1) (база) Дан автомат, который преобразует число. У него есть три команды:

1. Прибавить 3
2. Прибавить 7
3. Возвести в квадрат

Первая команда увеличивает число на 3. Вторая команда увеличивает число на 7. Третья команда возводит число в квадрат (например, для исходного числа 17 результатом работы команды будет 289).

Сколько существует различных последовательностей команд, в результате исполнения которых из числа 13 получится число 92 ?

Решение:

def F(start,end):  
 if start>end:  
 return 0  
 if start==end:  
 return 1  
 k=F(start+3,end)  
 k+=F(start+7,end)  
 k+=F(start\*\*2,end)  
 return k  
print(F(13,92)) #Ответ:25744

Объяснение:

В основе решения этого задания лежит рекурсивный алгоритм, который принимает на вход начальное число и то, которое мы должны получить на выходе. Для начла мы ставим условие start>end, ведь все наши команды могут лишь что то прибавлять, делать начальное число start больше, а следовательно если start в один момент станет больше end, они уже никогда не сравняются. Далее прописываем условие start==end, если оно исполнилось, значит мы смогли подобрать нужную последовательность команд. Затем нам осталось ввести счетчик К, который будут накапливать в себе количество всех подходящих последовательностей команд. Его мы и возвращаем в функции F.

1. (база) Дан автомат, который преобразует число. У него есть три команды:
2. Вычесть 2
3. Вычесть 5
4. Разделить нацело на 3 (остаток отбрасываем)

Первая команда вычитает из начального числа 2. Вторая вычитает 5. Третья уменьшает число в 3 раза (Результатом применения команды к числу 11 будет 3).

Сколько существует различных последовательностей команд, в результате исполнения которых из числа 117 получится число 2 ? Решение:

def F(start,end):  
 if end <start:  
 return 0  
 if start==end:  
 return 1  
 k=F(start,end-2)  
 k+=F(start,end-5)  
 if end%3==0:  
 k+=F(start,end//3)  
 return k  
print(F(2,34))#Ответ 493

Объяснение:

Отличительной чертой этого задания от предыдущего является то, что все команды уменьшают числа, следовательно мы будем рассматривать всю цепочку с конца, сводить end к start. Следует отметить, что если конечное число end не кратно 3, то его нельзя получить из другого числа умножением на 3.

3) (стандартная) Дан автомат, который преобразует число. У него есть три команды:

1. Прибавить 11
2. Прибавить 32
3. Возвести в куб

Первая команда увеличивает число на 11. Вторая команда увеличивает число на 32. Третья команда возводит число в куб (например, для исходного числа 8 результатом работы команды будет 512).

Сколько существует различных последовательностей команд, в результате исполнения которых из числа 4 получится число 211, при этом траектория вычислений содержит 59, но исключает 136 ?

Решение:

def F(start,end):  
 if start==136:  
 return 0  
 if start>end:  
 return 0  
 if start==end:  
 return 1  
 k=F(start+11,end)  
 k+=F(start+32,end)  
 k+=F(start\*\*3,end)  
 return k  
print(F(4,59)\*F(59,211))#42

Объяснение: Эта задача отличается от базовых только лишь тем, что тут есть ограничения на числа, которые либо должны присутствовать в наших вычислениях, либо нет. Если мы встречаем число 136 (if start==136), мы возвращаем 0, так как через него мы проходить не можем. Через 59 мы, напротив, проходить должны. Чтобы это отследить, мы можем посчитать количество команд из 4 в 59 и умножить его на количество команд приводящих из 59 в 211, так как взяв последовательность команд из 4 в 59 можно ее совместить с любой из последовательностей 59-211 и получить в итоге 211.

4) (стандарт) Дан автомат, который преобразует число. У него есть три команды:

1. Вычесть 2
2. Вычесть 11
3. Разделить нацело на 5 (остаток отбрасываем)

Первая команда вычитает из начального числа 2. Вторая вычитает 11. Третья уменьшает число в 5 раз (Результатом применения команды к числу 16 будет 3).

Сколько существует различных последовательностей команд, в результате исполнения которых из числа 112 получится число 5, причем траектория вычислений должна содержать 17, но не должна 34?

Решение:

def F(start,end):  
 if end==34:  
 return 0  
 if end<start:  
 return 0  
 if start==end:  
 return 1  
 k=F(start,end-4)  
 k+=F(start,end-11)  
 if end%5==0:  
 k+=F(start,end//5)  
 return k  
print(F(5,17)\*F(17,112))#Ответ 3030

5) (повышенная сложность) Дан автомат, который преобразует число. У него есть четыре команды:

1. Прибавить 15
2. Умножить на 4 и вычесть остаток от деления на 7
3. Умножить на 3 и нацело разделить на 2 (остаток отбрасывается)
4. Обнулить младший разряд в двоичной записи (если он и так равен нулю, ничего не делать)

Первая команда увеличивает число на 15. Вторая команда увеличивает число в 4 раза и вычитает его остаток от деления на 7.Третья команда умножает число на 3 и делит нацело на 2. Четвертая команда переводит число в двоичную систему счисления, обнуляет младший разряд и возвращает результат в десятичной системе счисления.

Сколько различных результатов работы автомата можно получить из числа 17, если известно, что длинна последовательности команд равна 7?

Решение:

def fourth(x):  
 ma=[]  
 while x>0:  
 ma.append(x%2)  
 x//=2  
 ma[0]=0  
 x1=0  
 for i in range(len(ma)):  
 x1+=ma[i]\*(2\*\*i)  
 return x1  
mas=set()  
def F(value,step):  
 if step>7:  
 return 0  
 if step==7:  
 mas.add(value)  
 F(value+15,step+1)  
 F(value\*4-value\*4%7,step+1)  
 F((value\*3)//2,step+1)  
 F(fourth(value),step+1)  
F(17,0)  
print(len(mas))#Ответ 1764

Объяснение: Функция fourth отвечает за выполнение четвертой команды, переводит число в двоичную с.с. и обратно в десятичную, обнуляя младший разряд. Отличительной чертой этого задания от предыдущих является то, что нам не нужно находить количество программ, а их итоговой результат. Множество mas будет собирать все возможные результаты работы программ длинной 7, так как это не список, дублироваться значения не будут. На вход рекурсивной функции подается два значения – текущее значение и шаги (количество уже выполненных команд). Если количество команд превышает 7, то мы возвращаем 0, дальнейшее рассмотрение нас не интересует. Когда step равен 7, мы записываем в множество текущее значение, если до этого мы такое не встречали, то оно запишется. Далее мы вызываем функцию F, изменяя текущее значение всеми возможными способами и прибавляя 1 к количеству шагов. После написания рекурсии следует ее вызвать (F(17,0)), тогда она отработает и запишет все нужные нам значения в список mas. Остается только вывести его длину.

6) ) (повышенная сложность) Дан автомат, который преобразует число. У него есть четыре команды:

1. Возвести в степень 1,5 (дробную часть отбрасываем)
2. Сделать четное (если уже четное, ничего не делаем)
3. Увеличить на 6
4. Подвергнуть разности систем счисления

Первая команда увеличивает число в корень из себя раз. Вторая команда добавляет 1 если число нечетное, ничего не делает если четное. Третья команда добавляет к числу 6. Четвертая команда заменяет младший разряд в семеричной записи числа на модуль разности заменяемой цифры и младшего разряда записи числа в троичной системе счисления.

Сколько различных результатов работы автомата можно получить из числа 222, если известно, что длинна последовательности команд равна 10?

Решение:

def even(x):  
 if x%2==0:  
 return x  
 else:  
 return x+1  
def difference(x):  
 ma=[]  
 su=0  
 dif=x%3  
 while x>0:  
 ma.append(x%7)  
 x//=7  
 ma[0]=abs(ma[0]-dif)  
 for i in range(len(ma)):  
 su+=ma[i]\*(7\*\*i)  
 return su  
mas=set()  
def F(value,step):  
 if step>10:  
 return 0  
 if step==10:  
 mas.add(value)  
 F(int(value\*\*1.5),step+1)  
 F(even(value),step+1)  
 F(value+6,step+1)  
 F(difference(value),step+1)  
F(222,0)  
print(len(mas))#Ответ 12549

Объяснение: Задача довольно сильно похожа на предыдущую, но есть значительные отличия в действиях. Функция even просто делает ровно то, что описано в условии. Функция difference переводит число в семеричную систему счисления и заменяет младший разряд на модуль разности значения этого разряда и остатка от деления первоначального числа х на 3, затем переводит число обратно в десятичную и возвращает его.

1. (Что то необычное)

В детстве Дмитрий очень любил смотреть «Терминатора», и однажды он решил, что его предназначение - спасти мир. Он узнал, что в Политехе появился суперкомпьютер, это была угроза. Дмитрий много учился, выиграл всерос, поступил в Политех и разнёс суперкомпьютер, теперь миром не будут править машины. Но вот незадача, именно после этого события Владимир Владимирович вежливо попросил ректора Политеха решить одну задачу государственной важности. Ректор был в ярости, когда узнал, что сделал Дмитрий. Теперь второму надо либо решить задачу, либо отправится в армию. Дмитрию надо бы помочь, ведь именно он обеспечил наше свободное, светлое будущее. Итак, задача такова:  
Для суперкомпьютера было приготовлено 5 команд:  
1) Побитовый сдвиг влево  
2) Счет двоичной суммы  
3) Семеричная прибавка  
4) Инверсия  
5) взять корень (дробную часть отбрасываем)

Первая команда совершает сдвиг в двоичной записи числа (10011 превращается в 00111). Вторая команда считает сумму единиц в двоичной записи числа и добавляет в ее конец остаток от деления суммы на 2. Третья команда прибавляет к числу 4, если в семеричной записи оно оканчивается на что то меньшее четырех. Четвертая команда переводит число в двоичную систему счисления и инвертирует старший в ней разряд (0 превращается в 1 и наоборот). Пятая команда берет корень из числа.

А результатом работы компьютера стал бы перебор всех чисел, которые получаются в ходе выполнения последовательностей из команд длинной 6 для первоначального значения 42,таких что траектория вычислений не содержит 0, и нахождение среди них наибольшего полного квадрата. Какое число должен был вывести суперкомпьютер ?

Решение:

def two\_to\_ten(ma):  
 ma.reverse()  
 su=0  
 for i in range(len(ma)):  
 su+=ma[i]\*(2\*\*i)  
 return su  
def inversion(x):  
 ma\_rev=[]  
 while x>0:  
 ma\_rev.append(x%2)  
 x//=2  
 ma\_rev.reverse()  
 if ma\_rev[0]==0:  
 ma\_rev[0]=1  
 else:  
 ma\_rev[0]=0  
 return two\_to\_ten(ma\_rev)  
def shift(x):  
 ma=[]  
 while x>0:  
 ma.append(x%2)  
 x//=2  
 ma.reverse()  
 ma.append(ma[0])  
 ma[0]=0  
 return two\_to\_ten(ma)  
def binary\_account(x):  
 ma=[]  
 while x>0:  
 ma.append(x%2)  
 x//=2  
 ma.reverse()  
 ma.append(sum(ma)%2)  
 return two\_to\_ten(ma)  
answer=[]  
def F(value,step):  
 if step>6:  
 return 0  
 if value==0:  
 return 0  
 if step==6:  
 answer.append(value)  
 F(shift(value),step+1)  
 F(binary\_account(value),step+1)  
 F(inversion(value),step+1)  
 if value%7<4:  
 F(value+4,step+1)  
 F(int(value),step+1)  
F(42,0)  
answer.sort(reverse=True)  
for x in answer:  
 if int(x\*\*0.5)\*\*2==x:  
 print(x)  
 break

Объяснение:

Суть задания примерно такая же, как и в усложнённом уровне, но главные проблемы доставляют нам сами 5 команд. Для начла напишем перевод из двоичной системы счисления в десятичную, эта функция пригодится в нескольких местах в дальнейшем. Далее напишем функцию инверсии, переводим число в двоичную систему счисления и изменяем его старший разряд, возвращаем число в десятичной с.с. Далее пропишем функцию побитового сдвига, сдвиг влево по сути представляет из себя копирование старшего разряда в конец записи. Ну и напишем последнюю функцию - счёт суммы единиц в двоичной записи и прибавление к записи остатка от деления этой суммы на 2. Далее, как в задачах сложного уровня, мы вызываем рекурсии от всех возможных команд и получаем список answer, который содержит все возможные числа, получаемые из команд длинной 6. Затем нам остается пробежаться по этому списку, предварительно отсортировав его по убыванию, и найти полные квадраты. Определить квадрат можно благодаря тому, что функция int откидывает дробную часть числа, рассмотрим пример 25 и 26, для 25 int(5.0)=5, но и int(26\*\*0.5)=5, но 5\*\*2=25 и не равно 26.