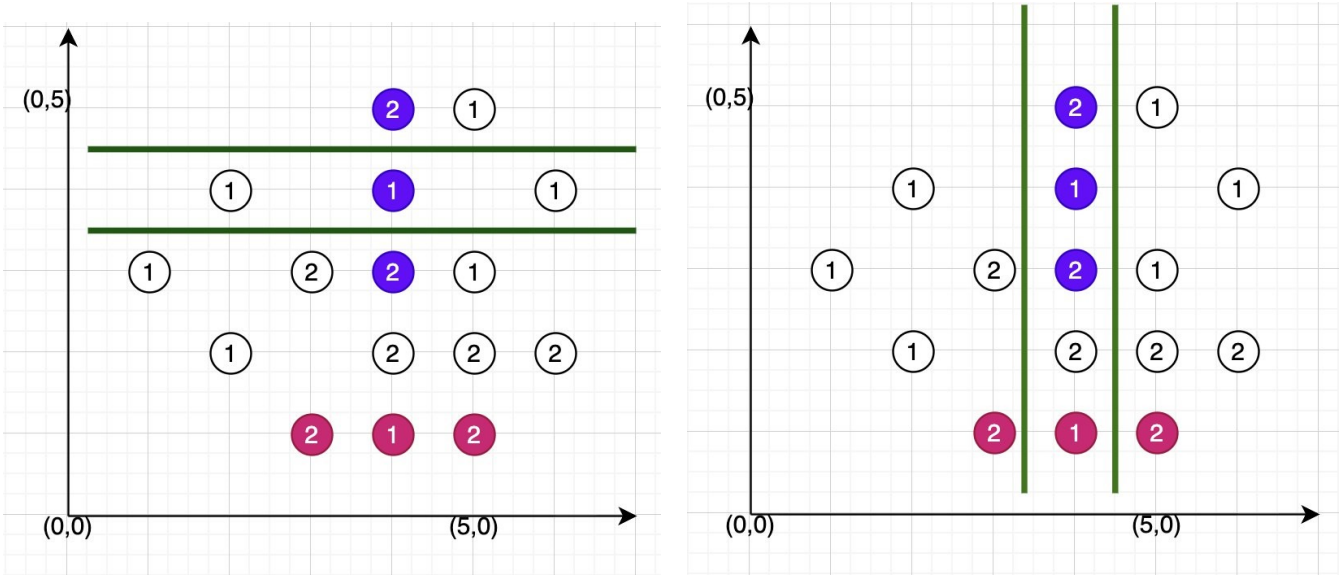
Рассмотрим три случая.

1. Все 3 фиолетовые точки классифицированы правильно. Тогда два условия на глубине 1 должны быть горизонтальными и разделять пары соседних фиолетовых точек. Эти условия нельзя дополнить третьим так, чтобы суммарная точность превысила 14/16.

2. Аналогично: если все 3 розовые точки классифицированы правильно, точность не превосходит 14/16.

3. Если неверно классифицирована хотя бы одна фиолетовая и хотя бы одна розовая точки, то суммарная точность окажется также не больше 14/16.

Существует конфигурация дерева, дающая 14/16 (пример на рисунке), значит, верхняя граница достижима.



Задание 4. Тикеты поддержки

В службе поддержки каждый тикет автоматически помечается вероятностью того, что он критический. Есть разные издержки: если пометим критическим обычный тикет, придётся зря отвлекать команду, а если не заметим критический тикет, понесём серьёзные потери.

Дан файл, который вы можете скачать в форматах XLSX, ODS или CSV. В каждой строке файла записаны четыре значения:

- `prob` — предсказанная вероятность того, что тикет критический, число из интервала [0,1];
- `label` — истинная метка: 0 (обычный тикет) или 1 (критический);
- `costFP` — штраф за ложное срабатывание;
- `costFN` — штраф за пропуск критического тикета.

Для каждой строки применяется следующее правило классификации:

$$\widehat{y} = \begin{cases} 1, \text{ если } p \geqslant \dfrac{\text{costFP}}{\text{costFP} + \text{costFN}}, \\ 0, \text{ иначе.} \end{cases} \tag{1}$$

Для скольких строк предсказание совпадёт с истинной меткой, то есть $\widehat{y} = \text{label}$?

Ответ: 819

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. Для каждой строки вычисляем порог

$$\text{thr} = \frac{\texttt{costFP}}{\texttt{costFP} + \texttt{costFN}},$$

после чего предсказываем класс

$$\hat{y} = \mathbf{1}\big[\texttt{prob} \geq \text{thr}\big].$$

Ответом является количество строк, в которых предсказание совпадает с истинной меткой: $\hat{y} = \texttt{label}$.

Код (pandas):

```
import pandas as pd

df = pd.read_csv("task7_cost_sensitive.csv")

thr = df["costFP"] / (df["costFP"] + df["costFN"])
answer = ((df["prob"] >= thr) == df["label"]).sum()

print(answer)
```

Задание 5. Кросс-валидация

Однажды Саша решил обучить классификатор. К сожалению, ресурсов на сложную нейронную сеть у него не было, поэтому он решил, что классификатор будет предсказывать класс, который чаще всего встречался в обучающих данных. Если таких было несколько, он будет предсказывать класс с наименьшим номером. Саша недавно узнал о кросс-валидации и поэтому решил оценить свою модель именно с помощью неё.

Всего у Саши есть n объектов, i-й из которых имеет класс c_i. Для кросс-валидации Саша разделил эти данные на части по t объектов (последняя часть при этом могла оказаться короче). Для каждой из частей классификатор обучается на данных из всех остальных частей и считает число правильных ответов модели по данным из текущей части.

В итоге он суммирует число правильных ответов для всех разбиений. Таким образом, получается некоторая оценка точности модели. Помогите Саше: найдите значение точности модели, посчитанное с помощью кросс-валидации.

Формат входных данных

В первой строке вводятся два целых числа n, t ($1 \leqslant n \leqslant 200000$, $1 \leqslant t \leqslant 1000$).
Во второй строке вводятся n целых чисел c_i — номера классов ($1 \leqslant c_i \leqslant 100$).

Формат выходных данных

Выведите одно целое число: суммарное количество правильных ответов по всем разбиениям в кросс-валидации.

Замечание

В первом тестовом примере данные будут разбиты на части {1, 1, 2}, {2, 2, 3}, {3}:

- при тестовом блоке {1, 1, 2}, самым частым среди оставшихся классов будет 2-й (по количеству он совпадает с 3-м, но модель выберет наименьший по номеру), поэтому для этого блока будет 1 правильное предсказание;
- при тестовой части {2, 2, 3} самым частым среди оставшихся классов будет 1-й, поэтому для этой части будет 0 правильных предсказаний;
- при тестовой части {3} самым частым среди оставшихся классов будет 2-й, поэтому для этой части будет 0 правильных предсказаний.

Примеры

стандартный ввод	стандартный вывод
7 3 1 1 2 2 2 3 3	1

Ответ: 1 правильное предсказание.
Критерий оценивания: точное совпадение ответа — 100 баллов
Максимальный балл за задание — 100

Решение

Сначала вычислим вспомогательный массив `cnt_all`[100], который хранит общее количество объектов каждого класса. Это можно сделать за один проход по исходному массиву за $O(n)$.

Далее, для каждого блока, на котором проводится тестирование, посчитаем массив `cnt`[100], где `cnt`[i] — количество элементов класса i в данном блоке.

Тогда самый часто встречающийся элемент в оставшемся массиве определяется как индекс максимального элемента массива `cnt_others`, где `cnt_others`[i] = `cnt_all`[i] − `cnt`[i]. Поиск максимума выполняется за $O(100)$.

Суммарно подсчёт массива `cnt` по всем блокам занимает $O(n)$. Количество блоков в разбиении равно $\lceil n/k \rceil$.

Итоговая асимптотическая сложность алгоритма равна $O(n + (n/k) \cdot 100)$.

```
n, k = map(int, input().split())
a = list(map(int, input().split()))

C = 100
cnt = [0 for _ in range(C)]
for i in range(n):
        a[i] -= 1
        cnt[a[i]] += 1


sum_ans = 0
for l in range(0, n, k):
        block_cnt = [0 for _ in range(C)]
        for i in range(l, min(l + k, n)):
                block_cnt[a[i]] += 1
                cnt[a[i]] -= 1

        mx = max(cnt)
        mx_a = cnt.index(mx)
        sum_ans += block_cnt[mx_a]
        for i in range(l, min(l + k, n)):
                cnt[a[i]] += 1

print(sum_ans)
```

Задание 6. Шаг K-Means

Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

Современные сервисы, такие как карты или маркетплейсы, часто используют кластеризацию — разбиение множества объектов на группы по сходству. Например, можно сгруппировать магазины по районам или фотографии по похожим цветам. Один из самых простых и популярных методов кластеризации — k-means.

Этот метод ищет группы точек, расположенных близко друг к другу. Основная идея состоит в том, что каждая группа описывается своим центром (средней точкой), а обучение происходит итерационно: точки относят к ближайшим центрам, а затем центры пересчитывают как среднее значение всех точек, попавших в группу. Эта простая идея лежит в основе многих систем машинного обучения и анализа данных.

В этой задаче вам предстоит реализовать один шаг обновления центров кластеров. Имеется множество точек на плоскости и несколько текущих центров. Необходимо выполнить стандартную операцию алгоритма k-means:

1. Каждая точка относится к ближайшему центру.

2. Координаты каждого центра заменяются на среднее арифметическое координат всех точек, отнесённых к нему.

Если к центру не относится ни одна точка, то его положение не меняется. Если точка одинаково близка к нескольким центрам, она относится к тому, у которого первая координата меньше, а при их равенстве — вторая координата меньше.

Координаты новых центров необходимо вывести в том же порядке, в котором заданы исходные центры.

Формат входных данных

Первая строка содержит два целых числа n и k — количество точек и количество кластеров ($1 \leqslant k \leqslant n \leqslant 10^5$).
Далее даны n строк по два целых числа x_i, y_i — координаты точек.
Затем следуют k строк два вещественных числа $c_{x,j}$, $c_{y,j}$, — координаты текущих центров ($|x_i|$, $|y_i|$, $|c_{x,j}|$, $|c_{y,j}| \leqslant 10^6$).

Формат выходных данных

Выведите k строк по два вещественных числа с точностью 10^{-4} — координаты новых центров кластеров после пересчёта — в том же порядке, что и во входных данных.

Система оценки

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

Подзадача	Баллы	Доп. ограничения
1	20	$k = 1$
2	80	$1 \leqslant nk \leqslant 10^6$

Замечание

Первые три точки ближе к первому центру $(0, 0)$ оставшиеся две — ко второму $(10, 0)$. После шага алгоритма центры пересчитываются как средние значения точек, относящихся к каждому кластеру: (1.0, 0.3333), (8.5, 2.5).

Примеры

стандартный ввод	стандартный вывод
5 2 0 0 1 1 2 0 8 3 9 2 0.0 0.0 10.0 0.0	1.0000 0.3333 8.5000 2.5000

Решение

Дан один шаг алгоритма K-Means. Он состоит из двух логических этапов: распределения точек по кластерам и пересчёта центров.

Сначала для каждой точки необходимо определить ближайший центр. Для этого для каждой пары «точка–центр» вычисляется квадрат евклидова расстояния $dist^2 = (x - c_x)^2 + (y - c_y)^2$. Извлекать квадратный корень не требуется, так как он не влияет на порядок сравнения расстояний.

Если точка оказывается на одинаковом расстоянии от нескольких центров, она относится к центру с меньшей первой координатой, а при равенстве — с меньшей второй координатой. Это правило обеспечивает однозначность распределения.

После того как все точки распределены по кластерам, для каждого центра пересчитываются координаты. Новое положение центра равно среднему арифметическому координат всех точек, отнесённых к данному кластеру: $c_{x,\text{new}} = \frac{\sum x_i}{m}$, $c_{y,\text{new}} = \frac{\sum y_i}{m}$, где суммирование ведётся по всем m точкам кластера. Если к центру не была отнесена ни одна точка, его координаты остаются без изменений.

Для каждой из n точек перебираются k центров, поэтому основной этап алгоритма работает за $O(n \cdot k)$. Пересчёт координат центров выполняется за $O(n)$. Таким образом, асимптотическая сложность одного шага алгоритма равна $O(n \cdot k)$.

```python
line = input().split()
n = int(line[0])
k = int(line[1])

points = []
for i in range(n):
    x, y = map(int, input().split())
    points.append([x, y])

centroids = []
for i in range(k):
    cx, cy = map(float, input().split())
    centroids.append([cx, cy])

cluster_sum_x = [0.0] * k
cluster_sum_y = [0.0] * k
cluster_count = [0] * k

for px, py in points:
    best_dist = float('inf')
    best_cluster = 0

    for j in range(k):
        cx, cy = centroids[j]
        dx = px - cx
        dy = py - cy
        dist = dx * dx + dy * dy

        if dist < best_dist:
            best_dist = dist
            best_cluster = j
        elif dist == best_dist:
            if cx < centroids[best_cluster][0]:
                best_cluster = j
            elif cx == centroids[best_cluster][0] and cy < centroids[best_cluster][1]:
                best_cluster = j

    cluster_sum_x[best_cluster] += px
    cluster_sum_y[best_cluster] += py
    cluster_count[best_cluster] += 1

for j in range(k):
    if cluster_count[j] > 0:
        print(cluster_sum_x[j] / cluster_count[j],
              cluster_sum_y[j] / cluster_count[j])
    else:
        print(centroids[j][0], centroids[j][1])
```

Максимальный балл за задание — 100