Задание №23, Рекурсия, количество программ

Теория

> *Базовую теорию по рекурсии см. в методичке по 16 заданию*

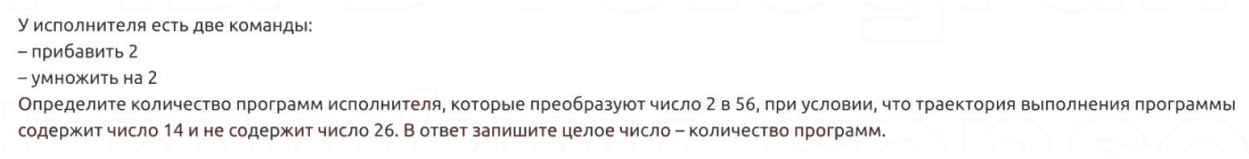
Задачу можно разделить на несколько типов по двум признакам:

1. В зависимости от операций:
    a. Увеличивающие ⇒ **if start > end: return 0**
    b. Уменьшающие ⇒ **if start < end: return 0**
2. В зависимости от условия, найти количество программ, преобразующие *n* в *m*:
    a. Без доп. условий ⇒ print(**F(n,m)**)
    b. Траектория должна содержать число *k* ⇒ **F(n,k)*F(k,m)**
    c. Траектория не должна содержать число *k* ⇒ **if n == k: return 0**
    d. Объединение b и c

- **Усложнённые, СтатГрад: ограниченное число операций, не более i подряд и тд.*

План решения

Объединенный тип:


У исполнителя есть две команды:
– прибавить 2
– умножить на 2
Определите количество программ исполнителя, которые преобразуют число 2 в 56, при условии, что траектория выполнения программы содержит число 14 и не содержит число 26. В ответ запишите целое число – количество программ.

1. Объявляем функцию:

```
def F(start,end):
```

2. Пишем тело функции:
   a. Условие выхода из рекурсии, start успешно дошёл до end:

```
if start == end:
    return 1
```

   b. Условие выхода из рекурсии, start не дошёл до end и никогда не дойдёт, в зависимости от типа операций ставим либо “**>**”, либо “**<**”

```
if start > end:
   return 0
```

   c. В остальных случаях возвращаем сумму рекурсивных вызовов для всех операций в условии:

```
return F(start + 2, end) + F(start * 2, end)
```

3. Если траектория по условию не должна содержать какое-то число , то прописываем это условие в пункте 2.b:

```
if start > end or start == 26:
    return 0
```

4. Завершив писать функцию, выводим то, что спрашивается в условии. Если траектория по условию должна содержать какое-то число, то выводим произведение функций от начального числа до заданного и от заданного до конечного:

```
print( F(2,14) * F(14,56) )
```

Вся функция:

```
def F(start, end):
    if start == end:
        return 1
    if start > end or start == 26:
        return 0
    return F(start + 2, end) + F(start * 2, end)
print( F(2,14) * F(14,56) )
```

Алгоритм решения

Решение руками (доп.):

У исполнителя есть две команды:
— прибавить 5;
— умножить на 2.
Первая из них увеличивает число на 5, вторая увеличивает число в 2 раза. Сколько существует программ, которые преобразуют исходное число 1 в число 24 и траектория вычисления которых содержит число 12?

Поставим начальному числу значение 1. Для начального и каждого последующего числа используем следующий алгоритм: к значения чисел, которые можно получить из текущего, будем прибавлять значение текущего. Сначала найдем количество программ, которые преобразуют исходное число

в обязательное, после чего количество программ, которые преобразуют обязательное число в конечное

1	2	3	4	5	6	7	8	9	10	11	12
1	1	0	1	0	1	1	1	1	0	1	+1
											+1
											2

12	13	14	15	16	17	18	19	20	21	22	23	24
2	0	0	0	0	2	0	0	0	0	2	0	2

Ответ: 2

СтатГрад, не более 3 операций подряд:

У исполнителя есть две команды:
— прибавить 1
— умножить на 3
Сколько существует программ, которые преобразуют исходное число 1 в число 30, и при этом никакая команда не повторяется более трёх раз подряд?

1. Пишем программный код как для обычной задачи по плану решения:

```
def F(n, m):
    if n > m:
        return 0
    if n == m:
        return 1
    return F(n + 1, m) + F(n * 3, m)
print(F(1,30))
```

2. Вводим счётчик k и j для операций сложения и умножения соответственно. Будем передавать в рекурсивные вызовы k + 1 для операции сложения и j + 1 для операции умножения, тем самым показывая, сколько раз эти операции были произведены. Также для

каждого рекурсивного вызова будем обнулять другой счётчик, передавая 0, тем самым счётчики будут показывать именно количество операций подряд. Если это количество станет нарушать условие, то просто вернём из функции значение 0, как для неподходящего пути. Весь код:

```
def F(n, m, k, j):
    if n > m or k > 3 or j > 3:
        return 0
    if n == m:
        return 1
    return F(n + 1, m, k + 1, 0) + F(n * 3, m, 0, j + 1)
print(F(1,30, 0, 0))
```