Математический тур. Разбор.

Никитин Тимофей • Чернов Владимир

15.01.2026

Математический тур

A. Игра Незнайки

Незнайка участвует в магической игре: он n раз подкидывает монетку с вероятностью орла p, изначально $p = 0.5$. Если выпадает орёл, в его кармане появляется тугрик и p уменьшается на 0.1. В противном случае Незнайка ничего не выигрывает и p увеличивается на 0.1. Сколько в среднем тугриков выиграет Незнайка? $n = 2025$.

Решение

Рассмотрим математическое ожидание выпадения орла на шаге $k + 1$:

$$\begin{aligned}\mathbb{E}[p_{k+1} \mid p_k] &= p_k \cdot (p_k - 0.1) + (1 - p_k)(p_k + 0.1) \\ &= p_k^2 - 0.1p_k + p_k + 0.1 - p_k^2 - 0.1p_k \\ &= 0.1 + 0.8p_k\end{aligned}$$

Теперь подставим начальную вероятность $p_1 = 0.5$ и получим $\mathbb{E}[\text{тугрика}]$ на втором шаге $= 0.5$, то есть вероятность выпадения орла равна $p_2 = 0.1 + \frac{0.8}{2} = 0.5$, по индукции получаем, что на каждом шаге вероятность выпадения орла (или получения тугрика) равна 0.5. Таким образом $\mathbb{E}[\text{число тугриков}] = \frac{n}{2}$

Ответ: 1012.5

Примечание: Обратите внимание, что в данной задаче просят вывести ответ с точностью до 9 знаков, но ответ 1012.5 тоже будет засчитан.

B. Максимум Энтропии

Во многих задачах по искусственному интеллекту важно оценивать, насколько модель уверена в своём ответе. Для этого смотрят на распределение вероятностей по вариантам: если один вариант почти наверняка верен, распределение сосредоточено, если же модель сильно сомневается, вероятности разных исходов ближе друг к другу. Такие числовые меры размытости распределения часто называют мерами энтропии и используют при обучении и настройке моделей.

Пусть задано дискретное распределение вероятностей $\{p_1, ..., p_n\}$, $\sum_{i=1}^{n} p_i = 1$. Известно, что для всех i выполняется $p_i \geq 0.01$, и существует по крайней мере один исход j с вероятностью $p_j = 0.2$.

Найдите максимальное возможное значение величины

$$H(p) = 1 - \sum_{i=1}^{n} p_i^2.$$

Решение

Заметим небольшой очевидный факт.

$$(p_i + p_j)^2 > p_i^2 + p_j^2$$

То есть когда нужно минимизировать сумму $\sum_{i=1}^{n} p_i^2$, нужно делать p_i меньше. Возьмем $p_1 = 0.2$ (как обязательное) и сделаем остальные вероятности насколько возможно маленькими, то есть $\sum_{i=2}^{n} p_i = 0.8$, значит $n \leq 80$ так как $p_i \geq 0.01$, значит мы нашли оптимум. По очевидному факту мы поняли, что если можно разбить число на 2 меньших, это нужно сделать.

Ответ: $1 - \left(\sum_{i=1}^{80} 0.01^2 + 0.04\right) = 1 - 0.048 = \mathbf{0.952}$

C. Индексы согласованности

Инженеры проводят серию измерений и фиксируют результаты в таблице. Для каждого измерения записаны четыре целых значения:

$$A, B, C, D,$$

где

- A и C — показания двух различных счётчиков;
- B и D — соответствующие шкалы (нормировочные параметры) этих счётчиков.

Каждый из параметров A, B, C, D принимает целые значения в диапазоне от 2 до 2000 включительно.

На основании этих данных инженеры вычисляли два контрольных индекса, используя значения каждой строки таблицы.

Индекс согласованности:

$$K = \frac{A+B}{B} + \frac{C+D}{D} - \frac{(A-B)\cdot D + (C-D)\cdot B}{B \cdot D}.$$

Индекс обратной согласованности:

$$E = \frac{A+B}{C} + \frac{B+C}{A} + \frac{C+A}{B}.$$

Требуется посчитать, в скольких строках таблицы значения и K, и E являются целыми числами.

Файлы: **excel** **csv** (Сразу пойдет скачивание)

Решение

Очевидно, что можно открыть excel и забить там эти строки, или сделать это с помощью python.

Предложим сначала немного подумать: давайте сначала приведем K к человеческому виду:

$$K = \frac{A+B}{B} + \frac{C+D}{B} - \frac{(A-B)\cdot D + (C-D)\cdot B}{B\cdot D}$$
$$= \frac{(A+B)D}{B\cdot D} + \frac{(C+D)B}{B\cdot D} - \frac{(A-B)\cdot D + (C-D)\cdot B}{B\cdot D}$$
$$= \frac{2BD+2BD}{BD} = 4$$

То есть K всегда целое число. Проверить на целость число E не составит проблем.

Ответ: 4613

D. Гипер-муравей

Рассмотрим 4-мерный гиперкуб Q_4. Его вершины — все двоичные строки длины 4 (0000, 0001, ..., 1111); две вершины соединены ребром, если они отличаются ровно в одном разряде.

Муравей стартует в вершине 0000. На каждом шаге он равновероятно (с вероятностью $\frac{1}{4}$ для каждого варианта) выбирает одно из рёбер, выходящих из его вершины, и переходит по нему в соседнюю вершину. Процесс продолжается, пока муравей впервые не попадёт в вершину, противоположную стартовой, то есть в 1111.

Требуется найти математическое ожидание числа шагов до первого попадания в 1111.

Решение

Введем ожидаемое количество ходов до вершины с t единицами как p_t, тогда $p_4 = 0$. Любым шагом муравей меняет положение единицы в числе на ноль или нуля на единицу. Будем рассматривать задачу относительно количества единиц в числе. Тогда муравей может пойти с вероятностью $\frac{4-t}{4}$ в вершину с большим количеством единиц и $\frac{t}{4}$ с меньшим, то есть:

$$p_t = 1 + \frac{4-t}{4}p_{t+1} + \frac{t}{4}p_{t-1}$$

Так как $p_4 = 0$,

$$p_3 = 1 + \frac{3}{4}p_2 + \frac{1}{4}p_4$$
$$p_2 = 1 + \frac{2}{4}p_3 + \frac{2}{4}p_1$$
$$p_1 = 1 + \frac{3}{4}p_2 + \frac{1}{4}p_0$$
$$p_0 = 1 + p_1$$

Это система линейных уравнений. Решения этой системы: $p_3 = 15$, $p_2 = 18\frac{2}{3}$, $p_1 = 20\frac{1}{3}$,$p_0 = 21\frac{1}{3}$. Количество шагов от 0000 до 1111 это и есть p_0

Ответ: 21.333333

E. Минимизация сложной функции

Найдите единственную точку минимума функции

$$F(x, y, z) = y^4 + 2y^2 + z^4 + 4z \cdot \cos(x) + 4yz \cdot \sin(x)$$

если $x \in [0; 2\pi), y \geq 0, z \geq 0$.

В качестве ответа выведите сумму $x + y + z$ в точке минимума.

Решение

Зафиксируем x и будем пытаться минимизировать по y, z:

$$\frac{\partial F}{\partial x} = 4z(y\cos(x) - \sin(x)) = 0$$

$$\frac{\partial F}{\partial y} = 4y^3 + 4y + 4z \cdot \sin(x) = 0$$

$$\frac{\partial F}{\partial z} = 4z^3 + 4\cos(x) + 4y \cdot \sin(x) = 0$$

Получаем систему:

$$\begin{cases} z(y\cos(x) - \sin(x)) = 0 \\ y^3 + y + z \cdot \sin(x) = 0 \\ z^3 + \cos(x) + y \cdot \sin(x) = 0 \end{cases}$$

Значит $z = 0$ или $y\cos(x) - \sin(x) = 0$.

1. В обеих точках при $y = 0, z = 0$ у нас получается значение функции **0**.
2. Понимаем, что $y = \tan(x)$, подставим $y\cos(x) = \sin(x)$ в третье выражение, получим что $\cos(x) = -\frac{z^3}{1+y^2}$. Тогда угол находится в III четверти, так как $y > 0$ и $\cos(x) < 0$. Тогда подставим во второе выражение $y\cos(x) = \sin(x)$. Получим $y((y^2 + 1) + z\cos(x)) = 0$

Отсюда есть два варианта:

1. $y = 0$: Получаем ответ $(\pi, 0, 1)$ и $F = -3$
2. $y^2 + 1 + z\cos(x) = 0$

Во втором варианте получаем $\cos(x) = -\frac{y^2+1}{z}$. Тогда справедливо равенство:

$$-\frac{y^2 + 1}{z} = -\frac{z^3}{1 + y^2}$$

То есть $z^4 = (1+y^2)^2$. Из $\tan(x) = y$ получаем $\cos^2(x) = \frac{1}{z^2}$, но также мы знаем, что $\cos(x) = -\frac{y^2+1}{z} = -\frac{z^2}{z} = -z$. Отсюда $z^2 = \frac{1}{z^2}$, то есть $z = 1$. То есть $y = 0$. Это первый вариант.

Ответ: $\pi + 0 + 1 = \pi + 1 \approx$ **4.141593**

Алгоритмическое решение

```python
import numpy as np

def F(x, y, z):
    return y**4 + 2*y**2 + z**4 + 4*z*np.cos(x) + 4*y*z*np.sin(x)

xs = np.linspace(0, 2*np.pi, 12, endpoint=False)  # 12 точек по x
ys = np.linspace(0, 1, 101)                       # шаг 0.01
zs = np.linspace(0, 1, 101)

min_val = np.inf
min_point = None

for x in xs:
    cosx = np.cos(x)
    sinx = np.sin(x)
    for y in ys:
        for z in zs:
            val = y**4 + 2*y**2 + z**4 + 4*z*cosx + 4*y*z*sinx
            if val < min_val:
                min_val = val
                min_point = (x, y, z)

min_val, min_point
```

```
(-3.0, (3.141592653589793, 0.0, 1.0))
```

F. Гладиаторы

Тимофей и Максим живут во времена Рима, у каждого из них по 5 гладиаторов. Силы гладиаторов Тимофея равны $1, 4, 9, 16, 25$, а силы гладиаторов Максима равны $1, 2, 3, 4, 20$.

Турнир проходит так: Тимофей и Максим по очереди выбирают гладиаторов, которые будут сражаться друг против друга. Затем два гладиатора сражаются насмерть (ничьих не бывает).

Если гладиаторы имеют силы x и y соответственно, то вероятность того, что гладиатор силы x победит, равна

$$\frac{x}{x+y}.$$

При этом победивший гладиатор наследует силу своего соперника. То есть, если гладиатор силы x побеждает гладиатора силы y, то его новая сила становится равной $x+y$.

Тимофей всегда выбирает первым для каждого боя из числа оставшихся у него гладиаторов. Затем Максим (если у него остались гладиаторы) выбирает своего для боя с выбранным гладиатором Тимофея.

Победителем турнира считается тот, у кого в конце остаётся хотя бы один гладиатор.

Предположим, что Тимофей и Максим играют оптимально. Какова вероятность Максима выиграть турнир?

Решение

Десять гладиаторов это довольно много, давайте попытаемся сначала понять что-то для 2 и 3. Для двух гладиаторов очевидно, что вероятность победы Максима будет равна $\frac{y}{x+y}$. Что происходит для трех гладиаторов:

$((x),(a,b))$ или $((a,b),y)$. Тогда вероятность победы Тимофея в первом случае равна $\frac{x}{x+a} \cdot \frac{x+a}{x+a+b} = \frac{x}{x+a+b}$, то есть вероятность победы Максима будет равна $\frac{a+b}{a+b+x}$. Во втором случае вероятность победы Максима будет равна $\frac{y}{y+a} \cdot \frac{y+a}{y+a+b}$, то есть $\frac{y}{y+a+b}$. Или другими словами просто $\frac{S_2}{S_1+S_2}$, где S_1 – сумма сил гладиаторов Тимофея, а S_2 – сумма сил гладиаторов Максима.

Попробуем обобщить найденную закономерность, а точнее докажем, что вероятность победы Максима будет равна $\frac{S_2}{S_1+S_2}$. Пусть n – суммарное количество гладиаторов.

База индукции: $n=2$ и $n=3$ работает.

Пусть для $n=k$ работает, докажем для $n=k+1$. Пусть M_1 — множество гладиаторов Тимофея, кроме одного, а M_2 — множество гладиаторов Максима, кроме одного. Тогда можно представить состояние игры так:

$$((M_1,x),(y,M_2))$$

Не умаляя общности, так как нам неважно какой гладиатор с каким сражается, скажем, что первыми сражаются гладиаторы с силами x, y. Тогда из начального состояния игры можно перейти в:

$$((M_1,x+y),(M_2)) \text{ или } ((M_1),(x+y,M_2))$$

Посчитаем вероятность победы в каждом из случаев. Так как теперь стало k гладиаторов. Тогда по предположению индукции вероятность победы в первом случае Максима равна $\frac{S_2-y}{S_2+S_1}$, а во втором случае равна $\frac{S_2+x}{S_2+S_2}$, где S_1 и S_2 – суммы сил гладиаторов в изначальном положении игры.

Учтем, что новые положения не равновероятны, тогда вероятность победы Максима будет равна:

$$\frac{x}{x+y} \cdot \frac{S_2-y}{S_2+S_1} + \frac{y}{x+y} \cdot \frac{S_2+x}{S_2+S_1} =$$

$$\frac{S_2x - yx + yS_2 + xy}{(x+y)(S_2+S_1)} = \frac{S_2(x+y)}{(x+y)(S_2+S_1)} =$$

$$\frac{S_2}{S_2+S_1}$$

Что и требовалось доказать, тогда в исходной задаче вероятность победы Максима равна

$$\frac{1+2+3+4+20}{(1+4+9+16+25)+(1+2+3+4+20)}$$

Ответ: 0.352941

Математический тур. Задачи на программирование

G. RMSE

Ограничение времени	12 секунд
Ограничение памяти	64 Мб

В секретном хранилище лежат важные данные, представляющие из себя точки на плоскости, но проказник Алексей получил доступ на редактирование этих данных. Он уверен, что если удалить k точек, то никто ничего не заметит. Помогите Алексею максимально испортить эти данные, удалив ровно k точек, а именно максимизируйте RMSE линейной модели на оставшихся $N - k$ точках.

Формат ввода

Первая строка: два целых числа N, k $(1 \le N \le 10^4)$ — количество точек в хранилище и количество точек, которое удалит Алексей. Далее N строк: по два вещественных числа x_i, y_i — координаты точек.

Гарантируется, что $|x_i|, |y_i| \le 10^9$.

Формат вывода

Выведите индексы (индексация начинается с нуля) k точек, которые вы бы удалили на месте Алексея, пытаясь максимизировать RMSE на оставшихся точках.

Система оценивания

Максимальный возможный балл за задачу — 50.

Ваш балл пропорционален среднему по всем тестам $RMSE^5$, где RMSE — значение среднеквадратичной ошибки линейной модели на не удалённых данных.

Примечания

Линейная регрессия — это «лучшая прямая» $\hat{y} = ax + b$, которую выбирают так, чтобы средний квадрат вертикальных отклонений точек от прямой был минимален.

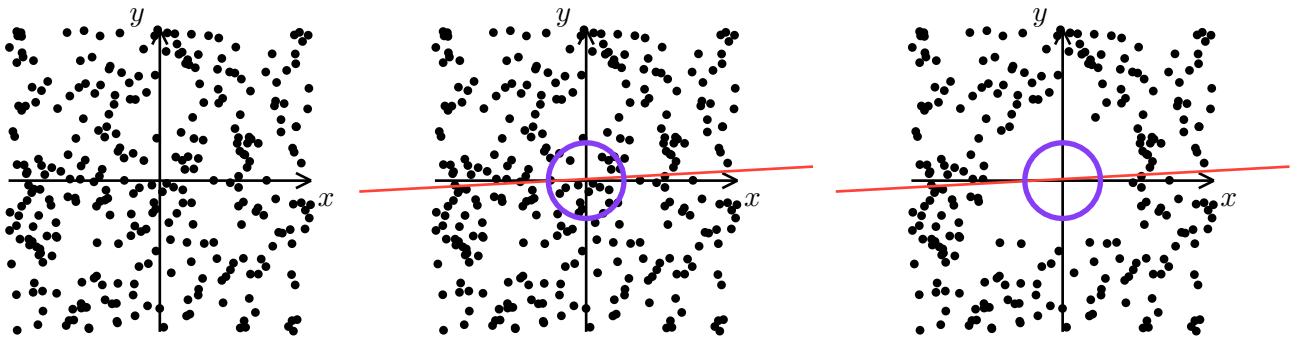
RMSE (среднеквадратичная ошибка) показывает, насколько сильно в среднем точки отклоняются от этой прямой:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - (ax_i + b))^2}$$

.

Решение

Представим какой-нибудь тест и попробуем его проанализировать, возможно из него мы сможем понять о задаче в целом. Вот пример на 300 точек. Пусть нужно удалить 50 точек. Попробуем здесь провести хорошую прямую: соответственно картинки а) и б)



а) Исходные данные б) С построенной прямой в) С удаленными точками

Несложно заметить, что прямая почти всегда будет проходить через фиолетовый круг. Но что это за фиолетовый круг? Это круг с центром в центре масс с каким-то радиусом.

Тогда довольно понятно, что можно удалить k ближайших к центру масс, и линейная регрессия не сильно поменяется, но RMSE только увеличится.

Такое решение набирает $\geq$ **43.9** баллов. Как действовать дальше?

Вспомним что такое центр масс – это центр кластера. Воспользуемся другим определением центра кластера – центр кластера – такая точка, что расстояние до нее от других точек минимально.

Если воспользоваться таким методом, вы получите $\geq$ **48.263** балла, что вполне достаточно. **Причем заметим, что не обязательно нужно удалять относительно центра кластера, а возможно относительно близких к нему точек**, поэтому нужно перебрать вокруг несколько точек и посчитать RMSE для каждой. Ответом будет удаление с наибольшим RMSE.

Небольшой интересный факт: если выбирать не ближайший к центру новый центр, а случайный (`random.randint(0, n - 1)`), то **может** получиться **49.343** баллов

Пример реализации:

```python
from math import sqrt
n,k = map(int,input().split())
points = [list(map(float,input().split())) for _ in range(n)]
def dist(A, B):
    return sqrt((A[0] - B[0]) ** 2 + (A[1] - B[1]) ** 2)

def rmse(inds): # супер простой подсчет rmse
    ppoints = [points[x] for x in inds]
    n = len(ppoints)
    xs = [p[0] for p in ppoints]
    ys = [p[1] for p in ppoints]

    mean_x = sum(xs) / n
    mean_y = sum(ys) / n

    # Оценки МНК для a и b
    num = sum((x - mean_x) * (y - mean_y) for x, y in ppoints)
    den = sum((x - mean_x) ** 2 for x in xs)
    # скажем, что den != 0
    a = num / den
    b = mean_y - a * mean_x
```

```
    # RMSE = sqrt( (1/n) * sum (y_i - (a*x_i + b))^2 )
    mse = sum((y - (a * x + b)) ** 2 for x, y in ppoints) / n
    rmse = sqrt(mse)

    return rmse

def find(points): # поиск центра кластера
    n = len(points)
    xs = [p[0] for p in points]
    ys = [p[1] for p in points]

    # предвычислим суммы и суммы квадратов
    sum_x = sum(xs)
    sum_y = sum(ys)
    sum_x2 = sum(x*x for x in xs)
    sum_y2 = sum(y*y for y in ys)

    best_ind = 0
    best_val = float('inf')

    for i in range(n):
        xi, yi = xs[i], ys[i]
        s = (
            n * (xi*xi + yi*yi)
            - 2 * xi * sum_x
            - 2 * yi * sum_y
            + sum_x2 + sum_y2
        )
        if s < best_val:
            best_val = s
            best_ind = i

    return best_ind

best = list(range(n))
best_rmse = 0

points_center = points[find(points)]
revert = sorted(list(range(n)), key = lambda x: dist(points[x], points_center))
points.sort(key = lambda x: dist(x, points_center))
for _ in range(min(1250, n)):
    if _ == 0: # Гарантируем себе ~43 балла
        center = (sum(x[0] for x in points)/n,sum(x[1] for x in points)/n)
    else:
        center = points[_ - 1]
    ans = list(range(n))
    ans.sort(key = lambda x: dist(points[x], center))
    RMSE = rmse(ans[k:])
    if RMSE > best_rmse:
        best_rmse = RMSE
        best = [revert[x] for x in ans[:k]]
print(*best)
```

H. Круги

Ограничение времени	10 секунд
Ограничение памяти	64 Мб

Дрон летит по кругу вокруг маяка, но его GPS «шумит». Вам дали набор снимков положения дрона. Известно, что без шума он двигался точно по окружности, однако измерения искажены гауссовским шумом, причём чем больше радиус, тем сильнее шум. Требуется восстановить координаты центра окружности.

Формальная постановка:

Существует неизвестный центр $C = (x^*, y^*) \in \mathbb{R}^2$ и неизвестный радиус $R > 0$. Наблюдения (x_i, y_i) формируются независимо по схеме

$$(x_i, y_i) = C + R(\cos\theta_i, \sin\theta_i) + \varepsilon_i, \quad i = 1, ..., n,$$

где:

- θ_i — угол, сгенерированный из произвольного распределения $\mathcal{D}$ на $[0, 2\pi)$ (может быть неравномерным и даже нулевым на некоторых интервалах);
- $\varepsilon_i \sim \mathcal{N}$ — гауссовский шум, который в среднем равен 0.

Необходимо по данным (x_i, y_i) восстановить центр C.

Формат ввода

Первая строка: целое число n $(3 \le n \le 10^5)$. Далее n строк: по два вещественных числа x_i, y_i — координаты наблюдений. Гарантируется, что $|x_i|, |y_i| \le 10^9$.

Формат вывода

Выведите два вещественных числа x и y — оценку координат центра C.

Система оценивания

Максимальный возможный балл за задачу — 50.

Ваш балл считается по среднему по всем тестам отклонению, нормированному на произведение радиуса и стандартного отклонения шума. То есть по всем тестам:

$$\text{avg} = \frac{\|(x_{\text{true}}, y_{\text{true}}) - (x_{\text{your}}, y_{\text{your}})\|}{\sigma R}$$

Итоговый балл вычисляется по формуле

$$\text{score} = 50 \cdot \max\left(0, \min\left(1 - \frac{\text{avg} - 1.5}{5}, 1\right)\right).$$

Решение

Представим окружность в стандартном, школьном виде:

$$(x_i - x_*)^2 + (y_i - y_*)^2 = R^2 + \varepsilon$$

Раскроем скобки и перенесем в правую часть константы и члены зависящие от y_* и x_*

$$x_i^2 + y_i^2 = 2x_*x_i + 2y_*y_i + R^2 - x_*^2 - y_*^2 + \varepsilon$$

Ничего не напоминает? Конечно, это линейная регрессия:

$$x_i^2 + y_i^2 = \underbrace{2x_*}_{\text{коэффициент A}} x_i + \underbrace{2y_*}_{\text{коэффициент B}} x_i + \underbrace{R^2 - x_*^2 - y_*^2}_{\text{коэффициент C}} + \underbrace{\varepsilon}_{\text{шум}}$$

То есть мы хотим предсказывать величину $x_i^2 + y_i^2$ через линейную регрессию $Ax_i + By_i + C$.

Ответом на эту задачу будет $\frac{A}{2}$ и $\frac{B}{2}$, потому что это будут $\frac{2x_*}{2}$ и $\frac{2y_*}{2}$ соответственно.

Пример реализации:

```python
import numpy as np

class CircleLikeRegression:
    def __init__(self):
        self.A = None
        self.B = None
        self.C = None

    def fit(self, x, y):
        x = np.asarray(x, dtype=float)
        y = np.asarray(y, dtype=float)
        n = x.size

        # Левая часть: z_i = x_i^2 + y_i^2
        z = x**2 + y**2

        # Матрица признаков: [x_i, y_i, 1]
        X = np.column_stack([x, y, np.ones(n)])

        # МНК: (X^T X) theta = X^T z, theta = [A, B, C]^T
        theta, *_ = np.linalg.lstsq(X, z, rcond=None)
        self.A, self.B, self.C = theta

n = int(input())
x, y = [], []
for _ in range(n):
    _x, _y = map(float, input().split())
    x.append(_x)
    y.append(_y)
model = CircleLikeRegression()
model.fit(x,y)
print(model.A/2, model.B/2)
```

Реализация специально запихана в класс, чтобы аналогично моделям можно было прописать просто `model.fit(...)` и получить ответ на задачу.

Эта реализация получает примерно **49.986** баллов. Если использовать некоторые более продвинутые способы можно получить **50** баллов.

Примечание: Обратите внимание, что результат ≥ 45 баллов уже хороший, и если есть нерешенная теоретическая математика стоит бросить добивать ≤ 4 балла. То есть воевать с лучшими, чем 45, баллами на задачах по программированию нужно ТОЛЬКО после решения всех теоретических задач.