Модуль №5. Задание №1

Задача №1: Факториал

Условие:
Напишите функцию factorial(n), которая принимает одно целое число n и возвращает факториал этого числа. Функция должна обрабатывать все допустимые значения n (неотрицательные целые числа).
Входные данные: Целое число n (0 ≤ n ≤ 20)
Выходные данные: Целое число, равное n!
Ограничение на время: 5 секунд
1. Оцените временную сложность алгоритма в лучшем, среднем и худшем случаях.

2. Напишите модульные тесты для проверки корректности функции в стратегиях: позитивные тесты, негативные и граничные.

Пример работы программы:

```
print(factorial(0))  # Ожидаемый вывод: 1

print(factorial(5))  # Ожидаемый вывод: 120

print(factorial(20)) # Ожидаемый вывод: 2432902008176640000
```

Задача №2: Числа Фибоначчи

Условие:
Напишите функцию fibonacci(n), которая принимает одно целое число n и возвращает список чисел Фибоначчи от 0 до n-го числа (включительно). Функция должна обрабатывать все допустимые значения n (n ≥ 0).
Входные данные: Целое число n (n ≥ 0)
Выходные данные: Список чисел Фибоначчи от 0 до n-го числа (включительно)
1. Оцените временную сложность алгоритма в лучшем, среднем и худшем случаях.

2. Напишите модульные тесты для проверки корректности функции в стратегиях: позитивные тесты, негативные и граничные.

Пример работы программы:

```
print(fibonacci(0))  # Ожидаемый вывод: [0]

print(fibonacci(5))  # Ожидаемый вывод: [0, 1, 1, 2, 3, 5]

print(fibonacci(10)) # Ожидаемый вывод: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

Задача №3: Подсчёт единиц в двоичном представлении

Условие:
Напишите функцию count_ones(n), которая принимает одно целое число n в десятичном представлении и возвращает количество единиц в его двоичном представлении. Функция должна обрабатывать все допустимые значения n (n ≥ 0).
Входные данные: Целое число n (n ≥ 0)
Выходные данные: Целое число, равное количеству единиц в двоичном представлении n
1. Оцените временную сложность алгоритма в лучшем, среднем и худшем случаях.

2. Напишите модульные тесты для проверки корректности функции в стратегиях: позитивные тесты, негативные и граничные.

Пример работы программы:

print(count_ones(0)) # Ожидаемый вывод: 0 (0 -> "0")

print(count_ones(5)) # Ожидаемый вывод: 2 (5 -> "101")

print(count_ones(15)) # Ожидаемый вывод: 4 (15 -> "1111")

Задача №4: Палиндром (без использования строк)

Условие:
Напишите функцию, которая принимает целое число x и возвращает true, если число является палиндромом, и false в противном случае. При этом использовать строки нельзя.
1. Оцените временную сложность алгоритма в лучшем, среднем и худшем случаях.

2. Напишите модульные тесты для проверки корректности функции в стратегиях: позитивные тесты, негативные и граничные.

Пример работы программы:

print(is_palindrome(121)) # Ожидаемый вывод: True

print(is_palindrome(-121)) # Ожидаемый вывод: False

print(is_palindrome(10)) # Ожидаемый вывод: False

print(is_palindrome(0)) # Ожидаемый вывод: True

Задача №5

Формализуйте следующую проблему заказчика и разработайте программу.

Привет!

Нам нужна помощь в анализе данных о посещаемости нашего сайта. У нас есть большой объем данных о том, сколько посетителей заходят на наш сайт каждый день, и мы хотим понять, какие дни недели и месяцы наиболее популярны. Мы думаем, что если у нас будет программа, которая сможет обрабатывать эти данные, это поможет нам лучше планировать маркетинговые кампании и улучшать контент.

Было бы здорово, если бы программа могла также показывать дни с наименьшей посещаемостью, чтобы мы могли разобраться, почему так происходит. Мы собираем данные в виде электронных таблиц, где указаны дата и количество посетителей. Нам важно, чтобы программа могла работать с этим форматом и давала нам четкую и понятную статистику.

Мы не совсем уверены, какие дополнительные данные могут понадобиться или какие могут быть ограничения, поэтому надеемся, что вы сможете предложить оптимальное решение.

Спасибо за вашу помощь!

Файл: data.csv

Веселые задачи повышенной сложности

1 задача: Быстрое возведение в степень (Exponentiation)

- **Описание:** Напишите функцию fast_power(a, n), которая вычисляет a^n (где a – число, а n – неотрицательное целое число) с использованием алгоритма быстрого возведения в степень (метод двоичного разложения показателя).
- **Анализ временной сложности:** ...?
- **Модульные тесты:** Позитивные, негативные (например, нецелые n или отрицательные степени, если не поддерживаются) и граничные случаи (например, n = 0).

2 задача: Проверка простоты числа

- **Описание:** Напишите функцию is_prime(n), которая принимает целое число n и возвращает true, если число является простым, и false в противном случае. Реализуйте оптимизацию с проверкой делителей до квадратного корня из n.
- **Анализ временной сложности:** ...?
- **Модульные тесты:** Позитивные тесты (простые и составные числа), негативные тесты (n ≤ 1) и граничные случаи (n = 2).

3 задача: Вычисление НОД (наибольшего общего делителя)

- **Описание:** Напишите функцию gcd(a, b), которая принимает два целых числа и возвращает их наибольший общий делитель, используя алгоритм Евклида.
- **Анализ временной сложности:** ...?
- **Модульные тесты:** Позитивные тесты (разные пары чисел), негативные тесты (отрицательные числа, если не поддерживаются) и граничные случаи (один из аргументов равен 0).