Конспект 3

Strim_25_3

Простые числа — это такие натуральные числа, которые имеют ровно два различных натуральных делителя: единицу и само число. Таким образом, простыми числами являются те числа, для которых существует всего два делителя. Важно отметить, что единица не является простым числом, поскольку у неё лишь один делитель — сама единица. Простые числа начинаются с двойки: 2, 3, 5, 7, 11, 13, 17 и т.д. Исходя из данного определения, можно легко определить, является ли число простым: достаточно проверить количество его делителей. Если их ровно два, то число является простым.

Задача № 1 (2583)

Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [7178551; 7178659], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241283?t=0h1m5s

Решение

Для нахождения простых чисел необходимо разработать функцию для поиска делителей. Эта функция должна перебирать числа от 1 до квадратного корня из исследуемого числа x, включая целые значения. В процессе перебора будут найдены все возможные пары делителей, а также, при необходимости, сам квадратный корень, если он является делителем. Для хранения всех делителей используется множество d.
Алгоритм следующий:

- Перебираются числа в диапазоне от 1 до целого значения квадратного корня из x, включительно.
- Если остаток от деления x на перебираемое число равен нулю, то оба делителя добавляются в множество.
- После завершения цикла возвращается отсортированный список делителей.

Далее выполняется перебор чисел в заданном диапазоне от 7178551 до 7178659 включительно. Для каждого числа выводится его значение и список его делителей. Простое число определяется наличием ровно двух делителей единицы и самого числа. Таким образом, формируется список простых чисел в указанном диапазоне.

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

```
k = 0
for x in range(7_178_551,7_178_660):
    if len(div(x))==2:
        k += 1
        print(k, x)
```

Ответ:

```
1 7178609
2 7178617
3 7178621
4 7178623
5 7178627
6 7178653
7 7178657
8 7178659
```

Задача №2 (2837)

Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [25317; 51237], которые имеют хотя бы 6 различных простых делителей. Делители 1 и само число не учитываются. Запишите в ответе для каждого найденного числа само число и его максимальный простой делитель.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241283?t=0h9m45s

Решение

Начнем с написания функции div, которая возвращает множество всех делителей числа, за исключением 1 и самого числа. Для этого мы перебираем возможные делители от 2 до $\sqrt{x}$ включительно. Если находим делитель, добавляем его и частное от деления в множество. Используем операцию объединения множеств, которая записывается, как |= и означает что к множеству d добавляется новое множество, состоящее из двух элементов: текущего делителя i и частного от деления числа x на i. Это делается для того, чтобы учесть все делители, включая те, которые могут быть больше x. Т.е. Когда мы находим делитель i, он может быть меньше квадратного корня из числа x, но при этом частное от деления x // i также является делителем и может оказаться больше квадратного корня. Например, если x = 36 и i = 2, то x // i = 18, и оба эти значения являются делителями числа 36. Таким образом, добавляя оба значения в множество, мы гарантируем, что найдем все делители числа. Также важно, что использование множества позволяет автоматически исключить повторяющиеся элементы, что помогает избежать дублирования делителей в итоговом результате. После завершения цикла возвращаем отсортированное множество делителей.

```
def div(x):
    d = set()
    for i in range(2, int(x**0.5) + 1):
        if x % i == 0:
```

```
            #объединяем текущее множество d с новым множеством, содержащим два элемента: сам делитель i и результат целочисленного деления x // i
            d |= {i, x // i}
    return sorted(d)
```

Далее нам нужно проверить, какие из найденных делителей являются простыми. Простое число – это число, которое делится только на 1 и на само себя. Чтобы проверить, является ли делитель простым, мы вызываем функцию div и проверяем, содержит ли она какие-либо делители, отличные от 1 и самого числа. Если таких делителей нет, то текущий делитель является простым.
Мы создаем список d, содержащий только простые делители числа x, и проверяем, что длина этого списка равна или превышает 6.
Если число удовлетворяет условиям задачи, выводим его и его наибольший простой делитель, который находится последним в отсортированном списке d.

```
for x in range(25_317, 51_238):
    d = [i for i in div(x) if len(div(i))==0]
    if len(d)>=6:
        print(x,d[-1])
```

Таким образом программа последовательно перебирает все числа в заданном диапазоне, находит их делители, фильтрует простые делители и выводит числа, удовлетворяющие условию задачи, вместе с их максимальными простыми делителями.
Весь код для решения задачи:

```
def div(x):
    d = set()
    for i in range(2,int(x**0.5)+1):
        if x%i==0:
            d = d | {i,x//i}
    return sorted(d)

for x in range(25_317, 51_238):
    d = [i for i in div(x) if len(div(i))==0]
    if len(d)>=6:
        print(x,d[-1])
```

Ответ:

```
30030 13
39270 17
43890 19
46410 17
```

Задача № 3 (4116)

Обозначим через S сумму простых делителей целого числа, не считая самого числа. Если таких делителей у числа нет, то считаем значение S равным нулю. Напишите программу, которая перебирает целые числа, большие 250000 в порядке возрастания и ищет среди них такие, для которых значение S не равно нулю и кратно 17. Программа должна найти первые 5 таких чисел. Для каждого из них в отдельной строке сначала выводится само число, затем значение S. Строки выводятся в порядке возрастания найденных чисел.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241283?t=0h21m50s

Решение

Создадим функцию div, которая будет искать все делители числа x. Она работает следующим образом:

- Перебираются возможные делители от 2 до $\sqrt{x}$.
- Если i является делителем x, то добавляем как i, так и x//i в множество делителей, чтобы избежать дублирования.
- Возвращается отсортированный список делителей.

```
def div(x):
    # Множество для хранения уникальных делителей
    d = set()

    # Перебор возможных делителей от 2 до sqrt(x)
    for i in range(2, int(x ** 0.5) + 1):
        if x % i == 0:
            # Добавляем i и x//i в множество делителей
            d |= {i, x // i}

    # Сортируем и возвращаем список делителей
    return sorted(d)
```

Теперь перейдем к основной части программы. Нам нужно перебирать числа, начиная с 250001, и проверять каждое число на выполнение условий задачи:

1. Найти простые делители числа.
2. Вычислить их сумму.
3. Проверить, делится ли эта сумма на 17.
4. Вывести первые 5 подходящих чисел.

```
for x in range(250_001,251_000):
    d = [i for i in div(x) if len(div(i))==0]
    S = sum(d)
    if S!=0 and S%17==0:
        print(x,S)
```

Так выглядит программа целиком:

```
def div(x):
    # Множество для хранения уникальных делителей
    d = set()
    # Перебор возможных делителей от 2 до sqrt(x)
    for i in range(2, int(x ** 0.5) + 1):
        # Если i является делителем x
        if x % i == 0:
            # Добавляем i и x//i в множество делителей
            d |= {i, x // i}
    # Сортируем и возвращаем список делителей
```

```
    return sorted(d)
# Перебираем числа от 250001 до 250999
for x in range(250_001, 251_000):
    # Находим простые делители числа x
    d = [i for i in div(x) if len(div(i)) == 0]
     # Суммируем простые делители
    S = sum(d)
    # Проверяем условие: S не равно 0 и кратно 17
    if S != 0 and S % 17 == 0:
        # Выводим число и сумму его простых делителей
        print(x, S)
```

Результат работы программы:

```
250003 19244
250015 1649
250028 62509
250059 170
250062 663
250124 8942
250160 119
250184 2856
... ... ...
```

В ответ запишем первые пять выведенных значений.

Ответ:

```
250003 19244
250015 1649
250028 62509
250059 170
250062 663
```

Задача № 4 (7960)

Обозначим через M(N) сумму максимального и минимального числа среди простых делителей целого числа N, не считая самого числа. Если таких делителей у числа нет, то считаем значение M(N) равным нулю. Найдите первые 6 чисел, больших 23 600 000, для которых значение M(N) при делении на 213 даёт в остатке 171. В ответе запишите найденные числа в порядке возрастания, справа от каждого запишите соответствующее значение M(N).

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241283?t=0h29m15s

Решение

Создадим функцию, которая находит все делители числа, за исключением самого числа и единицы, так как единица — непростое число. Затем следует перебрать числа в диапазоне от 23600001 до 23601000 и для каждого числа определить его простые делители. После этого вычисляется значение M(N) как сумма максимального и минимального из этих делителей. Важно отметить, что если у числа есть только один простой делитель, этот делитель одновременно считается и максимальным, и минимальным. Далее проверяется, дает ли M(N) остаток 171 при делении на 213. Если да, то такое число добавляется в список результатов.

```
def div(x):
   # Множество для хранения уникальных делителей
    d = set()
    # Перебор возможных делителей от 2 до sqrt(x)
    for i in range(2, int(x ** 0.5) + 1):
       # Если i является делителем x
        if x % i == 0:
           # Добавляем i и x//i в множество делителей
            d |= {i, x // i}
             # Сортируем и возвращаем список делителей
    return sorted(d)
# Перебираем числа от 23600001 до 23600999
for x in range(23_600_001, 23_601_000):
   # Отбираем только простые делители числа x
    d = [i for i in div(x) if len(div(i)) == 0]
    # Если у числа есть хотя бы один простой делитель
    if len(d) > 0:
        # Вычисляем M как сумму максимального и минимального делителя
        M = max(d) + min(d)
         # Проверяем условие деления на 213 с остатком 171
        if M % 213 == 171:
          # Выводим число и соответствующее значение M
            print(x, M)
```

Ответ:

```
23600182 694125
23600442 28713
23600478 357585
23600570 1449
23600838 135639
23600970 29139
```

Задача №5 (4211)

Обозначим через S сумму делителей числа, не являющихся простыми, кроме единицы и самого числа. Если таких делителей у числа нет, то S равно нулю. Напишите программу, которая перебирает нечетные целые числа, меньшие 912673, в порядке убывания и ищет среди них первые 5 чисел, которые кратны S. Для каждого из найденных чисел в отдельной строке сначала выводится само число, затем значение S. Строки выводятся в порядке убывания найденных чисел.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241283?t=0h35m35s

Решение

Для начала разберемся с задачей. Требуется найти нечетные числа, меньшие 912673, которые делятся на сумму своих непростых делителей. Непростыми называются делители, которые не являются простыми числами, но при этом не равны единице и самому числу. Наша цель – найти первые пять таких чисел и вывести их вместе с соответствующей суммой делителей.
Алгоритм решения включает несколько шагов:
1. Поиск всех делителей числа. Для этого используем функцию div, которая перебирает возможные делители от 2 до квадратного корня из числа. Если число делится на текущий делитель, то в множество добавляется как сам делитель, так и результат деления числа на этот делитель. Важно отметить, что используется множество, чтобы избежать дублирования делителей.
2. Фильтрация непростых делителей. Из списка всех делителей выбираются те, которые имеют больше двух делителей, то есть являются составными числами. Они сохраняются в отдельный список.
3. Суммирование непростых делителей. После фильтрации составляется сумма этих делителей.
4. Проверка условия. Если полученная сумма больше нуля и число делится на неё без остатка, то данное число соответствует условиям задачи.
Теперь перейдем непосредственно к коду:

```
def div(x):
    d = set()
    for i in range(2, int(x**0.5) + 1):
        if x % i == 0:
            d |= {i, x // i}
    return sorted(d)
 # Перебираем нечетные числа в убывающем порядке
for x in range(912_671, 100_001, -2):
   # Фильтруем непростые делители
    d = [i for i in div(x) if len(div(i)) > 0]
     # Суммируем непростые делители
    S = sum(d)
    # Проверяем условие
    if S > 0 and x % S == 0:
        # Выводим число и сумму его непростых делителей
        print(x, S)
```

Результат работы программы:

```
704969 7921
571787 6889
493039 6241
389017 5329
357911 5041
300763 4489
226981 3721
205379 3481
148877 2809
```

```
103823 2209
```

- Функция div принимает на вход число x и создает пустое множество d. В цикле от 2 до квадратного корня из x проверяется, является ли текущее число i делителем x. Если да, то в множество добавляются и i, и результат деления x на i. В итоге возвращаются все делители числа, за исключением единицы и самого числа.

· Основной цикл начинается с числа 912671 и движется вниз с шагом -2, чтобы проверять только нечетные числа. Для каждого числа вызывается функция div, после чего происходит фильтрация делителей, которые являются составными (имеют больше одного делителя). Сумма этих делителей сохраняется в переменной S.

· Если сумма S больше нуля и число x делится на неё без остатка, то выводится пара значений: само число и соответствующая ему сумма непростых делителей.

Ответ:

```
704969 7921
571787 6889
493039 6241
389017 5329
357911 5041
```

Задание №6 (5227)

Пусть N(k) = 500 000 000 + k, где k – натуральное число. Найдите пять наименьших значений k, при которых N(k) нельзя представить в виде произведения трёх натуральных чисел, больших 1. В ответе запишите найденные значения k в порядке убывания, справа от каждого значения запишите наибольший делитель N(k), не равный самому числу.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241283?t=0h45m30s

Решение

Чтобы определить, какое число невозможно представить в виде произведения трёх натуральных чисел (больше единицы), следует учитывать наличие у числа минимум трёх простых делителей. Например, число =2×3×5 соответствует этому условию, так как оно состоит из трёх различных простых чисел. Также к этой категории относится число =2×3×3 , хотя здесь одно простое число повторяется дважды.

Однако существуют числа, которые не могут быть представлены подобным образом. Примером таких чисел являются 6 (=2×3) и 7 (простое число). Эти числа имеют либо только один простой делитель, как 7, либо всего два разных простых делителя, как 6.

Для того чтобы понять, почему некоторые числа нельзя выразить через произведение трёх натуральных чисел, нужно выделить два ключевых условия:

1. Число имеет один или два уникальных простых делителя.
2. Произведение этих простых делителей должно давать само число.

Например, рассмотрим число 45. Оно разлагается на простые множители следующим образом: =3×3×5. Хотя это число содержит три простых делителя, оно не может быть представлено в виде произведения трёх натуральных чисел больше единицы, потому что произведение его простых делителей не даёт самого числа.

Согласно основной теореме арифметики, любое натуральное число можно однозначно разложить на простые множители. Чтобы выяснить, возможно ли представить данное число в виде произведения

трёх натуральных чисел, следует начать с разложения этого числа на простые множители. Если количество уникальных простых делителей меньше трёх, и их произведение равно исходному числу, тогда такое число удовлетворяет заданным условиям.

Алгоритм поиска

Для того чтобы решить задачу, необходимо проверить все числа вида N(k), начиная с k=1

1. Разложить каждое число N(k) на простые множители.
2. Проверить количество уникальных простых множителей.
3. Если их не более двух, проверить, что произведение этих множителей равно исходному числу.

```
# Функция для разложения числа на простые множители
def div(x):
   # Создаем пустой список для хранения простых множителей
    d = []
    # Начинаем проверку делимости с числа 2
    i = 2
    # Цикл продолжается, пока x больше 1
    while x > 1:
        # Если x делится на i без остатка
        while x % i == 0:
          # Добавляем i в список множителей
            d.append(i)
             # Делим x на i и обновляем x
            x = x // i
              # Переходим к следующему потенциальному делителю
        i += 1
    return d
# Основной цикл для проверки значений k
for k in range(1, 1000):
  # Рассчитываем значение N(k)
    x = 500_000_000 + k
    # Разлагаем x на простые множители
    d = div(x)
     # Выводим число x и его простые множители
    print(x, d)
```

Результат работы программы:

500000001 [3, 43, 983, 3943]

500000002 [2, 41, 41, 148721]

500000003 [500000003]

500000004 [2, 2, 3, 3, 7, 109, 109, 167]

500000005 [5, 17, 5882353]

500000006 [2, 11, 47, 79, 6121]

500000007 [3, 13, 103, 124471]

500000008 [2, 2, 2, 62500001]

500000009 [500000009]

500000010 [2, 3, 5, 19, 739, 1187]

500000011 [7, 181, 394633]

500000012 [2, 2, 125000003]

500000013 [3, 3, 3, 23, 805153]

500000014 [2, 131, 757, 2521]

500000015 [5, 643, 155521]

500000016 [2, 2, 2, 2, 3, 127, 82021]

500000017 [11, 45454547]

500000018 [2, 7, 7, 7, 37, 19699]

500000019 [3, 166666673]

500000020 [2, 2, 5, 13, 13, 29, 5101]

500000021 [1879, 266099]
……….

Ответ:

21 266099
19 166666673
17 45454547
9 1
3 1

Задание № 7(5307)

Пусть P(N) – сумма всех простых делителей числа N, а E(N) - сумма всех его чётных делителей. Обозначим M(N) = | P(N) – E(N) | (модуль разности). Найдите 5 наименьших чисел, больших 100 000 000, у которых количество простых делителей совпадает с количеством чётных делителей. В ответе запишите в первом столбце таблицы все найденные числа в порядке возрастания, а во втором столбце — соответствующие им значения M(N).
Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241283?t=1h18m0s

Решение

Чтобы решить эту задачу, необходимо осуществить последовательный перебор чисел, начиная с 100 000 001, и для каждого числа проверить выполнение условия о равенстве количества простых и чётных делителей. Это требует определения полного набора делителей данного числа, а затем разделения их на две группы: группу простых делителей и группу чётных делителей.
Алгоритм поиска можно описать следующим образом:
1. Определение функции для нахождения множества всех делителей числа. Эта функция должна возвращать полный список всех делителей заданного числа.

2. Разделение найденных делителей на две категории:
o Простые делители (то есть те, которые являются простыми числами).
o Четные делители (те, которые делятся на 2 без остатка).
3. Проверка условия совпадения количеств этих двух категорий делителей. Необходимо сравнить количество простых делителей с количеством чётных делителей. Если они равны, значит, данное число является одним из искомых.
4. Сохранение текущего числа как одного из искомых. Если условие выполнено, следует зафиксировать это число как одно из требуемых.
По завершении работы алгоритма мы получим список чисел, удовлетворяющих условиям задачи, из которого для ответа отберем первые пять.

```
def div(x):
    d = set()
    for i in range(1, int(x ** 0.5) + 1):
          if x % i == 0:
            d |= {i, x // i}
    return sorted(d)

# Перебираем числа начиная с 100 000 001 до 100 001 000
for x in range(100_000_001, 100_001_000):
    # Находим все делители числа x
    d = div(x)
    # Отбираем простые делители (те, у которых ровно два делителя)
    dp = [i for i in d if len(div(i)) == 2]
    # Отбираем чётные делители
    d0 = [i for i in d if i % 2 == 0]
    # Проверяем, совпадают ли количества простых и чётных делителей
    if len(dp) == len(d0):
        # Вычисляем M(N) как модуль разности суммы простых и чётных делителей
        print(x, abs(sum(dp) - sum(d0)))
```

Результат работы программы:

```
100000034 50000017
100000042 50000021
100000094 50000047
100000118 50000059
100000126 50000063
100000202 50000101
100000262 50000131
100000282 50000141
100000322 50000161
100000402 50000201
....
```

Ответ:

```
100000034 50000017
100000042 50000021
100000094 50000047
100000118 50000059
100000126 50000063
```

Задание №8 (5282)

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.
Найдите 7 наибольших чисел, меньших 10^7, которые кратны 217 и удовлетворяют маске 14?4*.
Выведите эти числа в порядке возрастания, справа от каждого числа выведите сумму его нечётных делителей.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241283?t=1h28m0s

Решение

Первым делом разберемся с маской. Маска 14?4* означает, что число должно начинаться с цифр 14, за которыми следует одна любая цифра, это обозначается символом ?, а затем могут следовать любые другие цифры, символ *. Например, число 14416 соответствует этой маске, поскольку начинается с 14, затем идет любая цифра 4, а дальше еще две цифры 16.
Теперь перейдем к поиску чисел, кратных 217. Начнем с числа 0 и будем двигаться с шагом 217, пока не достигнем значения 107107. Таким образом, мы будем рассматривать только те числа, которые делятся на 217 без остатка.
Когда мы нашли такое число, соответствующее нашим критериям, нужно проверить, подходит ли оно под нашу маску. Если подходит, то мы должны найти все его делители, выбрать среди них только нечётные и посчитать их сумму.

```
# Импортируем модуль fnmatch для работы с шаблонами строк
from fnmatch import *
def div(x):
    d = set()
    for i in range(1, int(x**0.5) + 1):
        if x % i == 0:
            d |= {i, x // i}
    return sorted(d)
# Начинаем цикл с 0 и шагаем с интервалом 217, чтобы проверять только те числа,
которые кратны 217
for x in range(0, 10**7, 217):
    # Проверяем соответствие числа маске '14?4*'
    if fnmatch(str(x), '14?4*'):
        # Находим все делители числа и фильтруем только нечётные
        d = [i for i in div(x) if i % 2 != 0]
        # Выводим само число и сумму его нечётных делителей
        print(x, sum(d))
```

Результат работы программы:

```
141484 41984
143437 169472
146475 317440
...
```

```
1484714 958464
1484931 2336768
1494045 3345408
1494262 964608
1494479 1806336
1494696 306432
1494913 1785088
```

Ответ:

```
1484714 958464
1484931 2336768
1494045 3345408
1494262 964608
1494479 1806336
1494696 306432
1494913 1785088
```

Задание №9(6427)

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 12300425.
Среди натуральных чисел, не превышающих 107, найдите все числа, соответствующие маске 12*4*8?, которые представляют собой произведение двух различных простых делителей, причём сумма этих делителей – простое число.
В ответе запишите все найденные числа в порядке возрастания, справа от каждого числа – сумму его простых делителей.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241283?t=1h36m0s

Решение

Для решения задачи необходимо пройти через несколько этапов:
1. Перебор всех чисел: Так как ограничений на шаг перебора нет, будем проверять каждое натуральное число от 1 до 107107.
2. Проверка маски: Для каждого числа проверяется соответствие шаблону 12*4*8?. Это означает, что число должно начинаться с цифр 124 и заканчиваться цифрой 8.
3. Нахождение простых делителей: Для числа, подходящего под маску, находим все его простые делители. Число должно иметь ровно два таких делителя, и их произведение должно давать само число.
4. Простота суммы делителей: Проверяем, чтобы сумма двух найденных простых делителей была простым числом.
Для реализации алгоритма воспользуемся функцией fnmatch для проверки соответствия числовой строки заданной маске, а также напишем функцию для нахождения всех делителей числа.

```
from fnmatch import *
# Функция для нахождения всех делителей числа
```

```
def div(x):
  # Используем множество для хранения уникальных делителей
    d = set()
     # Перебираем возможные делители до квадратного корня числа
    for i in range(1, int(x**0.5) + 1):
        if x % i == 0:
          # Добавляем сам делитель
            d.add(i)
            # Добавляем второй делитель (частное)
            d.add(x // i)
    return sorted(d)
# Основной цикл перебора чисел
for x in range(1, 10**7):
  # Проверяем соответствие маске
    if fnmatch(str(x), '12*4*8?'):
        # Находим простые делители числа
        d = [i for i in div(x) if len(div(i)) == 2]
        # Проверяем условия задачи
        if len(d) == 2 and d[0] * d[1] == x and len(div(d[0] + d[1])) == 2:
          # Выводим результат
            print(x, d[0] + d[1])
```

Ответ:

```
124282 62143
1204282 602143
1214182 607093
1224082 612043
1229482 614743
1244482 622243
1295482 647743
```

Задание №10(7781)

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 12300425.
Найдите все натуральные числа, принадлежащие интервалу [109; 1010], которые соответствуют маске ?*23*21 и имеют ровно пять натуральных делителей. В ответе запишите все найденные числа в порядке возрастания, справа от каждого числа запишите его третий по величине делитель.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241283?t=1h44m50s

Решение

Основная сложность задачи заключается в том, что диапазон очень большой — порядка миллиарда чисел. Перебирать каждое число вручную невозможно, поэтому необходимо использовать оптимизацию.

Ключевая фраза задачи — «ровно пять натуральных делителей». Это означает, что искомые числа должны быть квадратами простых чисел, поскольку только у таких чисел нечетное количество делителей. Таким образом, вместо перебора всех чисел из заданного диапазона, достаточно рассмотреть только квадраты натуральных чисел.
Для начала определим границы диапазона для перебора квадратов. Для этого найдем квадратный корень из 10^9 и 10^10:
√10^9≈31622.77
√10^10≈100000

Округляем полученные значения до целых чисел, чтобы убедиться, что квадраты будут находиться внутри требуемого диапазона. Получаем, что нам нужно перебирать квадраты чисел от 31623 до 100000.
Теперь реализуем программу, которая будет проверять соответствие каждого квадрата натуральному числу условиям задачи:

```
from fnmatch import *

# Функция для нахождения всех делителей числа
def div(x):
    # Используем множество для хранения уникальных делителей
    d = set()
    for i in range(1, int(x**0.5) + 1):
        if x % i == 0:
            d.add(i)
            d.add(x // i)
    return sorted(d)

# Основной цикл перебора квадратов
for i in range(31623, 100001):
    x = i ** 2
    if fnmatch(str(x), '?*23*21') and len(div(x)) == 5:
        # Выводим число и его третий по величине делитель
        print(x, div(x)[2])
```

В программе используется модуль fnmatch, который позволяет сравнивать строки с шаблонами. Для проверки количества делителей числа используем функцию div, которая находит все делители числа и возвращает их отсортированный список.
Запустив эту программу, получим два числа, удовлетворяющих всем условиям задачи.

Ответ:

```
5236114321 72361
6234839521 78961
```

Задание №11(2839)

Назовём нетривиальным делителем натурального числа его делитель, не равный единице и самому числу. Найдите все натуральные числа, принадлежащие отрезку [358633892; 535672891] и имеющие ровно три нетривиальных делителя. Для каждого найденного числа запишите в ответе само число и его наибольший нетривиальный делитель. Найденные числа расположите в порядке возрастания.
Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241283?t=1h53m10s

Решение

Чтобы решить задачу, важно понять, что требование наличия ровно трех нетривиальных делителей подразумевает наличие пяти делителей в целом, включая единицу и само число. Такое количество делителей возможно только у квадратов простых чисел, так как именно у них нечетное количество делителей.
Таким образом, вместо перебора всех чисел в указанном диапазоне, достаточно перебирать только квадраты натуральных чисел. Это значительно сокращает объем вычислений.
Первым шагом является определение границ диапазона для перебора квадратов. Для этого находим корни из граничных значений отрезка:
$\sqrt{358633892} \approx 18937.68$
$\sqrt{535672891} \approx 23143.93$
Округляя эти значения, получаем, что нужно перебирать квадраты чисел от 18938 до 23144.
Далее реализована функция для нахождения всех делителей числа, начиная с двойки, так как нас интересуют только нетривиальные делители (без учета единицы и самого числа).
Основной цикл программы перебирает квадраты чисел в указанном диапазоне и проверяет, соответствует ли количество нетривиальных делителей условию задачи. Если да, то выводится само число и его наибольший нетривиальный делитель.
Вот код программы:

```
def div(x):
    d = set()
    for i in range(2,int(x**0.5)+1):
        if x%i==0:
            d = d | {i,x//i}
    return sorted(d)

for i in range(18938, 23145):
    x = i**2
    if len(div(x))==3:
        print(x, div(x)[2])
```

Запуская этот код, мы получаем числа, соответствующие условиям задачи, в порядке возрастания, с указанием их наибольшего нетривиального делителя.

Ответ:

```
373301041 2685619
492884401 3307949
519885601 3442951
```

Задание №12(3779)

Найдите все натуральные числа, принадлежащие отрезку [78 000 000; 85 000 000], у которых ровно пять различных нечётных делителей (количество чётных делителей может быть любым). В ответе перечислите найденные числа, справа от каждого числа запишите его наибольший нечётный делитель.
Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241283?t=1h57m55s

Решение

Чтобы решить задачу, важно понимать структуру делителей числа. Любое натуральное число n можно представить в виде произведения простых чисел, где каждый простой множитель входит в степень, которая может быть равна нулю. Таким образом, число n можно записать как:
$n=2^a \cdot p_1^{b1} \cdot p_2^{b2} \cdots p_k^{bk}$,
где p_i — простые нечётные числа, а $a, b_1, b_2, \ldots, b_k$ — их соответствующие показатели степеней.
Для поиска нечётных делителей, мы можем избавиться от всех множителей 2, деля число на 2 до тех пор, пока это возможно. Тогда останется только произведение нечётных множителей:
$m=p_1^{b1} \cdot p_2^{b2} \cdots p_k^{bk}$.
Если число m имеет ровно пять различных делителей, то оно должно быть квадратом простого числа. Действительно, если m=p^4, то у него ровно пять делителей: $\{1, p, p^2, p^3, p^4\}$.
Таким образом, наша стратегия заключается в следующем:

1. Перебрать все числа в заданном диапазоне.
2. Избавиться от всех множителей 2.
3. Проверить, является ли оставшаяся часть числом вида p^4.
4. Если да, проверить, что у этой части ровно пять делителей.

```
def div(x):
    # Используем множество для хранения уникальных делителей
    d = set()
    for i in range(1, int(x**0.5) + 1):
        if x % i == 0:
            d.add(i)
            d.add(x // i)
    return sorted(d)

# Перебор всех чисел в диапазоне
for x in range(78_000_000, 85_000_001):
    # Получаем нечетную часть числа
    x1 = x
    while x1 % 2 == 0:
        x1 = x1 // 2
    # Проверяем, является ли x1 квадратом целого числа
    if int(x1**0.5)**2 == x1 and len(div(x1)) == 5:
        # Печатаем число и его наибольший нечётный делитель
        print(x, div(x1)[-1])
```

1. Функция div(x) находит все делители числа x. Она использует множество st), чтобы избежать дублирования делителей.
2. Основной цикл перебирает все числа в диапазоне от 78000000 до 84999999 включительно.
3. Внутри цикла мы избавляемся от всех множителей 2, пока это возможно, используя операцию деления нацело.
4. Затем проверяется, является ли оставшееся число квадратом целого числа. Если это так, и у этого числа ровно пять делителей, мы выводим исходное число и его наибольший нечётный делитель.

Важные моменты

· Основная теорема арифметики: каждое натуральное число можно разложить на простые множители единственным способом.
· Нечётные делители: после удаления всех множителей 2 остаются только нечётные делители.
· Квадрат простого числа: если число имеет ровно пять делителей, оно должно быть квадратом простого числа.
Ответ:

```
78074896 4879681
78675968 2401
80604484 20151121
81920000 625
84934656 81
```

Задание №13(3160)

Рассмотрим произвольное натуральное число, представим его всеми возможными способами в виде произведения двух натуральных чисел и найдём для каждого такого произведения разность сомножителей. Например, для числа 18 получим: 18 = 18*1 = 9*2 = 6*3, множество разностей содержит числа 17, 7 и 3. Подходящей будем называть пару сомножителей, разность между которыми не превышает 110. Найдите все натуральные числа, принадлежащие отрезку [1000000; 1500000], у которых есть не менее трёх подходящих пар сомножителей. В ответе перечислите найденные числа в порядке возрастания, справа от каждого запишите наибольший из всех сомножителей, образующих подходящие пары.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241283?t=2h7m25s

Решение

Для начала рассмотрим пример, который помогает понять суть задачи. Пусть дано число n=18. Его можно представить следующими способами в виде произведения двух натуральных чисел:
18=18×1=9×2=6×3
Разности между этими сомножителями равны соответственно:
17,7,3
Мы видим, что все эти разности удовлетворяют условию задачи, поскольку они не превышают 110.
Теперь перейдем к решению основной задачи. Поскольку нас интересуют только те числа, у которых есть не менее трёх подходящих пар сомножителей, мы должны проверить каждое число из заданного диапазона [1000000;1500000] на наличие таких пар.
Для этого создадим функцию div, которая будет находить все делители числа x и проверять, удовлетворяет ли каждая пара условие разности не более 110. Затем мы сохраним все подходящие делители в наборе set, чтобы избежать дублирования.
Оператор | используется для объединения двух множеств. Когда вы применяете этот оператор к двум множествам, он возвращает новое множество, которое состоит из всех уникальных элементов обоих исходных множеств.
В данном случае, когда добавляется пара делителей {i, x // i}, используется операция объединения множества d с этим множеством. Это позволяет добавлять новые элементы в множество без повторений, так как множества автоматически исключают дубликаты.

```
def div(x):
  # Множество для хранения подходящих делителей
    d = set()
     # Проходимся по возможным делителям
    for i in range(1, int(x**0.5)+1):
       # Проверяем, является ли i делителем и удовлетворяет ли условие разности
        if x % i == 0 and x // i - i <= 110:
            # Добавляем оба делителя в множество
            d = d | {i, x // i}
    return sorted(d)
```

После того как функция для нахождения подходящих делителей создана, мы можем пройтись по каждому числу в диапазоне от 1 000 000 до 1 500 000 и проверить, сколько подходящих пар оно имеет. Если таких пар не менее трёх, выводим соответствующее число вместе с максимальным делителем из подходящих пар.

```
for x in range(1_000_000, 1_500_001):
  # Находим подходящие делители
    d = div(x)
    # Проверяем, что подходящих пар не менее трёх (так как одна пара даёт два
делителя)
    if len(d) >= 6:
       # Выводим число и максимальный делитель
        print(x, d[-1])
```

Таким образом, программа находит все числа из указанного диапазона, имеющие не менее трёх подходящих пар сомножителей, и выводит их вместе с максимальными делителями из этих пар.

```
def div(x):
    d = set()
    for i in range(1, int(x**0.5)+1):
        if x%i==0 and x//i - i <= 110:
            d = d | {i,x//i}
    return sorted(d)

for x in range(1_000_000,1_500_001):
    d = div(x)
    if len(d)>=6:
        print(x, d[-1])
```

Ответ:

```
1113840 1105
1179360 1134
1208844 1148
1422720 1248
1499400 1275
```

Telegram: @fast_ege