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

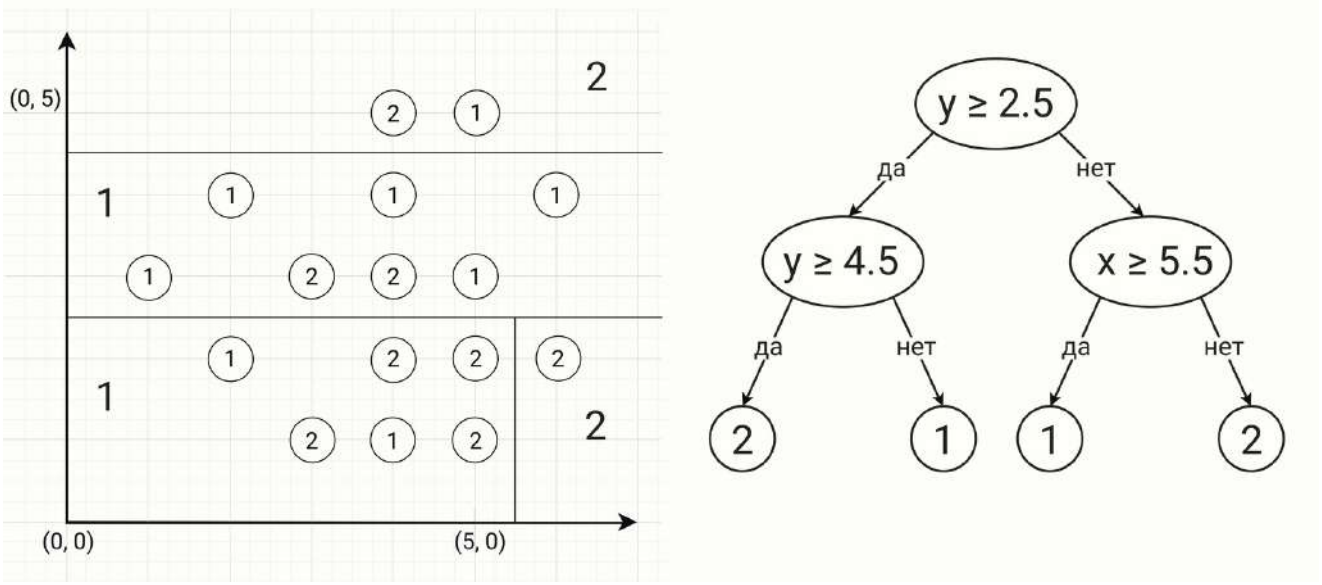
Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

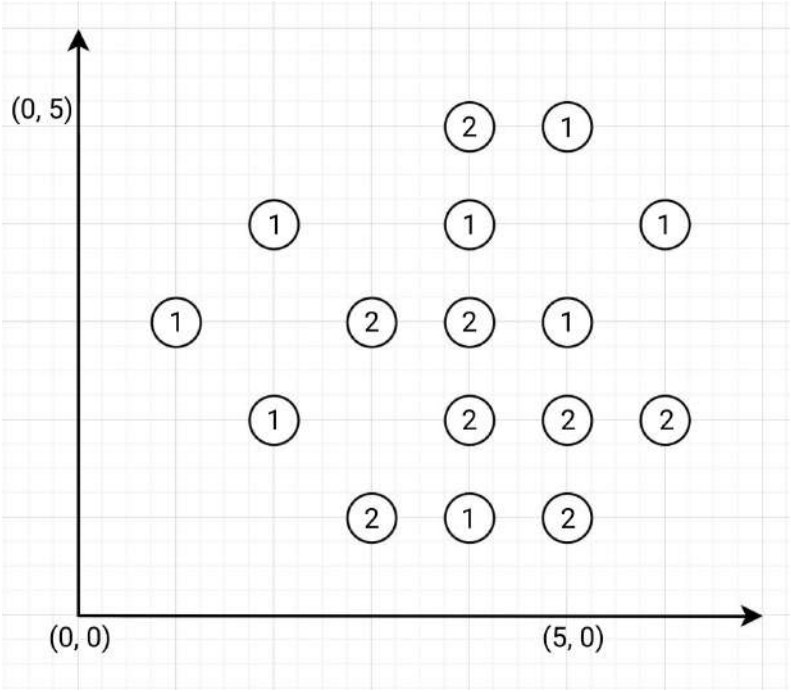
Решение. Ответим на оба вопроса. Легко видеть, что квадрат 24×24 разбивается на 144 квадратика 2×2, а из оставшейся каёмки можно вырезать ещё 12 прямоугольников из 4 клеток (отметим, что квадрат — тоже прямоугольник), итого 156 прямоугольников. Больше 156 прямоугольников вырезать нельзя, поскольку $157 \cdot 4 = 628$, 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.

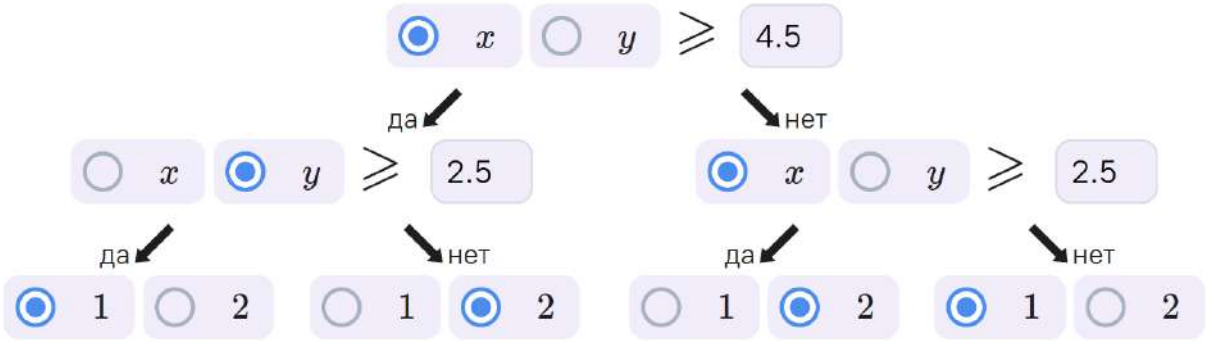


Даны 16 точек на плоскости, каждая принадлежит одному из классов 1, 2.



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

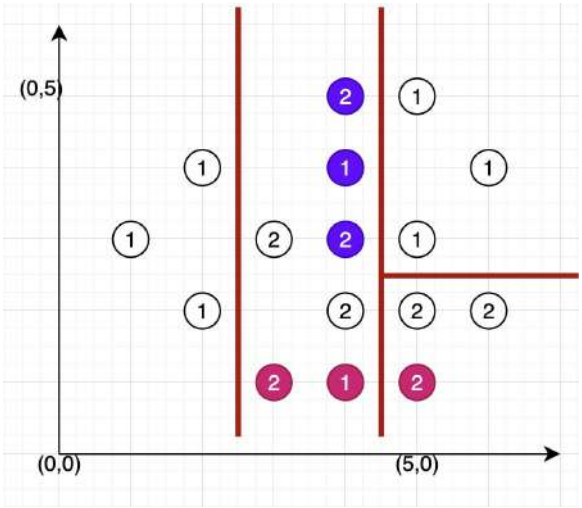
Ответ:



Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение.



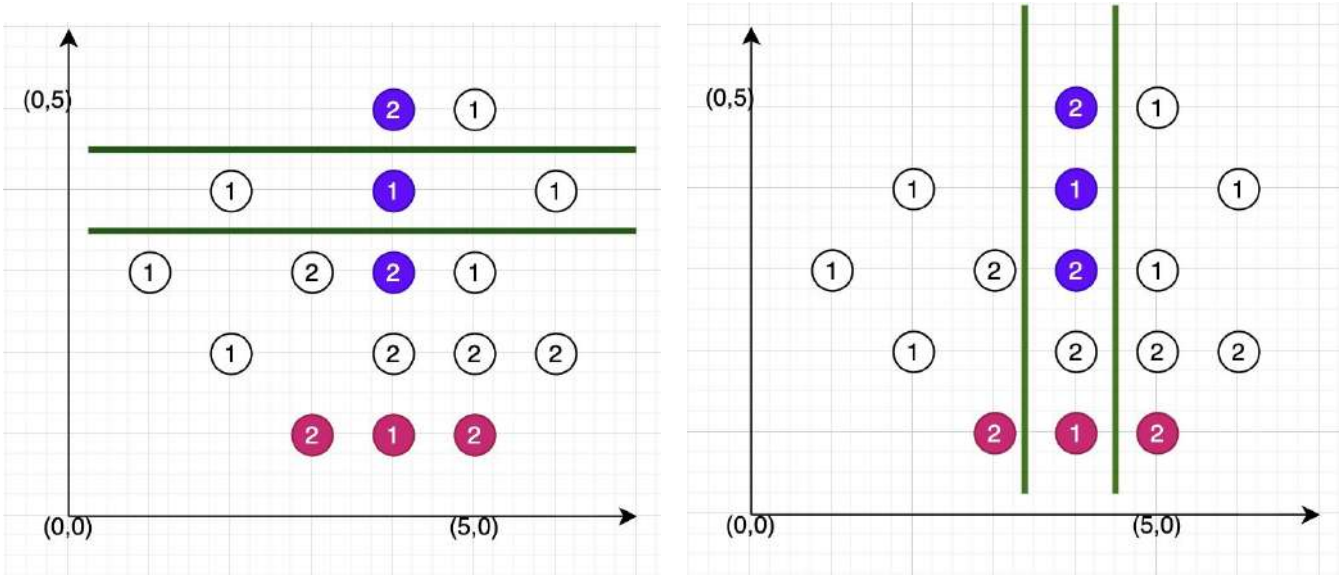
Рассмотрим три случая.

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

Максимальное количество баллов за олимпиаду — 600

Задание 1. Ошибающийся чат–бот

Поиск в интернете с помощью ИИ по запросу «Какое наибольшее число попарно пересекающихся двухэлементных подмножеств можно выбрать из девятиэлементного множества?» выдал правильный ответ на другой запрос «Какое наибольшее число попарно непересекающихся двухэлементных подмножеств можно выбрать из девятиэлементного множества?».

На сколько полученный ответ отличается от истинного?

Ответ: 4 или −4

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. Ясно, что ответ на второй вопрос равен 4.

Покажем, что ответ на первый вопрос — 8. Пример на 8 подмножеств получается, если взять все 8 подмножеств, содержащих один и тот же элемент. Пусть мы выбрали большее число подмножеств, выбранные множества A и B пересекаются по элементу, который будем называть первым. Тогда какое–то выбранное множество C первый элемент не содержит, то есть оно состоит из двух элементов множеств A и B, отличных от первого. Теперь легко видеть, что никакое другое двухэлементное подмножество не может пересекаться одновременно и с A, и с B, и с C, противоречие.

Задание 2. Расстояние между словами

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

- убрать из слова одну любую букву;
- добавить в любое место слова любую букву.

За какое наименьшее количество таких операций можно из слова ПРИКОЛ получить слово ПОИСК?

Ответ: 5

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. Пяти операций достаточно.

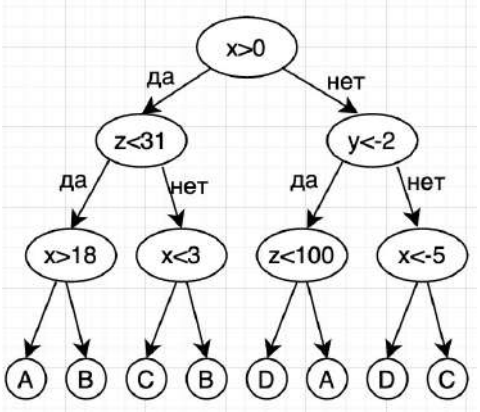
ПРИКОЛ — ПИКОЛ — ПИКО — ПИК — ПОИК — ПОИСК

Меньшего числа операций не хватит. Действительно, если мы удалили хотя бы 3 буквы, то должны были добавить хотя бы 2, итого хотя бы пять операций. В противном случае хотя бы 4 буквы не удалялись, тогда они общие в наших словах — П, О, И, К. Однако их порядок изменился, поэтому не удалять ни одну из них было нельзя.

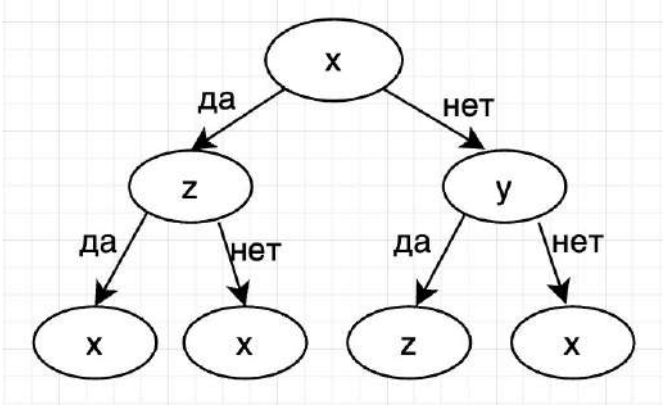
Задание 3. Красим решающее дерево

На доске нарисовали полное бинарное решающее дерево глубины 3, классифицирующее точки (x, y, z). У дерева 15 вершин: в 8 листьях записаны классы, а во всех 7 внутренних вершинах — линейные условия на одну из переменных (например, $y < -2$).

Кто–то стёр все листья, а в каждой внутренней вершине оставил только название переменной, участвовавшей в условии (x, y или z). После стирания выполнено требование: в каждой паре соединённых вершин «предок — потомок» переменные различаются.



До



После

Два результата стирания считаются различными, если существует хотя бы одна внутренняя вершина, содержащая различные переменные. При этом форма дерева и подписи да/нет на рёбрах зафиксированы.

Сколько различных деревьев (с точки зрения подписанных во внутренних вершинах переменных) могло остаться после стирания?

Ответ: 192

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. После стирания остаются 7 внутренних вершин полного бинарного дерева глубины 3. Корень можно подписать любой из трёх переменных. Каждая из остальных 6 внутренних вершин должна отличаться от своего предка и потому имеет 2 допустимых варианта. Итого: $3 \cdot 2^6 = 192$.

Задание 4. Рекомендательная система

Дан файл, который вы можете скачать в форматах XLSX, ODS и CSV. В каждой строке записаны значения `prob1`, `prob2`, `prob3`, `prob4`, `label`. Четыре значения `prob1` — `prob4` — это вероятности, которыми четыре независимых рекомендательных алгоритма оценили, понравится ли пользователю фильм; `label` — реальное мнение пользователя о фильме (1 — понравился, 0 — не понравился).

Считаем *достаточно высокими* вероятности $\geqslant 0.25$. Обозначим через `opt_avg` среднее по всем значениям вероятностей, не меньших 0.25. Если таких значений нет, то под `opt_avg` понимаем обычное среднее четырёх чисел `prob1`–`prob4`.

Пример. Пусть в строке `prob1` = 0.10, `prob2` = 0.90, `prob3` = 0.60, `prob4` = 0.30. Тогда три вероятности — `prob2`, `prob3`, `prob4` — *достаточно высокие* и

$$\texttt{opt\_avg} = \frac{0.90 + 0.60 + 0.30}{3} = 0.6.$$

Если $\texttt{opt\_avg} \geqslant 0.5$, предсказываем `pred` = 1. Иначе — `pred` = 0.

В скольких строках `pred` совпадает с `label`?

Ответ: 953

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. По каждой строке берём только те вероятности $\{p_i\}$, для которых $p_i \geq 0.25$, и усредняем их. Если таких значений нет, используем обычное среднее четырёх чисел. Получаем `opt_avg` и предсказываем

$$\texttt{pred} = \mathbf{1}[\texttt{opt\_avg} \geq 0.5].$$

Ответ — количество строк, в которых `pred` = `label`.

Код (pandas):

```
import pandas as pd

df = pd.read_csv("task8_adaptive_ensemble.csv")

probs = df[["prob1", "prob2", "prob3", "prob4"]]

selected = probs.where(probs >= 0.25)
opt_avg = selected.mean(axis=1)

fallback = probs.mean(axis=1)
opt_avg[opt_avg.isna()] = fallback[opt_avg.isna()]

answer = ((opt_avg >= 0.5) == (df["label"] == 1)).sum()
print(answer)
```

Задание 5. Активация нейрона

Ограничение по времени: 1 секунда

Ограничение по памяти: 256 мегабайт

Персептрон — это самая базовая архитектура нейронной сети. Она состоит из единственного нейрона. На вход нейрону подаётся k чисел a_i. Они суммируются с весами w_i. Таким образом, мы получаем $S = a_1w_1 + a_2w_2 + ... + a_kw_k$. Далее нейрон активируется, если $S \geqslant B$.

Вам поручили изучать зависимости активации нейронов от входных параметров a_i. Более конкретно, требуется при фиксированных a_1,..., a_{k-1}, w_1,..., w_k и B определить такое минимальное целое значение a_k, что нейрон активируется. Гарантируется, что $w_k > 0$.

Формат входных данных

В первой строке вводятся три целых числа k, B, w_k ($1 \leqslant k \leqslant 1000$, $-10^9 \leqslant B \leqslant 10^9$, $1 \leqslant w_k \leqslant 1000$).

В каждой из следующих $k - 1$ строк вводятся по 2 целых числа: a_i и w_i ($-1000 \leqslant a_i, w_i \leqslant 1000$).

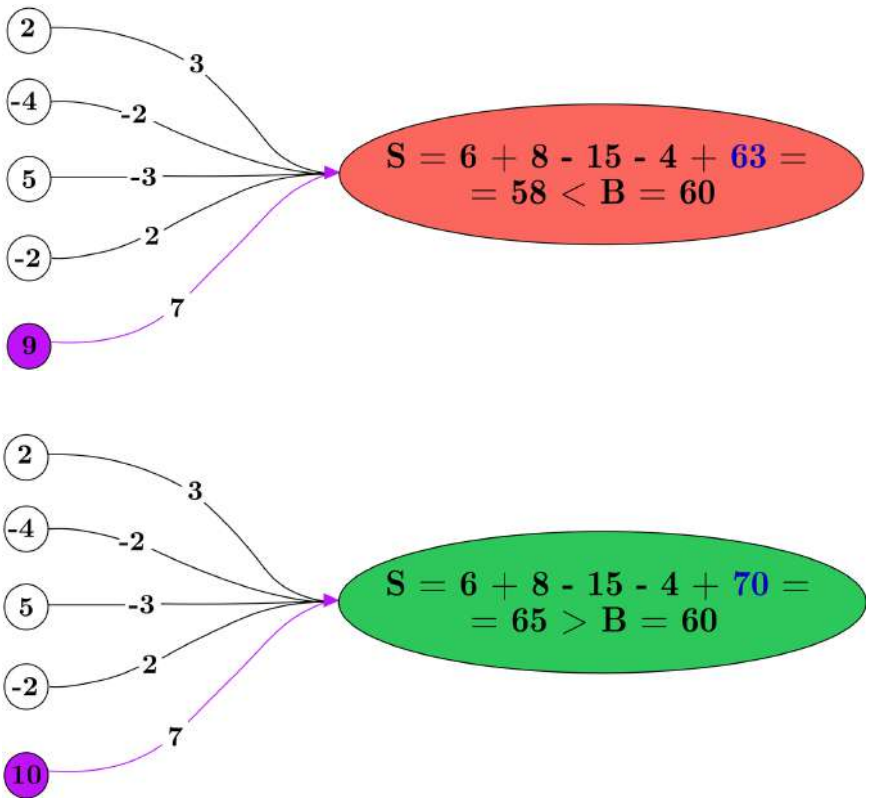
Формат выходных данных

Выведите одно целое число — минимальное значение a_k, при котором нейрон активируется.

Примеры

стандартный ввод	стандартный вывод
5 60 7 2 3 −4 −2 5 −3 −2 2	10

Замечание



В вершинах заданы a_i, на рёбрах — w_i. $B = 60$. Требуется минимизировать значение в выделенной вершине. Зелёный нейрон активирован, так как $65 = S > B = 60$. Красный нейрон не активирован, так как $58 = S < B = 60$.

Решение

От нас требовалось найти такое минимальное значение a_k, что $a_1w_1 + a_2w_2 + \cdots + a_kw_k \geq B$. Перенесём известную часть в правую сторону: $a_kw_k \geq B - (a_1w_1 + a_2w_2 + \cdots + a_{k-1}w_{k-1})$. Отсюда, непосредственно, получаем формулу

$$a_k = \left\lceil \frac{B - (a_1w_1 + a_2w_2 + \cdots + a_{k-1}w_{k-1})}{w_k} \right\rceil.$$

На практике сначала уменьшаем B, вычитая вклад уже известных коэффициентов, а затем считаем минимальное целое a_k, удовлетворяющее неравенству. Для корректного вычисления потолка используется стандартный приём $(B + w_k - 1)/w_k$ с дополнительным сдвигом, чтобы избежать проблем с отрицательными значениями.

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

using ll = long long;

int main() {
    int k;
    ll B, w0;
    cin >> k >> B >> w0;

    for (int i = 0; i < k - 1; ++i) {
        ll a, w;
        cin >> a >> w;
        B -= a * w;
    }

    ll N = 1000000000LL;
    cout << (B + w0 - 1 + w0 * N) / w0 - N << '\n';
    return 0;
}
```

Стоит заметить, что для вычисления $\lceil a/b \rceil$ при целочисленном делении в коде можно использовать выражение

```
    ans = (a + b - 1) / b;
```

Максимальный балл за задание — 100

Задание 6. Матрёшки

На столе стоит n матрёшек. Каждая матрёшка имеет цвет a_i и размер b_i. Матрёшку i можно вложить в матрёшку j, если $b_i < b_j$.
Для каждой матрёшки найдите количество различных цветов матрёшек, которые можно вложить в данную.

Формат входных данных

Первая строка содержит целое число n ($1 \leqslant n \leqslant 10^5$).
Вторая строка содержит n целых чисел a_i ($1 \leqslant a_i \leqslant 10^9$) — цвета матрёшек.
Третья строка содержит n целых чисел b_i ($1 \leqslant b_i \leqslant 10^9$) — размеры матрёшек.

Формат выходных данных

Выведите n целых чисел — ответы для каждой матрёшки. Порядок должен соответствовать их порядку во входных данных.

Примеры

стандартный ввод	стандартный вывод
4 10 4 8 4 4 2 5 8	1 0 2 3

Решение

Для каждой матрёшки требуется посчитать число меньших матрёшек, которые можно в неё вложить. Давайте отсортируем матрёшки по размеру b. В каждый момент времени в `set` будем поддерживать все цвета, которые встречались у матрёшек с b_i, строго меньшим, чем текущий b.

Удобно обрабатывать матрёшки группами одинакового размера. Пусть мы идём по отсортированному массиву. Тогда при переходе на новый размер (то есть когда $b_i > b_{i-1}$) нужно добавить в `set` цвета всех матрёшек предыдущего размера. Для этого будем накапливать цвета текущей группы в буфере, а когда размер меняется — переносить буфер в `set` и очищать его. Суммарно все такие переносы проходят каждый элемент ровно один раз, то есть занимают $O(n)$ времени.

Осталось заметить, что для каждой матрёшки ответом является размер `set` в момент её рассмотрения, потому что `set` содержит ровно цвета всех матрёшек строго меньшего размера.

```cpp
#include <bits/stdc++.h>
using namespace std;

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

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

    vector<pair<int,int>> p(n);
    for (int i = 0; i < n; ++i) p[i] = {b[i], a[i]};
    sort(p.begin(), p.end());

    vector<int> buf;
    map<int,int> col, ans;

    for (int i = 0; i < n; ++i) {
        if (i > 0 && p[i - 1].first != p[i].first) {
            for (int x : buf) col[x] = 1;
            buf.clear();
        }
        ans[p[i].first] = (int)col.size();
        buf.push_back(p[i].second);
    }

    for (int i = 0; i < n; ++i) {
        cout << ans[b[i]] << ' ';
    }
    return 0;
}
```

Максимальный балл за задание — 100

Максимальное количество баллов за олимпиаду — 600

Задание 1. Задача по геометрии

Саша нарисовал выпуклый многоугольник с помощью компьютерной программы и отправил модели ИИ запрос вычислить сумму его углов. Он получил неверный ответ 500°. Оказалось, что один из углов многоугольника модель учла дважды. Чему равен этот угол? Ответ выразите в градусах. Напомним, что в выпуклом многоугольнике все углы меньше 180°.

Ответ: 140

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. Если в Сашином многоугольнике хотя бы пять вершин, то сумма его углов хотя бы $180° \cdot 3 = 540°$, этот случай невозможен. Если это треугольник, сумма углов равна 180°, и учтенный дважды угол должен быть больше 180°, что невозможно. Значит, у Саши четырехугольник, его сумма углов равна 360°, и дважды был посчитан угол величиной $500° - 360° = 140°$.

Задание 2. Математика в чат-боте

Дима выбрал два натуральных числа a и b и затем отправил запрос модели ИИ найти значение выражения $a + b \cdot 2$ (он записал это выражение на бумажке, сфотографировал и загрузил полученную фотографию). Из-за неаккуратного почерка модель распознала записанное выражение как $a + b^2$ и в результате ответ модели оказался на 80 больше правильного. Найдите последнюю цифру числа $a \cdot b$.

Ответ: 0

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. Исходя из условия, мы получаем, что $b^2 - 2b = 80$, то есть $(b - 10)(b + 8) = 0$. Следовательно, $b = 10$, поскольку число b — натуральное, а тогда ab оканчивается нулём.

Задание 3. Градиентный спуск

На прямой изучают работу очень простого «искусственного интеллекта», который зависит всего от одного числа — параметра w. Для каждого целого w от 1 до 10 заранее посчитана ошибка $E(w)$ этого ИИ на обучающих примерах:

w	1	2	3	4	5	6	7	8	9	10
$E(w)$	9	5	2	4	6	7	3	1	2	4

ИИ обучают с помощью следующего алгоритма изменения параметра w (аналог градиентного спуска).

1. Сначала выбирают начальное целое значение параметра w от 1 до 10.

2. Рассматривают «соседей» текущего значения w:
   - слева — число $w - 1$ (если $w > 1$);
   - справа — число $w + 1$ (если $w < 10$).

3. Если среди существующих соседей есть такие, у которых ошибка строго меньше: $E(w_{\text{сосед}}) < E(w)$, то переходят к тому соседу, у которого ошибка наименьшая (среди соседей).

4. Затем снова выполняют шаг 2 и так далее, пока не окажется, что у всех существующих соседей ошибка не меньше текущей. В этот момент алгоритм останавливается; говорят, что он застрял в локальном минимуме.

Из таблицы видно, что наименьшее значение ошибки достигается при $w = 8$; это глобальный минимум.

В реальных задачах часто запускают обучение много раз из разных начальных точек, чтобы увеличить шанс попасть в глобальный минимум. Будем считать, что:

- в каждом запуске начальное значение w выбирается случайно и равновероятно из чисел 1, 2, ..., 10;
- разные запуски независимы;
- запуск считается успешным, если алгоритм в итоге остановился в глобальном минимуме при $w = 8$.

При каком наименьшем количестве запусков вероятность того, что хотя бы один запуск окажется успешным, будет не меньше 95%?

Ответ: 5

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. По таблице видно, что слева все траектории спуска (из $w = 1, 2, 3, 4, 5$) приходят в локальный минимум при $w = 3$, а справа все траектории (из $w = 6, 7, 8, 9, 10$) приходят в глобальный минимум при $w = 8$.

Значит, при случайном равновероятном выборе начального w из $\{1, \dots, 10\}$ вероятность успешного запуска (то есть попадания в $w = 8$) равна

$$p = \frac{5}{10} = \frac{1}{2}.$$

При N независимых запусках вероятность того, что *хотя бы один* из них успешен, равна

$$1 - (1 - p)^N = 1 - \left(\frac{1}{2}\right)^N.$$

Нужно, чтобы

$$1 - \left(\frac{1}{2}\right)^N \geq 0{,}95 \iff \left(\frac{1}{2}\right)^N \leq 0{,}05.$$

Подбором:

$$\left(\frac{1}{2}\right)^4 = \frac{1}{16} \approx 0{,}0625 > 0{,}05, \quad \left(\frac{1}{2}\right)^5 = \frac{1}{32} \approx 0{,}03125 < 0{,}05.$$

Минимальное N, при котором условие выполняется, равно 5.

Задание 4. Лидер продаж в категории

Интернет-магазин агрегирует товары по категориям. Данные находятся в файле, который вы можете скачать в форматах XLSX, ODS или CSV.

Для каждого товара известны три значения: `category`, `score`, `label`.

- `category` — категория товара;
- `score` — оценка интереса, число из диапазона $[0, 1]$;
- `label` — факт покупки: 0 или 1.

В каждой категории на витрине показываются все товары, у которых значение `score` является максимальным среди всех товаров той же категории. Для каждой категории найдите её максимальную оценку:

$\texttt{max\_score}(g) = \max\{\texttt{score} : \texttt{category} = g\}$.

Выберите все строки, где $\texttt{score} = \texttt{max\_score}(g)$. Если в категории несколько товаров делят максимум, выбираются все. Сколько выбранных строк имеют $\texttt{label} = 1$?

Ответ: 49

Критерий оценивания: точное совпадение ответа — 100 баллов

Максимальный балл за задание — 100

Решение. В каждой категории g находим максимальную оценку интереса

$$\texttt{max\_score}(g) = \max\{\texttt{score} : \ \texttt{category} = g\},$$

и выбираем все строки, где $\texttt{score} = \texttt{max\_score}(g)$. Ответ — сумма значений `label` среди выбранных строк.

Код (pandas):

```
import pandas as pd

df = pd.read_csv("task9_group_top_select.csv")

max_score = df.groupby("category")["score"].transform("max")
sel = df["score"] == max_score

answer = df.loc[sel, "label"].sum()
print(answer)
```

Задание 5. ICPC

Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

В машинном обучении ансамбли работают лучше, когда в них есть разнообразие моделей: смешивают разные архитектуры и источники признаков, чтобы усилить общий результат. Однородные ансамбли часто переобучаются и хуже обобщают, а разнородные — устойчивее и сильнее.

Рассмотрим команды как ансамбль людей. Скоро начнётся новый сезон ICPC, а значит, известному тренеру Михаилу необходимо собрать команду, которая его выиграет! Ранее такое уже случалось, получится и в этом году. Секрет прост — нужно, чтобы в команде были и математики, и программисты. Если в команду войдут только программисты

или только математики, результат хорошим не будет. Прямо сейчас у Михаила для распределения есть n математиков и m программистов.

Сколькими способами можно собрать ровно одну команду из трёх человек на ICPC?

Формат входных данных

Первая строка содержит два целых числа n и m ($1 \leqslant n,\ m \leqslant 10^5$) — количества математиков и информатиков соответственно.

Формат выходных данных

Выведите одно целое число — количество способов собрать одну успешную команду из трёх человек для участия в ICPC.

Примеры

стандартный ввод	стандартный вывод
2 3	9

Критерий оценивания: точное совпадение ответа — 100 баллов
Максимальный балл за задание — 100

Решение

Есть два непересекающихся способа собрать команду: два математика + программист и два программиста + математик.

Посчитаем общую формулу: сколько есть способов собрать команду, если двух людей нужно выбрать из группы из a человек, а одного человека нужно выбрать из группы из b человек. Двух людей можно выбрать $a(a-1)/2$ способами, одного человека — b способами. Для каждого выбора пары подходит любой выбор одиночного участника, значит всего способов $a(a-1)/2 \cdot b$.

Применяя это к нашей задаче, получаем итоговый ответ: $n(n-1)/2 \cdot m + m(m-1)/2 \cdot n$.

```cpp
#include <iostream>
#include <cstdint>
using namespace std;

int main() {
    int64_t n, m;
    cin >> n >> m;
    cout << m * n * (n - 1) / 2 + n * m * (m - 1) / 2;
    return 0;
}
```

Задание 6. Коллектив

Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

В эпоху больших данных обучение становится распределённым, а ресурсы — на вес золота. В случае распределённого обучения на разных устройствах какие-то операции возможно производить эффективно, только если необходимые модули находятся в одном кластере. В таких случаях очень важно уметь распределять вычисления для достижения наилучшего результата с данными ресурсами.

Эту идею можно продемонстрировать на взаимодействии людей. В исследовательском центре одной небезызвестной компании работает n сотрудников. Про каждого сотрудника известно, что он является выпускником вуза с номером b_i, а его личный вклад в решение сложных задач равен a_i.

Сотрудникам необходимо решить очень сложную задачу. Для большей эффективности они будут работать над задачей парами. Рассмотрим все пары различных сотрудников (i,j), где $1 \leqslant i < j \leqslant n$:

- пара сотрудников (i,j) считается допустимой, если они выпускники одного и того же вуза, то есть $b_i = b_j$;
- вклад допустимой пары (i,j) в решение равен $a_i + a_j$;
- если $b_i \neq b_j$, то такая пара не является допустимой и не даёт вклада в решение.

Каждый работник может входить в несколько допустимых пар с разными коллегами. При этом каждая конкретная пара сотрудников (i,j) учитывается не более одного раза. Требуется посчитать суммарный вклад всех допустимых пар сотрудников.

Формат входных данных

Первая строка входных данных содержит одно целое число n ($2 \leqslant n \leqslant 10^5$) — количество сотрудников.
Вторая строка содержит n целых чисел a_1, a_2, ..., a_n, где a_i ($1 \leqslant a_i \leqslant 10^6$) — вклад i-го сотрудника.
Третья строка содержит n целых чисел b_1, b_2, ..., b_n, где b_i ($1 \leqslant b_i \leqslant 10^6$) — номер вуза, который окончил i-й сотрудник.

Формат выходных данных

Выведите одно целое число — суммарный вклад всех допустимых пар сотрудников.

Замечание

В первом примере есть только один выпускник вуза 1, поэтому он не сможет образовать ни одной пары. При этом есть 3 выпускника вуза 2 и они образуют пары, которые дадут вклады $4+8$, $4+8$, $4+4$ в решение. Суммарный вклад будет $12 + 12 + 8 = 32$.

Примеры

стандартный ввод	стандартный вывод
4 4 8 1 4 2 2 1 2	32

Критерий оценивания: точное совпадение ответа — 100 баллов
Максимальный балл за задание — 100

Решение

В итоговый результат вклад дают только пары с одинаковым b. Давайте для каждого значения b посчитаем $\texttt{cnt}[b]$ — количество людей с таким b.

Заметим, что каждый человек i образует допустимую пару ровно с $\texttt{cnt}[b_i] - 1$ людьми. В каждую такую пару он даёт вклад a_i (вторую половину вклада даёт другой человек), поэтому суммарный вклад человека i в ответ равен $(\texttt{cnt}[b_i] - 1) \cdot a_i$. Ответ получается суммированием этой величины по всем людям.

```cpp
#include <bits/stdc++.h>
using namespace std;

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

    int n;
    cin >> n;
    vector<long long> a(n), b(n);
    for (int i = 0; i < n; ++i) cin >> a[i];
    for (int i = 0; i < n; ++i) cin >> b[i];

    const int MAXB = 1000000;
    vector<long long> cnt(MAXB + 1, 0);
    for (int i = 0; i < n; ++i) ++cnt[b[i]];

    long long ans = 0;
    for (int i = 0; i < n; ++i) {
        ans += a[i] * (cnt[b[i]] - 1);
    }

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