Конспект 2

Strim_17_2

Задача № 1 (7936)

В файле 17-426.txt содержится последовательность целых чисел, не превышающих по модулю 100 000. Определите количество троек последовательности, в которых хотя бы один элемент является пятизначным числом и оканчивается на 43, а сумма квадратов элементов тройки не больше квадрата максимального элемента последовательности, являющегося пятизначным числом и оканчивающегося на 43. Гарантируется, что такой элемент в последовательности есть. В ответе запишите количество найденных троек, затем минимальную из сумм квадратов элементов таких троек. В данной задаче под тройкой подразумевается три идущих подряд элемента последовательности.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241312?t=0h3m15s

Решение

Важно учитывать, что элементы последовательности могут быть как положительными, так и отрицательными числами, не превышающими по модулю 100 000.
Для начала необходимо найти максимальный элемент последовательности, соответствующий условию: он должен быть пятизначным и заканчиваться на 43. После этого следует пройтись по всем тройкам подряд идущих элементов и проверить выполнение двух условий:
1. Наличие хотя бы одного пятизначного числа, оканчивающегося на 43.
2. Проверить, что сумма квадратов элементов тройки не превышает квадрат максимального элемента.

Проверку условия мы вынесем в отдельную функцию, поскольку она достаточно сложная и её повторение в разных местах кода сделает программу менее читабельной и поддерживаемой. Использование же отдельной функции позволяет избежать дублирования кода и упрощает чтение основной части программы. Кроме того, это улучшает структурированность кода и облегчает его тестирование и отладку.

```
# Открываем файл и читаем его содержимое, преобразуя каждую строку в целое число
a = [int(x) for x in open('17-426.txt')]
# Определяем функцию для проверки, является ли число пятизначным и оканчивается ли оно на 43
def check(x):
    # Условие для проверки: число должно быть пятизначным и оканчиваться на 43
    return 10000 <= abs(x) < 100000 and abs(x) % 100 == 43

# Находим максимальный элемент, удовлетворяющий условию (пятизначное число, оканчивающееся на 43)
m = max([x for x in a if check(x)])
```

```
# Создаем пустой список для хранения сумм квадратов подходящих троек
ans = []

# Перебираем тройки последовательных элементов
for x, y, z in zip(a, a[1:], a[2:]):
    # Проверяем, что хотя бы одно число в тройке подходит под условие,
    # и что сумма квадратов элементов тройки не превышает квадрат максимального
элемента
    if (check(x) or check(y) or check(z)) and x**2 + y**2 + z**2 <= m**2:
        # Добавляем сумму квадратов в список
        ans.append(x**2 + y**2 + z**2)

# Выводим количество подходящих троек и минимальную сумму квадратов
print(len(ans), min(ans))
```

Ответ:

92 838850571

Задача №2 (7821)

В файле 17-418.txt содержится последовательность натуральных чисел, не превышающих 10000. Определите количество троек, для которых выполняются следующие условия: – остаток от деления на 11 ровно одного числа из тройки равен остатку от деления на 5 минимального элемента последовательности, который записывается в шестеричной системе счисления как четырёхзначное число; – остаток от деления на 7 ровно одного числа из тройки равен остатку от деления на 13 минимального элемента последовательности, который записывается в девятеричной системе счисления как трёхзначное число. В ответе запишите два числа: сначала количество найденных троек, затем минимальную величину суммы элементов таких троек. В данной задаче под тройкой подразумевается три идущих подряд элемента последовательности.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241312?t=0h23m55s

Решение

Сначала читаем данные из файла 17-418.txt и преобразуем их в список целых чисел. Затем создадим функцию для преобразования числа в другую систему счисления, подобную той, что была использована нами при решении задач из задания № 14:

```
def cc(x, n):
    a = []
    while x > 0:
        a.append(x % n)
        x //= n
    return a[::-1]
```

Эта функция принимает число x и основание системы счисления n, возвращает список цифр этого числа в новой системе счисления. Нам нужны два минимальных элемента:

· Минимальный элемент, который в шестеричной системе счисления занимает 4 цифры.

· Минимальный элемент, который в девятеричной системе счисления занимает 3 цифры.
Используя функцию cc(), найдем эти элементы

```
# Четырёхзначный в шестнадцатеричной системе
m1 = min([x for x in a if len(cc(x, 6)) == 4])
m2 = min([x for x in a if len(cc(x, 9)) == 3])
```

Также можно использовать эквивалентные проверки с использованием двойных неравенств:

```
# Находим минимальный элемент, который в шестеричной системе является четырехзначным числом
m1 = min([x for x in a if 6**3 <= x < 6**4])

# Находим минимальный элемент, который в девятеричной системе является трехзначным числом
m2 = min([x for x in a if 9**2 <= x < 9**3])
```

Проверка с использованием двойных неравенств основана на том, что любое число, записанное в определенной системе счисления, может быть представлено диапазоном значений в десятичной системе.

Далее определим две функции для проверки каждого из двух условий:

```
def check1(x, y, z):
    return (x % 11 == m1 % 5) + (y % 11 == m1 % 5) + (z % 11 == m1 % 5) == 1

def check2(x, y, z):
    return (x % 7 == m2 % 13) + (y % 7 == m2 % 13) + (z % 7 == m2 % 13) == 1
```

Эти функции проверяют, что ровно одно число из тройки соответствует условию.
Проходим по всем тройкам последовательных чисел в списке и проверяем выполнение обоих условий:

```
ans = []
for p, q, r in zip(a, a[1:], a[2:]):
    if check1(p, q, r) and check2(p, q, r):
        ans.append(p + q + r)
```

Здесь zip() позволяет легко пройтись по тройкам чисел.
После завершения цикла выводим количество подходящих троек и минимальную сумму элементов такой тройки.

Полный код:

```
a = [int(x) for x in open('17-418.txt')]
def cc(x,n):
    a = []
    while x>0:
        a.append(x%n)
        x = x//n
    return a[::-1]
```

```
m1 = min([x for x in a if len(cc(x,6))==4])
m2 = min([x for x in a if len(cc(x,9))==3])
#m1 = min([x for x in a if 6**3<=x<6**4])
#m2 = min([x for x in a if 9**2<=x<9**3])
def check1(x,y,z):
    return (x%11==m1%5)+(y%11==m1%5)+(z%11==m1%5)==1
def check2(x,y,z):
    return (x%7==m2%13)+(y%7==m2%13)+(z%7==m2%13)==1
ans = []
for p,q,r in zip(a,a[1:],a[2:]):
    if check1(p,q,r) and check2(p,q,r):
        ans.append(p+q+r)
print(len(ans), min(ans))
```

Ответ:

1490 126

Задача № 3 (7233)

В файле 17-390.txt содержится последовательность целых чисел. Элементы последовательности могут принимать целые значения от –100 000 до 100 000 включительно. Определите количество троек, для которых выполняются следующие условия: – в тройке есть трёхзначные числа, но не все числа трёхзначные; – в тройке больше чисел, кратных 11, чем чисел, кратных 3; – каждый элемент тройки больше среднего арифметического всех элементов последовательности, запись которых заканчивается на 271. (Гарантируется, что в последовательности есть хотя бы один элемент, запись которого заканчивается на 271.) В ответе запишите количество найденных троек, затем – минимальную из сумм элементов таких троек. В данной задаче под тройкой подразумевается три идущих подряд элемента последовательности.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241312?t=0h45m45s

Решение

Первым делом нужно загрузить все числа из файла и сохранить их в виде списка. Затем найдем все числа, которые заканчиваются на 271. Для этого используем проверку остатка при делении на 1000. Поскольку числа могут быть как положительные, так и отрицательные, применяем модуль к числу (abs()). Далее вычислим среднее арифметическое всех чисел, оканчивающихся на 271. Для проверки каждого из условий создадим три функции. Функция check1 будет проверять, что среди трёх чисел есть хотя бы одно трёхзначное, но не все числа трёхзначные. Это достигается путём подсчёта количества трёхзначных чисел среди трёх заданных. Сумма проверок должна быть больше 0 и меньше 3. Функция check2 проверяет, что в тройке больше чисел, кратных 11, чем чисел, кратных 3. Для этого считаем количество чисел, кратных каждому из делителей, и сравниваем результаты. Функция check3 проверяет, что каждое число в тройке больше среднего арифметического всех чисел, оканчивающихся на 271. Для поиска троек проходим по списку чисел и для каждой тройки проверяем выполнение всех трёх условий. Если все условия выполнены, добавляем сумму этой тройки в список ответов.
Полный код программы:

```
# Загрузка данных из файла
```

```python
a = [int(x) for x in open('17-390.txt')]

# Поиск чисел, оканчивающихся на 271
a271 = [x for x in a if abs(x) % 1000 == 271]

# Вычисление среднего арифметического
avg = sum(a271) / len(a271)

# Функция для проверки первого условия
def check1(x, y, z):
    return 0 < (100 <= abs(x) < 1000) + (100 <= abs(y) < 1000) + (100 <= abs(z) <
1000) < 3

# Функция для проверки второго условия
def check2(x, y, z):
    k11 = (x % 11 == 0) + (y % 11 == 0) + (z % 11 == 0)
    k3 = (x % 3 == 0) + (y % 3 == 0) + (z % 3 == 0)
    return k11 > k3

# Функция для проверки третьего условия
def check3(x, y, z):
    return x > avg and y > avg and z > avg

# Поиск троек, удовлетворяющих всем условиям
ans = []
for z, o, v in zip(a, a[1:], a[2:]):
    if check1(z, o, v) and check2(z, o, v) and check3(z, o, v):
        ans.append(z + o + v)

# Вывод результата
print(len(ans), min(ans))
```

Ответ:

31 11207

Задача № 4 (4719)

В файле 17-243.txt содержится последовательность целых чисел. Элементы последовательности могут принимать целые значения от 0 до 10 000 включительно. Определите количество пар чисел, в которых ровно один из двух элементов меньше, чем сумма цифр всех чисел в файле, делящихся на 49, а другой делится на 13. В ответе запишите два числа: сначала количество найденных пар, а затем – максимальную сумму элементов таких пар. В данной задаче под парой подразумевается два идущих подряд элемента последовательности.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241312?t=0h57m15s

Решение

Сначала находим все числа, которые делятся на 49. Далее преобразуем каждое такое число в строку и суммируем его цифры.

```
# Читаем файл и преобразуем строки в числа
a = [int(x) for x in open('17-243.txt')]
# Суммируем цифры чисел, кратных 49
sm = sum([int(d) for x in a for d in str(x) if x % 49 == 0])
```

Перебираем пары чисел с помощью функции zip. Проверяем две ситуации:

- Первое число меньше суммы цифр, второе делится на 13.
- Второе число меньше суммы цифр, первое делится на 13.

Если хотя бы одно из этих условий выполнено, добавляем сумму пары в список ответов.

```
# Список для хранения сумм подходящих пар
ans = []
# Проходим по парам чисел
for x, y in zip(a, a[1:]):
    if (x < sm and y >= sm and y % 13 == 0) or (y < sm and x >= sm and x % 13 == 0):
        # Добавляем сумму пары в список
         ans.append(x + y)
```

После того как все подходящие пары найдены, выводим количество таких пар и максимальную сумму среди них.
Полный код программы:

```
a = [int(x) for x in open('17-243.txt')]

sm = sum([int(d) for x in a for d in str(x) if x%49==0])

#sm = 0
#for x in a:
#    if x%49==0:
#        for d in str(x):
#            sm += int(d)

ans = []

for x,y in zip(a,a[1:]):
    if (x<sm and y>=sm and y%13==0) or (y<sm and x>=sm and x%13==0):
        ans.append(x+y)
print(len(ans),max(ans))
```

Ответ:

299 13397

Задача №5 (4422)

В файле 17-202.txt содержится последовательность целых чисел, которые принимают значения от -10000 до 10000 включительно. Тройка идущих подряд чисел последовательности называется уникальной, если только второе из них является положительным трёхзначным числом, заканчивающимся на 12. Определите количество уникальных троек чисел, а затем – максимальную из всех сумм таких троек

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241312?t=1h13m15s

Решение

Для начала разберёмся с условием уникальности тройки чисел. В условии сказано, что тройка считается уникальной, если только второе число удовлетворяет следующим критериям:

1. Оно положительно.
2. Является трёхзначным.
3. Заканчивается на 12.

При этом важно отметить, что первые и третьи числа в тройке могут быть любыми, главное, чтобы они не были положительными трёхзначными числами, оканчивающимися на 12.
Теперь рассмотрим, как проверять каждое из чисел на соответствие условиям. Для этого удобно использовать вспомогательную функцию check, которая будет принимать число и возвращать True, если оно соответствует всем трём требованиям одновременно:

```
def check(x):
    # Проверка, что число положительное
    # Проверка, что число трёхзначное (лежит между 100 и 999)
    # Проверка, что число заканчивается на 12 (остаток при делении на 100 равен 12)
    return x > 0 and 100 <= abs(x) < 1000 and abs(x) % 100 == 12
```

Важно, что проверка на трёхзначность выполняется через использование функции abs(). Это сделано для того, чтобы избежать лишних проверок знака числа, поскольку нас интересует только диапазон значений от 100 до 999.
Далее нужно пройтись по списку чисел и для каждой тройки проверить, является ли она уникальной. Для этого используем встроенную функцию zip, которая позволяет легко перебирать три последовательных элемента списка:

1. a — исходный список чисел.
2. a[1:] — тот же список, начиная со второго элемента (то есть первый элемент отбрасывается).
3. a[2:] — тот же список, начиная с третьего элемента (первые два элемента отбрасываются).

На каждом шаге цикла происходит проверка текущего набора чисел x, y и z на выполнение условия уникальности:

```
        if not check(x) and check(y) and not check(z):
```

Здесь функция check применяется к каждому из чисел:

1. Первое число x должно не соответствовать условиям функции check.
2. Второе число y должно соответствовать условиям функции check.
3. Третье число z должно не соответствовать условиям функции check.

Если все эти условия выполнены, то текущая тройка считается уникальной, и её сумма добавляется в список ans.
Итоговый код:

```
# Читаем данные из файла
a = [int(x) for x in open('17-202.txt')]

# Список для хранения уникальных троек
ans = []

# Перебираем тройки чисел
```

```
for x, y, z in zip(a, a[1:], a[2:]):
    # Проверяем, что только второе число удовлетворяет условию
    if not check(x) and check(y) and not check(z):
        # Добавляем сумму текущей тройки в список ответов
        ans.append(x + y + z)
print(len(ans),max(ans))
```

Ответ:

2 4961

Задание №6 (7718 kompege)

В файле содержится последовательность целых неотрицательных чисел, не превышающих 10000. Определите количество пар элементов последовательности, в которых либо сумма элементов кратна 18, либо произведение элементов кратно 18. В ответе запишите два числа: сначала количество найденных пар, затем максимальную сумму элементов этих пар. В данной задаче под парой подразумевается два различных элемента последовательности.

Как перебирать всевозможные пары элементов: https://vk.com/video-205546952_456241312?t=1h26m50s

Для поиска всех возможных пар элементов последовательности необходимо использовать перебор. Важно учитывать, что под парой подразумеваются элементы с разными индексами, а не обязательно разные по значению. Существует два основных способа перебора пар:

1. Перебор по индексам:

Для этого используем вложенные циклы for, где внешний цикл проходит по всем возможным индексам массива, начиная с первого, а внутренний цикл начинается с индекса, следующего за текущим внешним индексом, и доходит до конца массива. Таким образом, каждая пара будет рассматриваться ровно один раз.

2. Использование модуля itertools:

Модуль itertools предоставляет функцию combinations(), которая автоматически генерирует все уникальные комбинации заданной длины из исходного набора данных. Для нашей задачи нужно получить все комбинации по две элемента.

Мы будем использовать второй метод, поскольку он более лаконичен и удобен.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241312?t=1h36m30s

Решение

Открываем файл и считываем каждую строку, преобразуя её содержимое в целое число.
С использованием функции combinations() из модуля itertools мы генерируем все возможные пары элементов. Для каждой пары проверяем, выполняется ли хотя бы одно из условий: делимость суммы или произведения на 18. Если пара подходит, добавляем сумму её элементов в список. После завершения перебора выводим количество подходящих пар и максимальную сумму среди них.

```
from itertools import *

a = [int(x) for x in open('17_7718.txt')]

ans = []

for x,y in combinations(a,2):
    if ((x+y)%18==0)+(x*y%18==0)==1:
        ans.append(x+y)
print(len(ans), max(ans))
```

Ответ:

120400 19971

Задание № 7(решу ЕГЭ)

В файле содержится последовательность из 10 000 целых положительных чисел. Каждое число не превышает 10 000. Определите и запишите в ответе сначала количество пар элементов последовательности, у которых сумма нечётна, а произведение делится на 3, затем максимальную из сумм элементов таких пар. В данной задаче под парой подразумевается два различных элемента последовательности. Порядок элементов в паре не важен.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241312?t=1h39m40s

Решение

Для начала стоит отметить, что данная задача требует перебора всех возможных пар чисел в последовательности. Такой подход называется квадратичным перебором, поскольку время выполнения алгоритма растёт пропорционально квадрату количества элементов в списке. Это связано с тем, что при увеличении числа элементов в 10 раз, количество пар возрастает в 100 раз. Рассмотрим пример. Для 100 чисел существует 100×99/2=4950 различных пар. При увеличении числа до 1000, количество пар становится 1000×999/2=499500, что уже значительно больше. Аналогично, для 10 000 чисел количество пар составит 10000×9999/2=49 995 000. Таким образом, хотя для Python обработка 50 миллионов пар может показаться большой задачей, она всё ещё выполнима за приемлемое время.
Для реализации решения воспользуемся модулем itertools, который предоставляет функцию combinations для генерации всех уникальных комбинаций пар элементов без учёта порядка. Само решение состоит в том, чтобы пройти через все возможные пары чисел, проверить выполнение условий (нечётная сумма и делимость произведения на 3), сохранить подходящие суммы и вывести итоговые результаты.

```
from itertools import *
# Читаем данные из файла и преобразуем их в список целых чисел
a = [int(x) for x in open('17_37350.txt')]
# Создаем пустой список для хранения сумм подходящих пар
ans = []
# Перебираем все уникальные комбинации пар элементов
for x, y in combinations(a, 2):
    # Проверяем условия: сумма должна быть нечётной, а произведение делиться на 3
    if (x + y) % 2 != 0 and x * y % 3 == 0:
```

```
        # Добавляем сумму пары в список
        ans.append(x + y)
# Выводим количество найденных пар и максимальную сумму среди них
print(len(ans), max(ans))
```

Ответ:

13931722 19993

Задание №8 (5799)

В файле 17-346.txt содержится последовательность целых чисел. Элементы последовательности могут принимать целые значения от 1 до 200 000 включительно. Определите количество троек последовательности, для которых произведение всех цифр трёх чисел не превосходит 2·109 и удовлетворяет маске «43*6*». В качестве ответа укажите количество таких троек и наибольшее произведение их цифр. В данной задаче под тройкой подразумевается три идущих подряд элемента последовательности.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241312?t=1h47m10s

Решение

Для начала прочитаем все числа из файла 17-346.txt. Далее необходимо определить количество троек последовательных элементов, для которых произведение всех цифр трёх чисел не превышает 2×10^9 и соответствует маске '43*6*'. Чтобы эффективно вычислить произведение всех цифр числа, воспользуемся модулем math и функцией prod из этого модуля, которая позволяет быстро находить произведение списка чисел. Напишем вспомогательную функцию pr, которая будет возвращать произведение цифр переданного ей числа.

```
def pr(x):
    return prod([int(d) for d in str(x)])
```

Теперь создадим функцию check, которая принимает три числа и возвращает True, если их произведение цифр удовлетворяет условиям задачи: не превышает 2×10^9 и соответствует маске '43*6*'.

```
def check(x, y, z):
    p = pr(x) * pr(y) * pr(z)
    return p <= 2 * 10 ** 9 and fnmatch(str(p), '43*6*')
```

После того как функции готовы, начнём перебирать тройки чисел в исходной последовательности. Если тройка удовлетворяет условию, добавляем её произведение цифр в список ans.
Наконец, выведем количество подходящих троек и максимальное значение произведения их цифр.

```
from fnmatch import *
from math import *

a = [int(x) for x in open('17-346.txt')]
```

```
def pr(x):
    return prod([int(d) for d in str(x)])

def check(x,y,z):
    p = pr(x) * pr(y) * pr(z)
    return p<=2*10**9 and fnmatch(str(p), '43*6*')

ans = []

for x,y,z in zip(a,a[1:],a[2:]):
    if check(x,y,z):
        ans.append(pr(x) * pr(y) * pr(z))
print(len(ans), max(ans))
```

Ответ:
10 438939648

Задание №9(4367 kompege)

В файле содержится последовательность натуральных чисел. Элементы последовательности могут принимать целые значения от 1 до 100 000 включительно. Определите количество пар последовательности, в которых оба числа делятся на минимальный элемент последовательности, кратный 8, но не равный 8. Гарантируется, что такой элемент в последовательности есть. В ответе запишите количество найденных пар, затем среди таких пар определите пару с минимальной суммой элементов и запишите максимальное из чисел в ней (если пар с минимальной суммой элементов несколько, то в ответ следует выбрать максимальное число из первой пары).
В данной задаче под парой подразумевается два идущих подряд элемента последовательности.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241312?t=1h56m10s

Решение

Основная идея решения этой задачи заключается в том, чтобы сначала найти минимальное число в последовательности, которое делится на 8, но не равно 8. После этого необходимо определить количество пар, в которых оба числа делятся на это минимальное число, и найти пару с минимальной суммой элементов, выбрав максимальное число из этой пары.
Сначала мы открываем файл и считываем все числа в список a. Далее используем генератор списка для нахождения минимального числа, которое делится на 8, но не равно 8. Затем создаём пустую структуру ans, в которую будем добавлять информацию о подходящих парах. Используя функцию zip, мы объединяем соседние элементы списка в пары и проверяем, делятся ли оба числа на найденное минимальное число. Если деление проходит успешно, мы добавляем в ans список, состоящий из суммы элементов пары и максимального из этих элементов. Наконец, выводим количество таких пар и минимальное значение из списка ans.

```
# Читаем данные из файла и преобразуем каждую строку в целое число
a = [int(x) for x in open('17_4367.txt')]

# Находим минимальное число в списке, которое делится на 8, но не равно 8
m = min([x for x in a if x % 8 == 0 and x != 8])
```

```python
# Создаем пустой список для хранения информации о парах
ans = []

# Проходим по всем парам соседних элементов в списке
for x, y in zip(a, a[1:]):
    # Проверяем, делятся ли оба числа на минимальное число, кратное 8
    if x % m == 0 and y % m == 0:
        # Добавляем в список сумму элементов пары и максимальное из них
        ans.append([x + y, max(x, y)])

# Выводим количество подходящих пар и минимальное значение из списка ans
print(len(ans), min(ans))
```

Ответ:

3 [124160, 74280]

Telegram: @fast_ege