ЛАБОРАТОРНА РОБОТА No2

з курсу

ТЕОРЕТИКО-ЧИСЛОВІ АЛГОРИТМИ В КРИПТОЛОГІЇ

Застосування алгоритму дискретного логарифмування

Виконав: Приходько Юрій ФБ-12

1. Мета роботи

Ознайомлення з алгоритмом дискретного логарифмування Сільвера-Поліга-Геллмана. Практична реалізація цього алгоритму. Пошук переваг, недоліків та особливостей застосування даного алгоритму дискретного логарифмування. Практична оцінка складності роботи алгоритму.

2. Порядок виконання роботи

Написати програму, що розв’язує задачу дисктерного логарифму шляхом звичайного перебору. Рекомендується обмежити час роботи програми, наприклад — 5 хвилин.

Написати програму, що реалізовує алгоритм Сільвера-Поліга-Геллмана для груп типу Zp*

В результаті виконання лабораторної роботи для підвищення ефективності алгоритмів було вирішено переписати алгоритми для факторизації з першої лабораторної роботи, що в подальшому були використані при роботі алгоритму Сільвера-Поліга-Геллмана, а також власне алгоритм перебору та алгоритм Сільвера-Поліга-Геллмана мовою C.

При цьому було вирішено реалізувати їх як бібліотеку, а інтерфейс користування надати за допомогою мови python використовуючи вбудований модуль для виклику функцій бібліотеки ctypes.

В результаті написання програми було отримано наступні файли:

- Dlp.c - бібліотека мовою C, що включає в себе імплементацію всіх необхідних функцій для роботи алгоритмів зазначених в завданні,

включно з алгоритмами факторизації та допоможіними обрахунками.
- DlpModule.py - програмний код мовою python, що надає зручний інтерфейс користування функціями низько-рівневої бібліотеки

3. Створити Docker image, що містить програму, яка містить імплементацію алгоритму Сільвера-Поліга-Гелмана з кроку 3, та інструкцію щодо запуску та користування програмою в цьому Docker image.

Після докеризації вихідних кодів застосунку було отримано docker image що при запуску через CLI отримує тип алгоритму, значення чисел a, b та p і в результаті виконання виводить дискретний логарифм.

4. За допомогою команди docker run -it salo1d/nta_cp2_helper:2.0 запустити допоміжну програму, яка генерує задачі пошуку дискретного логарифма двох типів. Вхідним параметром програми є порядок простого числа p — модуля кільця лишків. Програма очікує від користувача розв’язку та повідомляє про його правильність чи не правильність. Час створення задачі дискретного логарифма не перевищує 10 хвилин. Програма очікує введення розв’язку 5 хвилин.

Час очікування при генерації задач досягав достатньо довгих відміток тому було вирішено автоматизувати процес. Для цього засобами мови python та модуля pwntools було написано два класи що відповідають за отримання та надсилання між двома докер-контейнерами (лабораторним та створеним мною), а також файл що автоматизує роботу почергово викликаючи запити до лабораторної середи, що тим самим дає змогу ефективно заміряти дані та зберегти максимальні відмітки довжини чисел для вирішених проблем.

Приклад роботи у звичайному (брудному) режимі. Тут викликається створення обох задач для довжини чисел від 3 до 10, і відповідне вирішення.

```
INFO:oracle:Starting communication with oracle with dec_len = 4
[+] 2024-05-19 16:25:11 - Oracle validated SPH result
INFO:oracle:Oracle validated SPH result
[+] 2024-05-19 16:25:11 - Starting Task 2
INFO:oracle:Starting Task 2
[+] 2024-05-19 16:25:12 - Oracle validated SPH result
INFO:oracle:Oracle validated SPH result
[+] 2024-05-19 16:25:12 - Starting communication with oracle with dec_len = 5
INFO:oracle:Starting communication with oracle with dec_len = 5
[+] 2024-05-19 16:25:13 - Oracle validated SPH result
INFO:oracle:Oracle validated SPH result
[+] 2024-05-19 16:25:13 - Starting Task 2
INFO:oracle:Starting Task 2
[+] 2024-05-19 16:25:13 - Oracle validated SPH result
INFO:oracle:Oracle validated SPH result
[+] 2024-05-19 16:25:13 - Starting communication with oracle with dec_len = 6
INFO:oracle:Starting communication with oracle with dec_len = 6
[+] 2024-05-19 16:25:14 - Oracle validated SPH result
INFO:oracle:Oracle validated SPH result
[+] 2024-05-19 16:25:14 - Starting Task 2
INFO:oracle:Starting Task 2
[+] 2024-05-19 16:25:15 - Oracle validated SPH result
INFO:oracle:Oracle validated SPH result
[+] 2024-05-19 16:25:15 - Starting communication with oracle with dec_len = 7
INFO:oracle:Starting communication with oracle with dec_len = 7
[+] 2024-05-19 16:25:16 - Oracle validated SPH result
INFO:oracle:Oracle validated SPH result
[+] 2024-05-19 16:25:16 - Starting Task 2
INFO:oracle:Starting Task 2
[+] 2024-05-19 16:25:17 - Oracle validated SPH result
INFO:oracle:Oracle validated SPH result
[+] 2024-05-19 16:25:17 - Starting communication with oracle with dec_len = 8
INFO:oracle:Starting communication with oracle with dec_len = 8
[+] 2024-05-19 16:25:19 - Oracle validated SPH result
INFO:oracle:Oracle validated SPH result
[+] 2024-05-19 16:25:19 - Starting Task 2
INFO:oracle:Starting Task 2
[+] 2024-05-19 16:25:19 - Oracle validated SPH result
INFO:oracle:Oracle validated SPH result
[+] 2024-05-19 16:25:19 - Starting communication with oracle with dec_len = 9
INFO:oracle:Starting communication with oracle with dec_len = 9
[+] 2024-05-19 16:25:20 - Oracle validated SPH result
INFO:oracle:Oracle validated SPH result
[+] 2024-05-19 16:25:20 - Starting Task 2
INFO:oracle:Starting Task 2
```

Модифікуємо програму для заміру часу та порівняння роботи алгоритму на обох типах задач. В результаті роботи було отримано наступну таблицю порівняннь.

Decimal digit length	Bruteforce Task 1	Bruteforce Task 2	SPH Task 1	SPH Task 2
3	0.000013	0.000016	0.00024	0.0001
4	0.000027	0.000031	0.00031	0.0049
5	0.00042	0.00010	0.00319	0.00244
6	0.000069	0.00138	0.00046	0.14622
7	0.01506	0.03465	0.00139	0.84529
8	0.37574	0.02981	0.00205	0.03789
9	0.47621	5.30599	0.00208	0.45153
10	0.92065	19.66147	0.01349	9.68497

11	Exceeded 5m limit	Exceeded 5m limit	0.04276	0.68399
12	-	-	0.03577	32.57668
13	-	-	0.03255	29.06656
14	-	-	0.00454	284.98155
15	-	-	0.09928	Too much memory alloc
16	-	-	0.13239	-
17	-	-	0.08206	-
18	-	-	Too much time generating (> 10m)	-

Відповідь алгоритму SPH під час задачі 2 з довжиною 15

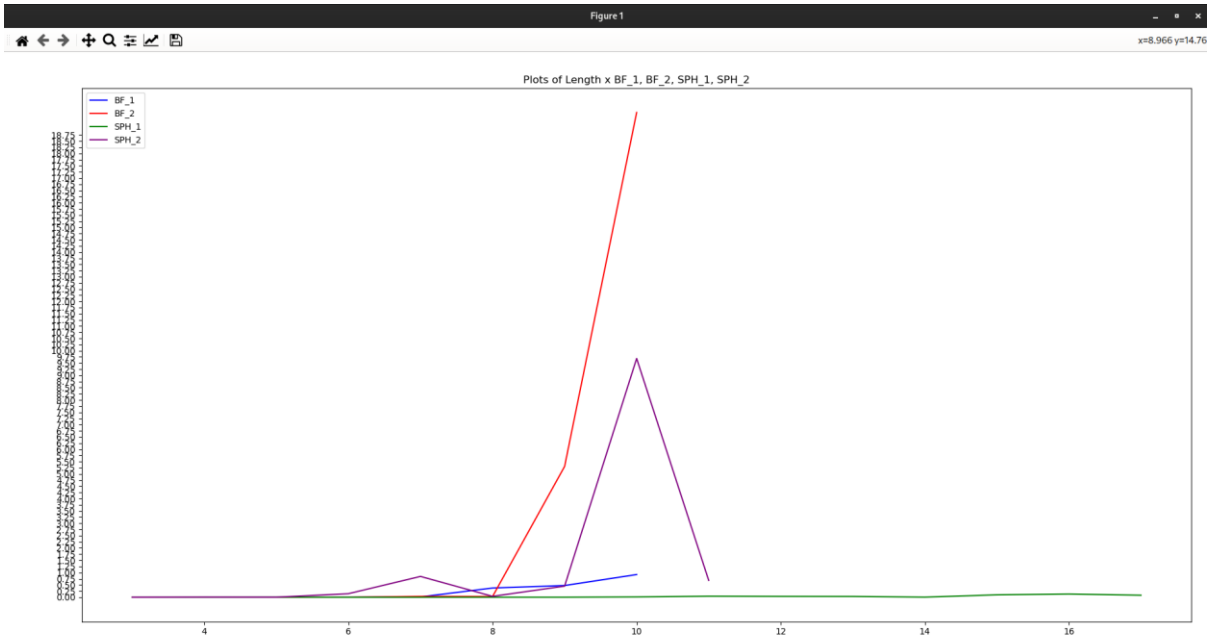
```
In [1]: from dlpModule import DLPSolver

In [2]: s = DLPSolver()

In [3]: a = 420167886330362; b = 118406672237771; p = 631232079024383;

In [4]: s.dlp_sph(a,b,p)
Error allocating memory for table[2]. Tried to allocate 437810779 MB
Error allocating memory
Time taken to solve: 0.0001823902130126953
Out[4]: 1
```

Додаткове тестування на різних довжинах біт, генерувалось власноруч за допомогою sage. На знімку продемонстровано відповідно довжина p - 40bit, 48bit (невдала спроба), 48bit (вдала спроба), 64bit

```
/run/media/gratigo/KINGSTON/term6/nta/NTA_2024_lab2/sources main ↑1 !5 ?11
docker run -it gratigo/nta_lab2:latest
Choose Method (BruteForce/SilverPohligHellman) [BF/SPH]> SPH
a = 635582021401
b = 558901518257
p = 643448513017
Time taken to solve: 0.14342713356018066
result = 298066455930

/run/media/gratigo/KINGSTON/term6/nta/NTA_2024_lab2/sources main ↑1 !5 ?11
docker run -it gratigo/nta_lab2:latest
Choose Method (BruteForce/SilverPohligHellman) [BF/SPH]> SPH
a = 65731619786799
b = 4071590069305
p = 224409487034027
Error allocating memory for table[3]. Tried to allocate 98982 MB
Error allocating memory
Time taken to solve: 0.0012760162353515625
result = 1

/run/media/gratigo/KINGSTON/term6/nta/NTA_2024_lab2/sources main ↑1 !5 ?11
docker run -it gratigo/nta_lab2:latest
Choose Method (BruteForce/SilverPohligHellman) [BF/SPH]> SPH
a = 155795934450786
b = 46207396669779
p = 209279736087931
Time taken to solve: 0.529052734375
result = 104539261745626

/run/media/gratigo/KINGSTON/term6/nta/NTA_2024_lab2/sources main ↑1 !5 ?11
docker run -it gratigo/nta_lab2:latest
Choose Method (BruteForce/SilverPohligHellman) [BF/SPH]> SPH
a = 7625797391709257917
b = 9556767054346367189
p = 10088630519426888101
Time taken to solve: 7.292999505996704
result = 7726154828589417880

/run/media/gratigo/KINGSTON/term6/nta/NTA_2024_lab2/sources main ↑1 !5 ?11
```

Далі все впирається в об’єм пам’яті необхідний для утримання таблиць.

Графіки залежності довжини від часу для кожного алгоритму:



5. Висновки

У ході виконання лабораторної роботи я ознайомився з двома методами розв’язку задачі дискретного логарифмування, а саме метод перебору, що гарно себе показує на маленький числах і методом Сільвера-Поліга-Геллмана що чудово себе показав навіть при великих числах. В результаті виконання було реалізовано ці алгоритми засобами мови C та python3.

В результаті тестувань та аналізів результатів роботи було доведено очевидну річ, алгоритм перебору працює добре тільки при достатньо невеликій довжині чисел, на відміну від алгоритму Сільвера-Поліга-Геллмана, який працює найкраще у випадку коли p-1 розбивається на малі прості дільники. Саме такий варіант був генерований за допомогою лабораторної програми у першому завданні (перший тип чисел). Другий тип чисел генерувався з одним чи декількома великими простими дільниками, що сильно сповільнює роботу алгоритму, і по факту наближає його час до часу роботи повного перебору.

Саме тому найбільш ефективний час роботи був показаний при роботі з першим типом чисел в алгоритмі Сільвера-Поліга-Геллмана.

Також була виконана спроба довести написану в ході виконання програму “до максимуму” власною генерацією чисел, при розмірі більшому за 64 біті (тестувались 72, 80) програма завжди упирається в виділення пам’яті, що потребує додаткової оптимізації для ефективного тестування

алгоритму. Час виконання алгоритму при максимальній досягнутій довжині числа в 64 біта - 7.3 секунди, результат правильний.