

СТРАТЕГИИ ПОДГОТОВКИ К ЕГЭ ПО ИНФОРМАТИКЕ

Сенчилова Ольга Константиновна,
учитель информатики
СОГБОУИ «Лицей имени Кирилла и Мефодия»

2 месяца до экзамена. Если подготовка почти завершена...

Приоритеты:

- Тайминг
- Альтернативные способы решения
- Планирование личной стратегии на экзамене
- Регулярные тренировки

Стратегия 1. Начальный уровень

База: минимальные теоретические основы, знакомство с языком программирования

- Простые задания с шаблонными рассуждениями

№№ 1, 4, 7, 16(без программы)

10, 6, 3, 9, 18

- Простые шаблонные программы

№№ 2, 8, 15(кроме отрезков), 14

25(на маски), 12(большая часть)

Важно:
внимательность!

Стратегия 2. Достаточная база знаний + навыки программирования

База: справляется с задачами начального уровня; умеет использовать ветвления и циклы, обрабатывать массивы и строки; знаком с типовыми алгоритмами (перевод в другую систему счисления, проверка простоты числа и т.п.)

- Уход от шаблонов к общим схемам решения
- Осознанный выбор инструментария
- Специальные приемы: возможности Python (!), функции электронных таблиц

№№ 1-23, 25, 27(A)

Систематичность и практика!

Полезные функции электронных таблиц

- Стандартные: МИН, МАКС, СУММ, СРЗНАЧ, СЧЁТ
- Логические, для единичных ячеек: ЕСЛИ, И, ИЛИ
- Для обработки диапазона
СУММЕСЛИ, СЧЁТЕСЛИ, СРЗНАЧЕСЛИ
НАИБОЛЬШИЙ(диапазон; k), НАИМЕНЬШИЙ(диапазон; k)
- возвращают k-ое по величине значение в диапазоне
- Математические
ABS, ОСТАТ, ЧАСТНОЕ, ОКРУГЛВВЕРХ, ОКРУГЛВНИЗ,
ОСНОВАНИЕ(число, основание, мин.длина) – перевод в
другую систему счисления (в виде текста)
ДЕС(текст, основание) – перевод из заданной системы
счисления в десятичное число

Специальные приемы в Python

Используем тип множество

- Удаление дубликатов из списка:

```
spisok = list(set(spisok))
```

- Проверка уникальности элементов:

```
if len(spisok) == len(set(spisok)) ...
```

Специальные приемы в Python

- Ускорение работы рекурсии

Используем автоматическое кэширование значений функции, т.е. реализуем метод динамического программирования с помощью встроенной библиотеки):

```
from functools import *
```

```
@lru_cache(None)
```

```
def func(<параметры>):
```

```
    <код функции>
```

Специальные приемы в Python

- Построение комбинаторных объектов

С помощью библиотеки `itertools` можно создать список всех возможных комбинаций, из которого далее отобрать нужное по условию.

Комбинации формируются в виде кортежей, для удобства работы из них лучше сформировать строки

```
from itertools import *  
# перестановки  
a = ["".join(p) for p in permutations('ABCD')]  
# размещения по 3 (с повторениями)  
b = ["".join(p) for p in product('ABCD', repeat=3)]
```


Стратегия 3. Осваиваем самые сложные задания

База: ученик хорошо подготовлен, справляется с задачами 1-23; уверенно программирует

- Идеи, структуры данных, приемы
- Разные способы решения одной задачи
- Тренировка на большом количестве задач

№№ 24, 25(делимость), 26, 27(В)???

Эти задания почти всегда обновляются!

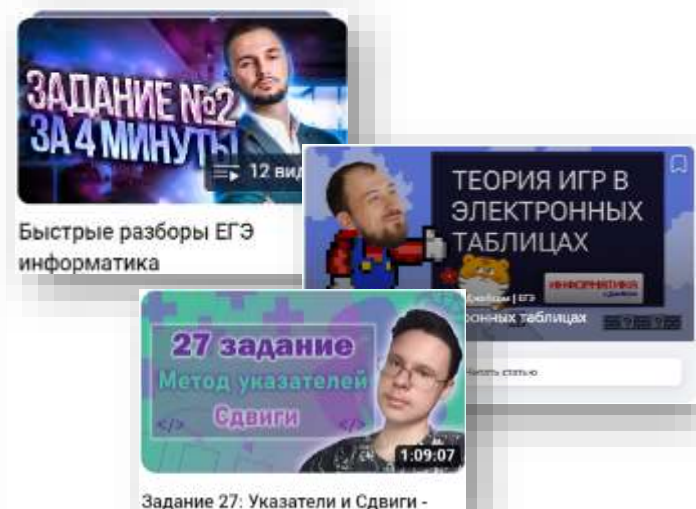
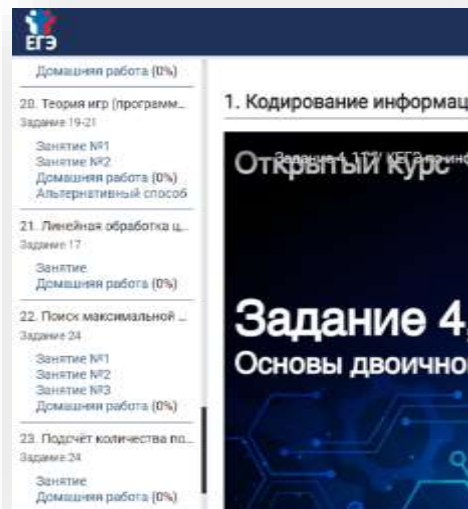
Стратегия 3. О задании 27(В)

- 27(А) нужно решать обязательно. Обычно для него достаточно простого перебора.
- 27(В) – очень индивидуально:
 - Каждый год обновляется тип задания
 - Для решения обычно требуется очень много времени
 - В сложном алгоритме большая вероятность допустить неточность и получить неверный ответ
 - При этом дает всего 1 первичный балл (т.е. 2 тестовых)

Ученик должен для себя определить, что ему выгоднее – оставить время на перепроверку остальных заданий или пытаться решить 27(В).

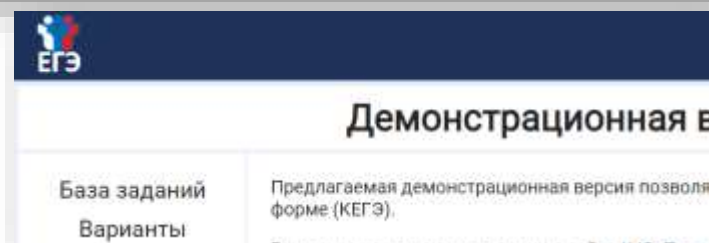
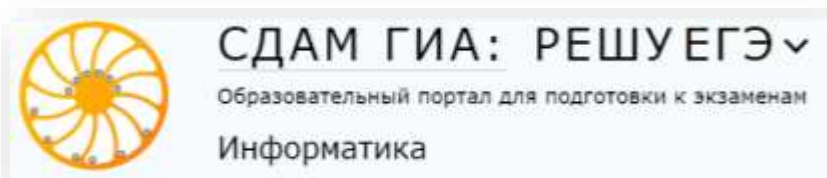
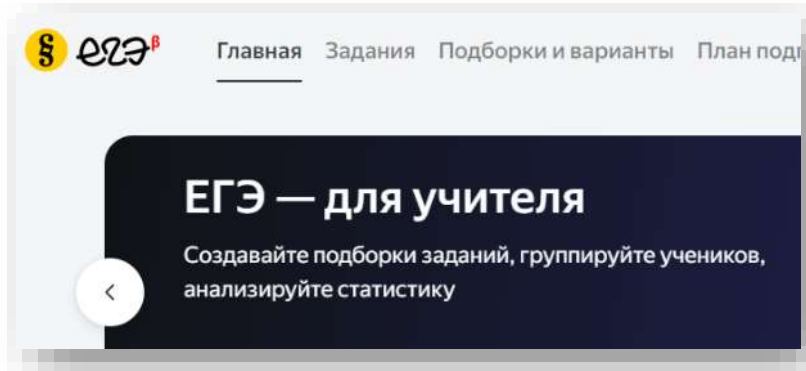
Где можно посмотреть? (теория, методы решения)

- Сайт К.Ю. Полякова (kpolyakov.spb.ru)
- Открытый курс А. Кабанова (Сайт komprege.ru и канал https://vk.com/ege_info_open)
- Видеоразборы и статьи в открытом доступе (А. Имаев, Е. Джобс, Л.Шастин и другие)



Где удобно тренироваться?

- Яндекс.Учебник (education.yandex.ru/ege)
- Сайт К.Ю. Полякова (kpolyakov.spb.ru)
- Сайт КЕГЭ (kompege.ru)
- Сайт sdamgia.ru



Советы по решению задач

- Решаются «вручную»: 1, 4, 7, 11, 15(с отрезками), 16(отношение или разность функций)
- Решаются «вручную» + использование ПК: 2, 13, 22
- Написание программы облегчает решение: 6, 8, 12, 15, 16(не всегда), 19-21
- Решаются в электронных таблицах: 3, 18
в текстовом редакторе 10
- На выбор, программой или в таблицах: 9, 26
- Решаются программой: 5, 14, 17, 23, 24, 25, 27

Решаем письменно

- №1. Опираемся на степени вершин
- №4. Строим дерево кодов

В №1 и №4 можно опираться на предположения: либо получится построить решение, либо получим противоречие и рассмотрим следующий вариант.

- №7. Две базовые формулы, умение преобразовывать единицы измерения информации, работать с процентами
- №11 – Простые формулы, но важно достичь понимания, когда нужно округлять с недостатком, а когда – с избытком (внимательно вычитываем условие!)

ПК как вспомогательный инструмент

- №2. Таблицу истинности быстрее построить на компьютере: с помощью программы или в ЭТ.

$$((x \vee z) \rightarrow (y \wedge x)) \wedge ((w \equiv z) \vee (w \rightarrow \neg y))$$

Шаблон программы:

```
print('x y z w F')
for x in (0, 1):
    for y in (0, 1):
        for z in (0, 1):
            for w in (0, 1):
                F = ((x or z) <= (y and x)) and ((w == z) or (w <= (not y)))
                if F == 1:
                    print(x, y, z, w, int(F))
```

and <= заменяет импликацию
or == заменяет эквиваленцию
not

Важно: порядок действий!
Каждую операцию лучше
взять в скобки

ПК как вспомогательный инструмент

- №13. Удобно в ЭТ: функции **ОСНОВАНИЕ** и **ДВ.В.ДЕС**

137.88 Маска: 255.255.240.0

92 . . . 137 . . . 88 IP

.10001001.01011000 IP

.10000000.00000000 Адрес сети

92 . . . 128 . . . 0 Адрес сети

111111.11110000.00000000 Маска

255 . . . 240 . 0 Маска

ра: 1001.01011000 = 100101011000₂ = 2392

D	E	F	G
.	137	.	88
=ОСНОВАНИЕ(E2;2;8)			01011000

Число

Основание

Минимальная длина

Формула:

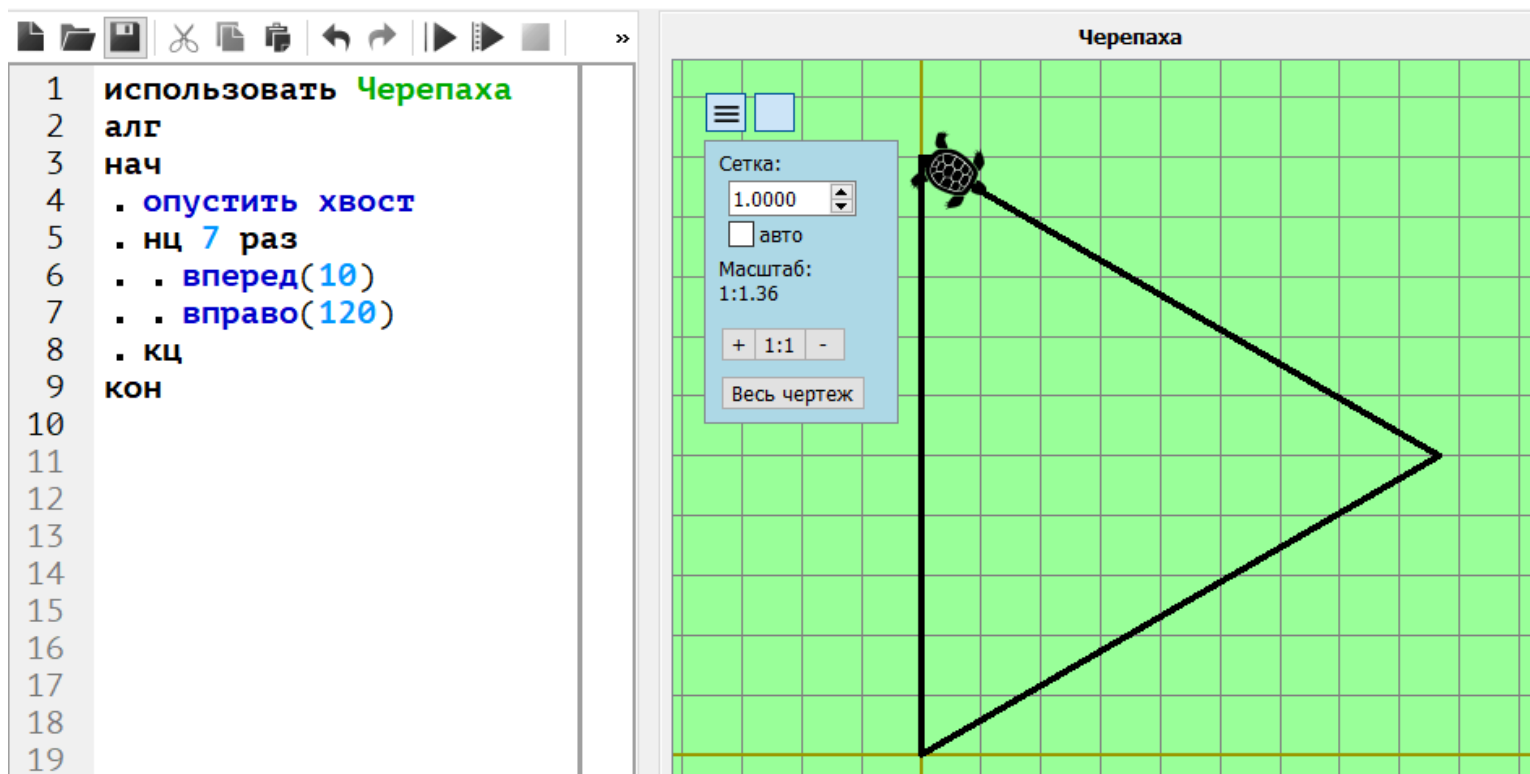
Результат:

Решать программой - удобно

- **№6.** Создаем аналогичную программу в **Кумир** или в **Python** с модулем **turtle**.

Алгоритм для Черепахи: Повтори 7 [Вперёд 10 Направо 120]

Программа в Кумир:



The screenshot displays the KUMIR programming environment. On the left, a code editor shows a program for a turtle named 'Черепаха' (Turtle). The code is as follows:

```
1 использовать Черепаха
2 алг
3 нач
4   . опустить хвост
5   . нц 7 раз
6     . вперед(10)
7     . вправо(120)
8   . кц
9 кон
```

On the right, the 'Черепаха' window shows a green grid with a turtle icon at the top. The turtle has drawn a triangle with a vertical left side, a horizontal bottom side, and a diagonal right side. A control panel is visible in the top-left corner of the drawing area, containing the following settings:

- Сетка: 1.0000
- авто
- Масштаб: 1:1.36
- + 1:1 -
- Весь чертеж

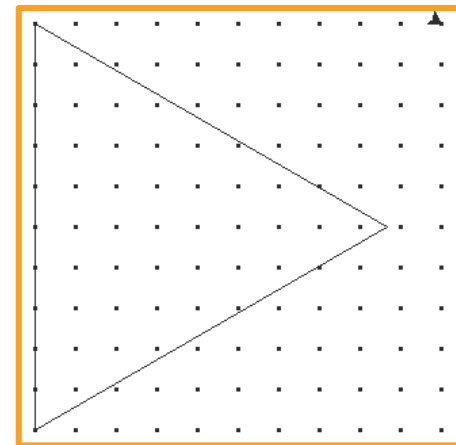
Решать программой - удобно

- **№6.** Создаем аналогичную программу в **Кумир** или в **Python с модулем turtle**.

Алгоритм для Черепахи: Повтори 7 [Вперёд 10 Направо 120]

Программа на Python:

```
from turtle import *
k = 30 # масштаб
speed(0) # для ускорения рисования
left( 90 ) # развернуть Черепаху "на север"
for i in range(7):
    forward( 10*k )
    right( 120 )
up()
for x in range(0, 11): # сетка точек
    for y in range(0, 11):
        goto( x*k, y*k )
        dot( 4, 'red' )
done() # для нормальной работы окна с Черепахой
```



Решать программой - удобно

- №8. Комбинаторика - но написать программу может оказаться быстрее и надежнее.

Определите количество пятизначных чисел, записанных в девятеричной системе счисления, которые не начинаются с нечётных цифр, не оканчиваются цифрами 1 или 8, а также содержат в своей записи не более одной цифры 3.

Способ 1. Вложенные циклы (не всегда легко)

```
ts = '012345678'
k = 0
for a1 in '2468':
    for a2 in ts:
        for a3 in ts:
            for a4 in ts:
                for a5 in '0234567':
                    s = a1 + a2 + a3 + a4 + a5
                    if s.count('3') <= 1:
                        k += 1
print(k)
```

Решать программой - удобно

- **№8.** Комбинаторика - но написать программу может оказаться быстрее и надежнее.

Определите количество пятизначных чисел, записанных в девятеричной системе счисления, которые не начинаются с нечётных цифр, не оканчиваются цифрами 1 или 8, а также содержат в своей записи не более одной цифры 3.

Способ 2. Функции из модуля `itertools`

```
from itertools import product

cmb = ["".join(p) for p in product('012345678', repeat=5)]
k = 0
for s in cmb:
    if s[0] in '2468' and s[-1] not in '18' and s.count('3') <= 1:
        k += 1
print(k)
```

Решать программой - удобно

- №8. Комбинаторика - но написать программу может оказаться быстрее и надежнее.

Вася составляет слова из букв слова АТТЕСТАТ. Код должен состоять из 8 букв, и каждая буква в нём должна встречаться столько же раз, сколько в заданном слове. Кроме того, в коде должны стоять рядом две гласные или две согласные буквы. Сколько различных слов может составить Вася?

```
from itertools import permutations

cmb = ["".join(p) for p in permutations('АТТЕСТАТ')]
cmb = list(set(cmb))      # убираем одинаковые слова
ans = []
for x in cmb:
    if ("АА" in x or "АЕ" in x or "ЕА" in x) or \
        ("ТТ" in x or "ТС" in x or "СТ" in x):
        ans.append(x)
print(len(ans))
```

Решать программой - удобно

- **№8.** Слова в лексикографическом порядке. Можно решать через системы счисления «вручную», а можно опять программой.

Маша составляет коды из букв, входящих в слово ЛЕОНАРД. Каждая буква должна входить в код ровно один раз. Все возможные коды Маша записывает в алфавитном порядке и нумерует. Начало списка выглядит так:

1. АДЕЛНОР
2. АДЕЛНРО
3. АДЕЛОНР

*...
Какой код будет записан под номером 4321?*

```
from itertools import permutations

cmb = ["".join(p) for p in permutations('ЛЕОНАРД')]
cmb.sort()
print(cmb[4321-1])
```

Программа может помочь

- №12. Команды исполнителя Редактор легко моделируются в Python:

нашлось (v) - v in s

заменить (v, w) - s = s.replace(v, w, 1)

- Для многих задач этого типа достаточно просто написать аналогичную программу
- Но! Часть таких задач проще решить рассуждениями, увидев закономерности в преобразованиях.

Программа может помочь

- №12. Моделируем Редактора в Python.

Дана программа для исполнителя Редактор. Программу применили к строке, состоящей из одного нуля и 45 стоящих справа от него единиц. Какое количество единиц будет в полученной строке?

```
НАЧАЛО
ПОКА нашлось(0) ИЛИ нашлось(01)
    ЕСЛИ нашлось(01)
        ТО заменить(01, 10)
        ИНАЧЕ заменить(0, 111)
    КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
КОНЕЦ
```

Шаблон программы:

```
s = "0" + "1"*45
while '0' in s or '01' in s:
    if '01' in s:
        s = s.replace('01', '10', 1)
    else:
        s = s.replace('0', '111', 1)
print(s)
print(s.count('1'))
```


Программа может помочь

- **№15.** Задачи этого типа на делимость, неравенства, поразрядную конъюнкцию легко программируются.
- Проверяем перебором в цикле не все, но достаточное количество целых точек для каждого значения параметра. Если условие ни разу не нарушилось – считаем, что такое значение параметра подходит.
- Задачи на отрезки быстрее решить вручную – но нужно уметь преобразовывать логические выражения.

Программа может помочь

- **№15.** Задачи этого типа на делимость, неравенства, поразрядную конъюнкцию легко программируются.

Для какого наибольшего натурального числа A формула $\neg \text{ДЕЛ}(x, A) \rightarrow (\text{ДЕЛ}(x, 6) \rightarrow \neg \text{ДЕЛ}(x, 9))$ тождественно истинна?

```
def DEL(x, d):  
    return x % d == 0  
  
def f(x, A):  
    return (not DEL(x, A)) <= (DEL(x, 6) <= (not DEL(x, 9)))  
  
for A in range(1, 1000):  
    flag = True  
    for x in range(1, 1000):  
        if not f(x, A):  
            flag = False  
            break  
    if flag:  
        print(A)
```

Программа может помочь

- **№16.** Последние версии задач на рекурсию не нужно, даже вредно программировать. Их цель – проверить понимание сути рекурсивных вызовов.

Программа не нужна:

$F(n) = 3$, если $n < 3$, $F(n) = 2n + 5 + F(n-2)$, если $n \geq 3$.

Чему равно значение выражения $F(3027) - F(3023)$?

- Для ряда задач этого типа можно попробовать написать по заданию рекурсивную функцию, но если объем вычислений велик, то нужно применять методы **динамического программирования**.
- Кэширования функции может быть недостаточно при большой глубине рекурсии, выход - заменять рекурсию вычислением в цикле.

В электронных таблицах

- **№3.** Поиск информации в связанных таблицах можно делать с помощью Фильтра. В сложных случаях помогает функция ВПР.

Важно: не стоит проводить суммирование по фильтрованной таблице, лучше по ее копии – тогда в подсчете точно не будут задействованы значения из свернутых строк.

- **№18.** При обычном копировании ячеек заменяется и оформление – стираются «стены». Выгодно использовать операцию «Вставить как...» – «Только формулы»
- **№22.** В таблицах удобно изобразить и изменять диаграмму Ганта для параллельных процессов, если уменьшить ширину ячеек и использовать заливку.

В таблицах или программой?

- **№9.** Приложен файл в формате таблиц. Но данные из него можно скопировать в текстовый файл и обрабатывать программой (считав как список списков).
- **№26.** Дан файл в формате .txt для обработки в программе. Но его можно открыть в электронной таблице и обрабатывать данные там, если позволяет условие задачи – это будет более наглядно.

Программируем

- №5, №14. Требуется работать с системами счисления.
- Для перевода из любой системы в десятичную можно использовать `int`, задавая второй параметр – основание исходной системы.

```
int('11001', 2)    или    int('9871', 15)
```

Перебор чисел с неизвестной цифрой 15-чной системы:

```
for x in '0123456789ABDCE':  
    a = '9897' + x + '21'  
    b = '12' + x + '023'  
    s = int(a, 15) + int(b, 15)  
    if s % 14 == 0:  
        print(x, s//x)  
        break
```

Программируем

- №5, №14. Требуется работать с системами счисления.
- Для перевода из десятичной системы в любую другую используем классический алгоритм.

```
n = 25
d = 2
s = ''
while n > 0:
    a = n % d
    s = str(a) + s
    n = n // d
print(s)
```

- Для систем 2, 8, 16 – можно встроенные функции `bin`, `oct` и `hex`, но не забываем убрать префикс

`bin(25)` это `'0b11001'`

`bin(25)[2:]` это `'11001'`

Программируем

- **№25**. Задание подразумевает оптимизацию алгоритма. Простой перебор выполняется слишком долго.
- В заданиях с масками можно использовать **срезы строк**, а можно использовать специальную библиотеку **fnmatch**. При этом срезы будут работать быстрее.

Программируем

- **№25.** Задание подразумевает оптимизацию алгоритма. Простой перебор выполняется слишком долго.

Пример: *Найти все числа, меньшие 10^8 , соответствующие маске $12*34?5$ и делящиеся без остатка на 2025.*

Главное – перебирать числа **с шагом** 2025. Без этого объем вычислений будет слишком большим.

Способ 1

```
for i in range(0, 10**8, 2025):
    x = str(i)
    if x[0:2]=="12" and x[-1]=="5" and \
        x[-4:-2]=="34":
        print(i, i // 2025)
```

Способ 2

```
from fnmatch import fnmatch

for i in range(0, 10**8, 2025):
    x = str(i)
    if fnmatch(x, "12*34?5"):
        print(i, i // 2025)
```

Успешной сдачи ЕГЭ вашим ученикам!

