



Задание №14, Системы счисления

Теория

Система счисления - это способ представления числа. Одно и то же число может быть представлено в различных видах. Например, число 200 в привычной нам десятичной системе может иметь вид 11001000 в двоичной системе, 310 в восьмеричной и C8 в шестнадцатеричной.

Для систем счисления, базис которых больше 10, после цифры 9 следуют буквы латинского алфавита, также обозначающие цифры. Например, в **шестнадцатеричной** системе счисления для записи числа используются цифры от 0 до 9 и буквы A,B,C,D,E,F, которые соответственно обозначают числа 10,11,12,13,14,15.

**Тип int предназначен только для десятичной системы счисления, поэтому другие типы храним в виде строки str*

Перевод десятичного числа в другую систему счисления

Руками

Чтобы перевести целое положительное десятичное число в систему счисления с другим основанием, нужно это число разделить на основание. Полученное частное снова разделить на основание, и дальше до тех пор, пока частное не окажется меньше основания. В результате записать в одну строку последнее частное и все остатки, начиная с последнего.

$$41_{10} = 101001_2$$

Кодом

Повторяем тот же алгоритм, что и руками. Для списка букв, обозначающих цифры больше 9, используем готовый латинский алфавит из класса string. Ниже представлена универсальная функция для перевода числа в другую систему счисления:

```
import string

def convert_to(n, base):
    digits = string.digits + string.ascii_uppercase
    res = ''
    while n > 0:
        res += digits[n % base]
```

```
n //= base
return res[::-1]
```

Для двоичной, восьмеричной и шестнадцатеричной систем счисления можно использовать встроенные команды Python: **bin()**, **oct()** и **hex()** соответственно. Важно учесть, что они ставят в начале 2 вспомогательных символа, которые можно убрать срезом [:2]

Перевод числа в десятичную систему счисления

Руками

Преобразовать число из любой системы счисления в десятичную можно следующим образом: каждый разряд числа необходимо умножить на X^n , где X - основание исходного числа, n - номер разряда. Затем суммировать полученные значения.

$$X_2 = A_n \cdot 2^{n-1} + A_{n-1} \cdot 2^{n-2} + A_{n-2} \cdot 2^{n-3} + \dots + A_2 \cdot 2^1 + A_1 \cdot 2^0$$

$$E8F(16) = 14 \cdot 16^2 + 8 \cdot 16^1 + 15 \cdot 16^0 = 3584 + 128 + 15 = 3727(10)$$

Кодом

В Python можно перевести число в десятичный вид из системы счисления с базисом, не превышающим 36, с помощью функции **int(x,y)**, где **x** - это строчное представление переводимого числа, а **y** - его система счисления

Алгоритм решения

1-ый тип

На перевод числа в другую систему счисления и подсчёт встречающихся цифр.

Найдите количество уникальных цифр в двадцатеричной записи числа $7777^{290} - 777^{29} + 77^2 - 7$.

1. Внимательно записываем число в переменную:

```
a = 77777 ** 290 - 777 ** 29 + 77 ** 2 - 7
```

2. Переводим число в двадцатеричную систему счисления с помощью универсальной функции выше, вызвав **convert_to(a, 20)**, либо пишем алгоритм перевода конкретно под эту задачу. Будем добавлять все остатки от деления этого числа на базис 20 в список. Элементы этого списка в обратном порядке и будут представлять собой 20-тиричную запись числа:

```
a_20 = []
while a > 0:
    a_20 += [a % 20] # к списку прибавляем список
    a //= 20
print(a_20[::-1])
# [1, 13, 10, 4, 2, 18, 0, 12, 6, 12, 16 ...
# Для сравнения:
# print(convert_to(77777 ** 290 - 777 ** 29 + 77 ** 2 - 7, 20))
# 1DA42I0C6CG...
```

3. Применим функцию **set()** к полученному списку a_20, чтобы получить только уникальные значения, а после применяем функцию **len()**, чтобы узнать их количество:

```
print(len(set(a_20)))
```

4. Получаем ответ **20**. Весь код:

```
a = 77777 ** 290 - 777 ** 29 + 77 ** 2 - 7
a_20 = []
while a > 0:
    a_20 += [a % 20]
    a //= 20
print(len(set(a_20)))
```

2-ой тип

На перебор x -числа для удовлетворения условия

DE4AE1

Значение арифметического выражения $7^{170} + 7^{100} - x$, где x – целое положительное число, не превышающее 2030, записали в 7-ричной системе счисления. Определите наибольшее значение x , при котором количество нулей в 7-ричной записи числа, являющегося значением данного арифметического выражения, максимально.

В ответе запишите число в десятичной системе счисления.

1. В начале введем переменную, которую будем использовать для хранения наибольшего найденного количества нулей и соответствующего x :

```
zero_count_max = (0, 0)
```

2. Запишем цикл для x для указанного диапазона $1 \leq x \leq 2030$ и перепишем выражение в переменную:

```
for x in range(1, 2030+1):  
    a = 7**170 + 7**100 - x
```

3. В цикле запишем алгоритм перевода числа в семеричную систему счисления:

```
a_7 = ""  
while a > 0:  
    a_7 += str(a % 7)  
    a //= 7  
a_7 = a_7[::-1]
```

4. Теперь будем считать количество нулей в семеричной записи, если оно окажется больше переменной (предыдущего максимума) или равно ему (т.к. нам нужен наибольший x), то мы будем обновлять переменную, записывая новый максимум и соответствующий x :

```
if zero_count_max[0] <= a_7.count('0'):
    zero_count_max = (a_7.count('0'), x)
```

5. В конце выводим эту переменную и получаем левое значение в качестве ответа: **1715**. Весь код:

```
zero_count_max = (0, 0)
for x in range(1, 2030+1):
    a = 7**170 + 7**100 - x
    a_7 = ''
    while a > 0:
        a_7 += str(a % 7)
        a //= 7
    a_7 = a_7[::-1]
    if zero_count_max[0] <= a_7.count('0'):
        zero_count_max = (a_7.count('0'), x)
print(zero_count_max) # (73, 1715)
```

3-ий тип

На перебор x -цифры для удовлетворения условия

412BA2

Операнды арифметического выражения записаны в системе счисления с основанием 19.

$$348x79643_{19} + 16x52_{19} + 43x7_{19}$$

В записи чисел переменной x обозначена неизвестная цифра из алфавита 19-ричной системы счисления. Определите наибольшее значение x , при котором значение данного арифметического выражения кратно 18. Для найденного x вычислите частное от деления значения арифметического выражения на 18 и укажите его в ответе в десятичной системе счисления. Основание системы счисления указывать не нужно.

1. Т.к. x - это цифра из 19-тиричной системы счисления, то она может принимать любое значение из этого набора: 0123456789ABCDEFGHI. Чтобы не писать его вручную, создадим переменную с цифрами вплоть

до 36-ой системы счисления (10 десятичных цифр + 26 букв латинского алфавита), взяв готовые строки из встроенного класса **string**:

```
import string
digits = string.digits + string.ascii_uppercase
# '0123456789ABCD...XYZ'
```

2. Создадим цикл в котором **x** будет принимать только допустимые значения для 19-тиричной системы счисления:

```
for x in digits[:19]:
```

3. Внутри цикла обозначим за **a** выражение из условия. Каждый член выражения необходимо привести к десятичному виду, чтобы произвести арифметическое сложение (в противном случае, мы будем просто склеивать строки), для этого используем функцию **int()** для базиса 19. С помощью **интерполяции строк** "внедряем" **x** прямо в выражение. Для этого перед строками пишем **f**, а саму переменную берём в фигурные скобки. Получаем такой результат:

```
a = int(f'98{x}79731', 19) + int(f'36{x}14', 19) + int(f'73{x}4', 19)
```

4. Дальше просто проверяем условие. Если оно выполняется, то выводим первый подходящий ответ (т.к. ищем наименьший **x**) и можем завершить цикл:

```
if a % 18 == 0:
    print(a // 18)
    break
```

5. Получаем ответ: **467926139**. Весь код:

```
import string
digits = string.digits + string.ascii_uppercase
print(digits[:19])
```

```
for x in digits[:19]:  
    a = int(f'98{x}79731', 19) + int(f'36{x}14', 19) + int(f'73{x}4', 19)  
    if a % 18 == 0:  
        print(a // 18)  
        break
```