# Методика решения задач высокой сложности на ЕГЭ 2024

Заболотнова Елена Юрьевна

24.11.23

# Задания высокого уровня сложности (5)

- Задание 24 - Умение создавать собственные программы (10–20 строк) для обработки символьной информации
- Задание 25 - Умение создавать собственные программы (10–20 строк) для обработки целочисленной информации
- Задание 26- Умение обрабатывать целочисленную информацию с использованием сортировки
- Задание 27 - Умение создавать собственные программы (20–40 строк) для анализа числовых последовательностей

# Система оценивания

Правильное выполнение каждого из заданий 1–25 оценивается 1 баллом.

Задание считается выполненным верно, если ответ записан в той форме, которая указана в инструкции по выполнению задания, и полностью совпадает с эталоном ответа.

За верный ответ на каждое из заданий 26 и 27 выставляется 2 балла.

Если числа в ячейках таблицы перепутаны местами ИЛИ в ячейках таблицы присутствует только одно верное число(второе неверно или отсутствует), ставится 1 балл. В остальных случаях – 0 баллов.

# Процент выполнения заданий 24-27 в Калининградской области в 2023 году

- 24 задание – 16 %
- 25 задание - 50 %
- 26 задание - 8%
- 27 задание – 7%

# Задание 24 (демоверсия)

Текстовый файл состоит из символов *T, U, V, W, X, Y* и *Z*.

Определите в прилагаемом файле максимальное количество идущих подряд символов (длину непрерывной подпоследовательности), среди которых символ *T* встречается ровно 100 раз.

Для выполнения этого задания следует написать программу.

# Решение 1

- Сначала в отдельном списке сохраняются значения индексов буквы T, чтобы обработать строки в начале и конце исходной строки в начало и конец исходной строки дописывается буква T.
- А затем результат определяется как максимальная разность индексов, между которыми находятся сто  минус одно значение.

# Текст программы

24.py - C:/Users/Elena/Documents/ЕГЭ_отчеты/САО ЕГЭ 2023/САО_информатика_2023/ИНФ/24.py (3.6.5)

File Edit Format Run Options Window Help

```
f=open('301_24.txt')
s=f.readline().strip()
s='T'+s+'T'
mi=[]
n=0
#  заполнение массива индексами T
for i in range(len(s)):
    if s[i]=='T': mi.append(i)
#  Определение максимальной длины отрезка  со 100 T
for j in range(100+1,len(mi)):
    l=mi[j]-mi[j-100-1]-1
    n=max(n,l)
print (n)
```

# Задание 25 (демоверсия)

Назовём маской числа последовательность цифр, в которой также могут встречаться следующие символы:

– символ «?» означает ровно одну произвольную цифру;

– символ «*» означает любую последовательность цифр произвольной длины; в том числе «*» может задавать и пустую последовательность.

Среди натуральных чисел, не превышающих $10^{10}$, найдите все числа, соответствующие маске 1?2157*4, делящиеся на 2024 без остатка.

В ответе запишите в первом столбце таблицы все найденные числа в порядке возрастания, а во втором столбце – соответствующие им результаты деления этих чисел на 2024.

# Решение 1

25-1.py - F:/Вебинар/25-1.py (3.6.5)

File Edit Format Run Options Window Help

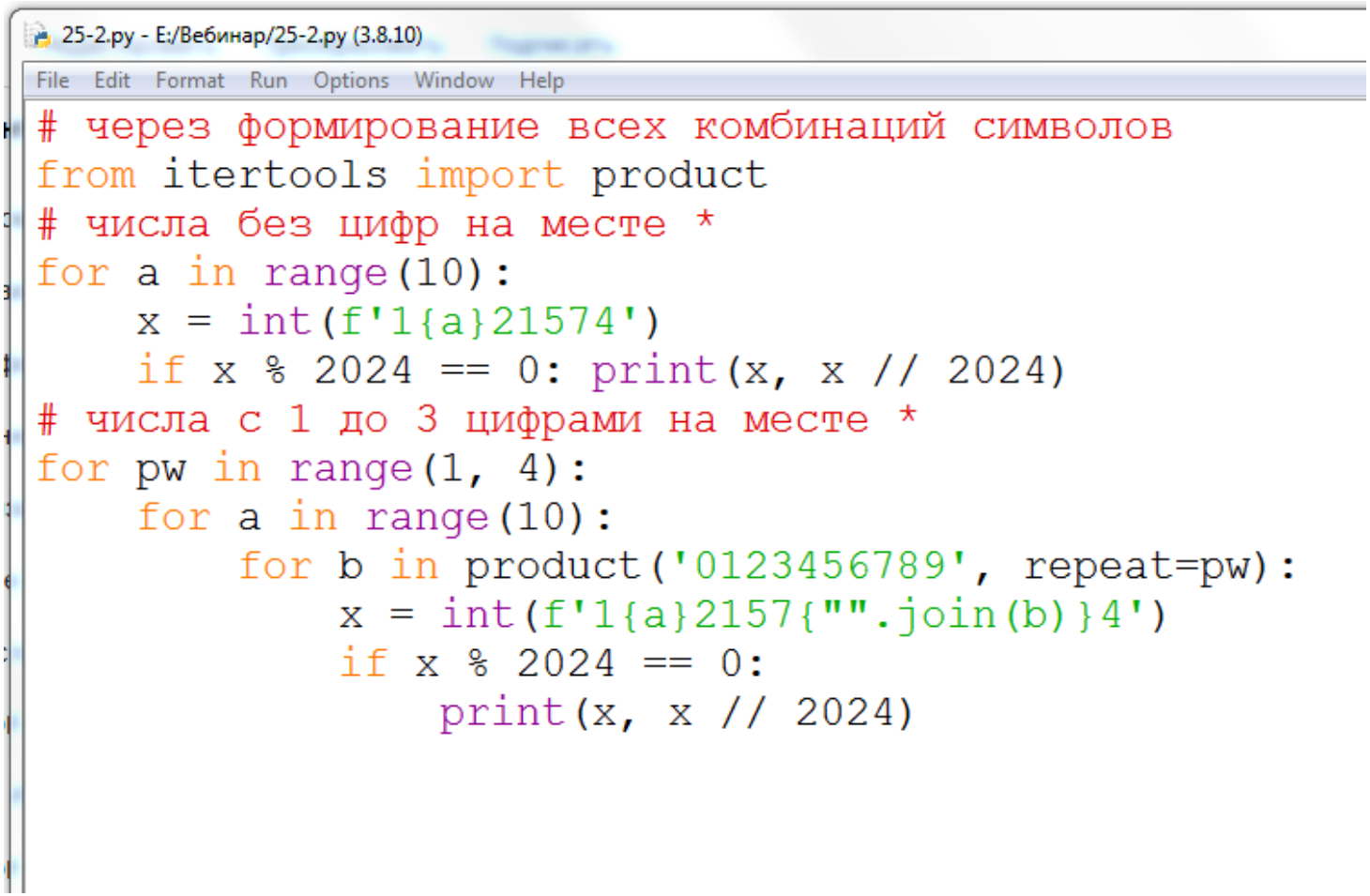
```
for x in range(2024, 10**10+1, 2024):
    s=str(x)
    if s[0]=='1' and s[-1]=='4' and s[2:6]=='2157':
        print (x,x//2024)
```

Python 3.6.5 Shell

File Edit Shell Debug Options Window Help

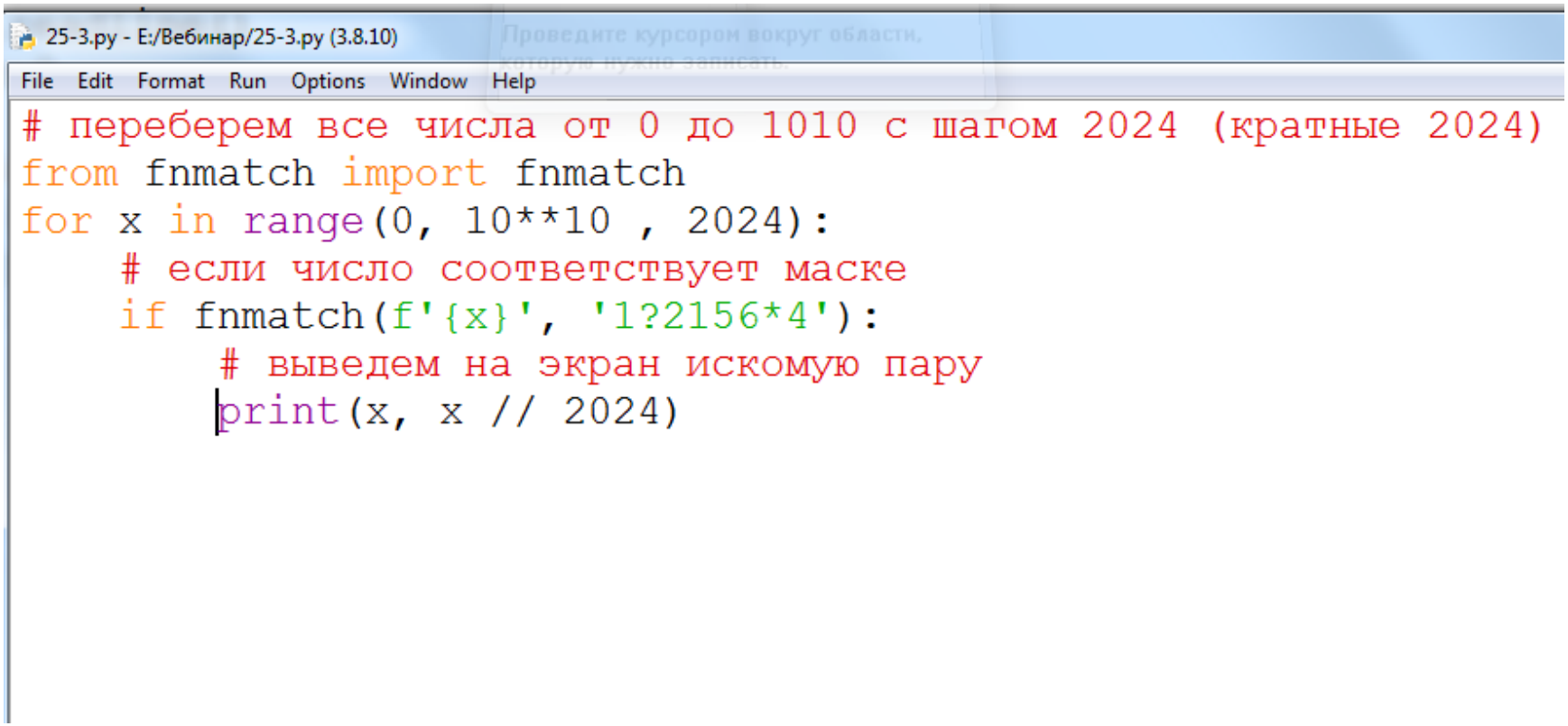
```
Python 3.6.5 (v3.6.5:f59c0
l)] on win32
Type "copyright", "credits
>>>
========================== R
142157664 70236
1021575544 504731
1121571264 554136
1221577104 603546
1321572824 652951
1421578664 702361
1521574384 751766
1621570104 801171
1721575944 850581
1821571664 899986
1921577504 949396
>>>
```

# Решение 2


25-2.py - E:/Вебинар/25-2.py (3.8.10)
File Edit Format Run Options Window Help
# через формирование всех комбинаций символов
from itertools import product
# числа без цифр на месте *
for a in range(10):
x = int(f'1{a}21574')
if x % 2024 == 0: print(x, x // 2024)
# числа с 1 до 3 цифрами на месте *
for pw in range(1, 4):
for a in range(10):
for b in product('0123456789', repeat=pw):
x = int(f'1{a}2157{"".join(b)}4')
if x % 2024 == 0:
print(x, x // 2024)

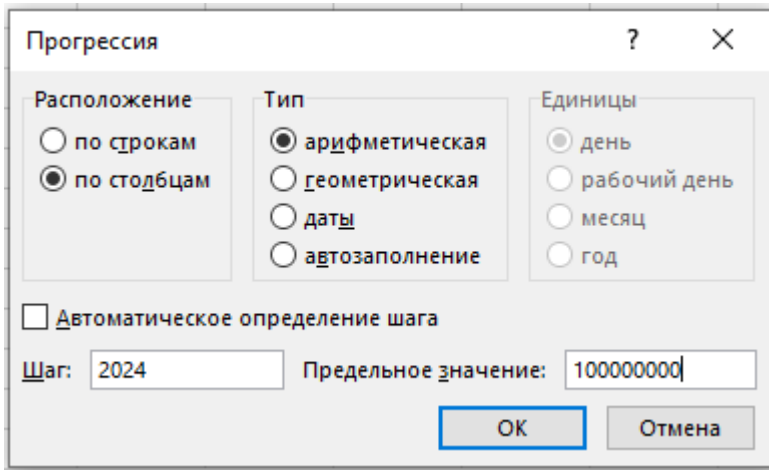
# Решение 3

[Функция fnmatch() модуля fnmatch в Python]
Функция `fnmatch()` модуля `fnmatch` проверяет, соответствует ли строка имени файла шаблонной строке, возвращая `True` или `False`.



# Решение 4, электронные таблицы

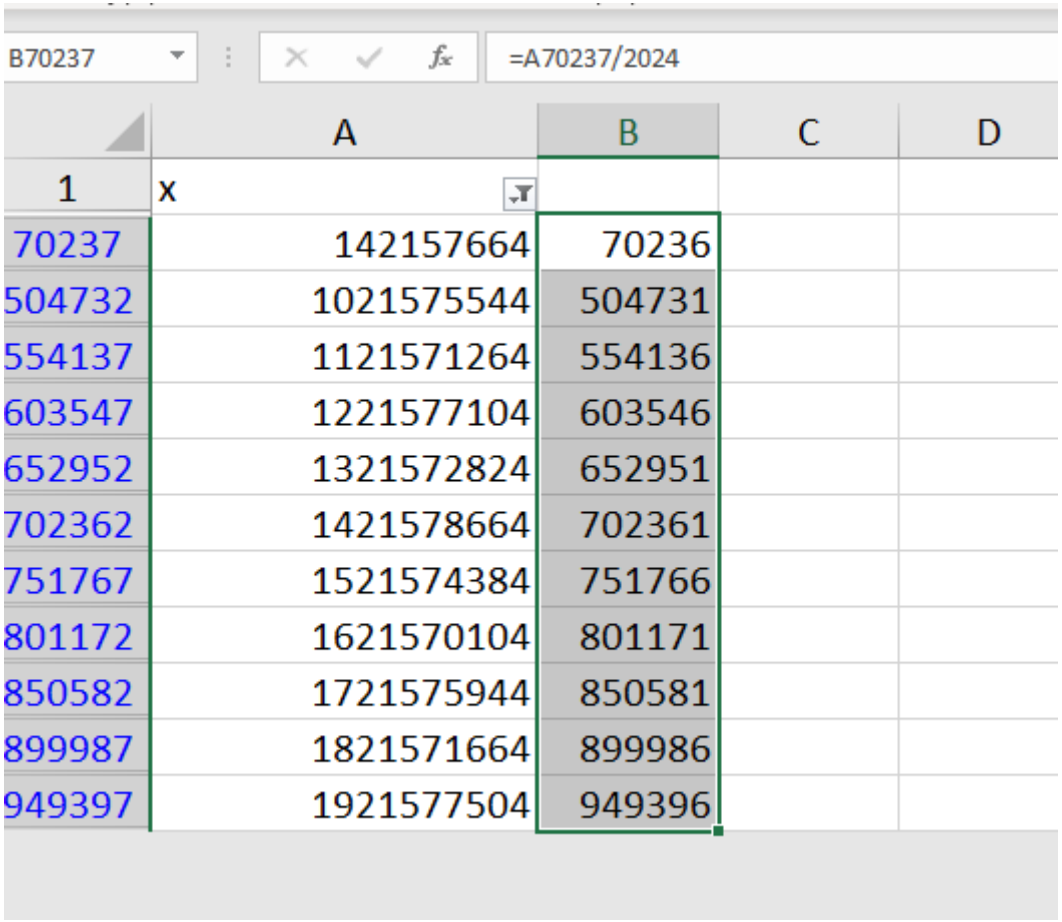
- В качестве первого числа последовательности указываем число 2024.
- Далее выбираем команды Заполнение-Прогрессия, в окне определяем тип прогрессии, шаг, предельное значение и расположение чисел прогрессии.



# Решение электронными таблицами

- Для отбора чисел, удовлетворяющих маске используем фильтры и вводим в строку Поиск, указанную в условии маску 1?2157*4
- Потом во втором столбце находим частное от деления на 2024.

# Результат



B70237 | =A70237/2024

| | A | B | C | D |
|---|---|---|---|---|
| 1 | x | | | |
| 70237 | 142157664 | 70236 | | |
| 504732 | 1021575544 | 504731 | | |
| 554137 | 1121571264 | 554136 | | |
| 603547 | 1221577104 | 603546 | | |
| 652952 | 1321572824 | 652951 | | |
| 702362 | 1421578664 | 702361 | | |
| 751767 | 1521574384 | 751766 | | |
| 801172 | 1621570104 | 801171 | | |
| 850582 | 1721575944 | 850581 | | |
| 899987 | 1821571664 | 899986 | | |
| 949397 | 1921577504 | 949396 | | |

# Задание 26 (демоверсия)

Входной файл содержит сведения о заявках на проведение мероприятий в конференц-зале. В каждой заявке указаны время начала и время окончания мероприятия (в минутах от начала суток). Если время начала одного мероприятия меньше времени окончания другого, то провести можно только одно из них. Если время окончания одного мероприятия совпадает со временем начала другого, то провести можно оба. Определите, какое максимальное количество мероприятий можно провести в конференц-зале и каков при этом максимально возможный перерыв между двумя последними мероприятиями.

# Алгоритм решения задачи

- Выгоднее провести мероприятие, которое закончится раньше всех, так как после него возможно будет провести большее количество мероприятий среди оставшихся. Следующее мероприятие находим по аналогичному рассуждению.
- Поэтому перед обработкой списка мероприятий отсортируем их по времени их окончания.
- Изначально в списке будет находиться первый элемент отсортированного списка.
- новые список, в который будем записывать первые, встреченные в отсортированном списке, мероприятия, начинающиеся позже последнего добавленного в список мероприятия.

# Решение 1 программа

26-2-1.py - E:\Вебинар\26-2-1.py (3.8.10)

File Edit Format Run Options Window Help

```
# считываем данные
with open('26.txt') as f:
    n = int(f.readline())
    events = []
    for _ in range(n):
        st, fn = map(int, f.readline().split())
        events.append([st, fn])
events.sort(key=lambda x: x[1])
# накапливаем список мероприятий
res = [events[0][1]]
for st, fn in events:
    if st >= res[-1][0]:
        res.append(fn)
# находим начало самого позднего мероприятия
mx_st = max(events)[0]
# разницу находим, как разность окончания предпоследнего
# и начала самого позднего мероприятия
print(len(res), mx_st - res[-2])
```

# Решение 2 Электронные таблицы

- Отсортируем данные по значению окончания мероприятия. В качестве окончания первого мероприятия определим значение в первой строке.
- Если в одной из следующих строк находится значение начала мероприятия больше завершения последнего, то в столбец С определяем значение его завершения.

  C2 =ЕСЛИ(A1>C1; B2; C1)

- Для подсчета количества мероприятий найдем все строки, значения в которых в столбце С сменяются. В этих строках увеличим счетчик на 1, начальное значение в первой строке зададим равным 1.

  D2 =ЕСЛИ(C1<>C2;D1+1;D1)

- Количество мероприятий будет равно максимальному значению в столбце D.

# Решение 2 (продолжение)

F1 =МАКС(D:D)

- Найдем время завершения предпоследнего мероприятия. Для этого найдем соответствующую строку.

  E2 =ЕСЛИ(И(D2=$F$1-1;C2<>C1);B2; "")

- И вычтем из максимального значения в столбце А значение в столбце Е (максимальное или минимальное, не важно, оно одно).

  F2 =МАКС(A:A) – МАКС(E:E)

H9

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | начало | конец | | | | | |
| 2 | 596 | 600 | 600 | 1 | | | |
| 3 | 575 | 601 | 600 | 1 | | | |
| 4 | 506 | 601 | 600 | 1 | | | |
| 5 | 438 | 603 | 600 | 1 | | 15 | |
| 6 | 545 | 605 | 600 | 1 | | | |
| 7 | 399 | 606 | 600 | 1 | | | |
| 8 | 521 | 606 | 600 | 1 | | | |
| 9 | 168 | 609 | 600 | 1 | | | |
| 10 | 135 | 614 | 600 | 1 | | | |
| 11 | 252 | 614 | 600 | 1 | | | |
| 12 | 338 | 615 | 600 | 1 | | | |
| 13 | 104 | 615 | 600 | 1 | | | |
| 14 | 347 | 615 | 600 | 1 | | | |
| 15 | 600 | 619 | 619 | 2 | | | |
| 16 | 46 | 620 | 619 | 2 | | | |
| 17 | 221 | 620 | 619 | 2 | | | |
| 18 | 603 | 621 | 619 | 2 | | | |

# Задание 27 ( демоверсия)

По каналу связи передаётся последовательность целых чисел – показания прибора. В течение *N* мин. (*N* – натуральное число) прибор ежеминутно регистрирует значение напряжения (в условных единицах) в электрической сети и передаёт его на сервер.

Определите три таких переданных числа, чтобы между моментами передачи любых двух из них прошло **не менее** *K* мин., а сумма этих трёх чисел была максимально возможной. Запишите в ответе найденную сумму.

# Вариант А

27-A.py - E:/Вебинар/27-A.py (3.8.10)

File Edit Format Run Options Window Help

```
with open('27_A.txt') as f:
    k = int(f.readline())
    n = int(f.readline())
    nums = [int(f.readline()) for _ in range(n)]
mx = float('-inf')
for i in range(n):
    for j in range(i+k, n):
        for m in range(j+k, n):
            mx = max(mx, nums[i] + nums[j] + nums[m])
print(mx)
```

# Вариант Б

27-2.py - F:/Вебинар/Десоверсия 2024_ноябрь/informatika2024/Доп_файлы/Задание 27/27-2

File Edit Format Run Options Window Help

```
f=open('27_B_2024.txt')
k=int(f.readline())
n=int(f.readline())
a=[]
for s in f:
    a.append(int(s))
maxsum,max1,max2=-10**10, -10**10,-10**10
for i in range(2*k, len(a)):
    max1=max(max1, a[i-k-k])
    max2=max(max2,a[i-k]+max1)
    maxsum=max(maxsum, a[i]+max2)
print (maxsum)
```

- Спасибо за внимание!