Профиль «Большие данные и машинное обучение» погружает участников в выполнение реальных задач, связанных с анализом больших объемов данных и разработкой реальных приложений для анализа данных.

Профиль включает в себя задачи по двум школьным предметам: **математика** и **информатика**.

§1 Первый отборочный этап

Первый отборочный тур проводится индивидуально в сети интернет, работы оцениваются автоматически средствами системы онлайн-тестирования. На решение задач первого отборочного этапа участникам давалось 3 недели. Решение каждой задачи дает определенное количество баллов. Для всех участников предлагается общий набор задач, но за решение задач участникам разных параллелей (9 класс или 10-11 класс) давались разные баллы. Решение задач по информатике подразумевало написание программ на языке Python. Баллы зачисляются в полном объеме за правильное решение задачи. Участники получают оценку за решение задач в совокупности по всем предметам данного профиля (математика и информатика) — суммарно от 0 до 20 баллов.

1.1. Задачи по математике

Задача 1.1.1 (1 балл)

В поисках внеземной жизни ученые обнаружили интересный живой организм - Камкохоб. Эксперименты в земных условиях показали, что Камкохоб размножается делением. То есть родительский организм исчезает, и образуются новые особи. При этом каждая особь либо делится ровно на 5 потомков, либо не размножается, а остается одной особью . Экспериментальный образец, привезенный на Землю размножался, некоторые его потомки тоже размножались. **Выберите в таблице**, какое количество потомков могло получиться в итоге, а какое не могло.

Количество потомков	Могло получиться	Не могло получиться
9		
15		
1001		
2016		

РЕШЕНИЕ:

Правильный ответ на поставленную задачу представлен в таблице. Подробнее ход решения рассматривается в следующей задаче.

Количество потомков	Могло получиться	Не могло получиться
9	да	
15		да
1001	да	
2016		да

Задача 1.1.2 (1 балл)

В поисках внеземной жизни ученые обнаружили интересный живой организм - Камкохоб. Эксперименты в земных условиях показали, что Камкохоб размножается делением. То есть родительский организм исчезает, и образуются новые особи. При этом каждая особь либо делится ровно на 5 потомков, либо не размножается, а остается одной особью. Экспериментальный образец, привезенный на Землю, размножался, некоторые его потомки тоже размножались. Какое количество потомков могло получиться? **Опишите всю серию возможных ответов общей формулой**, используя переменную x (где x - натуральное число).

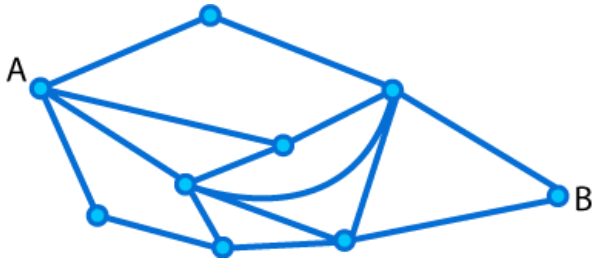
РЕШЕНИЕ:

Посмотрим, как меняется общее количество особей при одном размножении. Родительская особь исчезает, появляется 5 новых. То есть общее количество увеличивается на 4. Поскольку исходно у нас одна особь, то могло получиться только число, дающее остаток 1 от деления на 4.

Формула 4x+1, где x - натуральное число, описывает все возможные варианты. Действительно, чтобы получить 4x+1 особей, достаточно размножить любые x особей.

Задача 1.1.3 (1 балл)

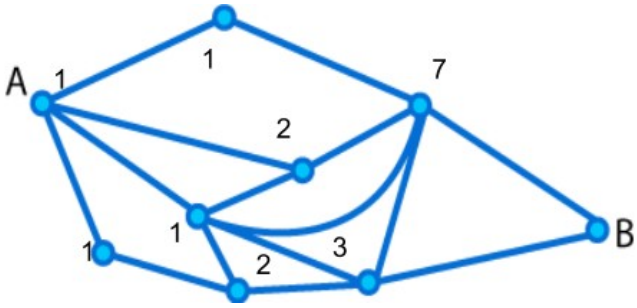
Автобус едет из пункта A в пункт B. При этом в любой момент времени он движется вправо, чтобы приближаться к пункту B иногда вправо вверх, иногда вправо вниз. Найдите количество способов доехать. На картинке приведена схема дорог между городами.



РЕШЕНИЕ:

Расставим количество способов доехать до каждой вершины, двигаясь слева

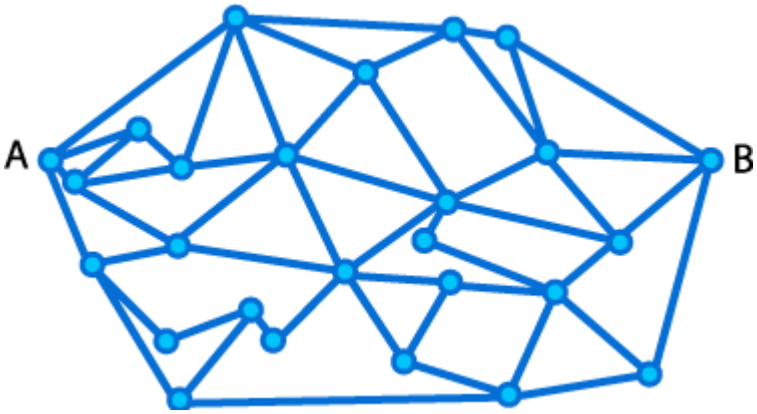
направо.



До вершины B 7+3=10 способов. Ответ: 10

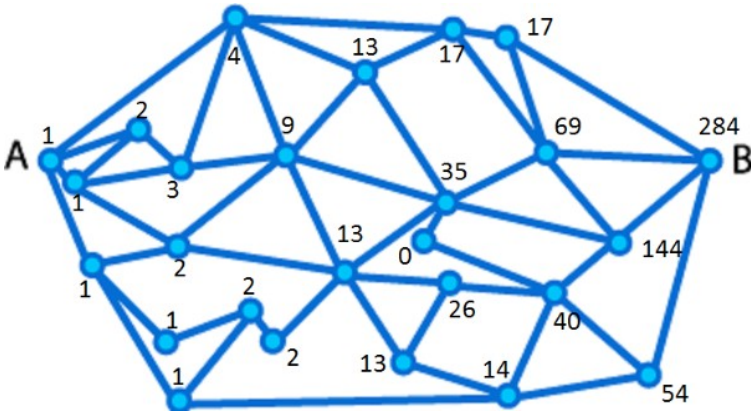
Задача 1.1.4 (1 балл)

Автобус едет из пункта A в пункт B. При этом в любой момент времени он движется вправо, чтобы приближаться к пункту B, иногда вправо вверх, иногда вправо вниз. Найдите количество способов доехать. На картинке приведена схема дорог между городами.



РЕШЕНИЕ:

Расставим количество способов доехать до каждой вершины, двигаясь слева направо.



Ответ: 284

Задача 1.1.5 (1 балл)

Незадачливый космонавт Иннокентий почувствовал себя плохо после центрифуги и не может определить направление, он находится в 5 метрах от комиссии и движется по прямой. Каждую секунду он с равной вероятностью либо приближается на метр к ней, либо отдаляется. Если он дошел до комиссии, то больше уже никуда не идет. Найдите вероятность попадания в руки комиссии не позже чем **на 6 секунде**.

Решение:

Пусть комиссия находится в пяти метрах справа от космонавта.

Любой путь космонавта за 6 секунд кодируется последовательностью из 6 символов П или Л (П, если он идет направо и Л - если налево). Всего таких последовательностей 64, из них нам годятся две: ПППППП и ПППППЛ.

Значит вероятность попасть к комиссии равна 2/64=1/32

Ответ: 1/32

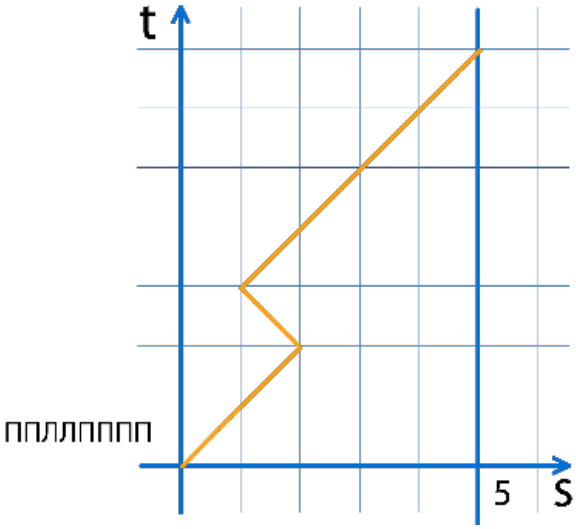
Задача 1.1.6 (1 балл)

Незадачливый космонавт Иннокентий почувствовал себя плохо после центрифуги и не может определить направление, он находится в 5 метрах от комиссии и движется по прямой. Каждую секунду он с равной вероятностью либо приближается на метр к ней, либо отдаляется. Если он дошел до комиссии, то больше уже никуда не идет. Найдите вероятность попадания в руки комиссии не позже чем **на 10-й секунде**.

РЕШЕНИЕ:

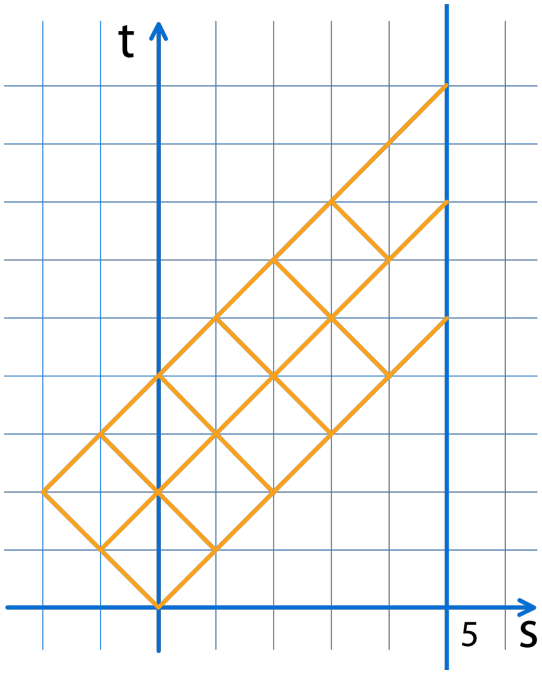
Изобразим путь космонавта на графике. По вертикальной оси отложим время, а по горизонтальной расстояние. Изначально космонавт находится в точке s=0, а комиссия в точке s=5.

На рисунке обозначен путь для последовательности ППЛПППП:

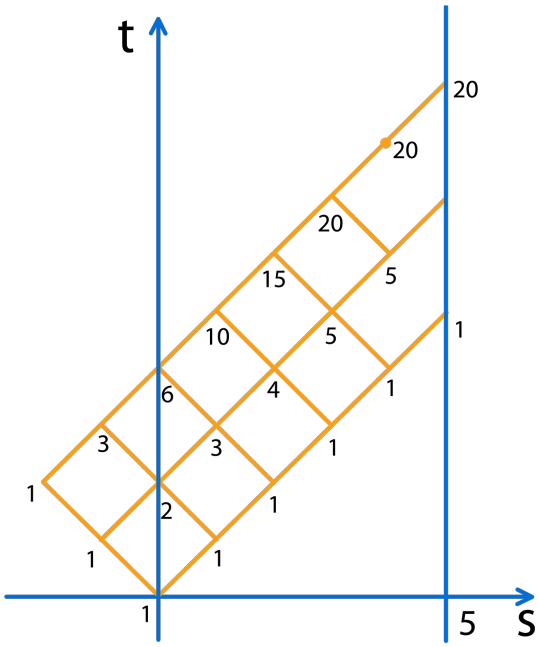


Заметим также, что космонавт может прийти к комиссии только на нечетных шагах. Таким образом, нас интересует, сколько существует путей, ведущих к комиссии за 5, 7 и 9 секунд.

Переформулируем задачу: сколько существует путей, по сетке на рисунке ниже, ведущих к прямой s=5. Ходить можно только двигаясь вверх.



Посчитаем количество таких путей для каждой точки, начиная от начальной.



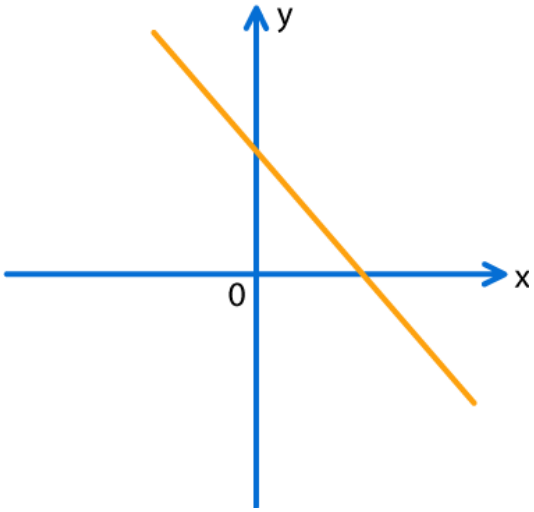
За 5 шагов приводит 1 путь, за 7 - 5 путей, за 9 - 20 путей.

Значит искомая вероятность равна 1/32+5/128+20/512 = 7/64

Ответ: 7/64

Задача 1.1.7 (1 балл)

На координатной плоскости задан график функции y=kx+b.



Найдите максимальное значение функции $y=kx^2+bx$.

РЕШЕНИЕ:

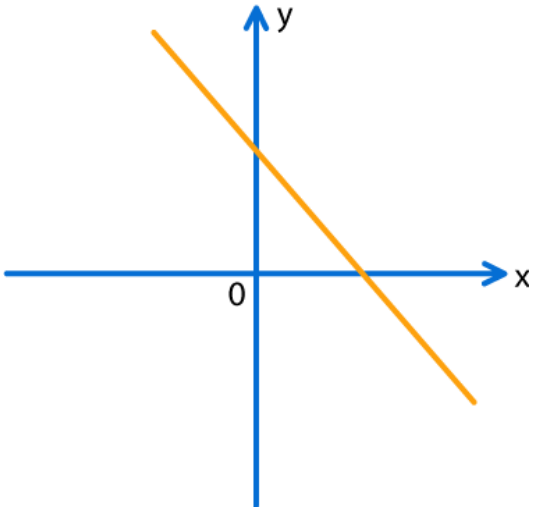
Поскольку y = kx+b убывает, k<0

$y = kx^2+bx = x(kx+b)$ парабола «рожками вниз», с нулями x = 0 и x = -b/k

Абсцисса оси параболы x= -b/2k подставим и найдем максимум $y=-b^2/4k$

Задача 1.1.8 (1 балл)

На координатной плоскости задан график функции $y=kx+b$.



Рассмотрите точки пересечения графика этой функции с графиком функции $y=kx^2+bx$. Выберите из этих точек точку с наименьшей абсциссой и выведите в ответе, чему равна эта абсцисса.

РЕШЕНИЕ:

Графиком квадратичной функции $y = x(kx + b)$ является парабола «рогами вниз», т.к. $k<0$

Один из нулей этой функции совпадает с нулем функции $y = kx + b$, а другой: $x = 0$. Между нулями расположена вторая точка пересечения графиков. Одним из корней уравнения $x(kx + b) = kx + b$ является $x = 1$, что даёт абсциссу второй точки пересечения данного и искомого графиков.

Ответ: 1

1.2. Задачи по информатике

Задача 1.2.1 «Кости» (1 балл)

В некоторых клетках квадратной доски 4×4 находятся четырёхгранные игральные кости. Кость снимается с поля, если количество соседних по стороне непустых клеток совпадает с числом, выпавшем на кости. Все кости, для которых выполняется это свойство, снимаются с поля одновременно. Если после снятия костей появляются новые кости, которые можно снять, то они снимаются по тем же правилам.

Нужно переставить одну кость так, чтобы снялось максимальное количество костей.

Формат входных данных:

На вход программе даётся четыре строки по четыре числа в каждом, разделённые одним пробелом. Всё числа — целые неотрицательные, не превышающие 4. Ноль означает отсутствие кости, любое положительное — наличие кости, на которой выпало указанное число.

Гарантируется, что на поле есть, по крайней мере, одна кость и одна свободная клетка

Пример ввода 1:

```
0 0 1 0
0 0 0 0
0 0 1 0
0 0 0 0
```

Пример ввода 2:

```
0 0 0 1
0 2 1 0
0 1 2 1
0 0 0 0
```

Формат выходных данных:

В виде ответа нужно вывести одно целое число — количество костей, которое можно снять с доски.

Пример вывода 1:

```
2
```

Пример вывода 2:

```
6
```

Пояснение. Во втором примере можно переставить единицу с первого ряда в четвёртый, под двойку. Тогда у левой верхней двойки будет два соседа, и она снимется. Так же с ней снимутся единица в четвёртом ряду и единица в четвёртом столбце, так как у них по одному соседу. После этого на доске останутся две единицы и одна двойка — их общий сосед. Все оставшиеся кости можно снять, так как правило снятия выполняется.

СПОСОБ ОЦЕНКИ РАБОТЫ:

Для генерации уникального условия и проверки результата используется следующий код на языке Python. Функция `generate` возвращает набор тестов и правильные ответы:

```
def generate():
      return [('0 0 1 0\n0 0 0 0\n0 0 1 0\n0 0 0 0\n', 2),
('0 0 0 1\n0 2 1 0\n0 1 2 1\n0 0 0 0\n', 6),
('0 3 3 2\n3 4 4 3\n3 4 4 2\n2 3 2 2\n', 15),
('1 1 1 2\n2 2 2 2\n2 2 2 1\n1 2 3 0\n', 15),
('1 0 0 0\n0 0 0 0\n0 0 0 0\n0 0 0 0\n', 0),
('4 4 4 4\n4 2 2 4\n4 2 1 4\n4 4 0 4\n', 0),
```

```
('0 3 0 2\n1 4 1 4\n4 0 4 0\n3 0 0 2\n', 3),
('2 4 1 1\n4 2 3 1\n4 3 1 1\n1 1 4 0\n', 4),
('2 3 1 0\n0 3 0 0\n1 1 1 4\n4 4 1 3\n', 5),
('0 3 3 3\n1 0 3 3\n2 2 0 0\n2 4 0 2\n', 6),
('2 0 1 2\n4 2 4 0\n0 3 0 4\n1 3 2 0\n', 7),
('3 1 4 1\n0 4 2 3\n4 2 0 1\n3 3 2 2\n', 8),
('2 4 0 4\n0 2 1 3\n3 2 1 0\n1 1 3 4\n', 9),
('4 3 3 3\n0 3 1 0\n2 2 4 1\n0 4 2 2\n', 10),
('4 3 3 1\n2 2 3 1\n1 1 4 3\n3 0 2 4\n', 11),
('0 3 2 1\n2 2 2 3\n1 0 2 1\n1 1 3 3\n', 12),
('1 3 1 2\n3 1 2 1\n2 2 0 3\n2 2 2 2\n', 13)]

def check(reply, clue):
    return int(reply) == int(clue)
```

РЕШЕНИЕ:

В этой задаче нужно аккуратно реализовать то, что указано в условии. Пример программы, реализующей данный алгоритм на языке Python:

```
1. import copy
2. import sys
3.
4. def neighbours(a, x, y):
5.     res = 0
6.     for dx, dy in [(1, 0), (0, 1), (-1, 0), (0, -1)]:
7.         xx = x + dx
8.         yy = y + dy
9.         res +=  int(-1 < xx < len(a) and -1 < yy < len(a[xx]) and a[xx][yy] !
= 0)
10.     return res
11.
12. def clear(a):
13.     b = []
14.     res = 0
15.     for x in range(len(a)):
16.         b.append([0] * len(a[x]))
17.         for y in range(len(a[x])):
18.             if a[x][y] == 0:
19.                 continue
20.             if a[x][y] != neighbours(a, x, y):
21.                 b[x][y] = a[x][y]
22.             else:
23.                 res += 1
24.     return (res, b)
25.
26.
27. def test(a, x1, y1, x2, y2):
28.     a = copy.deepcopy(a)
29.     a[x2][y2] = a[x1][y1]
30.     a[x1][y1] = 0
31.     res = 0
32.     while True:
33.         q, a = clear(a)
34.         if q == 0:
35.             return res
36.         res += q
37.
38. def solve(data):
39. a = list(map(lambda s: list(map(int, s.split())), filter(None, data.split('\
n'))))
40.     ans = 0
```

```
    for x in range(len(a)):
        for y in range(len(a[x])):
            if a[x][y] == 0:
                continue
            for xx in range(len(a)):
                for yy in range(len(a[x])):
                    if a[xx][yy] != 0:
                        continue
                    t = test(a, x, y, xx, yy)
                    if t > ans:
                        ans = t
    return str(ans)

solve(sys.stdin.read())
```

Задача 1.2.2 «Корни» (3 балла)

У мальчика Пети есть число *N*. Но оно ему не нужно, в отличие от числа *X*. Чтобы его получить Петя может брать целочисленный корень, умножать и складывать числа. Целочисленным корнем степени k из натурального числа n будем называть наибольшее натуральное число, для которого выполняется соотношение: $x^k \leq n$. Например, целочисленным корнем пятой степени из тысячи будет тройка, так как $3^5 = 243 \leq 1000$, а $4^5 = 1024 > 1000$. Обозначим это как $\sqrt[5]{1000} = 3$. Также будем считать, что степенью корня могут быть только натуральные числа.

Для Пети взятие целочисленного корня — тяжёлая задача, и он хочет минимизировать суммарную степень корней, которые встречаются в формуле получения *X*.

А ещё у Пети есть старший брат. Которого зовут Дима. Этот Дима решил добавить интереса задачке Пети, и дать такие ограничения:

1. Пете нельзя брать корни от чисел, отличных от *N*.
2. Можно перемножать только те числа, которые Петя получил с помощью операции взятия корня или умножения других чисел.
3. Складывать можно числа, которые получены как результат умножения, взятия корня или суммы других чисел.

Помогите Пете написать выражение, которое будет легко считаться и подходит под ограничения, наложенные Димой. Найдите минимальную сложность искомого выражения.

Формат входных данных:

В единственной строке даны два целых числа N и X ($1 \leq X, N \leq 1000$).

Пример ввода:

```
100 126
```

Формат выходных данных:

Выведите единственное натуральное число — ответ на задачу.

Пример вывода:

```
9
```

Пояснение к примеру: $126 = 100 + 10 + 4 \cdot 4 = \sqrt[1]{100} + \sqrt[2]{100} + \sqrt[3]{100} \cdot \sqrt[3]{100}$

СПОСОБ ОЦЕНКИ РАБОТЫ:

Для генерации уникального условия и проверки результата используется следующий код на языке Python. Функция `generate` возвращает набор тестов и правильные ответы:

```
def generate():
    return [('{} {}\n'.format(n, x), ans) for n, x, ans in [
(100, 126, 9),
(10, 10, 1),
(1000, 1000, 1),
(1, 1, 1),
(1, 1000, 1000),
(1000, 1, 10),
(1000, 999, 17),
(722, 966, 16),
(774, 717, 21),
(664, 177, 16),
(655, 657, 7),
(659, 65, 9),
(901, 559, 21),
(813, 314, 18),
(528, 131, 16),
(882, 258, 19),
(516, 583, 12),
(801, 767, 19),
(147, 222, 11),
(67, 743, 13),
(413, 335, 21),
(453, 467, 7),
(600, 104, 9),
(323, 209, 19),
(462, 822, 18),
(126, 743, 16),
(77, 917, 17),
(100, 999, 27),
(1, 999, 999),
(1000, 894, 29),
(999, 712, 28),
(123, 944, 24),
(432, 277, 24),
(945, 616, 28),
(100, 999, 27),
(1000, 894, 29),
(999, 712, 28),
(123, 944, 24),
(432, 277, 24),
(945, 616, 28)
]]

def check(reply, clue):
    return int(reply.strip()) == int(clue)
```

РЕШЕНИЕ:

Решение состоит из трёх этапов. На первом этапе нужно составить набор степеней и корней из N, которые может быть выгодно использовать. Если два корня с разной степенью равны, то нам выгодно использовать тот из них, степень которого меньше. Так как 210 > 1000, то для любого N будет не более 11 различных корней. На втором этапе методом динамического программирования получаем список чисел, которые можно получить перемножением корней с оптимальной суммарной степенью. На третьем этапе используя тот же метод получается список чисел, которые можно получить сложением чисел с предыдущего этапа с оптимальными суммами.

Пример программы, реализующей данный алгоритм на языке Python:

```
1. import sys
2.
3. INF = int(1e9)
4.
5. def getRoots(n, mx):
6.     ans = [INF, n]
7.     for x in range(n, 1, -1):
8.         while x ** len(ans) <= n:
9.             ans.append(x)
10.     ans.append(1)
11.     roots = dict()
12.     for i in range(len(ans)):
13.         if ans[i] != ans[i - 1] and ans[i] <= mx:
14.             roots[ans[i]] = i
15.     return roots
16.
17. def getProducts(roots, mx):
18.     ans = dict()
19.     ans[1] = 0
20.     for i in range(2, mx + 1):
21.         ans[i] = INF
22.         for k, v in roots.items():
23.             if i % k == 0:
24.                 d = i // k
25.                 if d in ans and ans[d] + v < ans[i]:
26.                     ans[i] = ans[d] + v
27.         if ans[i] == INF:
28.             ans.pop(i, None)
29.     ans[1] = roots[1]
30.     prods = [(k, v) for k, v in sorted(ans.items())]
31.     return prods
32.
33. def getSums(prods, mx):
34.     ans = [INF] * (mx + 1)
35.     ans[0] = 0
36.     for i in range(len(ans)):
37.         for k, v in prods:
38.             if k > i:
39.                 break
40.             if ans[i - k] + v < ans[i]:
41.                 ans[i] = ans[i - k] + v
42.     ans[0] = INF
43.     return ans
44.
45. def solve(dataset):
```

```
46.     n, x = list(map(int, dataset.strip().split()))
47.     roots = getRoots(n, x);
48.     prods = getProducts(roots, x)
49.     ans = getSums(prods, x)
50.     return str(ans[x])
51.
52. solve(sys.stdin.read())
```

Задача 1.2.3 «Прямая и точки» (3 балла)

На плоскости проведена прямая, проходящая через начало координат. Также на плоскости выбрано N точек, для каждой известно, с какой стороны от прямой она лежит. И M точек, для которых нужно ответить на аналогичный вопрос.

Формат входных данных:

В первой строке дано натуральное число N — число точек, для которых известно, с какой стороны прямой они лежат ($1 \le N \le 100000$). В следующих N строках идёт описание этих точек. Каждая строка состоит из трёх целых чисел, разделённых пробелом — координаты точки и указание, с какой стороны лежит точка. Все точки, для которых третье число равно нулю лежат по одну сторону от прямой, а единице — с другой. Других значений третьего параметра нет.

В следующей строке дано число M — число точек, для которых нужно указать сторону ($1 \le M \le 100000$). В следующих M строках пары целых чисел — координаты точек.

Координаты всех точек не превышают 1000000 по абсолютному значению. Гарантируется, что ни одна из точек не лежит на прямой.

Пример ввода:

```
4
1 0 1
0 1 1
-1 0 0
0 -1 0
2
1 1
-1 -1
```

Формат выходных данных:

Для каждой из M точек выведите 0 или 1 — сторону, с которой лежит точка, в том же смысле, как и во входных данных. Гарантируется, что ответ единственен.

Пример вывода:

```
1
0
```

СПОСОБ ОЦЕНКИ РАБОТЫ:

Для генерации уникального условия и проверки результата используется следующий код на языке Python. Функция `generate` возвращает набор тестов и правильные ответы:

```
def generate():
      return [("4\n1 0 1\n0 1 1\n-1 0 0\n0 -1 0\n2\n1 1\n-1 -1\n", [1, 0])]

def check(reply, clue):
      if type(clue) is str:
            clue = ast.literal_eval(clue)
      ans = list(map(int, filter(None, reply.split())))
      if len(ans) != len(clue):
            return False
      for i in range(len(ans)):
            if ans[i] != clue[i]:
                  return False
      return True
```

РЕШЕНИЕ:

Если отразить все точки, у которых «сторона» равна 1, относительно начала координат, то все точки попадут в некоторый сектор плоскости. Остаётся найти прямую, которая не пересекает этот сектор. Её можно найти следующим образом: находим два луча, которые образуют границы сектора (например, используя псевдоскалярное произведение). Искомая прямая будет перпендикулярна биссектрисе этих лучей.

Пример программы, реализующей данный алгоритм на языке Python:

```
1. import sys
2. import math
3.
4. def length(p):
5.     return math.sqrt(p[0] * p[0] + p[1] * p[1])
6.
7. def solve(dataset):
8.     lines = dataset.strip().split('\n')
9.     points = []
10.     n = int(lines[0])
11.     lines = lines[1:]
12.     for i in range(n):
13.         x, y, s = list(map(int, lines[i].split()))
14.         s = 2 * s - 1
15.         x *= s
16.         y *= s
17.         points.append((x, y))
18.     lines = lines[n:]
19.
20.     mn = points[0]
21.     mx = points[0]
22.     for x, y in points:
23.         if x * mn[1] - y * mn[0] > 0:
24.             mn = (x, y)
25.         if x * mx[1] - y * mx[0] < 0:
26.             mx = (x, y)
27.
28.     mnLen = length(mn)
29.     mxLen = length(mx)
30.
```

```
31.     a = mn[0] / mnLen + mx[0] / mxLen
32.     b = mn[1] / mnLen + mx[1] / mxLen
33.
34.     m = int(lines[0])
35.     lines = lines[1:]
36.     res = []
37.     for i in range(m):
38.         x, y = list(map(int, lines[i].split()))
39.         res.append(str(int(x * a + y * b > 0)))
40.     return '\n'.join(res)
41.
42. solve(sys.stdin.read())
```

Задача 1.2.4 «Черный ящик» (5 баллов)

Есть чёрный ящик с двумя вещественными входами и одним бинарным выходом. Нужно по заданному набору входных и выходных данных максимально точно предсказать ответы чёрного ящика на данные из другой выборки входных данных.

Формат входных данных:

В первой строке дано число N — количество входных данных, для которых известны ответы. В следующих N строках дано по два вещественных и одному натуральному числу, разделённых пробелами — входные параметры и ответ чёрного ящика, соответственно.

В следующей строке дано число M — количество входных данных, для которых нужно узнать ответ. В следующих M строках даны по два вещественных числа, разделённых пробелом — сами входные данные.

Формат выходных данных:

В ответ нужно вернуть M чисел — предполагаемые ответы чёрного ящика.

Примечание 1: В этой задаче Вам нужно скачать файл с тестом, решить задачу на вашем компьютере и вернуть ответ за пять минут. В процессе решения вы можете пользоваться любыми материалами, языками и программами. Через пять минут после начала решения задачи тест помечается как устаревший и ответ к нему не принимается. После этого можно запросить новый тест, при этом таймер запускается заново. Количество попыток не ограничено. Гарантируется, что все тесты получены с использованием одного чёрного ящика с незначительной заменой внутренних констант.

Примечание 2: Эта задача подразумевает возможность частичного решения. В случае, если правильные ответы даны менее чем для половины входных, то за задачу начисляется 0 баллов. Если процент правильных ответов превышает 95%, то начисляется 5 баллов. Иначе начисляется $5* \frac{100}{81}(2p-1)^2$, где p — доля правильных ответов.

СПОСОБ ОЦЕНКИ РАБОТЫ:

Для генерации уникального условия и проверки результата используется следующий код на языке Python. Функция `generate` возвращает набор тестов и правильные ответы:

```
def sign(x):
      return int(x > 0)

def sqr(x):
       return x * x

eps = 0.1

def getPoint(a, b, rot):
            phi = random.uniform(0, 2 * math.pi)
            r = random.uniform(0, 10)
            x = r * math.cos(phi)
            y = r * math.sin(phi)
            f = x * x * a - y * y * b - 2
            xx = x * math.cos(rot) - y * math.sin(rot)
            yy = x * math.sin(rot) + y * math.cos(rot)
            if abs(f) < eps:
                  return None
            return (xx, yy, sign(f))

def generate():
      mode = random.choice([0, 1])
      mid = math.pi / 100 * 29
      phi = random.uniform(mid - 0.1, mid + 0.1)
      train = 1000
      test = 500
      rot = random.uniform(0, 2 * math.pi)
      a = 1/math.cos(phi) ** 2
      b = 1/math.sin(phi) ** 2

      data = '{}\n'.format(train)

      i = 0
      while i < train:
            p = getPoint(a, b, rot)
            if p is None:
                  continue
            data += '{0:.10f} {1:.10f} {2}\n'.format(p[0], p[1], p[2] ^ mode)
            i += 1

      ans = []
      data += '{}\n'.format(test)
      i = 0
      while i < test:
            p = getPoint(a, b, rot)
            if p is None:
                  continue
            data += '{0:.10f} {1:.10f}\n'.format(p[0], p[1])
            ans.append(p[2] ^ mode)
            i += 1

      return (data, ans)

MAXRES = 5
def check(reply, clue):
      ans = list(map(int, reply.split()))
      if len(ans) != len(clue):
            return (0, 'баллов: 0')
```

```
    correct = 0
    for i in range(len(clue)):
        correct += int(clue[i] == ans[i])
    if correct * 2 <= len(clue):
        return (0, 'баллов: 0')
    correct /= len(clue)
    res = min(1, 100 * (2 * correct - 1) ** 2 / 81)
    return (res, 'баллов: {0:.2f}'.format(MAXRES * res))
```

РЕШЕНИЕ:

Если нарисовать данный тест на плоскости, считая пары чисел как координаты, то можно заметить, что есть одна зона, в которую попадают все точки с одним ответом и две, куда попадают остальные. Причём границы этих областей имеют простую структуру. Дальше можно было либо понять, что граница представляет собой гиперболу и восстановить её параметры, либо воспользоваться специальными инструментами, либо реализовать подходящий алгоритм машинного обучения, например, «метод трёх соседей»: выбираем для изучаемой точки три ближайших. Ответ для точки будет совпадать с преобладающим ответом среди соседей.

Пример программы, реализующей данный алгоритм на языке Python:

```
1. import sys
2. import math
3. import statistics
4.
5. def sqr(x):
6.     return x * x
7.
8. def solve(data, k = 3):
9.     lines = data.split('\n')
10.     n = int(lines[0])
11.     lines = lines[1:]
12.     p = [0] * n
13.     for i in range(n):
14.         p[i] = list(map(float, lines[i].split()))
15.     res = ''
16.     lines = lines[n:]
17.     m = int(lines[0])
18.     lines = lines[1:]
19.     for i in range(m):
20.         x, y = list(map(float, lines[i].split()))
21.         pp = sorted((sqr(x - xx) + sqr(y - yy), t) for xx, yy, t in p)
22.         ts = [t for d, t in pp[:k]]
23.         res += '{0:.0f}\n'.format(round(statistics.mean(ts)))
24.     return res
25.
26. solve(sys.stdin.read())
```

1.3. Критерии определения призеров и победителей

Количество баллов, набранных при решении всех задач суммируется. Призерам первого отборочного этапа необходимо набрать 7 баллов (для 9 класса) и 9 баллов (для 10-11 класса). Победители первого отборочного этапа должны набрать 20 баллов.