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.

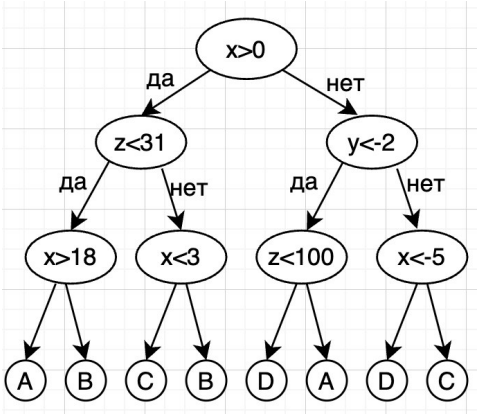
ПРИКОЛ — ПИКОЛ — ПИКО — ПИК — ПОИК — ПОИСК

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.

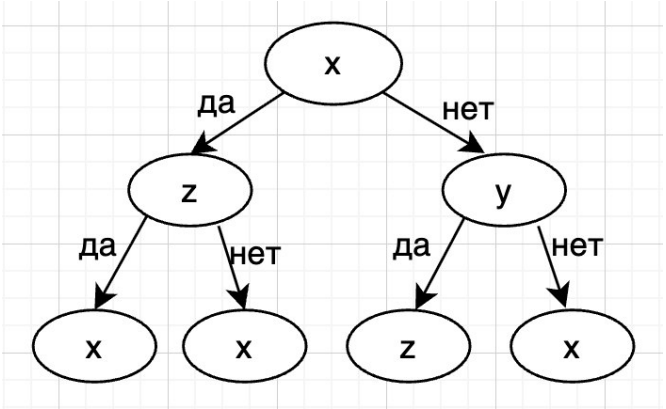
Задание 3. Красим решающее дерево

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 ($-1000 \leqslant a_i, w_i \leqslant 1000$).

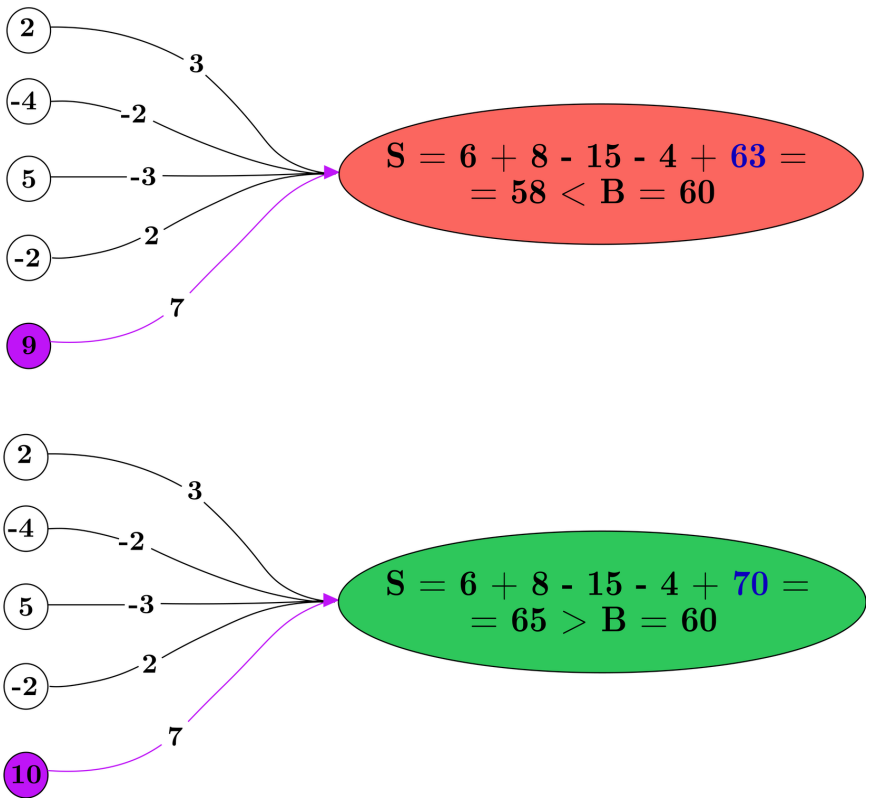
Формат выходных данных

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