Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА № 1**

**ПО КУРСУ «КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_К.В.Стасюк

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Крамаренко

**Задание:**

№ 12. Реализовать программный продукт нахождения функции эйлера от числа двумя способами(по определению и с помощью формулы). Сравнить эффективность алгоритмов для набора из 100 чисел, каждое из которых больше 10’000’000.

**Ход работы:**

**Функция Эйлера:**

Функция Эйлера (ϕ(n)) представляет собой количество положительных целых чисел, меньших n, взаимно простых с n. Взаимно простые числа имеют НОД (наибольший общий делитель) равный 1.

**Метод по определению:**

В первом методе euler\_phi\_definition, используется простой перебор чисел от 1 до n с проверкой их взаимной простоты с n при помощи функции math.gcd(n, i) – НОД .

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1 – метод по определениию.

**Метод с использованием стандартной формулы:**

Второй метод euler\_phi\_standard\_formula использует стандартную формулу Эйлера для эффективного вычисления функции Эйлера (ϕ(n)) для заданного числа n. Основные этапы:

1)Нахождение простых множителей:

Внутренний цикл с переменной i начинается с 2 и продолжается до n. Если i является делителем n, то i добавляется в множество prime\_factors, и n делится на i.Цикл повторяется, пока n делится на i.

2)Вычисление функции Эйлера:

Внешний цикл перебирает простые множители p из множества prime\_factors.

Каждый простой множитель p вносит свой вклад в вычисление функции Эйлера по стандартной формуле:

Изображение выглядит как Шрифт, рукописный текст, текст, белый

Автоматически созданное описание

Рисунок 2 – стандартная формула функции Эйлера.

​

Временная сложность:

Временная сложность этого метода зависит от количества простых множителей числа n. Если число имеет много простых множителей, время выполнения может быть существенным.

Этот метод является более эффективным по сравнению с методом по определению, но он все равно может быть неэффективным для больших чисел с большим количеством простых множителей.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 3 – Функция Эйлера с помощью стандартной формулы.

**Метод с оптимизированной стандартной формулой:**

Третий метод euler\_phi\_standard\_formula\_opt представляет собой оптимизированную версию второго метода за счет ограничения поиска простых множителей до квадратного корня из n.

В отличие от второго метода, внешний цикл в третьем методе продолжает выполнение только до тех пор, пока i \* i <= n.

Это означает, что мы ищем простые множители только до квадратного корня из n, что является математической оптимизацией, так как любой простой множитель p, который больше квадратного корня из n, будет уже учтен как делитель в предыдущих итерациях.

Таким образом, уменьшается количество проверок и делений, что делает метод более эффективным для больших чисел.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 4 – Функция Эйлера с помощью оптимизированной формулы.

**Измерение времени выполнения:**

Программа генерирует 100 случайных чисел и затем измеряет время выполнения каждого метода для каждого числа. Результаты выводятся в консоль, демонстрируя относительную эффективность оптимизированной стандартной формулы по сравнению с обычной стандартной формулой и методом по определению.

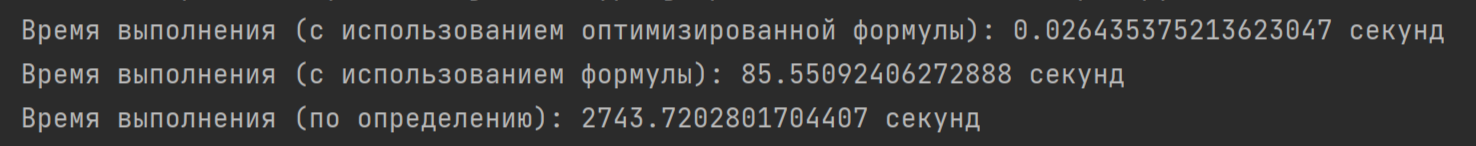


Рисунок 5 – Измерение времени.

Как можно заметить, оптимизированная функция показала лучшие результаты, функция с использованием стандартной формулы тоже значительно обогнала функцию по определению.

Код программы:

import time  
import random  
import math  
  
# Функция Эйлера представляет собой количество положительных целых чисел, меньших n, взаимно простых с  
  
  
# Функция для нахождения функции Эйлера по определению  
def euler\_phi\_definition(n):  
 count = 0  
 for i in range(1, n + 1):  
 #нод==1(взаимно простое)  
 if math.gcd(n, i) == 1:  
 count += 1  
 return count  
  
# Функция для нахождения функции Эйлера с использованием формулы  
def euler\_phi\_standard\_formula(n):  
 result = n  
 # Получаем список простых множителей числа n  
 prime\_factors = set()  
 i = 2  
 while i <= n:  
 #i-делитель  
 while n % i == 0:  
 prime\_factors.add(i)  
 #делим  
 n //= i  
 i += 1  
  
 # Вычисляем значение функции Эйлера по стандартной формуле  
 for p in prime\_factors:  
 result \*= (1 - 1 / p)  
  
 return round(result)  
  
# Функция для нахождения функции Эйлера с использованием формулы оптимизированно  
def euler\_phi\_standard\_formula\_opt(n):  
 result = n  
 # Получаем список простых множителей числа n  
 prime\_factors = set()  
 i = 2  
 while i\*i <= n:  
 #i-делитель  
 while n % i == 0:  
 prime\_factors.add(i)  
 #делим  
 n //= i  
 i += 1  
  
 # Вычисляем значение функции Эйлера по стандартной формуле  
 for p in prime\_factors:  
 result \*= (1 - 1 / p)  
  
 return round(result)  
  
  
# Генерация списка из 100 случайных чисел, каждое из которых больше 10,000,000  
numbers = [random.randint(10000000, 100000000) for \_ in range(100)]  
  
# Измерение времени выполнения для метода с использованием формулы  
start\_time = time.time()  
for num in numbers:  
 euler\_phi\_standard\_formula\_opt(num)  
end\_time = time.time()  
print(f"Время выполнения (с использованием оптимизированной формулы): {end\_time - start\_time} секунд")  
  
# Измерение времени выполнения для метода с использованием формулы  
start\_time = time.time()  
for num in numbers:  
 euler\_phi\_standard\_formula(num)  
end\_time = time.time()  
print(f"Время выполнения (с использованием формулы): {end\_time - start\_time} секунд")  
  
# Измерение времени выполнения для метода по определению  
start\_time = time.time()  
for num in numbers:  
 euler\_phi\_definition(num)  
end\_time = time.time()  
print(f"Время выполнения (по определению): {end\_time - start\_time} секунд")