Конспект 1

Strim_5_1

Пятая задача сводится к тому, что есть определённый алгоритм, который позволяет преобразовать одно число в другое. Этот процесс обычно включает несколько шагов:

1. Сначала мы берём исходное число и переводим его в какую-то другую систему счисления (например, двоичную, восьмеричную или шестнадцатеричную).
2. Затем над полученными цифрами выполняются определённые математические операции. Это могут быть сложение, вычитание, умножение, деление или какие-либо другие действия.
3. После выполнения всех необходимых операций результат вновь переводится обратно в десятичную систему счисления.
4. На основании полученных данных решается поставленная задача.

Таким образом, основная суть этой задачи состоит в переводе числа в другую систему счисления, выполнении некоторых арифметических операций с его цифрами и обратном переводе результата в десятичное представление.

Задача № 1 (1741)
На вход алгоритма подаётся натуральное число N. Алгоритм строит по нему новое число R следующим образом.
1. Строится двоичная запись числа N.
2. К этой записи дописываются справа ещё два разряда по следующему правилу:
а) складываются все цифры двоичной записи, и остаток от деления суммы на 2 дописывается в конец числа (справа). Например, запись 11100 преобразуется в запись 111001;
б) над этой записью производятся те же действия – справа дописывается остаток от деления суммы цифр на 2.
Полученная таким образом запись (в ней на два разряда больше, чем в записи исходного числа N) является двоичной записью искомого числа R. Какое наибольшее число, меньшее 70, может быть получено в результате работы автомата?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241249?t=0h2m40s

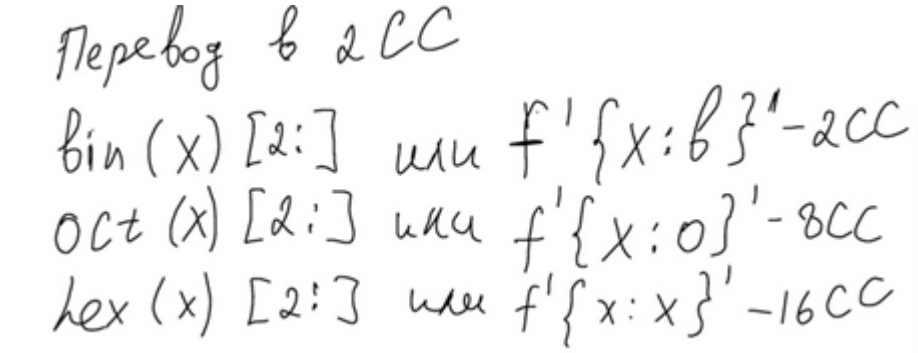
Решение

Сначала вспомним о переводе чисел в двоичную систему счисления.
Для удобства мы будем использовать f-строку, поскольку она позволяет легко манипулировать цифрами, а затем преобразовать обратно с помощью функции int. Поэтому мы будем стараться использовать строковую запись.
Когда речь идет о переводе числа в двоичное представление, у нас есть два основных метода.
Первый метод — стандартный, это использование функции bin().
Эта функция конвертирует число в двоичную форму. Мы знаем, что она добавляет префикс 0b к началу строки, поэтому мы обычно удаляем его. Два и двоеточие [2:] указывает на то, что мы отбрасываем первые два символа (так как они представляют собой 0b, которые нам не нужны).

Другой подход — использование f-строки. Она полезна для быстрого преобразования числа в двоичный формат. Для этого мы используем следующую запись:

```
b = f'{x:b}'
```

, где x — это число, которое нужно перевести.
Теперь поговорим о восьмеричной системе счисления. Для этой цели применяется функция oct(). Также в f-строке есть соответствующий модификатор o, который обозначает восьмеричную систему. Шестнадцатеричная система счисления представлена модификатором x. Здесь x является переменной, а следующее за ним x и o — модификаторы.



Итак, у нас есть три системы счисления: двоичная, восьмеричная и шестнадцатеричная.
Теперь перейдем к сумме цифр.
Сумму цифр числа можно вычислить следующим образом. В двоичной системе счисления всё довольно просто, поскольку здесь используются только 0 и 1. Нет никаких других чисел. Таким образом, мы можем просто подсчитать количество единиц, ведь ясно, что сумма единиц равна их количеству. Это одно и то же. Мы можем использовать выражение s.count('1'), которое представляет собой сумму цифр числа в двоичной системе.
В других системах счисления ситуация немного сложнее, так как необходимо суммировать значения всех цифр. Для этого применяется функция sum(int(d) for d in s), где s – строка, представляющая число. Она не обязана называться именно s, имя переменной может быть другим, но суть остаётся той же. Также можно воспользоваться функцией map:
sum(map(int, s)).

Теперь рассмотрим задачу подробнее и разберемся с каждым шагом алгоритма:

- Классический подход заключается в написании программы, которая будет выполнять указанные операции с различными числами n и проверять полученные результаты.
- Начинаем с цикла for, где перебираем натуральные числа n в диапазоне от 1 до 100. Поскольку максимальный результат должен быть меньше 70, этого диапазона кажется достаточным.

На каждом шаге цикла преобразуем текущее число n в двоичный формат. Создадим переменную b типа "двоичный" (binary), в которую будет записано двоичное представление числа n . Таким образом, получаем двоичное представление числа.

```
for n in range(1,100):
    b = f'{n:b}'
```

Далее вычисляем сумму единиц в двоичной записи числа, и если эта сумма четная, добавляем «0» в конец строки, иначе — «1». Этот процесс повторяем дважды.

```
    if b.count('1')%2==0:
        b = b + '0'
    else:
        b = b + '1'
```

После завершения преобразований переводим полученное двоичное число обратно в десятичный формат и сравниваем его с условием, что оно должно быть меньше 70.

```
    r = int(b,2)
```

```
    if r<70:
```

Все подходящие числа добавляются в заранее созданный список m.

```
        m.append(r)
```

Т.к. нам нужно наибольшее число в конце выведем максимальное значение из полученного списка.

```
m = []
for n in range(1,100):
    b = f'{n:b}'
    if b.count('1')%2==0:
        b = b + '0'
    else:
        b = b + '1'
    if b.count('1')%2==0:
        b = b + '0'
    else:
        b = b + '1'
    r = int(b,2)
    if r<70:
        m.append(r)
print(max(m))
```

Результат работы программы:

68

Ответ: 68

Задача №2 (1772)

На вход алгоритма подаётся натуральное число N. Алгоритм строит по нему новое число R следующим образом.
1. Строится двоичная запись числа N.
2. К этой записи дописывается (дублируется) последняя цифра.
3. Затем справа дописывается бит чётности: 0, если в двоичном коде полученного числа чётное

число единиц, и 1, если нечётное.
4. К полученному результату дописывается ещё один бит чётности.
Полученная таким образом запись (в ней на три разряда больше, чем в записи исходного числа N) является двоичной записью искомого числа R. Какое минимальное число R, большее 66, может быть получено в результате работы автомата?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241249?t=0h22m45s

Решение

Для решения задачи будем использовать строковое представление двоичных чисел, так как это позволяет легко манипулировать отдельными разрядами. Поскольку ожидаемый результат невелик, $r > 66$, будем рассматривать числа n в диапазоне от 1 до 100.
Рассмотрим пошаговый алгоритм:

1. Для каждого числа n из диапазона [1, 100]:
- Сформируем его двоичное представление.
- Добавим последнюю цифру к концу строки.
- Проверим чётность количества единиц в строке и добавим соответствующий бит чётности ('0' или '1') к её концу.
- Повторим проверку чётности и добавление бита ещё раз.
- Преобразуем получившуюся строку обратно в десятичное число.
2. Среди всех полученных значений выберем минимальное, превышающее 66.

```
m = []
for n in range(1,100):
    # Формируем двоичное представление числа n
    b = f'{n:b}'
    # Дублируем последнюю цифру
    b = b + b[-1]
    #Добавляем первый бит четности
    if b.count('1')%2==0:
        b = b + '0'
    else:
        b = b + '1'
    #Добавляем второй бит четности
    if b.count('1')%2==0:
        b = b + '0'
    else:
        b = b + '1'
    #Переводим результат в десятичную систему
    r = int(b,2)
    if r>66:
        #Сохраняем все значения r, которые больше 66
        m.append(r)
#Находим минимальное значение среди найденных
print(min(m))
```

Результат работы программы:

78

Ответ: 78

Задача № 3 (1776)

(Досрочный ЕГЭ-2018) На вход алгоритма подаётся натуральное число N. Алгоритм строит по нему новое число следующим образом.
1) Строится двоичная запись числа N.
2) К этой записи дописываются справа ещё два разряда по следующему правилу: если N чётное, в конец числа (справа) дописываются два нуля, в противном случае справа дописываются две единицы. Например, двоичная запись 1001 числа 9 будет преобразована в 100111.
Полученная таким образом запись (в ней на два разряда больше, чем в записи исходного числа N) является двоичной записью числа – результата работы данного алгоритма. Укажите минимальное число N, для которого результат работы алгоритма будет больше 115. В ответе это число запишите в десятичной системе счисления.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241249?t=0h30m40s

Решение

Важно отметить, что требуется именно минимальное такое число n, при котором результат выполнения данного алгоритма окажется больше 115. Для поиска решения можно перебрать значения n в диапазоне от 1 до 200, чтобы гарантированно охватить возможные варианты. Двоичная запись числа n может быть построена с помощью функции преобразования числа в строку, где форматированием задаётся двоичный вид числа.
Рассмотрим два возможных случая:
· Если число n чётное, то к его двоичной записи добавляется "00". Это можно проверить путём вычисления остатка от деления числа n на 2: если остаток равен 0, значит, число чётное.
· Если число n нечётное, то к его двоичной записи добавляется "11".
После добавления двух дополнительных битов к двоичному представлению числа, итоговая строка преобразовывается обратно в десятичное число. Например, если n = 9 , то его двоичная запись будет "1001", а после добавления двух единиц получится "100111" — это соответствует числу 39 в десятичной системе счисления 32 + 7.
Проверив работу алгоритма на конкретных примерах, убеждаемся, что он функционирует корректно. Теперь необходимо найти минимальное число n, для которого результат работы алгоритма превысит 115.
Для этого организуем цикл по всем числам от 1 до 200, будем добавлять к каждому числу дополнительные биты согласно правилу и проверять, превышает ли результат 115. Все подходящие числа n сохраняем в список, а затем находим среди них минимальное.

```
m = []
for n in range(1,200):
#Преобразование числа n в двоичную строку
    b = f'{n:b}'
# Проверка чётности числа n
if n%2==0:
# Добавляем два нуля, если число чётное
b = b + '00'
    else:
# Добавляем две единицы, если число нечётное
```

```
b = b + '11'
# Преобразовываем двоичную строку обратно в целое число
r = int(b,2)
    if r>115:
#Сохраняем подходящее число n в список
m.append(n)
# Находим минимальное число из подходящих значений
print(min(m))
```

Результат работы программы:

29

Результат выполнения программы показывает, что минимальным числом n, удовлетворяющим условию задачи, является 29. Этот результат был найден путём сохранения всех подходящих чисел n в список и последующего нахождения минимального элемента из него.

Ответ: 29

Задача № 4 (4219)

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

1) Строится двоичная запись числа N.
2) К этой записи дописывается ещё три или четыре разряда по следующему правилу: если N нечётное, то слева к нему приписывается "1", а справа - "11". В противном случае слева приписывается "11", а справа "00".
Например, $N = 5_{10} = 101_2 => 110111_2 = 55_{10} = R$

Полученная таким образом запись (в ней на три или четыре разряда больше, чем в записи исходного числа N) является двоичной записью искомого числа R. Укажите наибольшее число R, меньшее 127, которое может быть получено с помощью описанного алгоритма. В ответ запишите это число в десятичной системе счисления.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241249?t=0h35m50s

Решение

В данной задаче рассматривается интересное свойство: в зависимости от значения исходного числа, к нему добавляется различное количество цифр.

Алгоритму на вход подаётся натуральное число n. По этому числу алгоритм формирует новое число r следующим образом. Сначала создаётся двоичное представление числа, после чего к нему добавляются ещё 3 или 4 разряда (цифры), согласно определённому правилу. Если число n является нечётным, то слева от него записывается единица, а справа — комбинация "11". Таким образом, слева появляется одна единица, а справа — две.
Если же число n чётное, то слева добавляется последовательность "11", а справа — "00". Таким образом, в зависимости от чётности числа n, к его двоичной записи добавляются различные комбинации цифр. Обратите внимание, что из-за этого итоговые результаты получаются не последовательными. В отличие от предыдущих задач, где результаты следовали друг за другом по возрастанию, в этой задаче они будут перемешаны, так как иногда к числам добавляется три цифры, а иногда — четыре.
Наша цель — найти наибольшее значение r, которое будет меньше 127 и может быть получено при помощи вышеописанного алгоритма.
Как обычно, начнём с использования цикла for, пробегающего по всем возможным значениям n от 1 до некоторого большого числа, например, до 200.
Для каждого числа n сначала преобразуем его в двоичный формат. Для этого можно воспользоваться f-строкой. Затем проверим, является ли число n нечётным. Если оно нечётно, n % 2 != 0, то слева добавляем одну единицу, а справа — две "11. Если же число чётное, то слева добавляем "11", а справа — "00".
После этого переведём получившуюся двоичную строку обратно в десятичное число. Например, для числа 5 результатом будет 55.
Теперь нам необходимо найти максимальное значение r, которое меньше 127. Если найденное значение удовлетворяет данному условию, выводим на экран соответствующую пару чисел n и r.

```
for n in range(1,200):
    b = f'{n:b}'
    if n%2!=0:
        b = '1' + b + '11'
    else:
        b = '11' + b + '00'
    r = int(b,2)
    if r<127:
        print(n,r)
```

Результат работы программы:

```
1 15
2 56
3 31
4 112
5 55
6 120
7 63
9 103
11 111
13 119
```

Обратим внимание, что полученные значения идут не последовательно, как было в предыдущих задачах. Последний результат, равный 119, не является наибольшим. Максимальное значение — 120, и оно находится ближе к середине списка. Это связано с тем, что в зависимости от чётности числа n к нему добавляется разное количество цифр.
Чтобы найти наибольшее значение среди всех результатов, необходимо соберём подходящие значения в список и выберем максимальный элемент.

```
m = []
```

```
for n in range(1,200):
    b = f'{n:b}'
    if n%2!=0:
        b = '1' + b + '11'
    else:
        b = '11' + b + '00'
    r = int(b,2)
    if r<127:
        m.append(r)
print(max(m))
```

Результат работы программы:

120

Ответ:120

Задача №5 (4931)

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

1) Строится двоичная запись числа N.
2) К этой записи дописываются ещё несколько разрядов по следующему правилу:
   а) если N чётное, то к нему справа приписывается в двоичном виде сумма цифр его двоичной записи;
   б) если N нечётное, то к нему справа приписываются два нуля, а слева единица.

Например, двоичная запись числа 1101 будет преобразована в 1110100.
Полученная таким образом запись (в ней как минимум на один разряд больше, чем в записи исходного числа N) является двоичной записью искомого числа R.

Укажите наибольшее число N, для которого результат работы данного алгоритма меньше 1000. В ответе это число запишите в десятичной системе счисления.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241249?t=0h42m50s

Решение

Для решения задачи необходимо перебрать возможные значения n. Используем диапазон от 1 до 2000, чтобы гарантировать охват всех возможных значений, удовлетворяющих условию.
Алгоритм действий:

1. Для каждого числа n из указанного диапазона получаем его двоичное представление.
2. Проверяем четность числа n:

· Если n четное, вычисляем сумму цифр его двоичного представления и добавляем эту сумму в двоичном виде к правой части двоичной строки.

· Если n нечетное, добавляем "1" слева и "00" справа.

3. Преобразуем получившуюся строку обратно в целое число r.
4. Если $r < 1000$, сохраняем значение n в списке.
5. После завершения цикла находим максимальное значение среди сохраненных n.

```
m = []
for n in range(1,2000):
    b = f'{n:b}'
    if n%2==0:
        s = b.count('1')
        b = b + f'{s:b}'
    else:
        b = '1' + b + '00'
    r = int(b,2)
    if r<1000:
        m.append(n)
print(max(m))
```

Результат работы программы:
256
Результат выполнения программы показывает, что максимальное значение n, при котором $r < 1000$, равно 256.

Ответ: 256

Задание №6 (5368)

На вход алгоритма подаётся натуральное число N. Алгоритм строит по нему новое число R следующим образом.
1) Строится двоичная запись числа N.
2) К этой записи дописываются ещё несколько разрядов по следующему правилу:
а) если сумма цифр в двоичной записи числа чётная, то к этой записи справа дописывается 0, а затем два левых разряда заменяются на 10;
б) если сумма цифр в двоичной записи числа нечётная, то к этой записи справа дописывается 1, а затем два левых разряда заменяются на 11.
3) Результат переводится в десятичную систему и выводится на экран.
Например, для исходного числа $6 = 110_2$ результатом является число $1000_2 = 8$, а для исходного числа 4 = 1002 результатом является число $1101_2 = 13$.
Укажите максимальное число N, после обработки которого с помощью этого алгоритма получается число R, меньшее, чем 35.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241249?t=0h51m15s

Решение

Переберем числа с помощью цикла for от 1 до 100. Получив двоичное представление числа, проверим сумму его цифр. Если она чётная, допишем справа ноль и заменим два левых разряда на

один-ноль. Если сумма цифр нечётная, допишем справа единицу и заменим два левых разряда на один-ноль.
! Обратите внимание, что в строках нельзя напрямую изменять символы. Вместо этого создаётся новая строка на основе старой. Например, для замены двух первых символов строки на '10' можно использовать срез: '10' + b[2:].
После преобразования результата в десятичное число, выведем его на экран. Если результат меньше 35, запомним соответствующее значение. В итоге найдём максимальное, удовлетворяющее этому условию.

```
m = []
for n in range(1,100):
    b = f'{n:b}'
    if b.count('1')%2==0:
        b = b + '0'
        #убрали две первые цифры и вместо них поставили 10
        b = '10' + b[2:]
    else:
        b = b + '1'
        b = '11' + b[2:]
    r = int(b,2)
    if r<35:
        m.append(n)
print(max(m))
```

Результат работы программы:

24

Ответ: 24

Задание № 7()

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 количество значащих цифр в двоичной записи числа чётное, то к этой записи в середину дописывается 1;

б) если количество значащих цифр в двоичной записи числа нечётное, то запись не изменяется.

Полученная таким образом запись является двоичной записью искомого числа R. Например, для исходного числа $5_{10} = 101_2$, результатом является число $101_2 = 5_{10}$ а для исходного числа $2_{10} = 10_2$ результатом является число $110_2 = 610$.

Укажите минимальное число N, после обработки которого с помощью этого алгоритма получается число R, не меньшее, чем 26. В ответе запишите это число в десятичной системе счисления.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241249?t=1h10m35s

Решение

Для поиска минимального значения n, удовлетворяющего условию, используется следующий алгоритм:

1. Мы перебираем числа от 1 до 49 включительно.
2. Каждое число преобразуем в двоичную систему счисления.
3. Проверяем, является ли длина полученного двоичного представления чётной. Если да, добавляем "1" в середину этой строки.
4. Если длина двоичного представления нечётна, оставляем его без изменений.
5. После внесения возможных изменений переводим полученное двоичное число обратно в десятичную систему.
6. Сравниваем это новое десятичное число с числом 26. Если оно больше или равно 26, сохраняем соответствующее значение n.
7. Из всех найденных значений n выбираем минимальное.

```
#Создаём пустой список для хранения подходящих значений n
m = []
#Перебираем числа от 1 до 49 включительно
for n in range(1,50):
    # Преобразуем текущее число n в двоичный формат

    b = f'{n:b}'
    # Проверяем, является ли количество цифр в двоичном представлении чётным
    if len(b)%2==0:
        # Разделяем строку пополам и вставляем "1" в середину
        b = b[:len(b)//2] + '1' + b[len(b)//2:]
    else:
        # Если количество цифр нечётное, ничего не меняем
        ...
    # Переводим изменённую двоичную строку обратно в десятичный формат
    r = int(b,2)
    # Проверяем, удовлетворяет ли результат условию r >= 26
    if r>=26:
        # Если условие выполняется, сохраняем текущее значение n
        m.append(n)
# Находим и выводим минимальное подходящее значение n
print(min(m))
```

Результат работы программы:
12
После запуска программы получается, что минимальным подходящим значением n является 12.

Ответ: 12

** Чтобы дописать что-то в середину, необходимо строку разбить на две части*

Существует множество методов для представления чисел в различных системах счисления. Рассмотрим два основных случая: системы счисления с основанием, меньшим десяти, и системы счисления с основанием, большим десяти.
Системы счисления с основанием меньшим десяти
Для перевода числа в систему счисления с основанием менее десяти (например, в троичную или пятеричную) можно создать специальную функцию. Пример такой функции приведен ниже:

```
def cc(x, base):
    if x == 0:
        return '0'
    result = ''
    while x > 0:
        s = str(x%base) + s
        x //= base
    return s
```

Функция принимает два аргумента: x — это число, которое требуется преобразовать, а base — основание новой системы счисления.
Системы счисления с основанием большим десяти
При работе с системами счисления, у которых основание превышает десять, необходимо использовать буквы алфавита для обозначения цифр, которые соответствуют числам больше девяти. В этом случае мы можем воспользоваться модулем string, содержащим строку printable, которая включает все необходимые символы.
Пример функции для преобразования числа в систему счисления с основанием более десяти представлен здесь:

```
from string import printable
def cc(x, base):
    if x == 0:
        return '0'
    result = ''
    while x > 0:
        s = printable[x% base] + s
        x = x//base
    return s
```

Данная функция аналогична предыдущей, однако она использует строку printable для получения символов, представляющих цифры больше девяти. Это позволяет корректно отображать такие цифры с помощью соответствующих букв латинского алфавита.
Таким образом, обе эти функции предоставляют эффективные способы преобразования чисел в различные системы счисления, будь то системы с основанием меньше или больше десяти.
* Если функция принимает значение "ноль", рекомендуется предусмотреть возврат значения "ноль". В противном случае, при передаче функции аргумента "ноль" выполнение программы может быть прервано, и функция вернет пустую

строку. Это подчеркивает важность внимания к мелочам и указывает на ваш уровень программирования, говорит о том, что вы умеете учитывать разные ситуации, включая обработку нуля.

Задание №8 (6383)

Алгоритм получает на вход натуральное число N > 4 и строит по нему новое число R следующим образом:
1. Строится пятеричная запись числа N.
2. Далее эта запись обрабатывается по следующему правилу:
  а) если число N делится на 5, то в конец дописываются две последние цифры пятеричной записи числа;
  б) если число N на 5 не делится, то остаток от его деления на 5 умножается на 7, переводится в пятеричную запись и дописывается в конец числа.
Полученная таким образом запись является пятеричной записью искомого числа R.
Укажите минимальное число R, большее 200, которое может быть получено с помощью описанного алгоритма. В ответе запишите это число в десятичной системе счисления.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241249?t=1h19m10s

Решение

Для начала создадим функцию, которая переводит число из десятичной системы счисления в пятеричную. Эта функция будет принимать на вход целое число x и возвращать строку, представляющую собой пятеричную запись этого числа.

```
def cc5(x):
    if x==0: return '0'
    s = ''
    while x>0:
        s = str(x%5) + s
        x = x//5
    return s
```

Теперь реализуем основной алгоритм. Мы будем перебирать все натуральные числа начиная с 5, т.к. по условию задачи n > 4, строить их пятеричные представления, обрабатывать эти представления согласно правилам задачи и проверять, какое минимальное значение r превышает 200.

```
m = []
def cc5(x):
    if x==0: return '0'
    s = ''
    while x>0:
        s = str(x%5) + s
        x = x//5
    return s
for n in range(5,300):
```

```
    b = cc5(n)
    if n%5==0:
        # Дописываем две последние цифры
        b = b + b[-2] + b[-1]
    else:
       # Умножаем остаток на 7 и переводим в пятеричную систему
        b = b + cc5(n%5*7)
    # Переводим новую пятеричную запись обратно в десятичное число
    r = int(b,5)
    if r>200:
        m.append(r)
# Находим минимальное значение среди всех подходящих результатов
print(min(m))
```

Результат работы программы:

221

После выполнения программы мы увидим, что минимальным значением r , превышающим 200, является 221.

Ответ: 221

Задание №9(6884)

Алгоритм получает на вход натуральное число N > 11 и строит по нему новое число R следующим образом:

1. Строится запись числа N в системе счисления с основанием 12.
2. Далее эта запись обрабатывается по следующему правилу:
   а) если число N делится на 12, то в конец дописываются две последние цифры двенадцатеричной записи числа;
   б) если число N на 12 не делится, то остаток от его деления на 12 умножается на 9, переводится в систему счисления с основанием 12 и дописывается в конец числа.

Полученная таким образом запись является двенадцатеричной записью искомого числа R.

Укажите минимальное число R, большее 300, которое может быть получено с помощью описанного алгоритма. В ответе запишите это число в десятичной системе счисления.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241249?t=1h31m0s

Решение

Здесь мы работаем с двенадцатеричной системой счисления. В чем заключается её принципиальное различие с предыдущими системами? По существу, оно состоит лишь в основании системы — 12. Следует отметить, что прямой перенос алгоритмов здесь невозможен; потребуется использовать несколько иной метод для корректного применения цифр A и B.
Для начала необходимо импортировать модуль printable из библиотеки string.
Далее создадим функцию для преобразования чисел в двенадцатеричную систему. Назовём её cc_12. Если аргумент x равен нулю, функция вернёт нулевое значение. Затем создаётся пустая строка s, и пока значение x остаётся положительным, к этой строке добавляется цифра, которая соответствует результату деления x на 12 (printable[x % 12]), одновременно обновляется значение x путём целочисленного деления на 12. Так символы из списка printable будут последовательно добавлены в результирующую строку. По завершении цикла возвращается сформированная строка.
После этого переходим к обработке числа n. Для каждого значения n в диапазоне от 1 до 400 строится его представление в двенадцатеричной системе с помощью созданной функции. Если число делится на 12 без остатка, к концу строки присоединяются две последние цифры. Это может быть выполнено двумя методами: либо прямым добавлением этих цифр, либо извлечением среза последних двух символов.
Если остаток от деления n на 12 отличен от нуля, он умножается на 9, результат преобразуется в двенадцатеричное представление и добавляется к числу справа. Полученное число затем снова конвертируется в десятичную форму.
В завершение собираются все результаты, превышающие 300, в список m, после чего находится минимальное значение среди них.

```
from string import printable
# Функция для преобразования числа в строку по основанию 12.
def cc12(x):
    # Если число равно нулю, возвращаем строку '0'.
    if x==0: return '0'
    # Инициализируем пустую строку для хранения результата.
    s = ''
    # Пока число больше нуля, продолжаем цикл.
    while x>0:
        # Добавляем символ из строки printable,
        # соответствующий остатку от деления на 12.
        s = printable[x%12] + s
        # Обновить значение x целочисленным делением на 12
        x = x//12
    return s
m = []
for n in range(12,400):
    # Преобразуем число n в строку по основанию 12.
    b = cc12(n)
    if n%12==0:
        # Если число делится на 12 без остатка,
        # добавляем к строке последние два символа.
        b = b + b[-2:]
    else:
        # добавляем результат преобразования
        # остатка от деления на 12 умноженного на 9.
        b = b + cc12(n%12*9)
    # Преобразуем полученную строку обратно в десятичное число.
    r = int(b,12)
```

```
    if r>300:
        m.append(r)
print(min(m))
```

Результат работы программы:

309

Ответ: 309

* Если функция принимает значение "ноль", рекомендуется предусмотреть возврат значения "ноль". В противном случае, при передаче функции аргумента "ноль" выполнение программы может быть прервано, и функция вернет пустую строку. Это подчеркивает важность внимания к мелочам и указывает на ваш уровень программирования, говорит о том, что вы умеете учитывать разные ситуации, включая обработку нуля.

Задание №10 (7055)

На вход алгоритма подаётся натуральное число N. Алгоритм строит по нему новое число R следующим образом.
1. Строится троичная запись числа N.
2. К этой записи дописываются справа ещё несколько разрядов по следующему правилу:
  а) если N чётное, то к нему справа приписываются два нуля, а слева единица;
  б) если N нечётное, то к нему справа приписывается в троичном виде сумма цифр его троичной записи.
Полученная таким образом запись (в ней как минимум на один разряд больше, чем в записи исходного числа N) является троичной записью искомого числа R.
Например, исходное число $4_{10} = 113$ преобразуется в число $11100_3 = 117_{10}$, а исходное число $7_{10} = 21_3$ преобразуется в число $2110_3 = 66_{10}$.
Укажите такое наименьшее число N, для которого число R больше числа 168. В ответе запишите это число в десятичной системе счисления.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241249?t=1h46m0s

Решение

Мы строим троичное представление числа n. Затем к этому представлению добавляются дополнительные цифры согласно следующим правилам.

Если n чётное, то справа дописываются два нуля, а слева добавляется единица. Если n нечётное, то справа добавляется результат сложения всех цифр исходного троичного представления n, переведённый обратно в троичный формат.

Рассмотрим примеры чисел 4 и 7. Найдём минимальное значение n, при котором r превышает 168. Посмотрим, что получится, когда x умножить на 5%. Здесь x — это просто число, которое преобразуется в пятеричную систему счисления.

x % 5 означает остаток от деления на 5, который представляет собой цифру в пятеричной системе от 0 до 4. Из списка берутся строковые значения '0', '1', '2', '3' и '4'.

Теперь создадим функцию для преобразования числа в троичную систему. Она работает следующим образом:

- Если x = 0 , возвращаем "0".
- Пока x > 0 :
- Добавляем строку s к строке str(x % 3) + s .
- Делим x на 3.
- Возвращаем s.

Функция готова. Теперь будем проверять различные значения n от 1 до 200. Для каждого числа получаем его троичное представление. Если n чётно, добавляем справа два нуля и единицу слева. Иначе находим сумму цифр в троичной записи, переводим её в троичный формат и добавляем справа.
Сумму цифр вычислим через цикл, суммируя каждую цифру троичной записи. После этого добавляем эту сумму в троичном формате к исходной записи. Переводим итоговое число в десятичную систему, чтобы получить r .

```
def cc3(x):
    # Если x равно нулю, вернуть строку '0'
    if x==0: return '0'
    # Инициализировать пустую строку для хранения результата
    s = ''
   # Пока значение x больше нуля
    while x>0:
        # Добавить остаток от деления x на 3 к строке s
        s = str(x%3)+s
        # Обновить значение x целочисленным делением на 3
        x = x//3
    # Вернуть полученную строку
    return s
m = []
for n in range(1,200):
    b = cc3(n)
    if n%2==0:
        # Добавить к результату префикс '1' и суффикс '00',
        # чтобы получить новую строку
        b = '1' + b + '00'
    else:
        # Вычислить сумму цифр строки b как целого числа
        s = sum(int(d) for d in b)
        # Преобразовать полученное строковое представление
        # обратно в десятичное число
```

```
        b = b + cc3(s)
    r = int(b,3)
    if r>168:
        m.append(n)
print(min(m))
```

Результат работы программы:

10

Ответ : 10

Задание №11 (7667)

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

1. Число N переводится в систему счисления с основанием 19.
2. Далее эта запись обрабатывается по следующему правилу:
  а) согласные буквы (B, C, D, F, G, H) заменяются на 5;
  б) в начало полученной записи дописывается остаток от деления числа N на 19 в 19-ричной системе счисления;
  в) две последние цифры записи переставляются в начало (например, из строки 12345 получается 45123).
3. Действия а)-в) в п. 2. повторяются еще раз.

Полученная таким образом запись записью искомого числа R в системе счисления с основанием 19. Укажите максимальное число R с суммой цифр, кратной 7, которое может быть получено в результате работы алгоритма. Запишите его в ответе в десятичной системе счисления.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241249?t=1h52m25s

Решение

Сначала необходимо создать функцию для преобразования числа в систему счисления с основанием 19. Для этого мы перебираем все шестизначные числа от 100000 до 999999 (не включая миллион). Далее получаем представление

числа в системе счисления с основанием 19. Все согласные буквы в этой записи заменяются на цифру '5'. Например, буква 'b' меняется на '5'.

Затем к началу получившейся строки добавляется остаток от деления исходного числа на 19 в девятнадцатеричной системе. Таким образом, две последние цифры числа перемещаются в начало строки. После этого операция повторяется снова.

Далее полученное число преобразуется обратно в десятичное представление. Нам требуется найти максимальное значение среди всех таких чисел, сумма цифр которых делится на 7. Если сумма цифр кратна 7, добавляем это число в список m. В конце выводится максимальный элемент списка m.

Есть несколько вопросов, которые остаются неясными. Например, стоит ли учитывать возможные лидирующие нули, которые могут возникнуть после перемещения двух последних цифр в начало числа. Также непонятно, нужно ли выполнять перевод в десятичную систему перед проверкой условия делимости суммы цифр на 7. Проверим правильность реализации, запустив программу на миллионе чисел.

```
from string import printable
# Функция для преобразования числа в строку с основанием 19.
def cc19(x):
    if x==0: return '0'
    s = ''
    while x>0:
        # Добавляем символ из строки printable по индексу
        # (остаток от деления на 19).
        s = printable[x%19]+s
        # Обновить значение x целочисленным делением на 19
        x = x//19
    return s
m = []
# Проходим по числам от 100000 до 999999 включительно.
for n in range(100000,1000000):
   # Преобразуем число в строку с основанием 19.
    b = cc19(n)
    # Заменяем символы 'b', 'c', 'd', 'f', 'g' и 'h' на '5'.
    b = b.replace('b','5').replace('c','5').replace('d','5').replace('f','5')\
        .replace('g','5').replace('h','5')
    b = cc19(n%19) + b
    b = b[-2]+b[-1] + b[:-2]

    b = b.replace('b','5').replace('c','5').replace('d','5').replace('f','5')\
        .replace('g','5').replace('h','5')
    # Преобразуем остаток от деления числа на 19 в строку
    # с основанием 19 и добавляем его к строке b.
    b = cc19(n%19) + b
```

```
    # Меняем местами последние два символа строки b.
    b = b[-2]+b[-1] + b[:-2]
    # Преобразуем строку b обратно в целое число с основанием 19.
    r = int(b,19)
    # Проверяем, делится ли сумма цифр числа r на 7 без остатка.
    if sum(int(d) for d in str(r))%7==0:
        # Если да, добавляем число r в список m.
        m.append(r)
# Выводим максимальное значение из списка m.
print(max(m))
```

Результат работы программы:

893871724

Ответ : 893871724