Конспект 2

Strim_23_3

Рекурсивный проход траекторий

Мы создадим рекурсивную функцию, которая будет проходить по всем возможным путям вычислений от стартового значения к конечному, заданному нами. Эта рекурсия довольно проста и использует всего два параметра. Первый параметр – это текущее значение, которое изменяется при каждом вызове функции. Второй параметр определяет конечное значение, являющееся нашей целью.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241276?t=0h3m0s

Задача № 1 (2487)

Исполнитель Калькулятор преобразует число на экране. У исполнителя есть две команды, которым присвоены номера:

1. Прибавить 2
2. Умножить на 2

Программа для исполнителя Калькулятор – это последовательность команд. Сколько есть программ, которые число 1 преобразуют в число 24?

Решение

Чтобы решить эту задачу программным способом, мы напишем рекурсивную функцию. Назовём её f с двумя параметрами: текущим числом и конечным числом. Конечный аргумент задаёт число, к которому мы хотим прийти, 24, а текущий аргумент отслеживает состояние вычисления в каждый момент времени; он будет меняться после каждого действия.
В этой рекурсии есть три основных условия. Во-первых, если текущее число становится больше конечного, значит, эта ветвь вычислений ошибочна, и мы возвращаем 0, указывая, что такой путь не подходит. Во-вторых, если текущее число совпадает с конечным, значит, мы достигли цели, и функция возвращает 1, показывая, что найден подходящий путь. Наконец, если текущее число меньше конечного, то количество возможных путей вычисляется как сумма всех путей, которые могут возникнуть из двух последующих шагов: прибавления 2 и умножения на 2. Это означает, что мы должны сложить результаты вызовов функции f для значений c + 2 и c * 2.
Эта простая рекурсия пройдётся по всем возможным траекториям вычислений и подскажет итоговое количество решений. Всё, что остаётся сделать, — это вывести результат вызова функции с начальным значением 1 и конечным значением 24.

```
def f(curr, end):
    # Если текущее число превысило конечное, то данный путь неверный
    if curr > end:
        return 0
```

```
    # Если текущее число достигло конечного, то найдена одна программа
    if curr == end:
        return 1
    # Если текущее число еще меньше конечного, считаем варианты дальше
    if curr < end:
        return f(curr + 2, end) + f(curr * 2, end)
# Вызываем функцию с начальными условиями: текущее число = 1, конечное число = 24
print(f(1, 24))
```

С помощью сайта https://recursion.vercel.app/, мы можем визуализировать выполнение рекурсии в этой программе и наблюдать последовательность вызовов. Значение переменной C увеличивается на 2, пока не превысит 25, становясь слишком большим. Постепенно рекурсия исследует все возможные ветви вычислений. Видно, что рекурсия систематически проверяет различные пути, суммируя правильные и отметая неправильные. В итоге программа находит 32 различных способа преобразования числа 1 в 24.

Ответ: 32

Задача №2 (2480)

Исполнитель Калькулятор преобразует число на экране. У исполнителя есть три команды, которым присвоены номера:

1. Прибавить 1
2. Прибавить 2
3. Умножить на 2

Программа для исполнителя Калькулятор – это последовательность команд. Сколько существует программ, для которых при исходном числе 3 результатом является число 13 и при этом траектория вычислений не содержит число 8?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241276?t=0h15m15s

Решение

Основная идея заключается в том, чтобы исключить все возможные пути, проходящие через число 8. Для этого мы используем рекурсивную функцию, которая проверяет текущее состояние и возвращает 0, если текущие значения не подходят под ограничения задачи.
Основная функция f принимает два аргумента: текущее значение (curr) и целевое значение (end). Если текущее значение превышает целевое или равно 8, функция немедленно возвращает 0, исключая эту траекторию из дальнейшего рассмотрения.
Если текущее значение достигло целевого, возвращается 1, поскольку такая траектория считается успешной.
В остальных случаях функция вызывает себя трижды для каждого возможного действия (прибавление 1, прибавление 2 и умножение на 2) и суммирует результаты. Чтобы найти количество допустимых программ, начинаем с начального числа 3 и требуемого результата 13.

```
def f(curr, end):
    # Проверка условий выхода
```

```
    if curr > end or curr == 8: return 0
    if curr == end: return 1
    # Рекурсивный вызов для всех возможных действий
    if curr < end:
        return f(curr + 1, end) + f(curr + 2, end) + f(curr * 2, end)
print(f(3,13))
```

Таким образом, программа вернет количество различных последовательностей команд, приводящих от числа 3 к числу 13 без прохождения через число 8.

Ответ: 40

Задача № 3 (2481)

Исполнитель Калькулятор преобразует число на экране. У исполнителя есть три команды, которым присвоены номера:

1. Прибавить 1
2. Прибавить 2
3. Умножить на 3

Программа для исполнителя Калькулятор – это последовательность команд. Сколько существует программ, для которых при исходном числе 2 результатом является число 16 и при этом траектория вычислений содержит число 14?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241276?t=0h24m45s

Решение

Разобъём решение задачи на этапы:
1) Посчитать количество путей от 2 до 14.
Поскольку траектория должна проходить через 14, первым шагом будет подсчёт всех возможных способов достижения числа 14, начиная с 2.
2) Посчитать количество путей от 14 до 16.
После того как мы достигли числа 14, остаётся лишь перейти к числу 16.
3) Перемножить полученные результаты, так как каждая траектория от 2 до 14 может комбинироваться с любой траекторией от 14 до 16.
Реализация алгоритма:
Используем рекурсивную функцию для поиска количества траекторий. Функция принимает два параметра: текущее число c и конечное число e.

```
def f(c, e):
    # Если текущее число больше конечного, значит данная ветвь неверная
    if c > e: return 0
    # Если текущее число равно конечному, найден путь
    if c == e: return 1
```

```
    # Если текущее число меньше конечного, вызываем функцию для всех возможных шагов
    if c < e:
        return f(c + 1, e) + f(c + 2, e) + f(c * 3, e)
# Перемножим результаты, чтобы получить общее количество программ
print(f(2,14)*f(14,16))
```

Ответ: 558

Задача № 4 (2460)

Исполнитель Калькулятор преобразует число на экране. У исполнителя есть три команды, которым присвоены номера:

1. Прибавить 1
2. Умножить на 2
3. Умножить на 3

Программа для исполнителя Калькулятор – это последовательность команд. Сколько существует программ, для которых при исходном числе 5 результатом является число 52, и при этом траектория вычислений содержит число 15 и не содержит число 29?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241276?t=0h29m40s

Решение

Рассмотрим два этапа: переход от числа 5 к числу 15 и затем от числа 15 до числа 52. Важно отметить, что на пути выполнения программы нельзя попасть в число 29.
Сначала разберёмся с первым этапом: как перейти из числа 5 в число 15. Для этого будем использовать рекурсивную функцию, которая будет проверять выполнение условий. Если текущее число становится равным 29, функция немедленно возвращает 0, так как такая траектория считается недопустимой. Если же текущее число превышает конечное значение или совпадает с ним, возвращается 1, что означает успешный путь. Иначе продолжаем суммировать возможные варианты путей через прибавление 1, умножение на 2 и умножение на 3.
Теперь перейдем ко второму этапу: как перейти из числа 15 в число 52. Здесь также важно избегать попадания в число 29. Используем ту же самую логику, что и на первом этапе.
Объединяя оба этапа, получаем общее количество возможных программ, удовлетворяющих условиям задачи, путем перемножения количества вариантов первого и второго этапов.

```
def f(c, e):
    # Проверка условий выхода из функции
    if c > e or c == 29: return 0
    if c == e: return 1
    # Рекурсивная обработка возможных шагов
    if c < e:
        return f(c + 1, e) + f(c * 2, e) + f(c * 3, e)
# Вычисляем количество программ для каждого этапа
print(f(5, 15) * f(15, 52))
```

Ответ: 75

Задача №5 (7940)

У исполнителя имеются три команды, которые обозначены латинскими буквами:
A. Вычти 1
B. Вычти 6
C. Найди целую часть от деления на 2
Первая команда уменьшает число на экране на 1, вторая команда уменьшает это число на 6, третья команда делит число нацело на 2. Программа для исполнителя – это последовательность команд. Сколько существует таких программ, которые исходное число 34 преобразуют в число 6, и при этом траектория вычислений содержит числа 19 и 29 и не содержит числа 24?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241276?t=0h34m10s

Решение

Для решения задачи о количестве программ, преобразующих число 34 в число 6 через определенные промежуточные значения, важно учесть особенности работы с исполнителем, который может выполнять три различные операции над числом: вычитание 1, вычитание 6 и деление нацело на 2. Задача усложняется условиями, согласно которым программа должна проходить через числа 19 и 29, но избегать числа 24.
В отличие от других подобных задач, где обычно идут в направлении увеличения числа, здесь необходимо двигаться в обратном порядке — к уменьшению числа. Это требует особого подхода к условиям программы, так как теперь вместо проверки превышения определенного порога следует проверять, не стал ли результат слишком маленьким.
Рассмотрим программу пошагово:
1. Возвращение нуля: В программе предусмотрены два случая, когда функция возвращает 0. Первый случай — когда встречается число 24, второй — когда текущее значение становится меньше требуемого результата. Таким образом, если текущее число c меньше целевого e, либо равно 24, функция немедленно завершает работу и возвращает 0.
2. Возвращение единицы: Если текущее число совпадает с целевым, значит, данная траектория является корректной, и функция возвращает 1.
3. Рекурсивный вызов: Если текущее число все ещё больше целевого, то вызываются рекурсивные функции для уменьшения числа тремя возможными способами: вычесть 1, вычесть 6 или поделить нацело на 2. Эти результаты суммируются, поскольку каждая ветвь представляет собой одну из возможных траекторий.
4. Для нахождения количества всех подходящих программ, ведущих от 34 к 6 через 29 и 19 последовательно находим количество путей между всеми необходимыми точками маршрута: от 34 до 29, затем от 29 до 19, от 19 до 6 и перемножаем их.

```
def f(c,e):
    if c<e or c==24: return 0
    if c==e: return 1
    if c>e: return f(c-1,e)+f(c-6,e)+f(c//2,e)
print(f(34,29)*f(29,19)*f(19,6))
```

Запустив программу, получим итоговый ответ: 115.

Ответ: 115

Задание №6 (6062)

Исполнитель Калькулятор преобразует число, записанное на экране. У исполнителя есть три команды, которым присвоены номера:

1. Прибавь 1
2. Прибавь 2
3. Умножь на 2

Выполняя первую из них, исполнитель увеличивает число на экране на 1, выполняя вторую – увеличивает на 2, выполняя третью – увеличивает в 2 раза. Программой для исполнителя называется последовательность команд. Сколько существует программ, которые преобразуют исходное число 1 в число 38 так, что траектория вычисления не содержит чисел, в которых есть цифра 6?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241276?t=0h40m45s

Решение

Наша цель — найти все возможные пути преобразования числа 1 в 38 таким образом, чтобы ни одно промежуточное значение не содержало цифры 6.
Первым шагом будет проверка наличия цифры 6 в числе. Для этого удобно использовать строку, представляющую число, поскольку работа со строками позволяет легко проверять наличие символов. Например, если число представлено строкой '123', мы можем просто проверить, содержится ли в этой строке символ '6'.
Теперь перейдем к реализации функции, которая считает количество возможных путей от числа с до числа ее. Если текущее число с превышает конечное число е, или если оно содержит цифру 6, мы возвращаем 0, так как этот путь невозможен. Если же текущее число совпадает с конечным числом, мы возвращаем 1, потому что достигли цели. В противном случае, мы рекурсивно вызываем функцию для трех возможных действий: прибавляем 1, прибавляем 2 и умножаем на 2.

```
def f(c, e):
    # Проверяем, не превышает ли текущее число конечное и не содержит ли оно цифру 6
    if c > e or '6' in str(c):
        return 0
    # Если текущее число достигло конечного, возвращаем 1
    if c == e:
        return 1
    # Рекурсивный вызов для всех возможных действий
    if c < e:
        return f(c + 1, e) + f(c + 2, e) + f(c * 2, e)
print(f(1,38))
```

Программа выводит количество траекторий, ведущих от 1 к 38, избегая чисел с цифрой 6.

Ответ: 727785

Задание № 7(3097)

У исполнителя Калькулятор две команды, которым присвоены номера:

1. прибавь 1
2. умножь на 1,5

Первая из них увеличивает на 1 число на экране, вторая увеличивает это число в 1,5 раза, если число чётное. К нечётным числам вторая команда неприменима. Сколько есть программ, которые число 2 преобразуют в число 22?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241276?t=0h51m35s

Решение

В задаче требуется определить количество программ, которые могут преобразовать число 2 в число 22 при помощи двух команд: «прибавь 1» и «умножь на 1,5». Вторая команда применима только к чётным числам, поэтому необходимо учитывать это условие при построении программы.
Начнем с написания функции, которая будет считать количество возможных программ. Функция принимает два аргумента: текущее число c и конечное число e. Если текущее число больше конечного, программа завершает работу и возвращает 0, так как дальнейшее выполнение невозможно. Если текущее число равно конечному, функция возвращает 1, так как найдено решение. Если текущее число меньше конечного, происходит рекурсивный вызов функции для различных команд. Здесь важно учесть, что если число чётное, доступны обе команды, а если нечётное — только одна.
Если число чётное, выполняются оба действия: прибавляется 1 и умножается на 1.5. Если число нечётное, выполняется только одно действие — прибавление 1. Чтобы узнать количество программ, преобразующих число 2 в число 22, нужно вызвать функцию с соответствующими аргументами.

```
def f(c,e):
    if c>e: return 0
    if c==e: return 1
    if c<e and c%2==0: return f(c+1,e)+f(c*1.5,e)
    if c<e and c%2!=0: return f(c+1,e)

print(f(2,22))
```

Результат выполнения кода покажет, что существует 44 программы, которые могут преобразовать число 2 в число 22, соблюдая заданные правила.

Ответ: 44

Задание №8 (7793)

У исполнителя имеются две команды, которые обозначены латинскими буквами:
A. Вычти 2
B. Если число чётное, раздели на 2, иначе вычти 3
Программа для исполнителя – это последовательность команд. Сколько существует программ, которые преобразуют исходное число 98 в число 1 и при этом траектория вычислений не содержит число 28?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241276?t=0h57m0s

Решение

Нам необходимо найти количество программ, которые преобразуют число 98 в число 1 таким образом, чтобы траектория вычислений не содержала число 28.
Для начала рассмотрим возможные пути преобразования числа. Поскольку мы движемся от большего числа к меньшему, важно учитывать, какие операции доступны для каждого типа числа, чётного или нечётного.
Мы можем использовать рекурсивный подход для подсчета количества возможных путей. Для этого определим функцию f(c, e), где c — текущее число, а e — целевое число, в нашем случае 1.
Функция будет возвращать количество возможных путей из числа c в число e.

- Если c < e, то невозможно достичь цели, поэтому возвращаем 0.
- Если c == e, значит, мы достигли цели, и возвращаем 1.
- Если c > e:
  - Если c чётное (c % 2 == 0), то возможны два пути: вычитание 2 и деление на 2. Поэтому возвращаем сумму результатов вызова функции для этих двух вариантов.
  - Если c нечётное (c % 2 != 0), то возможны два пути: вычитание 2 и вычитание 3. Поэтому возвращаем сумму результатов вызова функции для этих двух вариантов.
- Если c == 28, то такой путь недопустим, поэтому также возвращаем 0.

```
def f(c, e):
    # Если текущее число меньше целевого или равно 28, возвращаем 0
    if c < e or c == 28:
        return 0
    # Если текущее число равно целевому, возвращаем 1
    if c == e:
        return 1

    # Если текущее число больше целевого и оно чётное
    if c > e and c % 2 == 0:
        return f(c - 2, e) + f(c // 2, e)
    # Если текущее число больше целевого и оно нечётное
    if c > e and c % 2 != 0:
        return f(c - 2, e) + f(c - 3, e)
# Вызываем функцию для исходного числа 98 и целевого числа 1
print(f(98, 1))
```

Этот код считает количество всех возможных программ, которые преобразуют число 98 в число 1 без прохождения через число 28.

Ответ: 12318

Задание №9(6653)

У исполнителя Калькулятор имеются три команды, которым присвоены номера:

1. Прибавь 1
2. Прибавь 2
3. Умножь на 3

Выполняя первую из них, исполнитель увеличивает число на экране на 1, выполняя вторую – увеличивает на 2, выполняя третью – умножает на 3. Сколько существует программ, для которых при исходном числе 6 результатом является число 25, и при этом траектория вычислений содержит либо число 15, либо 21, но не оба сразу?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241276?t=1h2m50s

Решение

Основная идея заключается в том, чтобы разделить задачу на два случая: когда траектория проходит через число 15, но не проходит через 21, и наоборот — когда она проходит через 21, но не через 15. Затем необходимо просуммировать результаты обоих случаев.
Для реализации этой идеи создадим рекурсивную функцию f, которая считает количество возможных путей от числа c до числа e. В зависимости от того, какое число запрещено проходить, 15 или 21, будем изменять условие остановки функции.

```
def f(c, e):
    # Если текущее значение больше целевого или равно запрещенному числу, возвращаем 0
    if c > e or c == 21:
        return 0
    # Если достигли целевого значения, возвращаем 1
    if c == e:
        return 1
    # Иначе продолжаем рекурсию, добавляя к текущему значению 1, 2 или умножая на 3
    if c < e:
        return f(c + 1, e) + f(c + 2, e) + f(c * 3, e)
# Считаем количество путей от 6 до 15 и затем от 15 до 25
k1 = f(6, 15) * f(15, 25)
# Меняем запрет на прохождение через 15 вместо 21
def f(c, e):
    if c > e or c == 15:
        return 0
        if c == e:
        return 1
    if c < e:
        return f(c + 1, e) + f(c + 2, e) + f(c * 3, e)
# Считаем количество путей от 6 до 21 и затем от 21 до 25
k2 = f(6, 21) * f(21, 25)
# Суммируем результаты двух случаев
print(k1 + k2)
```

Ответ: 2700

Задание №10 (7379)

У исполнителя Калькулятор имеются три команды, которые обозначены латинскими буквами:
A. Прибавить 1
B. Прибавить 4
C. Умножить на 2
Программа для исполнителя – это последовательность команд, каждая из которых изменяет число. Требуется найти количество таких программ, которые преобразуют исходное число 1 в число 50, и

при этом траектория вычислений содержит ровно одно из чисел 8, 16, или 32.
Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241276?t=1h10m30s

Решение

Для решения задачи требуется определить количество программ, состоящих из команд 'A' (прибавить 1), 'B' (прибавить 4) и 'C' (умножить на 2), которые преобразуют число 1 в число 50 так, чтобы в процессе вычисления встречалось ровно одно из чисел: 8, 16 или 32. Рассмотрим каждый случай отдельно.
Случай 1, только число 8.
Мы определяем функцию f, которая считает количество путей от числа c до числа e. Если текущее значение больше e или равно 16/32, функция возвращает 0, поскольку такие пути недопустимы. Если значение равно e, возвращается 1, так как найден правильный путь. В остальных случаях вызывается рекурсивная функция для каждого возможного шага (прибавление 1, прибавление 4, умножение на 2).
Аналогично первому случаю, определяем функцию f, но теперь исключаем числа 8 и 32 и рассчитываем количество путей от 1 до 16 и от 16 до 50:
Опять же, определяем функцию f, но теперь исключаем числа 8 и 16:
Рассчитываем количество путей от 1 до 32 и от 32 до 50.
Суммируем полученные значения для всех трех случаев.

```
# Функция для подсчета количества путей от числа c до числа e,
# учитывая ограничения на прохождение через определенные числа.
def f(c, e):
    # Если текущее число больше целевого или равно числу, которое нельзя проходить,
    # возвращаем 0, так как такой путь невозможен.
    if c > e or c == 16 or c == 32:
        return 0
    # Если текущее число достигло целевого, значит, найден верный путь.
    if c == e:
        return 1
    # Если текущее число еще меньше целевого, продолжаем искать возможные пути.
    if c < e:
        # Рекурсивный вызов функции для каждого возможного шага:
        # прибавляем 1, прибавляем 4, умножаем на 2.
        return f(c + 1, e) + f(c + 4, e) + f(c * 2, e)

# Рассчитываем количество путей от 1 до 8 и от 8 до 50.
k1 = f(1, 8) * f(8, 50)
# Переписываем функцию для второго случая, когда нужно пройти через 16.
def f(c, e):
    if c > e or c == 8 or c == 32:
        return 0
    if c == e:
        return 1
    if c < e:
        return f(c + 1, e) + f(c + 4, e) + f(c * 2, e)
# Рассчитываем количество путей от 1 до 16 и от 16 до 50.
k2 = f(1, 16) * f(16, 50)
# Переписываем функцию для третьего случая, когда нужно пройти через 32.
```

```
def f(c, e):
    if c > e or c == 8 or c == 16:
        return 0
    if c == e:
        return 1
    if c < e:
        return f(c + 1, e) + f(c + 4, e) + f(c * 2, e)
# Рассчитываем количество путей от 1 до 32 и от 32 до 50.
k3 = f(1, 32) * f(32, 50)
# Суммируем все три результата, чтобы получить общее количество путей.
print(k1 + k2 + k3)
```

Ответ : 6370599

Задание №11 (4494, kompege)

У исполнителя Калькулятор есть три команды, которым присвоены номера:
1. Прибавить 1
2. Прибавить 2
3. Прибавить 4

Определите число, для получения которого из числа 15 существует 4930 программ.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241276?t=1h19m35s

Решение

Для начала заметим, что поскольку каждая команда увеличивает текущее значение, то бессмысленно рассматривать числа, меньшие 15, так как ни одна программа не сможет привести к таким числам, начиная с 15. Поэтому будем искать число, большее 15.
Нам необходимо написать функцию, которая считает количество возможных программ для достижения заданного числа ее из начального значения с. Эта функция должна учитывать три возможные операции прибавления: +1, +2 и +4.
Затем, чтобы найти искомое число, переберем различные значения от 16 до некоторого разумного предела (например, до 100), пока не найдем такое число, для которого количество программ равно 4930.

```
def f(c, e):
    # Если текущее значение превышает целевое, значит, эта ветвь невозможна
    if c > e: return 0
    # Если текущее значение достигло целевого, значит, найдена одна возможная
программа
    if c == e: return 1
    # В противном случае рекурсивно считаем количество программ для каждого возможного
шага
    if c < e:
        return f(c + 1, e) + f(c + 2, e) + f(c + 4, e)
# Перебор значений от 16 до 99
for x in range(16, 100):
    if f(15, x) == 4930:
```

```
        print(x)
        break
```

После выполнения этой программы мы получим число 31, потому что именно для него существует 4930 различных программ, ведущих от 15 к этому числу.

Ответ : 31

Telegram: @fast_ege