Задача A. Котики на картинках

Эта задача с открытыми тестами. Ее решением является набор ответов, а не программа на языке программирования. Тесты указаны в самом условии, от вас требуется лишь ввести ответы на них в тестирующую систему.

В общий набор картинок с животными каждый день попадают изображения из двух источников: S1 и S2. Источники равновероятны.

В каждом источнике доля картинок с котом разная:

- из S_1 кот на картинке встречается в 70% случаев;
- из S_2 кот встречается в 30% случаев.

Перед тем как раздать картинки ученикам, детектор ставит техническую метку «CAT», если считает, что на изображении кот. Качество детектора по источникам известно:

Для картинок с котом:

- из S_1 он ставит «CAT» в 85% случаев;
- из S_2 – в 75% случаев;

Для картинок без кота:

- из S_1 он по ошибке ставит «CAT» в 10% случаев;
- из S_2 – в 5% случаев.

Нужно найти:

1. вероятность того, что на картинке действительно кот, если детектор уже поставил метку «CAT»;
2. вероятность того, что картинка с меткой «CAT» пришла из источника S_1.

Пояснение записи. Обозначение P(A | B) читается так: вероятность события «A», если известно, что произошло событие «B».

Например, P(кот | «CAT») – вероятность, что на картинке кот, при условии что детектор поставил «CAT».

Формат выходных данных

Два числа через пробел, каждое округлите до 3 знаков после точки (разделитель – точка): сначала P(кот | «CAT»), затем P(S_1 | «CAT»).

Пример возможного ответа

0.194 0.783

Правильный ответ (для экспертизы)

0.927 0.706

Задача B. Заметки. Нормализация признаков

Эта задача с открытыми тестами. Ее решением является набор ответов, а не программа на языке программирования. Тесты указаны в самом условии, от вас требуется лишь ввести ответы на них в тестирующую систему.

Система раскладывает короткие заметки по двум темам: «A» и «B». У заметки два признака: x_1 – «уровень теории», x_2 – «уровень практики».

Перед сравнением с темами система делает три шага.

Шаг 1. Нормализация признаков.

- $z_1 = (x_1 - m_1) / s_1$,
- $z_2 = (x_2 - m_2) / s_2$,
- где $m_1 = 1$, $s_1 = 2$; $m_2 = 0$, $s_2 = 1$.

Шаг 2. Преобразование.

- $y_1 = z_1 + 2 \cdot z_2 + 1$
- $y_2 = -z_1 + z_2$

Шаг 3. Усечение отрицательных вкладов (ReLU) и сравнение по косинусу:

- $r_1 = \max(0, y_1)$, $r_2 = \max(0, y_2)$

Ориентиры тем (эталонные векторы):

- $c_A = (4, 0)$, $c_B = (3, 4)$.

Косинусное сходство двух векторов $u = (u_1,u_2)$ и $v = (v_1,v_2)$:

- $cos(u, v) = (u_1 \cdot v_1 + u_2 \cdot v_2)/(\sqrt{u_1^2 + u_2^2} \cdot \sqrt{v_1^2 + v_2^2})$

Для заметки известно: $x_1 = 2$, $x_2 = 1$.

Вычислите $\cos(r, c_A)$ и $\cos(r, c_B)$, где $r = (r_1, r_2)$.

Формат выходных данных

Два числа через пробел – сначала $\cos(r, c_A)$, затем $\cos(r, c_B)$, каждое округлить до 3 знаков после точки (разделитель – точка).

Пример возможного ответа

0.462 0.931

Правильный ответ (для экспертизы)

0.990 0.707

Задача C. Метод k-средних

Эта задача с открытыми тестами. Ее решением является набор ответов, а не программа на языке программирования. Тесты указаны в самом условии, от вас требуется лишь ввести ответы на них в тестирующую систему.

Платформа группирует похожие задания по двум числовым признакам:
x – «тема/теоретичность», y – «практичность/сложность».

Команда решила, что вторая координата важнее в 2 раза, поэтому сравнение «кто ближе к центру» идёт по взвешенному квадрату расстояния:

$$d^2((x, y), (a, b)) = (x - a)^2 + 2 \cdot (y - b)^2$$

Алгоритм k-means (k=3) на одном шаге делает:

1. Назначение. Отнести каждую точку к ближайшему центру по формуле выше.
   Если расстояния ровно равны – точка идёт к центру с меньшей x-координатой, а при равенстве x – к центру с меньшей y.
2. Пересчёт центров. Для каждого кластера новый центр – это среднее его точек по x и по y. (Если кластер пуст, его центр не меняется.)

Начальные центры:
M_1=(1, 1), M_2=(6, 1), M_3=(8, 8)
Точки:
P_1=(0, 0), P_2=(2, 0), P_3=(0, 2), P_4=(5, 1), P_5=(6, 0), P_6=(8, 8), P_7=(9, 7), P_8=(7, 9)
Выполните один полный шаг (назначение → пересчёт) с указанной метрикой.
Что вывести (в одной строке через пробел):

1. новые координаты центров M_1, M_2, M_3 в этом порядке, каждую координату округлить до 2 знаков после точки;
2. затем размеры кластеров (число точек) n_1 n_2 n_3;
3. затем взвешенную сумму квадратов ошибок (SSE) после шага, округлённую до 2 знаков после точки.

Формат выходных данных

x_M_1 y_M_1 x_M_2 y_M_2 x_M_3 y_M_3 n_1 n_2 n_3 SSE

Пример возможного ответа

0.12 3.45 6.78 9.01 2.34 5.67 3 2 3 15.50

Правильный ответ (для экспертизы)

0.67 0.67 5.50 0.50 8.00 8.00 3 2 3 15.50

Задача D. Качество детектора дорожного знака

Эта задача с открытыми тестами. Ее решением является набор ответов, а не программа на языке программирования. Тесты указаны в самом условии, от вас требуется лишь ввести ответы на них в тестирующую систему.

Городская система распознает на фото знак «Пешеходный переход». Каждый кадр детектор либо помечает как «знак есть», либо как «знака нет».
После проверки экспертами собрана статистика за день.

Что означают показатели:

- TP (True Positive, верно найдено) – детектор сказал «знак есть», и знак действительно был.
  Пример: на фото виден знак, и модель его пометила – это TP.
- FP (False Positive, ложная тревога) – детектор сказал «знак есть», но знака не было.
  Пример: модель перепутала рекламу с дорожным знаком – это FP.
- FN (False Negative, пропуск) – детектор сказал «знака нет», но знак был.
  Пример: знак закрыла ветка, модель его не заметила – это FN.
- TN (True Negative, верно не сработал) – детектор сказал «знака нет», и знака действительно не было.

Сводка за день:

- TP = 36 (верно нашли знак)
- FP = 9 (ложно сработали)
- FN = 12 (пропустили знак)
- TN = 43 (верно не сработали)

Что нужно посчитать:

1. Точность (precision) – доля верных срабатываний среди всех срабатываний:
   precision = TP / (TP + FP).
2. Полнота (recall) – доля найденных знаков среди всех кадров, где знак был:
   recall = TP / (TP + FN).
3. F1-мера – объединяет точность и полноту:
   F1 = 2 * precision * recall / (precision + recall).

По данным выше вычислите F1-меру.

Формат выходных данных

Одно число с двумя знаками после точки, разделитель – точка.

Пример возможного ответа

0.12

Правильный ответ (для экспертизы)

0.77

Задача E. Добавление плашек

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Максимальное время работы на одном тесте:	*0.5 секунд*
Максимальный объем используемой памяти:	*64 мегабайта*

Вася решил обучить большую языковую модель, используя компьютер с «**C**» гигабайтами оперативной памяти. Однако оказалось, что большая языковая модель называется так не зря – ведь для её обучения требуется целых «**B**» гигабайт оперативной памяти.

Вася хочет оперативно разобраться с проблемой и купить себе ещё несколько плашек оперативной памяти, каждая из которых имеет объём «**A**» гигабайт. Вася хочет потратить как можно меньше денег на обучение модели.

Подскажите ему, какое минимальное число плашек потребуется купить, чтобы успешно обучить модель. Гарантируется, что это возможно.

Формат входных данных

В первой строке задано число $0 \leqslant A \leqslant 10^9$.

Во второй строке задано число $0 \leqslant B \leqslant 10^9$.

В третьей строке задано число $0 \leqslant C \leqslant 10^9$.

Формат выходных данных

Выведите единственное число– ответ на задачу

Пример

Примеры входных данных	Примеры выходных данных
9 16 6	2

Задача F. Градиентный спуск

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Максимальное время работы на одном тесте:	*1 секунд*
Максимальный объем используемой памяти:	*256 мегабайта*

Градиентный спуск – способ по шагам находить минимальное значение функции (например, наименьшую ошибку модели). Как он работает:

1. Вычисляем **градиент** – направление, в котором функция растёт быстрее всего;
2. Делаем **маленький шаг в противоположную сторону** (там функция убывает);
3. Повторяем шаги, пока изменения почти не исчезнут.

Размер шага называется **скоростью обучения**: слишком большой – будем «скакать» мимо минимума, слишком маленький – идти очень медленно. В машинном обучении градиентным спуском подбирают параметры модели, чтобы ошибка на данных стала как можно меньше.

Серёжа только учится пользоваться градиентным спуском, поэтому решил, в качестве тренировки, найти локальные минимумы его любимой функции. График любимой функции Серёжи это ломаная с вершинами в точках (1, a_1), (2, a_2), ..., (N, a_N). Локальным минимумом назовем такую точку i($1 < i < N$), что $a_i < a_i-1$ и $a_i < a_i+1$.

Помогите Серёже проверить себя, для этого найдите номера всех точек, являющихся локальными минимумами.

Формат входных данных

В первой строке задано число $1 \leqslant N \leqslant 100000$ – количество вершин любимой функции Серёжи. В последующих N строках заданы числа $a_1,a_2,...a_N$. Для всех i выполнено $1 \leqslant a_i \leqslant 10^6$.

Формат выходных данных

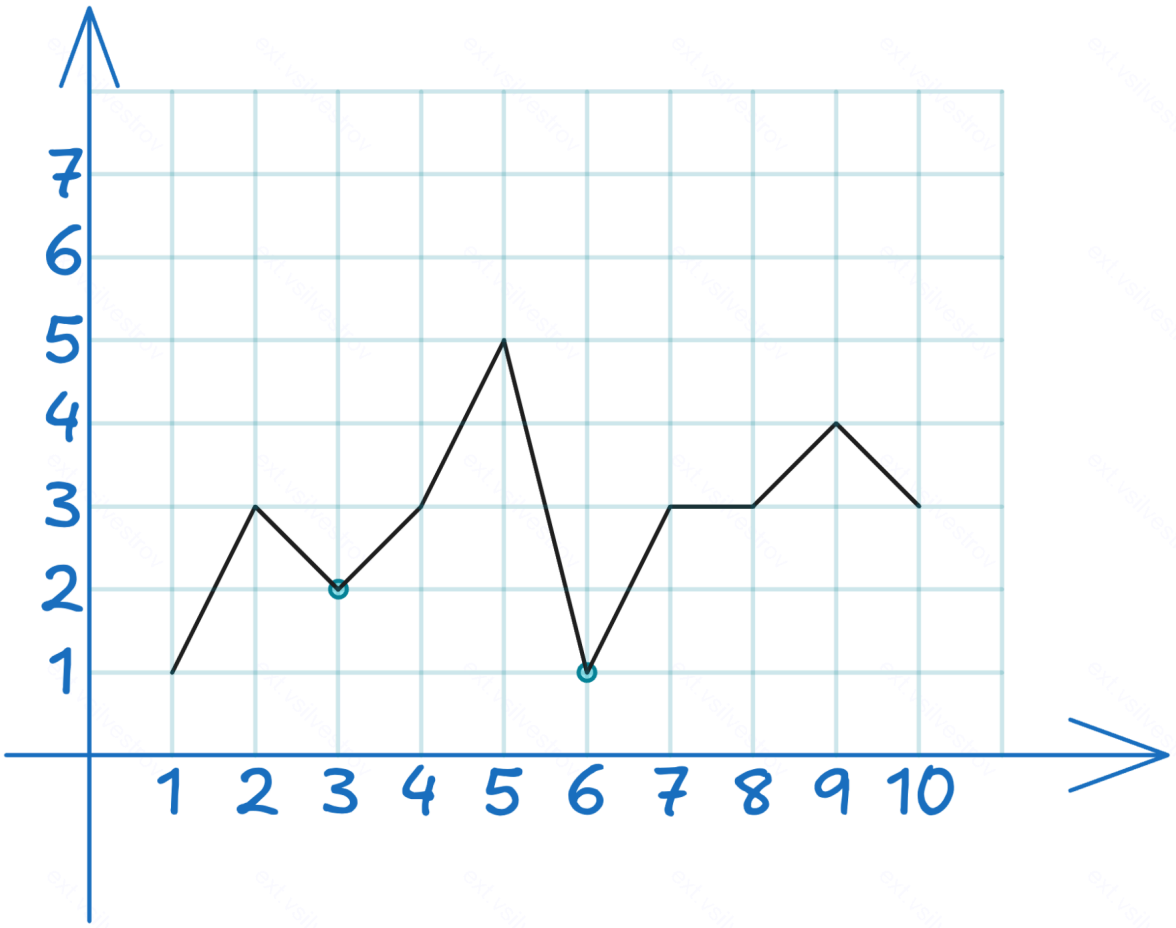
Выведите по возрастанию номера всех точек, являющихся локальными минимумами. Каждое число выводите в отдельной строке.

Пример

Примеры входных данных	Примеры выходных данных

10 1 3 2 3 5 1 3 3 4 3	3 6

Замечание



Любимая функция Серёжи из Примера 1 и её локальные минимумы.