Конспект 2

Strim_5_2

Задача № 1 ()

Автомат обрабатывает десятичное натуральное число N по следующему алгоритму:
1) Строится двоичная запись числа N.
2) К полученному числу справа дописывается 0, если в числе единиц больше, чем нулей; иначе дописывается 1.
3) Из середины двоичного числа убирается 2 разряда, если количество разрядов получилось четным, и 3 разряда, если нечетное.
4) Результат переводится в десятичную систему.
Пример. Дано число N = 11. Алгоритм работает следующим образом.
1) Двоичная запись числа N: 11 = 10112
2) Единиц больше, чем нулей, новая запись 101102.
3) Длина начётная, удаляем три средних разряда, новая запись 102.
4) Десятичное значение полученного числа 2.
Для скольких различных значений N в результате работы автомата получается число 58?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241254?t=0h2m10s

Решение

Основная задача заключается в удалении цифр из середины числа.
Ранее были рассмотрены методы добавления символов в середину числа путем деления его на две части и вставки дополнительного элемента между ними. В данном случае будет применен аналогичный подход.
Рассмотрим пример работы метода на строке чисел. Пусть имеется строка «12345678», состоящая из восьми цифр. Необходимо удалить две центральные цифры. Для начала разделим строку на две равные половины с использованием среза:
b[:len(b)//2]
Это даст нам первую половину строки («1234»). Вторая половина будет извлекаться следующим образом:
B[len(b)//2:]
Получаем правую часть строки («5678»).
Для удаления цифр по краям изменяем параметры среза таким образом, чтобы исключить одну цифру с каждой стороны. Вместо b[:len(b)//2] используем b[:len(b)//2-1], что исключит последний символ первой половины. Аналогичным образом для правой половины применяем срез b[len(b)//2+1:].
Объединение полученных частей дает результат «123678». Таким образом, удается удалить по одной цифре с каждого конца строки, при этом исключив две центральные цифры.
Теперь рассмотрим случай с нечётным количеством символов. Возьмём строку "123456789" длиной девять символов. Наша цель — удалить три центральных символа (цифры 4, 5 и 6). Разделим строку на две половины:
Левая половина: "1234". Правая половина: "56789".

Так как длина строки нечётная, правая половина оказывается длиннее левой на один символ. Чтобы решить задачу, уберём одну цифру из левой половины и две цифры из правой. Для этого воспользуемся следующими срезами:
Левую половину возьмём без последнего символа (b[:len(b)//2-1]), а правую — без двух первых символов (b[len(b)//2+2:]).
Таким образом, получаем строку "123789". Как видно, удалены три центральных цифры.
Если бы число было ещё короче, например, "12345", результатом выполнения аналогичного действия стало бы "15". При этом также удаляются три центральных цифры.
Можно попробовать другой вариант срезов, например, минус 2 слева и плюс 1 справа. Тогда результатом станет "45". Однако этот способ удаляет цифры ближе к началу строки, а не из её центра.
Поэтому важно корректно подбирать параметры срезов, чтобы добиться нужного результата.
Теперь осталось применить эти подходы в нашем решении. Начнем с цикла for n in range. Возьмем диапазон до 10000. Выполним двоичное представление числа f'{n:b}'.Затем проверим количество единиц в строке b. Если их больше, чем нулей, дописываем 0 к числу. Иначе добавляем 1. Это соответствует второму пункту задачи.
После добавления проверяем длину получившейся строки. Если она четная, удаляем две цифры из середины. Если нечетная – три цифры. Реализуем это через соответствующие срезы.

```
k = 0
for n in range(10,10000):
    b = f'{n:b}'
    if b.count('1')>b.count('0'):
        b = b + '0'
    else:
        b = b + '1'
    if len(b)%2==0:
        b = b[:len(b)//2-1] + b[len(b)//2+1:]
    else:
        b = b[:len(b)//2-1] + b[len(b)//2+2:]
    r = int(b,2)
    if r==58:
        print(n,r)
        k += 1
print(k)
```

Результат работы программы:

113 58

117 58

121 58

125 58

229 58

233 58

237 58

241 58

245 58

249 58

253 58

11

В задаче требуется количество уникальных значений n, которые дают результат 58. В выводе результата работы программы мы видим соответствующие условию задачи числа 117,121, 125 и т.д. Количество чисел можно подсчитать вручную. В результате получаем 11 чисел. Другой способ — использовать счётчик, увеличивающийся на 1 каждый раз, когда находится новое значение, что также даёт итоговый результат 11. Ещё одним вариантом решения может стать использование списка для подсчёта количества элементов. Все элементы можно собрать в список, а затем определить его длину. Этот метод тоже достаточно эффективен.

Основная особенность задачи состоит в том, что цифры удаляются не с краёв числа, а из его середины. Число разделяется на две части — левую и правую половину, причём одна часть может быть чуть меньше другой. Средние символы удаляются.

Ответ: 11

Задача №2 ()

На вход алгоритма подаётся натуральное число N. Алгоритм строит по нему новое число R следующим образом:

1. Строится двоичная запись числа N.
2. Далее эта запись обрабатывается по следующему правилу:
   а) если сумма цифр в двоичной записи числа чётная, то 4 младших бита инвертируются, т.е. 0 изменяется на 1, а 1 на 0;
   б) если сумма цифр в двоичной записи числа нечётная, то инвертируются 4 бита в двоичных разрядах 1-4 (нумерация разрядов справа налево, начиная с 0).
3. Полученная таким образом запись является двоичной записью искомого числа R.

Например, для исходного числа 125 = 11111012 результатом является число 114 = 11100102 а для исходного числа 227 = 111000112 результатом является число 253 = 111111012.
Укажите число N, большее 63, после обработки которого с помощью этого алгоритма получается минимальное число R. В ответе запишите число в десятичной системе счисления.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241254?t=0h16m20s

Решение

По условию задачи требуется найти число $n > 63$, при обработке которого получится минимальное число r . Ответ должен быть представлен в десятичной системе счисления.
Сложность задачи состоит в том, что требуется изменить определенные цифры. То есть не менять все цифры на противоположные, а лишь некоторые из них. Давайте рассмотрим пример с числом

125. Попытаемся понять, как это делается. У нас есть число 125. Из примера берем его двоичное представление: 1111101.Т.к. сумма цифр в числе четная, необходимо заменить последние четыре его цифры.
Разобьем строку на две части. Можно сделать это так: выделить конкретные две части. Первая часть – это последние четыре цифры. Как получить последние четыре цифры? Это можно сделать с использованием отрицательной индексации. Мы знаем, что индексы -1, -2, -3, -4 отсчитываются с конца строки. Если взять символы от -4 до конца, то получу именно последние четыре цифры. Например, срез b[-4:] вернет нам "1101".
Далее мы можем взять эти цифры, инвертировать их и затем заменить их на противоположные и потом заменить в оригинальной строке те, которые были.
Рассмотрим, как это будет выглядеть в коде. В качестве примера возьмем число 125. На текущий момент переменная n равна 125. Добавим двоичное представление числа b, используя форматирование строки: b = f'{n:b}'. Наша цель — изменить последние четыре цифры на противоположные. Для этого нам нужно извлечь эти последние четыре цифры из b, которые мы обозначим как b1. Вот соответствующий фрагмент кода:

```
n=125
b = f'{n:b}'
if b.count('1')%2==0:
        b1 = b[-4:]
else:
        b1 = b[-5:-1]
print(b,b1)
```

Результат работы программы:

```
1111101 1101
```

Теперь у нас есть b1, содержащий последние четыре цифры. Нам необходимо инвертировать их. Однако просто замена нулей на единицы не подходит, так как после первого изменения все цифры станут единицами, что приведет к потере данных. Поэтому мы будем действовать иначе: сначала заменим нули на двойки, чтобы сохранить исходные нули, потом изменим единицы на нули, а оставшееся значение "двойки" переведем в единицы.

```
b1 = b1.replace('0','2').replace('1','0').replace('2','1')
```

При этом последовательность 1101 сначала превратится в 1121 (заменив 0 на 2), затем 0010 (превращая 1 в 0), и наконец, 0010 (переводя 2 в 1).

```
n=125
b = f'{n:b}'
if b.count('1')%2==0:
        b1 = b[-4:]
else:
        b1 = b[-5:-1]
b1 = b1.replace('0','2').replace('1','0').replace('2','1')
print(b,b1)
```

Запустив код, вы увидите, что 1101 успешно изменится на 0010. Все цифры будут правильно инвертированы без потерь информации.

Далее необходимо произвести замену указанной части. Для этого следует извлечь исходную строку, удалить из неё последние четыре символа, оставив всё, кроме этих четырёх символов. Это достигается путём выделения всех символов строки, начиная с первого и заканчивая четвёртым символом с конца. Важно отметить, что здесь присутствуют два различных элемента: первые — последние четыре символа, вторые — вся оставшаяся часть без учёта последних четырёх символов. Правый элемент представляет собой последние четыре символа, левый — всю остальную часть.

Затем к левой части присоединяется результат выполненной операции, добавляется b1. В результате формируется новая строка b, в которой последние четыре символа теперь имеют значение 0010.

Таким образом, из двоичного представления выделяются четыре символа, которые помещаются в отдельную переменную. Эти символы преобразуются в противоположные значения, после чего они заменяют исходные символы. Старые символы удаляются из числа, а на их место вставляются новые.

Таким образом, необходимо последовательно выполнить следующие действия: выделить указанный фрагмент, отдельно изменить его, и затем заменить старый фрагмент новым.

Процесс может быть упрощен, если вместо использования строки применить список. Какова разница между строкой и списком?

Список позволяет изменять отдельные элементы напрямую. То есть, в отличие от строки, где невозможно изменить отдельные символы, в списке это возможно. Таким образом, мы можем воспользоваться этим подходом. Рассмотрим второй метод. Мы снова используем двоичное представление, но обернем его в список. Мы создаем список, а не строку. Различие заключается в том, что каждый символ теперь находится отдельно в списке, а не в строке.

Преимущество этого подхода в том, что мы можем изменить любой элемент на противоположное значение. Например, заменить единицу на ноль, а ноль на единицу. Это можно выразить следующим образом:

```
n = 125
b = list(f'{n:b}')
if b[-4]=='1':
    b[-4] = '0'
else:
    b[-4] = '1'
print(b)
```

Результат:

```
['1', '1', '1', '0', '1', '0', '1']
```

После запуска программа корректно выполняет изменения без ошибок, благодаря использованию списка.

Всё условие можно записать в одну строку.

```
n = 125
b = list(f'{n:b}')
b[-4] = '1' if b[-4]=='0' else '0'
##if b[-4]=='1':
##    b[-4] = '0'
##else:
##    b[-4] = '1'
print(b)
```

Такая форма записи носит название **тернарного оператора**. Она позволяет компактно представлять условные конструкции. Основная идея заключается в том, что при выполнении указанного условия b[i] == '0' переменной присваивается первое значение '1', в противном случае – второе '0'. Эти два способа выражения имеют идентичный смысл, однако использование тернарной формы делает запись более лаконичной и удобной для чтения.

```
n = 125
b = list(f'{n:b}')
b[-4] = '1' if b[-4]=='0' else '0'
b[-3] = '1' if b[-3]=='0' else '0'
b[-2] = '1' if b[-2]=='0' else '0'
b[-1] = '1' if b[-1]=='0' else '0'
```

Получили все инвертированные разряды:

```
['1', '1', '1', '0', '0', '1', '0']
```

Т.к. это однотипные команды, их возможно оформить в виде цикла. И объединить измененные символы обратно в строку с помощью метода `join(b)`. Теперь у нас получилась строка, в которой последние четыре цифры были изменены.

```
n = 125
b = list(f'{n:b}')
for i in range(-4,0):
    b[i] = '1' if b[i]=='0' else '0'
b = ''.join(b)
print(b)
```

В результате получена инвертированная строка:

```
1110010
```

Запишем код для решения задачи:

```
#создаём пустой список для ответов
m = []
#т.к., по условию задачи, n > 63, переберем n от 64 до 1000
for n in range(64,1000):
    #Создаем двоичное представление числа n через f-строку '{n:b}'
    b = f'{n:b}'
    #Проверяем четность количества единиц в строке b.
    if b.count('1')%2==0:
        # Берем последние четыре символа строки b и сохраняем их в переменную b1.
        b1 = b[-4:]
       #Меняем символы в строке b1: '0' заменяется на '2', '1' на '0', а '2' на '1'.
        b1 = b1.replace('0','2').replace('1','0').replace('2','1')
        #Формируем новую строку b, объединяя первые символы (до последних четырех) с измененной строкой b1.
        b = b[:-4] + b1
        #Закомментированный код для работы со списком символов вместо строки.
        #b = list(b)
        #for i in range(-4,0):
        #    b[i] = '1' if b[i]=='0' else '0'
        #b = ''.join(b)
    else:
        b1 = b[-5:-1]
        b1 = b1.replace('0','2').replace('1','0').replace('2','1')
        b = b[:-5] + b1 + b[-1]
        #b = list(b)
        #for i in range(-5,-1):
        #    b[i] = '1' if b[i]=='0' else '0'
        #b = ''.join(b)
    # Преобразуем строку b обратно в целое число по основанию 2.
    r = int(b,2)
    #Добавляем пару [r, n] в список m.
```

```
    m.append([r,n])
print(min(m))
```

Результат работы программы:

[64, 94]

Минимальное число n, из которого получается минимальный результат это 64.

Ответ: 64

Задача № 3 ()

Алгоритм получает на вход натуральное число N и строит по нему новое число R следующим образом:

1. Строится двоичная запись числа N.
2. Если сумма цифр десятичной записи заданного числа нечётна, то в конец двоичной записи дописывается 1, если чётна - 0.
3. Пункт 2 повторяется для вновь полученных чисел ещё два раза.
4. Результатом работы алгоритма становится десятичная запись полученного числа R.

Пример. Дано число N = 17. Алгоритм работает следующим образом:

1. Строим двоичную запись: $17_{10} = 10001_2$.
2. Сумма цифр числа 17 - чётная, дописываем к двоичной записи 0, получаем $100010_2 = 34_{10}$
3. Сумма цифр числа 34 - нечётная, дописываем к двоичной записи 1, получаем $1000101_2 = 69_{10}$
4. Сумма цифр числа 69 - нечётная, дописываем к двоечной записи I, получаем $10001011_2 = 139_{10}$
5. Результат работы алгоритма R = 139.

Определите наименьшее возможное значение R> 1028, которое может получиться в результате работы алгоритма.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241254?t=0h48m55s

Решение

Проанализируем поставленную задачу. Основная цель состоит в том, чтобы просуммировать цифры десятичной записи числа, однако ключевая сложность заключается в необходимости повторять данную операцию с каждым полученным промежуточным результатом. Из приведенного примера видно, что сначала берется исходное число, находится сумма его цифр, а затем аналогичная операция проводится с новым числом.

Рассмотрим различные значения n в диапазоне от 1 до 300.

Первым шагом является построение двоичного представления числа. Далее определяется сумма цифр его десятичной записи.

На начальном этапе необходимо найти сумму цифр десятичной записи числа. Для этого число преобразуется в строку, после чего производится обход каждого символа (цифры) строки, их преобразование в целое значение и последующее суммирование.

Если итоговая сумма оказывается нечетной, к концу двоичной записи добавляется единица, если четной — ноль. Таким образом завершается первый этап процесса.

Процедура должна быть выполнена еще два раза. Следует учитывать, что суммироваться будут уже не цифры исходного числа, а цифры числа, изменившегося после добавления новой цифры. Так как само число изменяется, соответственно меняется и сумма его цифр.

Как организовать выполнение данной процедуры? Процесс циклически повторяется. Те же самые действия выполняются повторно, но теперь объектом обработки становится не n, а десятичная версия числа b. На этом этапе записывается b2 — новое двоичное число, полученное после преобразования. Оно снова преобразуется в десятичную форму, рассчитывается новая сумма цифр и добавляется очередная цифра. Процедура повторяется третий раз.
Результаты можно проверить. Выполнив заданные преобразования трижды, получаем конечный ответ. При n=17 ожидается получить 139.

```
for n in range(1,300):
    b = f'{n:b}'
    sm = sum(int(d) for d in str( int(b,2) ))
    if sm%2!=0:
        b = b + '1'
    else:
        b = b + '0'
    sm = sum(int(d) for d in str( int(b,2) ))
    if sm%2!=0:
         b = b + '1'
    else:
        b = b + '0'
    sm = sum(int(d) for d in str( int(b,2) ))
    if sm%2!=0:
        b = b + '1'
    else:
        b = b + '0'
    r = int(b,2)
    if n==17:
       print(r)
```

Результат:
139
Проведенная проверка подтверждает корректность результата.
Минимизируем запись кода, записав три повтора действий внутри цикла. Необходимо определить минимальное значение R, когда R > 1028.
Код для решения задачи:

```
m = []
for n in range(1,300):
    b = f'{n:b}'
    for i in range(3):
        #Суммируем цифры числа в двоичной системе счисления.
        sm = sum(int(d) for d in str( int(b,2) ))
        if sm%2!=0:
            b = b + '1'
        else:
            b = b + '0'
    r = int(b,2)
    if r>1028:
        m.append(r)
print(min(m))
```

Результат работы программы:

1035

Полученные значения показывают, что минимальным результатом является 1035.

Ответ: 1035

Задача № 4 ()

Алгоритм получает на вход натуральное число N> 1 и строит по нему новое число R следующим образом:

1. Строится двоичная запись числа N.
2. Вычисляется количество единиц, стоящих на чётных местах в двоичной записи числа N без ведущих нулей, и количество нулей, стоящих на нечётных местах. Места отсчитываются слева направо (от старших разрядов к младшим, начиная с единицы).
3. Результатом работы алгоритма становится модуль разности полученных двух чисел.

Пример. Дано число N= 39. Алгоритм работает следующим образом:

1. Строится двоичная запись: $39_{10} = 100111_2$.
2. На чётных местах стоят две единицы, нечётных - один ноль.
3. Модуль разности равен 1.

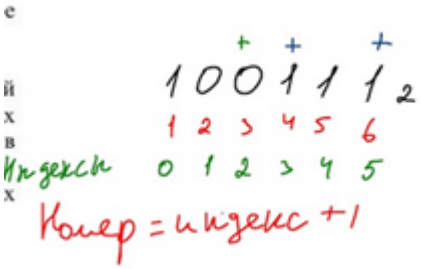
Результат работы алгоритма R = 1.
При каком наименьшем N в результате работы алгоритма получится R = 5?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952456241254?t=0h57m50s

Решение

Рассмотрим пример для N=39. Его двоичное представление равно 100111. Пронумеруем позиции слева направо: 123456.

Мы используем нумерацию слева направо, начиная с единицы. Таким образом, позиции будут пронумерованы следующим образом: 1 , 2 , 3 , 4 , 5 , 6.

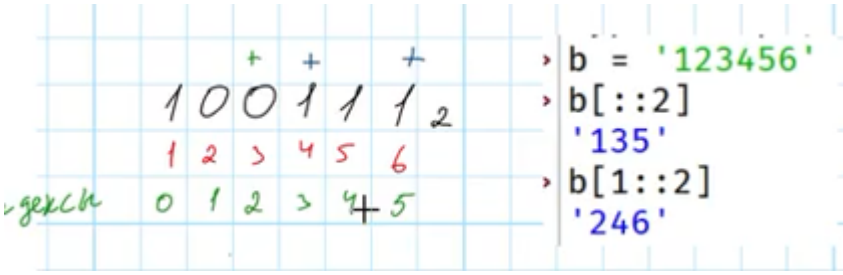


Рассмотрим, какие значения (единицы или нули) находятся на чётных и нечётных позициях. На чётных местах стоят две единицы, нечётных - один ноль.
Таким образом, у нас два случая, когда единица стоит на чётной позиции, и один случай, когда ноль стоит на нечётной позиции.
Модуль разности между количеством единиц на чётных позициях и количеством нулей на нечётных позициях составляет |2 - 1| = 1. Это означает, что разница между этими двумя значениями равна

единице. Ключевой момент заключается в том, как правильно разделить четные и нечетные позиции. Для этого нужно понимать, как работают операции со срезами (или индексированием).

Рассмотрим наш пример, полученное двоичное число 100111 с индексами 1, 2, 3, 4, 5, 6. Нам необходимо извлечь элементы с индексами 1, 3, 5, то есть через один, т.е. требуется получить элементы строки через каждую вторую позицию.
При использовании срезов обычно указывают одно двоеточие. Однако существует еще одна возможность, которая часто игнорируется. Она позволяет задать начальную, конечную позиции и шаг. Например, если нужно начинать с первого элемента и двигаться до конца с шагом 2, это означает выбор элементов через одну позицию: 1, 3, 5. Если начать с первой позиции, результат будет содержать элементы на четных позициях.

Таким образом, можно указать, что необходимы не последовательные элементы, а те, которые находятся через определенный интервал. При выборе шага 3 будут получены элементы через каждые два символа: 1, 4.



Таким образом, для выбора четных или нечетных символов используется третий параметр среза. Он позволяет извлекать элементы не по порядку, а с заданным шагом: через один, два, три и так далее, в зависимости от требований задачи.
Третий параметр среза не всегда указывается явно, однако он становится необходимым, когда требуется извлекать элементы не последовательно, а с определенным шагом.
Запишем код для проверки примера приведенного в задаче с использованием срезов:

```
for n in range(2,10000):
    b = f'{n:b}'
   #Подсчет количества единиц во всех позициях строки 'b' с нечетными индексами (начиная с первого символа).
    a1 = b[1::2].count('1')
   #Подсчет количества нулей во всех позициях строки 'b' с четными индексами (начиная с первого символа).
    a2 = b[::2].count('0')
   #Вычисление абсолютной разницы между количеством единиц в нечетных индексах и количеством нулей в четных индексах.
    r = abs(a2-a1)
    if n==39:
        print(n,r)
```

Результат:

```
39 1
```

Запишем этот код с учетом условия задачи:

```
for n in range(2,10000):
    b = f'{n:b}'
```

```
    a1 = b[1::2].count('1')
    a2 = b[::2].count('0')
    r = abs(a2-a1)
    if r==5:
        print(n,r)
        break
```

Результат:
1023 5
Решение задачи может быть получено и без использования срезов. Но следует учитывать, что индексация элементов массива начинается с нуля, однако нумерация позиций в условии задачи идет с единицы. Т.е., для определения номера позиции необходимо к индексу добавить единицу. Анализируем следующую ситуацию: если выражение i + 1 (где i – текущий индекс) представляет собой четную позицию, и значение b[i] равно 1, то переменная a1 должна быть увеличена на единицу. Если же порядковый номер позиции оказывается нечетным, а значение равно 0, то следует увеличить значение переменной a2 на единицу.

```
for n in range(2,10000):
    #Создаем строку с двоичным представлением числа n
    b = f'{n:b}'
    #a1 = b[1::2].count('1')
    #a2 = b[::2].count('0')
    #Инициализируем счетчики для подсчета единиц и нулей
    a1, a2 = 0, 0
    #Проходим по каждому символу строки b
    for i in range(len(b)):
       #Если индекс нечетный и текущий символ '1', увеличиваем a1
        if (i+1)%2==0 and b[i]=='1': a1 += 1
        #Если индекс четный и текущий символ '0', увеличиваем a2
        if (i+1)%2!=0 and b[i]=='0': a2 += 1
    #Вычисляем абсолютную разницу между количеством единиц и нулей
    r = abs(a2-a1)
    if r==5:
        print(n,r)
        break
```

Результат работы программы:

1023 5

Ответ: 1023

Задача №5 ()

Автомат получает на вход пятизначное число. По этому числу строится новое число по следующим правилам.

1. Складываются отдельно первая, третья и пятая цифры, а также вторая и четвёртая цифры.
2. Полученные два числа записываются друг за другом в порядке неубывания без разделителей.

Пример. Исходное число: 63 179. Суммы: 6 + 1 + 9 = 16; 3 + 7 = 10. Результат: 1016.

Укажите наименьшее число, при обработке которого автомат выдаёт результат 621.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952456241254?t=1h16m50s

Решение

Автомат принимает на входе пятизначное число. Важно отметить, что оно состоит ровно из пяти цифр. Далее по этому числу создается новое согласно таким правилам: складываются первая, третья и пятая цифра, затем вторая и четвертая. Эти две суммы объединяются в одно число в порядке возрастания. Если рассмотреть пример, где сумма первой группы цифр равна 16, а второй – 10, то итоговое число будет выглядеть так: 1016 (вначале меньшая сумма, затем большая). Требуется найти минимальное пятизначное число, которое после обработки даст результат 621. Здесь стоит учесть, что мы работаем непосредственно с цифрами самого числа, а не с его двоичным представлением. Нам необходимо сложить определенные цифры. Для решения задачи будем перебирать все возможные пятизначные числа от 10000 до 99999. Чтобы работать с отдельными цифрами числа, мы преобразуем его в список цифр. Например, если дано число 12345, то мы получим список [1, 2, 3, 4, 5]. Затем мы сможем суммировать нужные элементы списка.

```
for n in range(10000,100000):
    #Преобразование числа 'n' в строку и последующее преобразование каждого    символа строки в целое число.
    d = [int(c) for c in str(n)]
    #Суммирование элементов списка с индексами 0, 2 и 4 (первые три цифры).
    a1 = d[0] + d[2] + d[4]
    #Суммирование элементов списка с индексами 1 и 3 (вторые две цифры).
    a2 = d[1] + d[3]
      #Проверка условия: если сумма первых трех цифр меньше суммы двух последних,
    if a1<a2:
```

```
        r = int( str(a1)+str(a2) )
    else:
       #В противном случае конкатенация строковых представлений сумм в обратном
порядке.
        r = int( str(a2)+str(a1) )
    #Если полученное значение равно 621, выводим число 'n' и результат 'r'.
    if r==621:
        print(n,r)
```

Этот код выполняет следующие шаги:

1. Перебирает числа от 10000 до 99999.
2. Для каждого числа создает список его цифр.
3. Рассчитывает сумму первых трех цифр a1 и сумму двух последних a2.
4. Сравнивает эти суммы и формирует новое число r, объединяя значения a1 и a2.
5. Если это новое число равно 621, выводит исходное число n и результат r.

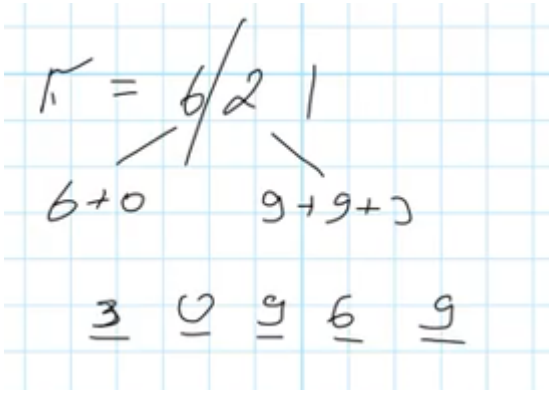
Результат работы программы:

```
30969 621
31959 621
32949 621
33939 621
34929 621
```

...

Минимальное n, из которого мы получаем нужный нам результат. 621, равно 30969.

Ручная проверка:



Ответ: 30969

Задание №6 ()

Автомат получает на вход трёхзначное число. По этому числу строится новое число по следующим правилам.

1. Вычисляются суммы квадратов первой и второй, а также второй и третьей цифр исходного числа.
2. Полученные два числа записываются друг за другом в порядке невозрастания (без разделителей).

Пример. Исходное число: 621. Суммы квадратов цифр: 62 + 22 = 40; 22 + 12 = 5.
Результат: 405.
Укажите наибольшее число, при обработке которого автомат выдаст число 9752

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952456241254?t=1h27m10s

Решение

Под "невозрастанием" понимается такой порядок, при котором первое число больше или равно второму.
Для решения задачи следует перебрать трёхзначные числа. Мы будем перебирать числа в диапазоне от 100 до 999 включительно. Для каждого числа необходимо получить его цифры и вычислить требуемые суммы квадратов. Получив числовые значения каждой цифры, мы можем рассчитать нужные суммы. Первая сумма — это сумма квадратов первой и второй цифр. Вторая сумма — это сумма квадратов второй и третьей цифр. Обратите внимание, что вторая цифра участвует в обоих расчётах. После того как получены обе суммы, их нужно объединить в порядке невозрастания. Если первая сумма больше второй, то она записывается первой, после чего добавляется вторая. В противном случае, сначала записывается меньшая сумма, а затем большая.

```
for n in range(100,1000):
    d = [int(c) for c in str(n)]
    a1 = d[0]**2 + d[1]**2
    a2 = d[1]**2 + d[2]**2
    if a1 >= a2:
        r = str(a1) + str(a2)
    else:
        r = str(a2) + str(a1)
    if r == '9752':
        print(n,r)
```

Результат работы программы:
649 9752
946 9752

Очевидно, что наибольшее число при котором в результате работы программы мы получим 9752 будет 946.

Ответ:

946

Задание № 7()

Алгоритм получает на вход натуральное число N> 1 и строит по нему новое число R следующим образом:
1. Вычисляется сумма чётных цифр в десятичной записи числа N. Если чётных цифр в записи нет, сумма считается равной нулю.

2. Вычисляется сумма цифр, стоящих на чётных местах в десятичной записи числа N без ведущих нулей. Места отсчитываются слева направо (от старших разрядов к младшим, начиная с единицы). Если число однозначное (цифр на чётных местах нет), сумма считается равной нулю.
3. Результатом работы алгоритма становится модуль разности полученных двух сумм.

Пример. Дано число N = 2021. Алгоритм работает следующим образом:

1. Чётные цифры в записи: 2, 0, 2, их сумма равна 4.
2. Цифры на чётных местах: 0, 1, их сумма равна 1.
3. Модуль разности полученных сумм равен 3.

Результат работы алгоритма R = 3.

При каком наименьшем N в результате работы алгоритма получится R = 9?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952456241254?t=1h34m0s

Решение

Чтобы решить задачу, последовательно проверим числа от 2 до 1000.

Для каждого числа извлекаются его цифры и делятся на две группы: четные цифры и цифры, находящиеся на четных позициях.

Для получения четных цифр создается список всех цифр числа, и затем проходит цикл, выбирающий только четные элементы. Эти цифры суммируются.

Далее определяется сумма цифр, расположенных на четных позициях. Для этого применяется срез списка, позволяющий выбрать каждый второй элемент, начиная с первого.

Когда обе суммы найдены, вычисляется модуль их разности, что и является конечным значением r.

```
for n in range(2,10000):
    d = [int(c) for c in str(n)]
    a1 = sum(c for c in d if c%2==0)
    a2 = sum(d[1::2])
    r = abs(a2-a1)
    if n==2021:
        print(n,r)
```

Результат:
2021 3
Эти значения соответствуют результату работы алгоритма приведенного в примере.
Дополним алгоритм заданным условием задачи:

```
for n in range(2,10000):
    d = [int(c) for c in str(n)]
    a1, a2 = 0, 0
    a1 = sum(c for c in d if c%2==0)
    a2 = sum(d[1::2])
    r = abs(a2-a1)
    if r==9:
        print(n,r)
```

Результат работы программы:

```
19 9
39 9
59 9
79 9
99 9
190 9
191 9
...
```

Рассмотрев выведенные числа определяем минимальное из них число, это число 19.

Рассмотрим способ решения этой задачи через перебор индексов. Реализуем следующим образом:

```
for n in range(2,10000):
    d = [int(c) for c in str(n)]
    a1, a2 = 0, 0
      # Проходим по каждой цифре числа
    for i in range(len(d)):
            # Если цифра находится на четной позиции, добавляем её к первой сумме
        if d[i]%2==0: a1 += d[i]
            # Если индекс цифры является нечетным, добавляем её ко второй сумме
        if (i+1)%2==0: a2 +=d[i]
    #a1 = sum(c for c in d if c%2==0)
    #a2 = sum(d[1::2])
    r = abs(a2-a1)
    if r==9:
        print(n,r)
```

Результат работы программы будет таким же

```
19 9
39 9
59 9
79 9
99 9
190 9
191 9
...
```

Ответ: 19

Задание №8 ()

Алгоритм получает на вход натуральное число N>10 и строит по нему новое число R следующим образом:

1. Все пары соседних цифр в десятичной записи N рассматриваются как двузначные числа (возможно, с ведущим нулём).
2. Из списка полученных на предыдущем шаге двузначных чисел выделяются наибольшее и наименьшее.
3. Результатом работы алгоритма становится разность найденных на предыдущем шаге двух чисел.

Пример. Дано число N= 2022. Алгоритм работает следующим образом:

1. В десятичной записи выделяем двузначные числа: 20, 02, 22.
2. Наибольшее из найденных чисел 22, наименьшее 02.
3. 22 - 02 = 20.

Результат работы алгоритма R = 20.
При каком наименьшем N в результате работы алгоритма получится R = 44?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952456241254?t=1h41m30s

Решение

Здесь нецелесообразно рассматривать двузначные числа, поскольку среди них только одна пара, и разность между двумя одинаковыми цифрами всегда будет равна нулю. Поэтому стоит начать с трёхзначных чисел. Нам понадобятся десятичные цифры. Мы будем склеивать их попарно, как показано в примере. Чтобы получить пары символов, воспользуемся строковой записью числа N. Рассмотрим все возможные пары соседних цифр. Их две: первая пара (первая и вторая цифры) и вторая пара (вторая и третья цифры). Таким образом, будем получать две пары: (d0 + d1) и (d1 + d2).
Далее находим максимальное и минимальное значения среди полученных пар и вычисляем их разность. После преобразования пар в числа, ищем максимальную и минимальную пару. Разница между максимальным и минимальным числом даст нам искомую величину.
Если результат равен 44, выводим число.

```
for n in range(100,1000):
    d = str(n)
    a = [int(d[0]+d[1]), int(d[1]+d[2])]
    r = max(a) - min(a)
    if r==44:
        print(n,r)
```

Результат работы программы:
159 44
160 44
271 44
382 44
493 44
506 44
617 44
728 44
839 44
840 44
951 44

Найдено подходящее число: 159. Разница между 59 и 15 действительно равна 44.
Для четырёхзначных чисел пришлось бы добавить ещё одну пару цифр, но поскольку решение найдено при рассмотрении трёхзначного числа, дальнейший перебор не требуется.

Ответ: 159

Задание №9()

Автомат получает на вход трёхзначное число. По этому числу строится новое число по следующим правилам.

1. Из цифр, образующих десятичную запись N, строятся наибольшее и наименьшее возможные двузначные числа (числа не могут начинаться с нуля).
2. На экран выводится разность полученных двузначных чисел.

Пример. Дано число N = 351. Наибольшее двузначное число из заданных цифр – 53, наименьшее – 13. На экран выводится разность 53 – 13 = 40.
Чему равно наименьшее возможное трёхзначное число N, в результате обработки которого на экране автомата появится число 60?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952456241254?t=1h48m5s

Решение

Немного о комбинаторике. Сколько разных пар можно составить из трех цифр? Если цифры уникальны, существует шесть вариантов (3 x 2).
Рассмотрим числа от 100 до 999. Нас интересуют их цифры. Запишем все возможные двузначные числа, состоящие из любой пары цифр. Например, такими могут быть следующие комбинации: (d0 + d1), (d0 + d2), (d1 + d0) и так далее, где каждая пара создает уникальное число.
Стоит учесть, что порядок важен: 35 и 53 — это два разных числа. В итоге получается шесть уникальных сочетаний двух цифр из трех. После формирования всех возможных комбинаций, преобразуем их в числа для определения максимальных и минимальных значений, а затем вычислим их разность. Убедимся, что найденные числа соответствуют условию задачи: они должны быть двузначными, то есть начинаться не с нуля.
Таким образом, исключаются однозначные числа, остаются только те, которые действительно являются двузначными. После этого определяется разница между наибольшим и наименьшим значениями среди этих чисел. Т.к. нам нужно найти наименьшее возможное трёхзначное число N, в результате обработки которого на экране автомата появится число 60, запишем соответствующее условие:

```
for n in range(100,1000):
    #Преобразование числа n в строку и сохранение результата в переменной 'd'
    d = str(n)
    #Создание списка 'a', состоящего из сумм двух соседних цифр числа n.
    a = [int(d[0]+d[1]), int(d[0]+d[2]), int(d[1]+d[0]), int(d[1]+d[2]),
         int(d[2]+d[0]), int(d[2]+d[1])]
    #Фильтрация списка 'a' с сохранением только тех элементов, которые больше или
равны 10.
    a = [x for x in a if x>=10]
    #Вычисление разности между максимальным и минимальным элементами списка 'a'.
    r = max(a) - min(a)
    #Проверка условия: если разность равна 60, выводим число n и значение r.
    if r == 60:
        print(n,r)
```

Результат работы программы:

493 60
517 60
528 60
539 60

571 60
...
Минимальным числом является число 493
Ответ: 493

Задание №10 ()

На вход алгоритма подается натуральное число N. Алгоритм строит по нему новое число R следующим образом.

1. Строится двоичная запись числа N.
2. В этой записи последний ноль заменяется на первые две цифры полученной записи. Если нуля нет, алгоритм аварийно завершается.
3. Запись записывается справа налево (в обратную сторону).

Полученная таким образом запись является двоичной записью искомого числа R.
Для какого минимального значения N в результате работы алгоритма получится число 123?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952456241254?t=1h58m55s

Решение

Возможен вариант решения задачи с использованием функции replace():

```
for n in range(2,100):
    b = f'{n:b}'
    #Проверяем наличие символа '0' в строке b
    if '0' in b:
           #Берем первые два символа строки b и объединяем их в одну переменную b1
        b1 = b[0]+b[1]
           #Переворачиваем строку b
        b = b[::-1]
        #Находим первую позицию символа '0' и заменяем его на перевернутый вариант
первых двух символов (b1[::-1])
        b = b.replace('0', b1[::-1] ,1)
    r = int(b,2)
    if r==123:
        print(n)
```

Или с использованием метода find()

```
for n in range(2,100):
    b = f'{n:b}'
    #Проверяем наличие символа '0' в строке b
    if '0' in b:
        #Находим последний индекс символа '0'
        i = b.rfind('0')
        #Меняем местами символы после последнего нуля
        b = b[:i]+ b[0]+b[1] + b[i+1:]
        #Инвертируем строку (меняем порядок символов)
        b = b[::-1]
        #b1 = b[0]+b[1]
        #b = b[::-1]
```

```
        #b = b.replace('0', b1[::-1] ,1)
        #Преобразуем строку в целое число по основанию 2
        r = int(b,2)
            #Если результат равен 123, выводим число n
        if r==123:
            print(n)
```

Результат работы обеих программ будет одинаков:
47
51
53
54
Минимальное значение, при котором в результате работы алгоритма получится число 123 равно 47.

Ответ : 47