Задача 1. Сложный уровень. Подробный разбор.

У исполнителя Дмитрия Андреевича есть мешок с известным количеством орехов. Он преобразует число, полученное после пересчёта орехов в мешке.

У исполнителя есть *3 команды*, которым присвоены номера:

1. **Прибавить 1 орех**
2. **Вычесть корень из предыдущего квадратного значения и прибавить корень из следующего квадратного значения**
3. **Прибавить сумму цифр числа**

Первая из них увеличивает количество орехов на 1, вторая уменьшает количество орехов на корень из предыдущего квадратного значения (для числа 6 предыдущим квадратным значением является число 4), а затем увеличивает на корень из следующего квадратного значения (для числа 12 следующим квадратным значением является число 16). Если число является квадратным, то увеличиваем значение на единицу. Третья команда прибавляет сумму цифр числа. Программа для исполнителя Дмитрия Андреевича – это последовательность команд.

Сколько различных результатов можно получить из исходного числа ***7*** после выполнения программы, содержащей ровно ***7*** команд?

Ответ: *45*

Решение:

```
#Task 1
s = set()
def f(a, n):
    if n > 7:
        return 0
    if n == 7:
        s.add(a)
    k = 1
    while True:
        if (a + k) ** 0.5 == float(int((a + k) ** 0.5)):
            sl = int((a + k) ** 0.5)
            break
        k += 1
    while True:
        if (a - k) ** 0.5 == float(int((a - k) ** 0.5)):
            pr = int((a - k) ** 0.5)
            break
        k += 1
    sc = 0
    for i in str(a):
        sc += int(i)
    f(a + 1, n + 1)
    if a ** 0.5 == float(int(a ** 0.5)):
        f(a + 1, n + 1)
    else:
        f(a - pr + sl, n + 1)
    f(a + sc, n + 1)
f(7, 0)
print(len(s))
```

Решение основано на применении рекурсивной функции.

На вход функция получает исходное значение, и начальную глубину рекурсии (0), которую мы будет увеличивать на единицу при каждом “углублении”.

Глубина рекурсии – это количество команд в программе. Если представить рекурсию в виде дерева (смотрите ниже), то можно обозначить за глубину количество уровней в дереве, где нулевой уровень будет иметь значение глубины рекурсии равное единице.

В пятой строке при выполнении условия n > 7 функция будет возвращать 0, т.к. нас не интересуют значения, получаемые в результате выполнения программ, содержащих более 7 команд.

В шестой строке при выполнении условия ***n == 7*** функция будет добавлять в множество значение, которое было получено в результате выполнения программы ровно из 7 команд. Главным свойством множества является то, что оно может хранить только уникальные значения, поэтому оно не будет содержать двух одинаковых результатов выполнения разных программ.

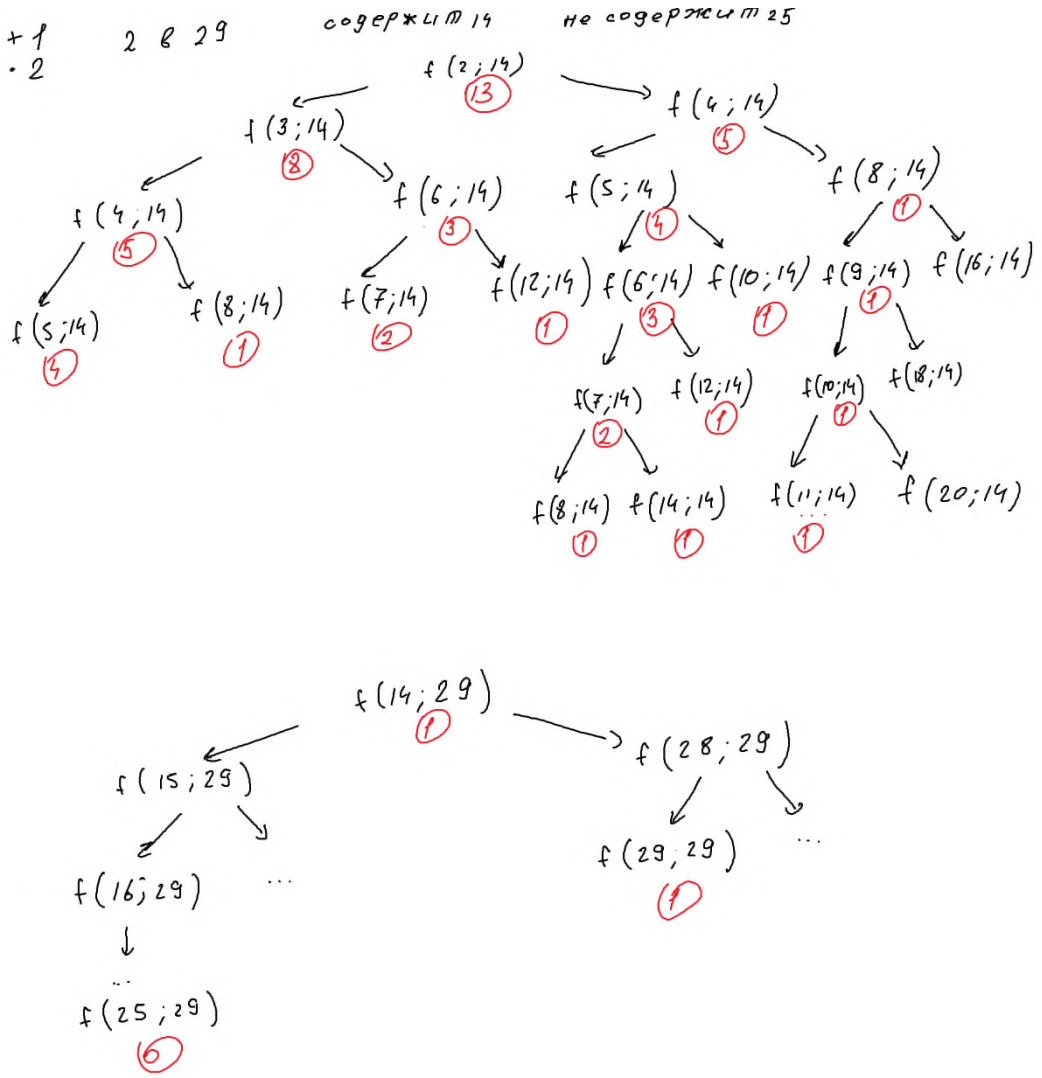
При помощи двух циклов while мы вычисляем корни из следующего и предыдущего квадратных чисел.

Затем в отдельную переменную мы записываем сумму цифр числа, после чего прописываем сами элементы рекурсии для каждой из команд. Таким образом функция будет вызывать саму себя (строки 22, 24 и 26, 27)

Отдельно прописывает условие, в котором при условии, что само число является квадратным, то увеличиваем его на единицу, а в ином случае вычитаем переменную *pr* и прибавляем переменную *sl* (обе были получены выше в результате выполнения циклов while)

После того, как мы прописали функцию, запускаем её и выводим длину множества, которое было получено после выполнения рекурсивной функции

Визуализируем принцип работы рекурсии на примере задачи с преобразованием числа ***2*** в число ***29*** с использованием команд ***+1*** и ****2,*** при условии, что траектория вычислений будет содержать число ***14*** и не будет содержать число ***25***.



$13 \cdot 1 = 13$

Ответ: 13

Задача 2. Сложный уровень.

У исполнителя Дмитрия Андреевича есть мешок с известным количеством орехов. Он преобразует число, полученное после пересчёта орехов в мешке.

У исполнителя есть *три команды*, которым присвоены номера:

1. **Прибавить 1 орех**
2. **Прибавить предыдущее количество орехов и вычесть из этого числа остаток от деления глубины рекурсии на 16**
3. **Дописать к количеству орехов справа остаток от деления квадрата числа на 3**

Первая из них увеличивает количество орехов на экране на ***1***, вторая увеличивает его на предыдущее количество орехов (к числу 5 прибавляется 4, к числу 4 прибавляется 3 и т.д.) и вычитает остаток от деления глубины рекурсии на ***16***, третья дописывает к количеству орехов справа остаток от деления квадрата числа на ***3***. Программа для Дмитрия Андреевича – это последовательность команд.

Сколько существует таких программ, которые преобразуют исходное число ***123*** в число ***321,*** и при этом траектория вычислений программы не содержит число ***222***?

Ответ: 45

Решение:

```
def f(a, b, k):
    if a > b or a == 222:
        return 0
    if a == b:
        return 1
    if a < b:
        s = str(a)+str(a**2 % 3)
        return f(a + 1, b, k + 1) + f(a + (a - 1) - k % 16, b, k + 1) + f(int(s), b, k + 1)
print(f(123, 321, 0))
```

Решение основано на применении рекурсивной функции. На вход получаем два значения: исходное и конечное числа, а также начальное значение счётчика глубины функции (*0*). Функция возвращает ноль при *a* > *b* и при *a* == *222* (траектория программы не должна содержать это число). При *a* == *b* возвращает единицу, и при *a* < *b* возвращает количество возможных программ из заданного набора команд.

Задача 3. Обычный уровень.

У исполнителя Дмитрия Андреевича есть мешок с известным количеством орехов. Он преобразует число, полученное после пересчёта орехов в мешке.

У исполнителя есть *три команды*, которым присвоены номера

1. **Изъять 1 орех**
2. **Изъять 3 ореха**
3. **Из каждого разряда числа изъять 1 орех**

Первая из них уменьшает количество орехов на экране на 1, вторая уменьшает его на 3, третья вычитает из каждого разряда 1. Программа для Дмитрия Андреевича – это последовательность команд.

Сколько существует таких программ, которые преобразуют исходное число ***32*** в число ***16***, и при этом траектория вычислений программы содержит число ***25*** и не содержит числа ***19***?

Ответ: 63

Решение:

```
1  def f(a, b):
2      if a < b or a == 19:
3          return 0
4      if a == b:
5          return 1
6      if a > b:
7          return f(a - 1, b) + f(a - 3, b) + f(a - 11, b)
8  print(f(32,25)*f(25,16))
```

Решение основано на применении рекурсивной функции. На вход получаем два значения: исходное и конечное числа. Функция возвращает ноль при $a < b$ и при $a == 19$ (траектория программы не должна содержать это число). При $a == b$ возвращает единицу, и при $a > b$ возвращает количество возможных программ из заданного набора команд. Стоит заметить, что так как диапазон вычислений состоит только из двузначных чисел, то команда “Из каждого разряда вычесть 1” равносильна формулировке “Вычесть 11”. При запуске функции в 8 строчке не забываем “выколоть” точку, которую должна содержать траектория вычислений.

Задача 4. Обычный уровень.

У исполнителя Дмитрия Андреевича есть мешок с известным количеством орехов. Он преобразует число, полученное после пересчёта орехов в мешке.

У исполнителя есть три команды, которым присвоены номера:

1. **Прибавить 1 орех**
2. **Произвести умножение количества орехов на 2**
3. **Дублировать справа последнюю цифру количества орехов**

Первая из них увеличивает количество орехов на экране на 1, вторая увеличивает его в 2 раза, третья дублирует последнюю цифру количества орехов (4 станет 44). Программа для Дмитрия Андреевича – это последовательность команд.

Сколько существует таких программ, которые преобразуют исходное число ***3*** в число ***333***, и при этом траектория вычислений программы содержит число ***110*** и не содержит число ***220***?

Ответ: 246344

Решение:

```
1  def f(a, b):
2      if a > b or a == 220:
3          return 0
4      if a == b:
5          return 1
6      if a < b:
7          return f(a + 1, b) + f(a * 2, b) + f(int(str(a)+str(a)[-1:]), b)
8  print(f(3,110)*f(110,333))
```

Решение основано на применении рекурсивной функции. На вход получаем два значения: исходное и конечное числа. Функция возвращает ноль при *a* > *b* и при *a* == *220* (траектория программы не должна содержать это число). При *a* == *b* возвращает единицу, и при *a* < *b* возвращает количество возможных программ из заданного набора команд. Дублирование последней цифры числа осуществлено через конкатенацию (соединение) строк и срез, отделяющий последнюю цифру числа. При запуске функции в 8 строчке не забываем “выколоть” точку, которую должна содержать траектория вычислений.

Задача 5. Усложнённый уровень.

У исполнителя Дмитрия Андреевича есть мешок с известным количеством орехов. Он преобразует двоичное число, полученное после пересчёта орехов в мешке.

У исполнителя есть *три команды*, которым присвоены номера

1. **Прибавить 1 орех**
2. **Умножить количество орехов на 3**
3. **Если сумма единиц в числе является чётной, то дописать к числу 0, в противном случае дописать 1**

Первая из них увеличивает количество орехов на 1, вторая умножает его на 3, третья убирает последнюю цифру числа, если оно чётное. Программа для Дмитрия Андреевича – это последовательность команд.

Сколько существует таких программ, которые преобразуют исходное число ***111*** в число ***1111111***, и при этом траектория вычислений программы содержит число ***60*** и не содержит число ***74***?

Ответ: 608

Решение:

```
1   def f(a, b):
2       if int(a, base = 2) > int(b, base = 2) or a == bin(74)[2:]:
3           return 0
4       if int(a, base = 2) == int(b, base = 2):
5           return 1
6       if int(a, base = 2) < int(b, base = 2):
7           if a.count("1") % 2 == 0:
8               s = a + "0"
9           else:
10              s = a + "1"
11          return f(bin(int(a, base = 2) + 1)[2:], b) + f(bin(int(a, base = 2) * 3)[2:], b) + f(s, b)
12  print(f("111", bin(60)[2:]) * f(bin(60)[2:], "1111111"))
```

Решение основано на применении рекурсивной функции. На вход получаем два значения: исходное и конечное числа. Функция возвращает ноль при $a > b$ и при a равном двоичной записи числа 74 (траектория программы не должна содержать это число). При $a == b$ возвращает единицу, и при $a < b$ возвращает количество возможных программ из заданного набора команд, проверяя на чётность количество единиц в числе a и записывая в отдельную переменную s результат работы команды 3 с переменной a (конкатенация). При запуске функции в 12 строчке не забываем “выколоть” точку, которую должна содержать траектория вычислений.

Задача 6. Усложнённый уровень.

У исполнителя Дмитрия Андреевича есть мешок с известным количеством орехов. Он преобразует число, полученное после пересчёта орехов в мешке.
У исполнителя есть *три команды*, которым присвоены номера

1. **Прибавить 1 орех**
2. **Умножить количество орехов на 3 и нацело разделить на 2**
3. **Прибавить сумму цифр числа, умноженную на 4**

Первая из них увеличивает количество орехов на 1, вторая умножает его на 3 и нацело делит на 2, третья увеличивает разряд десятков на 4. Программа для Дмитрия Андреевича – это последовательность команд.

Сколько различных результатов можно получить из исходного числа ***7*** после выполнения программы, содержащей ровно ***5*** команд?

Ответ: 120

Решение:

```
s = set()
def f(a, n):
    if n > 5:
        return 0
    if n == 5:
        s.add(a)
    f(a + 1, n + 1)
    f(a * 3 // 2, n + 1)
    t = 0
    for i in str(a):
        t += int(i)
    f(a + t * 4, n + 1)
f(7, 0)
print(len(s))
```

Решение основано на применении рекурсивной функции. На вход функция получает исходное значение, и начальную глубину рекурсии (0), которую мы будет увеличивать на единицу при каждом “углублении”. При глубине функции *5* мы добавляем в множество последнее значение переменной *a*, а так как множества хранят только уникальные значения, то все результаты будут различными. В отдельной переменной *t* я посчитал сумму цифр переменной *a*, что и использовал в описании параметров функции для третьей команды. Далее мы отдельно запускаем функцию, а в ответ выводим длину полученного множества.

Задание 7. Авторская задача.

Однажды на мир обрушилась катастрофа: с небес на огромных летающих тарелках прилетели инопланетяне, представители одной из множества государств галактики - федерации Лакония. Под угрозой уничтожения они поставили человечеству сложную задачу, которая в рамках галактического договора о защите молодых цивилизаций служит для проверки людей на то, достойны ли они перейти в космическую эру и путешествовать за пределами планеты, получить передовые знания многочисленных цивилизаций в составе галактического альянса, а также вести сотрудничество с инопланетными расами. На плечи героя - парламентёра Дмитрия Андреевича водрузили огромную ответственность по спасению планеты. Помогите ему решить задачу инопланетян: преобразуйте предоставленное инопланетянами двоичное число.

У Дмитрия Андреевича есть три команды, которым присвоены номера:

1. **Вычесть 1**
2. **Если число является квадратным, то извлечь из него корень**
3. **Если вторая цифра числа является единицей, то убрать первую цифру слева.**

Первая из них уменьшает число на 1, вторая извлекает из него корень при условии того, что число является квадратным (4, 9, 16 и т.д.) третья убирает первую цифру числа при условии того, что вторая цифра является единицей. Программа – это последовательность команд.

Помогите Дмитрию Андреевичу спасти мир, посчитав количество различных результатов, которые можно получить из исходного числа, содержащего единицу и ***17*** нулей (100...00), после выполнения программы, содержащей ровно ***15*** команд?

Ответ: 16

Решение:

```python
array = set()
def f(a,k):
    if k > 15:
        return 0
    if k == 15:
        array.add(a)
    f(bin(int(a, base=2)-1)[2:], k+1)
    if int(a, base=2) ** 0.5 == float(int(int(a, base=2) ** 0.5)):
        f(bin(int(int(a, base=2)**0.5))[2:], k+1)
    if a[1] == "1":
        f(a[1:], k+1)
f("1"+17*"0", 0)
print(array, len(array))
```

Решение основано на применении рекурсивной функции. На вход получаем значение, и начальную глубину рекурсии (0), которую мы будет увеличивать на единицу при каждом “углублении”. При глубине функции ***15*** мы добавляем в множество последнее значение переменной ***a***, а так как множества хранят только уникальные значения, то все результаты будут различными. Проверка числа на то, что оно квадратное происходит в десятичной системе (переводим двоичную в десятичную функцией с двумя параметрами: int(a, base = 2), а десятичную в двоичную с помощью функции bin(a)[2:] где значение в квадратных скобочках (срез) обрезает полученное число на ***2*** знака слева, т.к. они после перевода являются служебными), после чего мы можем извлекать из него корень. Обрезание числа на одну цифру слева так же происходит через срез. В ответ мы выводим длину полученного множества.