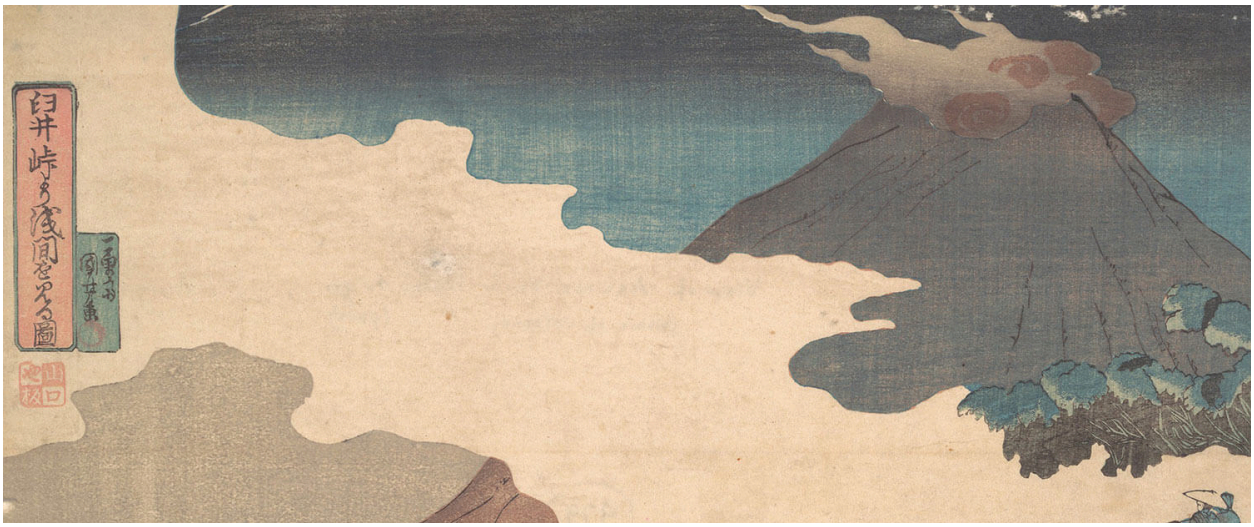

# Задание №25, Делители и Маски

## Теория

В 25 задании есть 2 основных типа заданий: на маски и на делители. Решаются они абсолютно по-разному.

**Делитель** целого числа **n** — это такое число, на которое можно поделить **n** без остатка.

**Маска** числа — это шаблон, который определяет, какие цифры или символы допускаются в числовой последовательности

*Демовариант 2025 года:*

**25** Пусть $M$ – сумма минимального и максимального натуральных делителей целого числа, не считая единицы и самого числа. Если таких делителей у числа нет, то считаем значение $M$ равным нулю.
Напишите программу, которая перебирает целые числа, бóльшие 800 000, в порядке возрастания и ищет среди них такие, для которых $M$ оканчивается на 4. В ответе запишите в первом столбце таблицы первые пять найденных чисел в порядке возрастания, а во втором столбце – соответствующие им значения $M$.

*Например,* для числа 20 $M = 2 + 10 = 12$.

Количество строк в таблице для ответа избыточно.

Ответ:

| | |
|---|---|
| … | … |
| | |

**ИЛИ**

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.

*Например,* маске 123*4?5 соответствуют числа 123405 и 12300405.

Среди натуральных чисел, не превышающих $10^{10}$, найдите все числа, соответствующие маске 3?12?14*5, делящиеся на 1917 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 1917.
Количество строк в таблице для ответа избыточно.

Ответ:

| | |
|---|---|
| … | … |
| | |

# План решения

## Делители

Пусть $M$ – сумма минимального и максимального натуральных делителей целого числа, не считая единицы и самого числа. Если таких делителей у числа нет, то считаем значение $M$ равным нулю.
Напишите программу, которая перебирает целые числа, бóльшие 800 000, в порядке возрастания и ищет среди них такие, для которых $M$ оканчивается на 4. В ответе запишите в первом столбце таблицы первые пять найденных чисел в порядке возрастания, а во втором столбце – соответствующие им значения $M$.

*Например,* для числа 20 $M = 2 + 10 = 12$.

Количество строк в таблице для ответа избыточно.

Ответ:

| | |
|---|---|
| … | … |
| | |

1. Пишем функцию нахождения делителей **dividers(n)**. Обычный перебор от 1 до **n** может быть слишком долгим, поэтому используем эффективный алгоритм поиска делителей от 1 до **√n**, использующий свойство: если **d** — делитель числа **n**, то число **n/d** тоже является делителем числа **n**. Так как в задании не учитываются делители, равные единице и самому числу, будем искать делители, начиная с 2. Чтобы для чисел, являющихся квадратами какого-либо числа, не записывался дважды один и тот же делитель, используем для хранения делителей **set()**, который хранит только уникальные элементы. Для удобства возвращаем отсортированное множество.

```
def dividers(n):
    dividers = set()
    for d in range(2, int(n**0.5) + 1):
        if n % d == 0:
            dividers.add(d)
            dividers.add(n//d)
    return sorted(dividers)
```

2. Пишем цикл перебора чисел от 800 000 (из условия) до достаточно большого числа — например, до 10^10. В этом же цикле проверяем условие для **M**, равного сумме минимального и максимального делителей числа. Проверку на то, что **M** оканчивается на 4, можно реализовать как через «**srt(M)[-1] == '4'**», так и через «**M % 10 == 4**». Т.к. нам нужно только первые 5 подходящих чисел, то также вводим **cnt_answers**, подсчитывающий количество выведенных ответов и останавливаем цикл, когда их количество становится равным 5.

```
cnt_answers = 0
for n in range(800000,10**100):
    divs = dividers(n)
    if not divs: #если нет делителей кроме 1 и самого числа
        continue #пропускаем это n и берём следующее
    M = min(divs) + max(divs)
    if M % 10 == 4:
        print(n, M)
        cnt_answers += 1
    if cnt_answers == 5:
        break
```

3. Запускаем код и получаем 5 пар значений **n** и **M**, которые и будут являться ответом. В зависимости от конкретной задачи меняется только определение **M** и условие, которое нужно для него записать.

   800004 400004
   800009 114294
   800013 266674
   800024 400014
   800033 61554

## Маски

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.

*Например,* маске 123*4?5 соответствуют числа 123405 и 12300405.

Среди натуральных чисел, не превышающих $10^{10}$, найдите все числа, соответствующие маске 3?12?14*5, делящиеся на 1917 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 1917.
Количество строк в таблице для ответа избыточно.

## Неоптимальный алгоритм

Данный алгоритм записывается буквально в несколько строчек и решает любую задачу с реального ЕГЭ за допустимое время, но он не является эффективным, поэтому он может работать очень долго.

Заключается он в простом переборе всех чисел, делящихся на указанное в условии число, и соотнесение их с маской через функцию **fnmatch** из библиотеки **fnmatch**.

1. Импортируем всё из библиотеки **fnmatch:**

   ```
   from fnmatch import *
   ```

2. Пишем цикл перебора всех чисел, делящихся на 1917 и не превышающих 10^10. Для этого начинаем цикл с числа 1917, и третьим параметром в качестве шага итерации в **range** также указываем 1917:

   ```
   for x in range(1917, 10**10+1, 1917):
   ```

3. Если fnmatch возвращает True по совпадению числа с маской из условия, то выводим это число и его частное от 1917. Получаем ответ.

   ```
   if fnmatch(str(x), '3?12?14*5'):
       print(x, x//1917)
   ```

351261495 183235
3212614035 1675855
3412614645 1780185
3712414275 1936575
3912414885 2040905

4. Задача решена. Весь код:

```
from fnmatch import *
for x in range(1917, 10**10+1, 1917):
    if fnmatch(str(x), '3?12?14*5'):
        print(x, x//1917)
```

У данного алгоритма помимо неэффективности можно выделить ещё один недостаток: если хоть немного поменять определение знаков «?» и «*» в условии или добавить какие-то новые обозначения в маску, то передать под эти изменения проверку fnmatch не получится.

## Оптимальный алгоритм

Заключается алгоритм в переборе только подходящих под заданную маску чисел, из которых отбираются только делящиеся на нужное число. Под «?» перебираем одну десятичную цифру, под «*» перебираем всевозможные комбинации цифр с повторениями (функция **product**) длины от 0 до такой, чтобы полученное число не превышало указанное в условии.

1. Импортируем **product** из **itertools** и **string** для списка чисел. Создаём список результатов **res**, в который будем добавлять подходящие под ответ числа. Лучше его сделать **set()**, ведь при нескольких «*» алгоритм может перебирать одни и те же числа несколько раз.

```
from itertools import *
import string
res = set()
```

2. Пишем вложенные циклы для символов «?» в маске по описанному выше алгоритму:

```
for a1 in string.digits:
    for a2 in string.digits:
```

3. Для «*» определяем сначала максимально допустимую длину последовательности цифр. Нам нельзя превышать 10^10 по условию, поэтому максимальная длина числа может быть 10. Учитывая «?», обозначающие одну цифру, в маске «3?12?14*5» уже как минимум 8 цифр. Значит, допустимая длина для «*» — от 0 до 2 включительно:

```
        for l in 0,1,2:
            for a3 in product(string.digits, repeat=l):
```

4. **a3** приводим к строковому виду (**product** возвращает кортеж значений) через **join** и следующей строчкой получаем подходящее под маску число, подставив вместо знаков в маске соответствующие им переменные **a1**, **a2** и **a3**:

```
                a3 = ''.join(a3)
                a = int(f'3{a1}12{a2}14{a3}5')
```

5. Проверяем это число на делимость и, если оно подходит, добавляем в список результатов:

```
                if a % 1917 == 0:
                    res.add(a)
```

6. После завершения цикла перебираем добавленные числа из отсортированного списка подходящих чисел и выводим их вместе с их результатами деления на 1917:

```
for a in sorted(res):
    print(a, a//1917)
```

7. Получаем такой же ответ, как и в алгоритме выше. Весь код:

```
from itertools import *
import string
res = set()
for a1 in string.digits:
    for a2 in string.digits:
        for l in 0,1,2:
            for a3 in product(string.digits, repeat=l):
                a3 = ''.join(a3)
                a = int(f'3{a1}12{a2}14{a3}5')
                if a % 1917 == 0:
                    res.add(a)
for a in sorted(res):
    print(a, a//1917)
```

# Алгоритм решения

## Досрок 2025. Есть делитель, оканчивающийся на 7

Напишите программу, которая перебирает целые числа, бóльшие 1 125 000, в порядке возрастания и ищет среди них такие, у которых
есть натуральный делитель, оканчивающийся на цифру 7 и не равный ни самому числу, ни числу 7. В ответе запишите в первой строке таблицы первые пять найденных чисел в порядке возрастания, а во втором столбце – наименьший делитель для каждого из них, оканчивающийся цифрой 7, не равный ни самому числу, ни числу 7.

Количество строк в таблице для ответа избыточно.

1. Весь код из плана решения остаётся неизменным, кроме написания условия. Теперь вместо суммы минимального и максимального делителей нам нужно найти минимальный делитель, который оканчивается на 7, но не равен 7. Для этого создадим переменную **min_div_end_in_7** со значением по умолчанию -1 (либо какое-либо другое очевидно не подходящее под условие число). Затем перебираем все делители в цикле (важно, что **divs = dividers(n)** отсортирован по возрастанию — это гарантирует нахождение минимального делителя). Как только находим подходящий делитель, выходим из цикла. Далее проверяем, изменилось ли значение переменной с -1 — если да, значит мы нашли подходящий делитель. После этого выводим найденные числа.

```
    min_div_end_in_7 = -1
    for d in divs:
        if d != 7 and d % 10 == 7:
            min_div_end_in_7 = d
            break
    if min_div_end_in_7 != -1:
        print(n, min_div_end_in_7)
        cnt += 1
```

2. Получаем ответ:

```
1125003 467
1125006 97
1125009 17
1125011 3187
1125012 177
```

3. Весь код:

```
def dividers(n):
    dividers = set()
    for x in range(2, int(n**0.5) + 1):
        if n % x == 0:
            dividers.add(x)
            dividers.add(n//x)
    return sorted(dividers)
cnt = 0
for n in range(1125000, 10 ** 100):
    divs = dividers(n)
    if not divs:
        continue
    min_div_end_in_7 = -1
    for d in divs:
        if d != 7 and d % 10 == 7:
            min_div_end_in_7 = d
            break
```

```
    if min_div_end_in_7 != -1:
        print(n, min_div_end_in_7)
        cnt += 1
    if cnt == 5:
        break
```

# ЕГКР 19.04.2025, 2 звёздочки

Назовём маской числа последовательность цифр, в которой также могут встречаться следующие символы:
- символ «?» означает ровно одну произвольную цифру;
- символ «*» означает любую последовательность произвольной длины; в том числе «*» может задавать и пустую последовательность.
Например, маске 123*4?5 соответствуют числа 123405 и 12300405.
Среди натуральных чисел, не превышающих $10^{10}$, найдите все числа, соответствующие маске 4*4736*1, которые делятся на 7993 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 7993.
Количество строк в таблице для ответа избыточно.

Единственная сложность этой задачи заключается в том, что чем больше длина для первого знака «*» в маске, тем меньше должна быть длина второго знака «*», чтобы не превысить лимит. В алгоритм аналогичен плану решения.

```
import string
from itertools import product
res = set()
for l1 in range(5):
    for a1 in product(string.digits, repeat=l1):
        for l2 in range(5-l1):
            for a2 in product(string.digits, repeat=l2):
                a1 = ''.join(a1)
                a2 = ''.join(a2)
                a = int(f'4{a1}4736{a2}1')
                if a % 7993 == 0:
                    res.add(a)
for r in sorted(res):
    print(r,r//7993)
```

Неоптимальным алгоритмом задача решается без каких либо проблем:

```
from fnmatch import fnmatch
for a in range(7993,10**10,7993):
    if fnmatch(str(a), '4*4736*1'):
        print(a, a//7993)
```