**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕАВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ» Московский̆ институт электроники и математики**

**Иванилов Леонид Михайлович группа БИВ231**

**ПРОГРАММА “Программное обеспечение для поиска всех простых чисел в отрезке [1; n]”**

**Курсовая работа**

**по направлению 09.03.01 Информатика и вычислительная техника студентов образовательной̆ программы бакалавриата «Информатика и вычислительная техника»**

**Руководитель ученая степень, звание (при наличии) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Консультант ученая степень, звание (при наличии) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва 2023-24г.**

# Введение

Этот отчет представляет собой детальное описание приложения для поиска простых чисел в заданном диапазоне, реализованного на языке программирования Python. Приложение использует библиотеку Tkinter для создания графического интерфейса, что позволяет пользователям легко взаимодействовать с программой. Целью проекта является предоставление пользователям возможности выбора различных алгоритмов для поиска простых чисел и сравнения их эффективности. В этом введении подробно рассматриваются используемые технологии и объясняются причины их выбора.

## Используемые технологии и причины их выбора

### Библиотека Tkinter

* **Причины выбора**: Tkinter входит в стандартную библиотеку Python и предоставляет мощные инструменты для создания графического интерфейса (GUI). Он легко интегрируется с основным кодом на Python и поддерживает создание кроссплатформенных приложений.
* **Преимущества**: Простота использования, хорошая документация, возможность быстрого создания прототипов, поддержка кроссплатформенности (работает на Windows, macOS и Linux).

### Библиотека Matplotlib

* **Причины выбора**: Matplotlib была выбрана для визуализации данных, поскольку она предоставляет мощные инструменты для создания графиков и диаграмм. Она широко используется в научных исследованиях и позволяет легко интегрировать графики в приложения на Python.
* **Преимущества**: Широкие возможности для создания различных типов графиков, хорошая интеграция с другими библиотеками Python, активное сообщество разработчиков.

### Библиотека Time

* **Причины выбора**: Библиотека Time используется для измерения времени выполнения алгоритмов, что позволяет анализировать и сравнивать их эффективность.
* **Преимущества**: Встроенная в Python, что обеспечивает легкую доступность и стабильность, простота использования.

# Архитектурный подход

Приложение построено с использованием объектно-ориентированного подхода, что обеспечивает модульность и расширяемость. Основные компоненты приложения включают классы PrimeCalculator и PrimeApp, каждый из которых выполняет конкретные задачи и взаимодействует с пользователем через графический интерфейс.

## PrimeCalculator

Этот класс отвечает за реализацию различных алгоритмов поиска простых чисел, записи в файл и выполнение алгоритмов. Каждый метод в этом классе реализует определенный алгоритм и позволяет пользователю выбирать наиболее подходящий для их задач. К основным методам относятся:

* **Sieve of Eratosthenes (Решето Эратосфена)**: Классический алгоритм, который эффективно работает для больших чисел. Временная сложность: O(n log log n).
* **Simple Check (Простой перебор)**: Наивный метод, проверяющий каждое число на простоту путем деления. Временная сложность: O(n sqrt n).
* **Segmented Sieve (Сегментированное решето)**: Оптимизированная версия решета Эратосфена для больших диапазонов. Временная сложность: O(n log log n) для предварительного решета и O((r-l) log log r) для сегментированного поиска.
* **Trial Division (Пробное деление)**: Метод, проверяющий деление числа на все целые числа до его квадратного корня. Временная сложность: O(n sqrt n).
* **Wheel Factorization (Факторизация колеса)**: Методика, уменьшающая количество делений путем исключения заранее известных простых множителей. Временная сложность: O(n / log n) для подготовки колеса и O(n sqrt n) для поиска простых чисел в заданном диапазоне.
* **Seive of Atkin (Решето Аткина)**: Оптимизированный алгоритм поиска простых чисел. Эффективен для больших чисел. Временная сложность: O(n / log log n)

### PrimeApp

Этот класс отвечает за базовый графический интерфейс и взаимодействие с пользователем для вычисления простых чисел. Он предоставляет пользователю возможность ввода диапазона чисел для анализа и отображает результаты. Основные функции включают:

* **Инициализация интерфейса:** Создание и настройка графических элементов, таких как поля ввода для начального и конечного значений диапазона, кнопок для запуска вычислений и области для отображения результатов.
* **Ввод данных:** Обеспечение пользовательского ввода для определения начального и конечного значения диапазона, в котором будут находиться простые числа.
* **Запуск вычислений:** Обработка событий, таких как нажатие кнопок, и вызов соответствующих методов для выполнения вычислений простых чисел.
* **Отображение результатов:** Представление пользователю списка найденных простых чисел и времени выполнения алгоритма.

### NormalPrimeApp

Этот класс расширяет функциональность PrimeApp, добавляя дополнительные алгоритмы для вычисления простых чисел и анализ их производительности. Основные функции включают:

* **Расширенный выбор алгоритмов:** Предоставление пользователю возможности выбора между различными алгоритмами, такими как Решето Эратосфена, Решето Аткина и метод пробного деления.
* **Выполнение алгоритмов:** Реализация и запуск выбранных алгоритмов для нахождения простых чисел в заданном диапазоне.
* **Сравнение производительности:** Сравнение времени выполнения различных алгоритмов для заданного диапазона и отображение этой информации пользователю.
* **Улучшенный интерфейс:** Предоставление пользователю более детализированной информации о процессе вычисления и результатах, включая графическое представление производительности.

### StepPrimeApp

Этот класс добавляет функциональность пошагового выполнения алгоритмов для вычисления простых чисел. Основные функции включают:

* **Пошаговое выполнение:** Обеспечение возможности пошагового выполнения алгоритмов, позволяя пользователю видеть каждый шаг процесса нахождения простых чисел.
* **Визуализация процесса:** Графическое представление шагов алгоритма, показывающее, какие числа проверяются и как они маркируются как простые или составные.
* **Интерактивность:** Обеспечение интерактивного интерфейса, позволяющего пользователю управлять выполнением алгоритма, например, переходить к следующему шагу или возвращаться назад.
* **Обучение и демонстрация:** Использование пошагового выполнения для образовательных целей, помогая пользователям лучше понять работу алгоритмов поиска простых чисел.

### AnalysisApp

Этот класс фокусируется на анализе и сравнении эффективности различных алгоритмов поиска простых чисел. Основные функции включают:

* **Сбор данных:** Сбор данных о времени выполнения и использовании ресурсов различными алгоритмами для нахождения простых чисел.
* **Анализ производительности:** Сравнение производительности различных алгоритмов в зависимости от размера диапазона и других параметров.
* **Генерация отчетов:** Создание подробных отчетов, содержащих графики и таблицы с результатами анализа производительности.
* **Предоставление рекомендаций:** Предоставление рекомендаций пользователю по выбору наиболее эффективного алгоритма для заданного диапазона чисел на основе результатов анализа.
* **Визуализация данных:** Графическое представление данных анализа, включая диаграммы, графики и другие визуальные элементы для наглядного сравнения алгоритмов.

# Пример использования

Приложение позволяет пользователям вводить начальное и конечное значения диапазона и выбирать алгоритмы для поиска простых чисел. После выполнения выбранных алгоритмов пользователю отображаются результаты, включая найденные простые числа и время выполнения каждого алгоритма. Если количество найденных простых чисел превышает 20, результаты записываются в файл для удобства просмотра.

# Особенности и функции приложения

## Графический интерфейс

* Ввод начального и конечного значений диапазона.
* Выбор одного или нескольких алгоритмов для выполнения.
* Отображение результатов и времени выполнения каждого алгоритма.
* Возможность записи результатов в файл, если количество найденных простых чисел превышает 20.

## Рекомендации по выбору алгоритмов

* Для небольших чисел рекомендуется использовать методы простого перебора и пробного деления.
* Для больших чисел лучше всего подходят решето Эратосфена и сегментированное решето.

## Анализ и визуализация результатов

* Построение графиков для сравнения времени выполнения различных алгоритмов.
* Возможность анализа эффективности каждого метода в зависимости от размера диапазона.

**Выполнение подсчета с шагом**

* Возможность самому задать шаг и более детально разобраться в работе алгоритма и в его временной сложности

# Детальное описание алгоритмов

## Решето Эратосфена

* **Описание**: Алгоритм эффективно находит все простые числа до заданного предела. Он начинает с массива, в котором все элементы отмечены как простые, и последовательно помечает кратные каждого простого числа как составные.
* **Временная сложность**: O(n log log n).
* **Рекомендация**: Используйте для больших чисел.

## Простой перебор

* **Описание**: Этот наивный метод проверяет каждое число на простоту путем деления его на все предыдущие числа до квадратного корня этого числа.
* **Временная сложность**: O(n sqrt n).
* **Рекомендация**: Используйте для небольших чисел.

## Сегментированное решето

* **Описание**: Оптимизированная версия решета Эратосфена, которая позволяет находить простые числа в больших диапазонах. Алгоритм делит диапазон на сегменты и применяет решето Эратосфена к каждому сегменту.
* **Временная сложность**: O(n log log n) для предварительного решета и O((r-l) log log r) для сегментированного поиска.
* **Рекомендация**: Подходит для всех размеров чисел.

## Пробное деление

* **Описание**: Метод основан на проверке деления числа на все целые числа в диапазоне от 2 до квадратного корня этого числа. Если число не делится нацело ни на одно из этих чисел, оно является простым.
* **Временная сложность**: O(n sqrt n).
* **Рекомендация**: Используйте для маленьких чисел.

## Факторизация колеса

* **Описание**: Алгоритм использует факторизацию колеса для нахождения простых чисел на определённом интервале. Этот метод уменьшает количество делений, исключая заранее известные простые множители.
* **Временная сложность**: O(n / log n) для подготовки колеса и O(n sqrt n) для поиска простых чисел в заданном диапазоне.
* **Рекомендация**: Используйте для больших чисел.

# Заключение

Проект поиска простых чисел в заданном диапазоне имеет высокую актуальность для исследовательских и образовательных целей. Приложение предоставляет пользователям возможность изучения и сравнения различных алгоритмов поиска простых чисел, что является важной задачей в области теории чисел и криптографии.

## Образовательные цели

Студенты и преподаватели могут использовать приложение для изучения основ алгоритмов поиска простых чисел и их эффективности. Это помогает лучше понять, как работают различные алгоритмы, и научиться выбирать наиболее подходящий метод для конкретных задач.

## Научные исследования

Исследователи могут использовать приложение для проведения экспериментов и анализа различных алгоритмов, что может привести к улучшению существующих методов или разработке новых. Это особенно важно в областях, где эффективный поиск простых чисел играет ключевую роль, например, в криптографии.

## Практические задачи

Приложение может быть полезным инструментом для разработчиков программного обеспечения, занимающихся криптографией и другими областями, где поиск простых чисел является важной задачей. Оно предоставляет удобный способ для проверки и сравнения различных алгоритмов, что может помочь в выборе наиболее эффективного решения для конкретных нужд.

Таким образом, данное приложение не только предоставляет удобный инструмент для поиска простых чисел, но и служит полезным ресурсом для углубленного изучения и исследования алгоритмов в этой области.