Отчёт по лабораторной работе №6. Разложение числа на множители

Дисциплина: Математические основы защиты информации и информационной безопасности

Манаева Варвара Евгеньевна

Содержание

Список иллюстраций

1 Общая информация о задании лабораторной работы

1.1 Цель работы

Ознакомиться с алгоритмами разложения числа на множители.

1.2 Задание [1]

1. Задание.

2 Теоретическое введение [2]

2.1 Разложение на множители

pho-метод Полланда (или $\rho - 1$ метод Полларда) является одним из алгоритмов для факторизации целых чисел, который особенно эффективен для нахождения малых простых делителей. Он основан на свойствах чисел и использует последовательности, чтобы вычислить делители.

2.1.1 Основные этапы метода

1. Подготовка:
    - **Выбор числа n:** Начинаем с целого числа n, которое необходимо факторизовать;
    - **Выбор параметров:** Выбираем небольшое целое число a и границу B, которая будет использоваться для ограничения множителей.
2. Генерация последовательности: Создаем последовательность чисел по формуле: $x_{k+1} = (x_k^2 + a)$.
3. Вычисление НОД: На каждом шаге вычисляем наибольший общий делитель (НОД) между n и разностью двух членов последовательности.
4. Проверка результата: Если найденный НОД d больше 1 и меньше n, то это делитель числа n. Если $d = n$, то алгоритм не дал результата, и его можно повторить с другими параметрами. Если $d = 1$, то повторяем действия со второго шага.

5. Завершение: Процесс продолжается до тех пор, пока не будет найден делитель или не исчерпаются все возможные варианты.

2.1.2 Применение метода

Метод Полланда эффективен для нахождения малых простых делителей, особенно когда число имеет структуру, позволяющую выделить такие делители. Он также может быть использован в сочетании с другими методами факторизации для повышения общей эффективности.

3 Выполнение лабораторной работы [1]

Исходный код написан на языке Julia [3]. Код функции, осуществляющей разложение числа на множители, представлен ниже.

```
function metodPollarda(n, c, any_func::Function)
    if n % 2 == 0
        return 2, round(Int, n/2)
    end
    a = c; b = c
    i = 0
    while i < 100
        a = any_func(a)
        b = any_func(any_func(b))
        d = evklidBin(a-b, n)
        # println(a, "\t", b, "\t", d)
        if d > 1
            return d, round(Int, n/d)
        end
        i += 1
    end
    return "Делитель не найден"
end
```

Разберём подробно работу функции.

На вход функция принимает 3 параметра:

- `n` – число, которое необходимо факторизовать;
- `c` – число, которое используется в качестве начала отсчёта;
- `any_func::Function` – функция, по которой рассчитывается каждая следующая итерация.

Функцию саму можно поделить на несколько смысловых частей:

1. Предобработка;
2. Входящие параметры для цикла;
3. Цикл работы функции;
4. Вывод при неудачном наборе входящих данных.

3.0.1 1. Предобработка

Если число, которое необходимо факторизовать, делится на 2, то оно не подходит под действие алгоритма (на вход даётся только нечётное число), в связи с чем можно сразу вывести делители этого числа.

```
if n % 2 == 0
    return 2, round(Int, n/2)
end
```

3.0.2 2. Входящие параметры для цикла

Первым шагом алгоритма является подготовка двух промежуточных значений (a и b), которые будут представлять x_i и x_{2i} в рамках работы алгоритма. Также задаётся счётчик для ограничения числа итераций работы функции.

```
a = c; b = c
i = 0
```

3.0.3 3. Цикл работы функции

Основный цикл работы функции, включающий в себя шаги 2-4 работы алгоритма.

```
while i < 100
    a = any_func(a)
    b = any_func(any_func(b))
    d = evklidBin(a-b, n)
    # println(a, "\t", b, "\t", d)
    if d > 1
        return d, round(Int, n/d)
    end
    i += 1
end
```

3.0.4 4. Вывод при неудачном наборе входящих данных

Возвращение значения “Делитель не найден” при завершении работы цикла в связи с превышением числа итераций.

```
return "Делитель не найден"
```

3.0.5 Проверка работы функции

```
n = 1359331
c = 1
metodPollarda(n, c, x -> (x^2 + 5) % n)
```

Результат работы кода представлен ниже (рис. 3.1).

```
[122]: n = 1359331
       c = 1
       metodPollarda(n, c, x -> (x^2 + 5) % n)

       6       41      1
       41      123939  1
       1686    391594  1
       123939  438157  1
       435426  582738  1
       391594  1144026 1
       1090062 885749  1181

[122]: (1181, 1151)
```

Рис. 3.1: Результат работы реализованной функции разложения числа на множители

3.1 Разложение крупного числа на множители

```
n = 135956347

c = 1

metodPollarda(n, c, x -> (x^2 + 13) % n)
```

Результат работы кода представлен ниже (рис. 3.2).

```
[9]: n = 135956347
     c = 1
     metodPollarda(n, c, x -> (x^2 + 13) % n)

[9]: (5591, 24317)
```

Рис. 3.2: Результат работы реализованной функции разложения числа на множители

4 Выводы

В результате работы мы ознакомились с алгоритмом разложения чисел на множители и реализовали его на языке программирования `Julia`.

Также были записаны скринкасты:

На RuTube:

- Весь плейлист
- Запись создания шаблона отчёта и презентации для заполнения
- Выполнения лабораторной работы
- Запись создания отчёта
- Запись создания презентации
- Защита лабораторной работы

На Платформе:

- Весь плейлист
- Запись создания шаблона отчёта и презентации для заполнения
- Выполнения лабораторной работы
- Запись создания отчёта
- Запись создания презентации
- Защита лабораторной работы

Список литературы

1. Лабораторная работа №6. Разложение числа на множители [Электронный ресурс]. RUDN, 2024. URL: https://esystem.rudn.ru/pluginfile.php/2368516/mod_folder/content/0/lab06.pdf.
2. Математика криптографии и теория шифрования [Электронный ресурс]. URL: https://intuit.ru/studies/courses/552/408/info.
3. Julia 1.10 Documentation [Электронный ресурс]. 2024. URL: https://docs.julialang.org/en/v1/.