Основы теории чисел и их использование в криптографии

Выполнил: Макаров Алексей Игоревич, 3 курс 4 группа 2 подгруппа

2024

**Ход работы**

В результате выполнения прошлой лабораторной работы мы выяснили,

что энтропия зависит от статических характеристик самих алфавитов и сообщений (энтропия по Шеннону и по Хартли).

Каждое натуральное число, большее единицы, делится по крайней мере на два числа: на 1 и на само себя. Если число не имеет делителей, кроме самого себя и единицы, то оно называется простым, а если у числа есть еще делители, то составным.

Вариант 4, значит значения m и n имеют соответственные значения 421 и 457. Таким образом, все простые числа в интервале [2, n] представлены на рисунке 1.

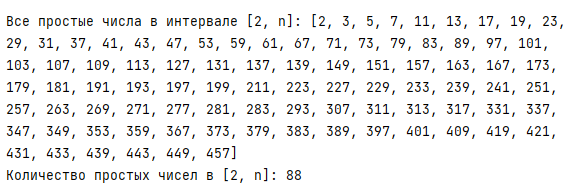


Рисунок 1 – Результат поиска всех простых чисел в интервале [2, n]

Всего простых чисел получилось 88, так же можно вручную посчитать примерное количество чисел с помощью следующей формулы n/ln(n) = 457/ln (457) ≈ 74.6.

Все простые числа в интервале [m, n] или [421, 457] представлены на рисунке 2.

A white background with black letters

Description automatically generated

Рисунок 2 – Результат поиска всех простых чисел в интервале [m, n]

Теперь сравним полученный результат с «ручными» вычислениями,

используя «решето Эратосфена»: то есть наименьшее 21.

Шаг 1. Запишем числа из заданного диапазона: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21.

Шаг 2. Удалим из списка числа с учетом s =2: 2, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21.

Шаг 3. Удалим из списка числа с учетом s =3: 2, 3, 5, 7, 11, 13, 17, 19.

Шаг 4. Запишем все числа, которые не делятся на числа из списка выше: 421, 431, 433, 439, 443, 449, 457

Числа m и n в виде произведения простых множителей (форма записи – каноническая) представляют из себя произведения двух чисел: самих себя и 1, так как m и n, при значениях 421, 457 соответственно – простые. Результат представления чисел представлен на рисунке 3.



Рисунок 3 – Результат представления чисел в виде произведения простых множителей

Проверка на то, является ли число, состоящее из конкатенации цифр m и n показана на рисунке 4.



Рисунок 4 – Результат проверки числа

Код приложения представлен в листинге 1.

|  |
| --- |
| *import math*  *# Функция для нахождения всех простых чисел в интервале [2, n]*  *def find\_primes(n):*  *primes = []*  *is\_prime = [True] \* (n + 1)*  *is\_prime[0] = is\_prime[1] = False*  *for i in range(2, int(math.sqrt(n)) + 1):*  *if is\_prime[i]:*  *primes.append(i)*  *for j in range(i \* i, n + 1, i):*  *is\_prime[j] = False*  *for i in range(int(math.sqrt(n)) + 1, n + 1):*  *if is\_prime[i]:*  *primes.append(i)*  *return primes*  *# Функция для вычисления НОД двух чисел*  *def gcd(a, b):*  *while b != 0:*  *a, b = b, a % b*  *return a*  *# Функция для факторизации числа на простые множители*  *def prime\_factors(n):*  *factors = []*  *divisor = 2*  *while n > 1:*  *while n % divisor == 0:*  *factors.append(divisor)*  *n //= divisor*  *divisor += 1*  *return factors*  *# Функция для конкатенации чисел и проверки на простоту*  *def is\_concatenated\_prime(m, n):*  *concatenated = int(str(m) + str(n))*  *if concatenated < 2:*  *return False*  *for i in range(2, int(math.sqrt(concatenated)) + 1):*  *if concatenated % i == 0:*  *return False*  *return True*  *# Поиск простых чисел в интервале [m, n]*  *def find\_primes\_in\_range(m, n):*  *primes = []*  *for num in range(m, n + 1):*  *if all(num % i != 0 for i in range(2, int(math.sqrt(num)) + 1)):*  *primes.append(num)*  *return primes*  *# Значения m и n*  *m = 667*  *n = 703*  *# 1. Найти все простые числа в интервале [2, n]*  *primes\_1 = find\_primes(n)*  *print("Простые числа в интервале [2, n]:", primes\_1)*  *# Подсчитать количество простых чисел в указанном интервале*  *count\_primes\_1 = len(primes\_1)*  *print("Количество простых чисел в интервале [2, n]:", count\_primes\_1)*  *# Сравнить это число с n/ln(n)*  *n\_div\_ln\_n = n / math.log(n)*  *print("n/ln(n):", n\_div\_ln\_n)*  *print("Отношение количества простых чисел к n/ln(n):", count\_primes\_1 / n\_div\_ln\_n)*  *# 2. Повторить п. 1 для интервала [m, n]*  *primes\_2 = find\_primes\_in\_range(m, n)*  *print("Простые числа в интервале [m, n]:", primes\_2)*  *# Сравнить результаты с ручными вычислениями, используя решето Эратосфена*  *# Здесь просто используем количество простых чисел в интервале [2, n], так как уже реализовано решето Эратосфена*  *print(*  *"Количество простых чисел в интервале [m, n] (с решетом Эратосфена):", len(primes\_2)*  *)*  *# 3. Записать числа m и n в виде произведения простых множителей*  *factors\_m = prime\_factors(m)*  *factors\_n = prime\_factors(n)*  *print("Простые множители числа m:", factors\_m)*  *print("Простые множители числа n:", factors\_n)*  *# 4. Проверить, является ли число, состоящее из конкатенации цифр m и n, простым*  *concatenated\_prime = is\_concatenated\_prime(m, n)*  *if concatenated\_prime:*  *print("Число, состоящее из конкатенации цифр m и n, является простым")*  *else:*  *print("Число, состоящее из конкатенации цифр m и n, не является простым")*  *# 5. Найти НОД (m, n)*  *gcd\_m\_n = gcd(m, n)*  *print("Наибольший общий делитель (НОД) m и n:", gcd\_m\_n)*  *# Выполнить задания по условиям п. 1 и 2.*  *primes\_between\_2\_and\_n = find\_primes(n)*  *primes\_between\_m\_and\_n = find\_primes\_in\_range(m, n)*  *print("Простые числа в интервале [2, n]:", primes\_between\_2\_and\_n)*  *print("Количество простых чисел в интервале [2, n]:", len(primes\_between\_2\_and\_n))*  *print("Простые числа в интервале [m, n]:", primes\_between\_m\_and\_n)*  *print("Количество простых чисел в интервале [m, n]:", len(primes\_between\_m\_and\_n))* |

Листинг 1 – Код приложения

Проверка работы приложения на двух и трех числах (m = 142, n = 178, k = 122) представлена на рисунке 5.

A screenshot of a computer

Description automatically generated

Рисунок 4 – Результат работы приложения на двух и трех числах

Таким образом, была разработана программа для вывода и подсчета наибольшего общего знаменателя чисел, рассмотрен алгоритм «решето Эратосфена».