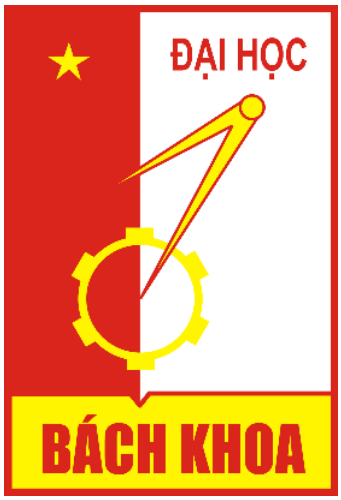
ĐẠI HỌC BÁCH KHOA HÀ NỘI

KHOA TOÁN – TIN



BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN KỸ THUẬT LẬP TRÌNH (MI3310)

Chủ đề 2:

XÂY DỰNG "THƯ VIỆN" PHÉP TOÁN VỚI SỐ NGUYÊN LỚN

Giảng viên hướng dẫn:	TS. NGUYỄN THÀNH NAM
Sinh viên thực hiện:	NGUYỄN TRUNG KIÊN
Mã số sinh viên:	20227180
Mã lớp học:	150327

HÀ NỘI, T6/2024

MỤC LỤC

LỜI MỞ ĐẦU

Trong thời đại công nghệ thông tin ngày nay, việc xử lý số liệu lớn đang ngày càng trở nên cấp thiết hơn bao giờ hết, đặc biệt trong bối cảnh sự phát triển mạnh mẽ của khoa học máy tính và ứng dụng rộng rãi của nó trong các lĩnh vực như mật mã học, tính toán khoa học và tài chính. Để đáp ứng nhu cầu này, việc xử lý các phép toán với số nguyên lớn trở thành một thách thức không thể tránh khỏi.

Bài báo cáo được thực hiện nhằm nghiên cứu và phát triển một thư viện phục vụ cho các phép toán cơ bản trên số nguyên lớn, với mục tiêu chính là đảm bảo tính chính xác và hiệu suất. Qua quá trình này, em hy vọng không chỉ nâng cao được hiểu biết về cách thức xử lý số liệu lớn mà còn cung cấp nền tảng quan trọng cho việc phát triển các ứng dụng phức tạp trong tương lai.

Báo cáo sẽ trình bày các khái niệm cơ bản về số nguyên lớn và tầm quan trọng của chúng, song là phần triển khai chi tiết về cấu trúc thư viện và các kỹ thuật được áp dụng để đảm bảo hiệu quả xử lý. Cuối cùng, bài báo cáo sẽ tổng kết các kết quả thực nghiệm và đánh giá hiệu suất của thư viện, cùng những nhận xét và kết luận rút ra từ quá trình nghiên cứu và phát triển này.

Mong rằng bài báo cáo sẽ mang đến những giá trị hữu ích và khơi dậy sự hứng thú và niềm đam mê nghiên cứu trong lĩnh vực lập trình và khoa học máy tính, không chỉ đối với các bạn sinh viên mà còn *"những công dân thời đại số"* nói chung.

Em xin chân thành cảm ơn!

DANH MỤC TỪ VIẾT TẮT

IT	Information Technology *(Công nghệ thông tin)*
FFT	Fast Furier Transform *(Thuật toán chuyển đổi nhanh)*
MPI	Message Passing Interface *(Giao diện truyền tin nhắn)*
BigInt	Big Integer *(Số nguyên lớn)*
API	Application Programming Interface *(Giao diện lập trình ứng dụng)*
GUI	Graphical User Interface *(Giao diện đồ họa người dung)*
IDE	Integrated Development Environment *(Môi trường phát triển tích hợp)*
OOP	JavaScript Object Notation *(Lập trình hướng đối tượng)*
JSON	Human Resources Management *(Đối tượng JavaScript được mã hóa)*
APL	Array Programming Language *(Ngôn ngữ lập trình mảng)*
DSA	Data Structures and Algorithms *(Cấu trúc dữ liệu và thuật toán)*
GMP	GNU Multiple Precision Arithmetic Library *(Thư viện đa số lớn)*
LTM	LibTomMath *(Đa số nhỏ)*
SAGE	Software for Algebra and Geometry Experimentation *(Phần mềm toán học tích hợp)*

DANH MỤC TÌNH HUỐNG KIỂM THỬ

DANH MỤC HÌNH ẢNH MINH HOẠ

PHẦN 1: TỔNG QUAN VỀ SỐ NGUYÊN LỚN

1.1. ĐỊNH NGHĨA

SỐ NGUYÊN LỚN LÀ GÌ?

Theo *Wikipedia*:

> "Về bản chất, không có định nghĩa nào cho số nguyên lớn. Trong thực tiễn khi nói hay xét đến số nguyên lớn, ta thường xét đến các số cực lớn so với các số sử dụng trong thường ngày.
>
> Dùng tiêu chuẩn bằng đơn vị đếm cao nhất của Việt Nam (tỉ), đặt nghìn tỉ (1000 tỉ = 10^{12}) làm quy ước giới hạn cho số lớn, ta sẽ chỉ xét:
>
> – Các số lớn hơn giới hạn 10^{12} trên.
>
> – Các dãy số có thể bắt đầu bằng rất ít số nhỏ, để mô tả một số lớn như vậy."

(Nguồn: https://vi.wikipedia.org/wiki/Thể_loại:Số_nguyên_lớn)

Xét về khía cạnh toán học nói chung và ngành nghiên cứu số học nói riêng, ta có những nhận xét chung như sau:

- Số nguyên lớn, thường được gọi là BigInt *(Big Integer)*, là các số nguyên mà giá trị của chúng vượt quá phạm vi lưu trữ của các kiểu dữ liệu số nguyên cơ bản trong hầu hết các ngôn ngữ lập trình. Các kiểu dữ liệu số nguyên chuẩn như `int` trong C hay Java thường có giới hạn kích thước cố định.
- *Ví dụ:*
  - `Từ -2,147,483,648 → 2,147,483,647` (đối với `32-bit`)
  - `Từ -9,223,372,036,854,775,808 → 9,223,372,036,854,775,807` (đối với `64-bit`).

Trong báo cáo lần này, chúng ta sẽ xét đến các số trong phạm vi đến 1000 chữ số và các phép toán cơ bản: Tổng, Hiệu, Tích, Thương (Lấy kết quả số thực) của chúng.

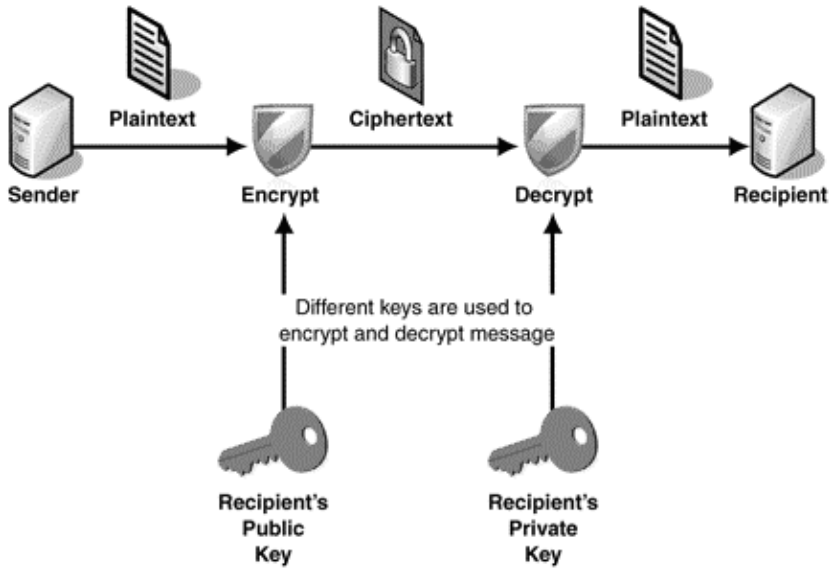
1.2. VAI TRÒ VÀ TẦM QUAN TRỌNG

Số nguyên lớn đóng vai trò quan trọng trong nhiều lĩnh vực công nghệ và khoa học, bao gồm:

- ***Mật mã học:*** An ninh thông tin phụ thuộc vào các thuật toán mã hóa sử dụng số nguyên lớn. Việc mã hóa và giải mã thông tin an toàn yêu cầu xử lý các phép toán với số nguyên có hàng trăm hoặc hàng ngàn chữ số.
- ***Tính toán khoa học:*** Nhiều mô phỏng và tính toán trong vật lý, hóa học, và các lĩnh vực khoa học khác đòi hỏi xử lý số liệu lớn với độ chính xác cao. Số nguyên lớn giúp đảm bảo kết quả chính xác và đáng tin cậy.
- ***Tài chính:*** Trong các hệ thống tài chính, xử lý số liệu lớn là cần thiết để đảm bảo tính chính xác và an toàn của các giao dịch. Các phép toán với số nguyên lớn giúp ngăn chặn sai sót và đảm bảo tính toàn vẹn của dữ liệu tài chính.
- ***Phát triển phần mềm:*** Khả năng xử lý số nguyên lớn mở ra nhiều cơ hội cho các nhà phát triển phần mềm trong việc xây dựng các ứng dụng phức tạp và tiên tiến hơn. Điều này giúp nâng cao khả năng cạnh tranh và sáng tạo trong ngành công nghiệp phần mềm.

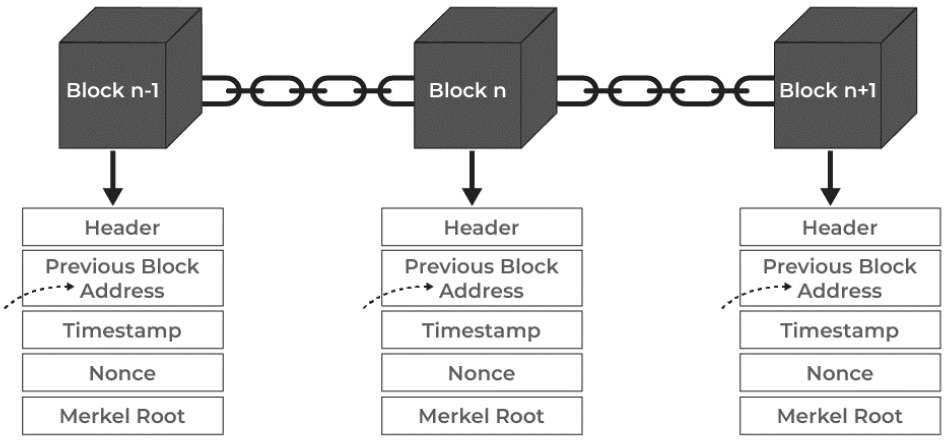
1.3. ỨNG DỤNG THỰC TIỄN

- ***RSA (Rivest-Shamir-Adleman)*** là một trong những thuật toán mã hóa công khai phổ biến nhất, sử dụng các số nguyên lớn trong các bước tạo khóa và mã hóa. Một ví dụ cụ thể là việc tạo khóa RSA, trong đó cần chọn hai số nguyên tố lớn và tính toán các phép nhân và modulo trên các số này để tạo ra khóa công khai và khóa bí mật.



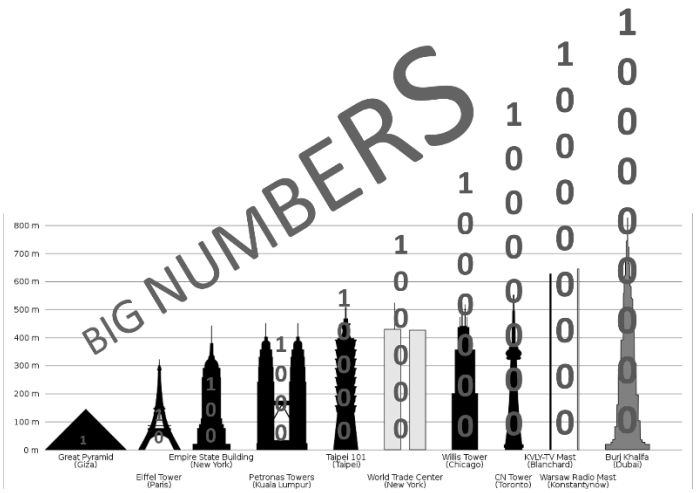
Hình 1: Sơ lược thuật toán RSA (Rivest-Shamir-Adleman)

- ***Blockchain và tiền điện tử:*** Bitcoin và các loại tiền điện tử khác sử dụng các số nguyên lớn trong các thuật toán băm và chữ ký số. Ví dụ, việc xác minh các giao dịch trong blockchain yêu cầu các phép toán trên các số nguyên lớn để đảm bảo tính toàn vẹn và an toàn của dữ liệu.



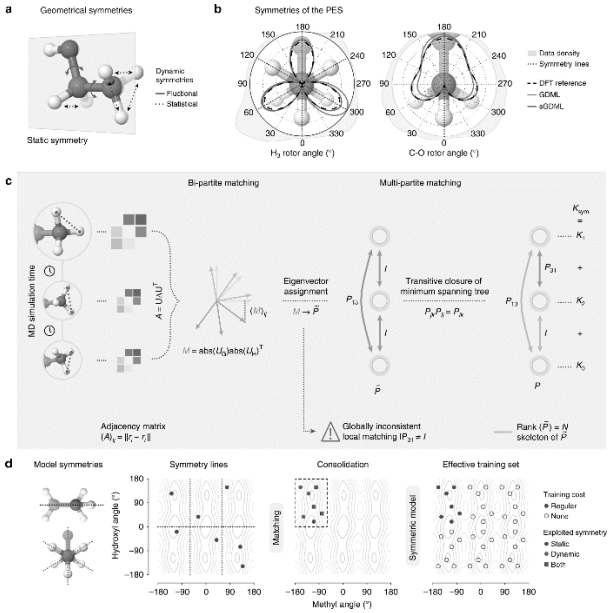
Hình 2: Ví dụ minh hoạ cấu trúc các lớp Blockchain

- ***Phân tích số học trong khoa học máy tính:*** Các nhà khoa học thường sử dụng các số nguyên lớn để phân tích các hiện tượng tự nhiên và toán học phức tạp. Chẳng hạn, việc tìm kiếm các số nguyên tố lớn hoặc các số hoàn hảo đòi hỏi khả năng xử lý và lưu trữ các số nguyên có hàng triệu chữ số.



Hình 3: Minh hoạ bài toán xử lý số nguyên lớn trong Khoa học máy tính

- ***Mô phỏng vật lý và hóa học:*** Trong các mô phỏng động lực học phân tử, các nhà khoa học cần thực hiện các phép toán trên các số liệu rất lớn để mô phỏng và dự đoán hành vi của các hệ thống vật lý và hóa học. Điều này giúp trong nghiên cứu và phát triển vật liệu mới, thuốc, và nhiều ứng dụng khoa học khác.



Hình 4: Minh hoạ xử lý số nguyên lớn trong mô phỏng động lực học phân tử

- *Và còn nhiều ứng dụng phục vụ các lĩnh vực khác trong đời sống…*

1.4. THÁCH THỨC VÀ GIẢI PHÁP

1.4.1. Thách thức về hiệu suất

- **Mô tả:**

  Xử lý số nguyên lớn đòi hỏi nhiều tài nguyên tính toán và bộ nhớ hơn so với số nguyên thông thường. Các phép toán cơ bản như cộng, trừ, nhân và chia trở nên phức tạp và tốn kém hơn về mặt thời gian và không gian.

- **Giải pháp:**
  - ***Thuật toán tối ưu:*** Sử dụng các thuật toán tối ưu để cải thiện hiệu suất. Ví dụ, thuật toán *Karatsuba* cho phép nhân hai số lớn với độ phức tạp $O(n^{\log_2 3})$ thay vì $O(n^2)$.
  - ***Cấu trúc dữ liệu hiệu quả:*** Lưu trữ các số lớn dưới dạng mảng hoặc danh sách các chữ số với cơ số lớn, chẳng hạn như 2^{32} hoặc 10^9, để giảm thiểu số lượng các phần tử cần xử lý.
  - ***Sử dụng thư viện chuyên dụng:*** Sử dụng các thư viện đã được tối ưu hóa như GMP *(GNU Multiple Precision Arithmetic Library)* hoặc các thư viện tương tự trong ngôn ngữ lập trình tương ứng.

1.4.2. Thách thức về chính xác

- **Mô tả:**

  Đảm bảo tính chính xác của các phép toán trên số nguyên lớn là rất quan trọng. Sai sót trong tính toán có thể dẫn đến kết quả không chính xác, ảnh hưởng đến các ứng dụng sử dụng chúng.

- **Giải pháp:**
  - ***Sử dụng thư viện chính xác cao:*** Sử dụng các thư viện như *Python's* `int`, *Java's* `BigInteger`, hoặc *C++'s GMP* để đảm bảo tính chính xác của các phép toán.
  - ***Kiểm thử và xác minh:*** Triển khai các kiểm thử đơn vị và kiểm thử tích hợp để xác minh tính chính xác của các phép toán trên số nguyên lớn.

- ***Xử lý tràn số:*** Kiểm soát và xử lý các tình huống tràn số khi thực hiện các phép toán, đảm bảo các phép toán không dẫn đến mất dữ liệu.

1.4.3. Thách thức về bộ nhớ

- **Mô tả:**

  Số nguyên lớn đòi hỏi nhiều bộ nhớ để lưu trữ. Quản lý bộ nhớ hiệu quả là một thách thức lớn, đặc biệt khi làm việc với các hệ thống có giới hạn bộ nhớ.

- **Giải pháp:**
  - ***Cấu trúc dữ liệu nén:*** Sử dụng các cấu trúc dữ liệu nén để lưu trữ số nguyên lớn.

    Ví dụ: Sử dụng mảng hoặc danh sách với các phần tử là các khối dữ liệu thay vì từng chữ số riêng lẻ.
  - ***Thuật toán tối ưu bộ nhớ:*** Áp dụng các thuật toán tối ưu bộ nhớ để giảm thiểu việc sử dụng bộ nhớ không cần thiết. Ví dụ, sử dụng thuật toán FFT *(Fast Fourier Transform)* trong nhân số lớn.
  - **Quản lý bộ nhớ động:** Sử dụng các kỹ thuật quản lý bộ nhớ động để phân bổ và giải phóng bộ nhớ một cách hiệu quả, giảm thiểu rủi ro tràn bộ nhớ.

1.4.4. Thách thức về tính toán phân tán

- **Mô tả:**

  Khi xử lý các số nguyên lớn trong môi trường tính toán phân tán hoặc đa lõi, việc phân chia công việc và đồng bộ kết quả là một thách thức.

- **Giải pháp:**
  - ***Phân chia công việc hợp lý:*** Sử dụng các thuật toán phân chia công việc hiệu quả để phân chia các phép toán lớn thành các phép toán nhỏ hơn, có thể thực hiện song song.
  - ***Đồng bộ hóa:*** Sử dụng các kỹ thuật đồng bộ hóa để đảm bảo kết quả của các phép toán song song được kết hợp chính xác. Ví dụ, sử dụng khóa hoặc các cơ chế đồng bộ khác.

- ***Sử dụng framework phân tán:*** Sử dụng các framework phân tán như *Apache Spark, Hadoop* hoặc *MPI (Message Passing Interface)* để quản lý và thực hiện các phép toán trên số nguyên lớn trong môi trường phân tán.

1.4.5. Thách thức về bảo mật

- **Mô tả:**

  Trong các ứng dụng mật mã học, việc xử lý các số nguyên lớn phải đảm bảo an toàn và bảo mật, ngăn chặn các tấn công như phân tích thời gian hoặc tấn công kênh phụ.

- **Giải pháp:**
  - ***Thuật toán bảo mật:*** Sử dụng các thuật toán mật mã học an toàn và đã được kiểm chứng để thực hiện các phép toán trên số nguyên lớn.
  - ***Kỹ thuật bảo mật:*** Áp dụng các kỹ thuật bảo mật như padding, tránh phân tích thời gian để bảo vệ các phép toán khỏi bị tấn công.
  - ***Kiểm thử bảo mật:*** Triển khai các kiểm thử bảo mật để xác định và khắc phục các lỗ hổng bảo mật trong việc xử lý số nguyên lớn.

1.4.6. Thách thức về tương thích và tính di động

- **Mô tả:**

  Đảm bảo tính tương thích và tính di động của các ứng dụng xử lý số nguyên lớn trên các hệ thống và nền tảng khác nhau.

- **Giải pháp:**
  - ***Sử dụng thư viện đa nền tảng:*** Sử dụng các thư viện đa nền tảng như GMP hoặc các thư viện tiêu chuẩn của ngôn ngữ lập trình để đảm bảo tính tương thích.
  - ***Kiểm thử trên nhiều nền tảng:*** Thực hiện kiểm thử trên nhiều hệ điều hành và nền tảng khác nhau để đảm bảo ứng dụng hoạt động đúng trên tất cả các môi trường.
  - ***Thiết kế kiến trúc mô-đun:*** Thiết kế các ứng dụng với kiến trúc mô-đun, cho phép dễ dàng thay thế hoặc cập nhật các thành phần để tương thích với các hệ thống khác nhau.

PHẦN 2: XỬ LÝ SỐ NGUYÊN LỚN VÀ CÁC THƯ VIỆN PHỔ BIẾN

2.1. XỬ LÝ SỐ NGUYÊN LỚN

Xử lý số nguyên lớn đòi hỏi việc sử dụng các cấu trúc dữ liệu và thuật toán đặc biệt, vì các kiểu dữ liệu số nguyên cơ bản của ngôn ngữ lập trình không thể chứa đựng những giá trị lớn này. Dưới đây là một số kỹ thuật và phương pháp phổ biến:

2.1.1. Cấu trúc dữ liệu

- **Mảng chữ số:**

  Số nguyên lớn có thể được biểu diễn dưới dạng mảng các chữ số hoặc mảng các khối số. Ví dụ, số 123456789 có thể được lưu trữ trong mảng [1, 2, 3, 4, 5, 6, 7, 8, 9].

- **Cơ số lớn:**

  Thay vì lưu từng chữ số trong mảng, ta có thể lưu trữ theo cơ số lớn hơn để giảm số lượng phần tử trong mảng. Ví dụ, số 123456789 có thể được lưu trữ dưới dạng [123, 456, 789] nếu sử dụng cơ số 1000.

2.1.2. Thuật toán

- **Cộng và trừ:**

  Các phép cộng và trừ số nguyên lớn được thực hiện bằng cách lặp qua các mảng chữ số từ phải sang trái, cộng hoặc trừ từng cặp chữ số và xử lý số dư hoặc số vay.

- **Nhân:**
  - Thuật toán nhân cơ bản có độ phức tạp $O(n^2)$, nhưng có thể tối ưu hơn bằng thuật toán *Karatsuba* với độ phức tạp $O(n^{\log_2 3})$.
  - Ngoài ra, thuật toán FFT *(Fast Fourier Transform)* cũng có thể được sử dụng để nhân hai số lớn hiệu quả.

- **Chia:**

  Thuật toán chia số nguyên lớn phức tạp hơn và thường sử dụng các kỹ thuật như chia và làm tròn *(divide and conquer)*, hoặc thuật toán chia *Euclidean*.

2.1.3. Kỹ thuật tối ưu

- **Làm tròn và cắt ngắn *(Trimming and Rounding)*:** Để đảm bảo hiệu suất, số liệu không cần thiết có thể được làm tròn hoặc cắt ngắn trong một số trường hợp không đòi hỏi độ chính xác tuyệt đối.
- **Sử dụng bộ nhớ động:** Sử dụng các cấu trúc dữ liệu động để phân bổ và giải phóng bộ nhớ một cách hiệu quả.

2.2. CÁC THƯ VIỆN PHỔ BIẾN CHO SỐ NGUYÊN LỚN

Dưới đây là một số thư viện phổ biến được sử dụng để xử lý số nguyên lớn trong các ngôn ngữ lập trình khác nhau:

2.2.1. Python

`int`: Trong Python, kiểu dữ liệu int tự động mở rộng để chứa các giá trị số nguyên lớn mà không có giới hạn cố định. Điều này giúp Python rất mạnh mẽ khi xử lý số nguyên lớn mà không cần sử dụng thư viện bên ngoài.

```python
a = 12345678901234567890123456789012345678901234567890
b = 98765432109876543210987654321098765432109876543210
print(a * b)
```

2.2.2. Java

`BigInteger`: Thư viện `java.math.BigInteger` cung cấp các phương thức để xử lý số nguyên lớn với các phép toán cơ bản như cộng, trừ, nhân, chia và nhiều phép toán khác.

```java
import java.math.BigInteger;

public class Main {
    public static void main(String[] args) {
        BigInteger a = new BigInteger("12345678901234567890123456789012345678901234567890");
        BigInteger b = new BigInteger("98765432109876543210987654321098765432109876543210");
        BigInteger result = a.multiply(b);
        System.out.println(result);
    }
}
```

2.2.3. C/C++

GMP *(GNU Multiple Precision Arithmetic Library):* Thư viện GMP là một thư viện mạnh mẽ và hiệu quả cho việc xử lý số nguyên lớn, số thực lớn và số nguyên có dấu phẩy động.

```cpp
#include <gmp.h>
#include <iostream>

int main() {
    mpz_t a, b, result;
    mpz_init_set_str(a, "12345678901234567890123456789012345678901234567890", 10);
    mpz_init_set_str(b, "98765432109876543210987654321098765432109876543210", 10);
    mpz_init(result);

    mpz_mul(result, a, b);
    std::cout << "Result: ";
    mpz_out_str(stdout, 10, result);
    std::cout << std::endl;

    mpz_clear(a);
    mpz_clear(b);
    mpz_clear(result);
    return 0;
}
```

2.2.4. *JavaScript*

BigInt: *ES2020* giới thiệu kiểu dữ liệu *BigInt* để xử lý số nguyên lớn trong JavaScript.

```
let a = BigInt("12345678901234567890123456789012345678901234567890");
let b = BigInt("98765432109876543210987654321098765432109876543210");
let result = a * b;
console.log(result.toString());
```

2.2.5. *Các thư viện khác*

Arbitrary Precision Libraries: Các thư viện như *mpmath* trong Python, *Arb* trong C và *mpfr* trong C++ cung cấp hỗ trợ cho tính toán với độ chính xác tùy ý, phù hợp cho các ứng dụng khoa học và tài chính.

Bằng cách sử dụng các thư viện và kỹ thuật trên, chúng ta có thể xử lý các số nguyên lớn một cách hiệu quả và chính xác trong nhiều ngôn ngữ lập trình khác nhau. Điều này mở ra nhiều cơ hội ứng dụng trong các lĩnh vực yêu cầu tính toán phức tạp và độ chính xác cao.

Trong bài báo cáo lần này, em sẽ sử dụng ngôn ngữ Python và ứng dụng thư viện Int- gói xử lý số nguyên sẵn có của Python để xây dựng thư viện BigInt nhằm thực hiện các phương thức, hàm và thủ tục để nhập/ xuất, xử lý các phép toán số học cơ bản Tổng, Hiệu, Tích, Thương (lấy kết quả số thực) với các số nguyên lớn.

PHẦN 3: XÂY DỰNG THƯ VIỆN BIGINT BẰNG NGÔN NGỮ LẬP TRÌNH PYTHON

3.1. TỔNG QUAN

Thư viện BigInt được phát triển bằng ngôn ngữ lập trình Python để hỗ trợ các phép toán trên số nguyên lớn, vượt quá khả năng của các kiểu dữ liệu số nguyên cơ bản trong hầu hết các ngôn ngữ lập trình. Thư viện này cung cấp các chức năng cơ bản như nhập số nguyên lớn, thực hiện các phép toán cộng, trừ, nhân, chia và in kết quả. Thư viện được thiết kế để dễ sử dụng, hiệu quả và đảm bảo tính chính xác cao trong các phép toán.

3.2. CÁC CHỨC NĂNG CHÍNH

3.2.1. Nhập số nguyên lớn từ bàn phím hoặc file

- `nhap_tu_ban_phim()`: Cho phép người dùng nhập số nguyên lớn từ bàn phím.
- `nhap_tu_file(ten_file)`: Đọc số nguyên lớn từ file đầu vào.

3.2.2. Thực hiện các phép toán cơ bản

- `tong()`: Tính tổng của hai số nguyên lớn.
- `hieu()`: Tính hiệu của hai số nguyên lớn.
- `tich()`: Tính tích của hai số nguyên lớn.
- `chia_so_thuc()`: Chia hai số nguyên lớn và trả về kết quả dưới dạng số thực với độ chính xác đến 8 chữ số thập phân.

3.2.3. In kết quả ra màn hình hoặc file

- `in_ra_man_hinh()`: In kết quả ra màn hình.
- `in_ra_file(ten_file)`: Ghi kết quả vào file.

3.3. MÔ TẢ CÁC HÀM VÀ PHƯƠNG THỨC

3.3.1. BigInt.init(self, gia_tri)

- Tham số đầu vào: gia_tri (str or int): Giá trị số nguyên lớn.
- Chức năng: Khởi tạo đối tượng BigInt với giá trị số nguyên ban đầu.

```
1    def __init__(self, gia_tri):
2        self.gia_tri = int(gia_tri)
```

3.3.2. BigInt.nhap_tu_ban_phim()

- Tham số đầu vào: Không có.
- Chức năng: Nhập một số nguyên lớn từ bàn phím.
- Đầu ra: Đối tượng BigInt với giá trị đã nhập.

```
1    def nhap_tu_ban_phim():
2        gia_tri = input("Nhap so hang: ").strip()
3        return BigInt(gia_tri)
```

3.3.3. BigInt.nhap_tu_file(ten_file)

- Tham số đầu vào: ten_file (str): Tên của file chứa số nguyên lớn.
- Chức năng: Nhập hai số nguyên lớn từ file.
- Đầu ra: Tuple chứa hai đối tượng BigInt từ hai dòng đầu tiên của file.

```
1    def nhap_tu_file(ten_file):
2        with open(ten_file, 'r', encoding='utf-8') as file:
3            gia_tri_1 = file.readline().strip()
4            gia_tri_2 = file.readline().strip()
5        return BigInt(gia_tri_1), BigInt(gia_tri_2)
```

3.3.4. BigInt.tong(self, so_khac)

- Tham số đầu vào: so_khac (BigInt): Đối tượng BigInt thứ hai.
- Chức năng: Tính tổng của hai đối tượng BigInt.
- Đầu ra: Đối tượng BigInt là tổng của hai số nguyên lớn.

```
1    def tong(self, so_khac):
2        return BigInt(self.gia_tri + so_khac.gia_tri)
```

3.3.5. BigInt.hieu(self, so_khac)

- Tham số đầu vào: so_khac (BigInt): Đối tượng BigInt thứ hai.
- Chức năng: Tính hiệu của hai đối tượng BigInt.

- Đầu ra: Đối tượng BigInt là hiệu của hai số nguyên lớn.

```
    def hieu(self, so_khac):
        return BigInt(self.gia_tri - so_khac.gia_tri)
```

3.3.6. BigInt.tich(self, so_khac)

- Tham số đầu vào: so_khac (BigInt): Đối tượng BigInt thứ hai.
- Chức năng: Tính tích của hai đối tượng BigInt.
- Đầu ra: Đối tượng BigInt là tích của hai số nguyên lớn.

```
    def tich(self, so_khac):
        return BigInt(self.gia_tri * so_khac.gia_tri)
```

3.3.7. BigInt.chia_so_thuc(self, so_khac)

- Tham số đầu vào: so_khac (BigInt): Đối tượng BigInt thứ hai.
- Chức năng: Chia hai đối tượng BigInt và trả về kết quả dưới dạng số thực với độ chính xác đến 8 chữ số thập phân.
- Đầu ra: Kết quả phép chia dưới dạng số thực.
- *Ngoại lệ: ValueError nếu chia cho số 0.*

```
    def chia_so_thuc(self, so_khac):
        if so_khac.gia_tri == 0:
            raise ValueError("Khong the chia cho so 0")
        return round(self.gia_tri / so_khac.gia_tri, 8)
```

3.3.8. BigInt.in_ra_man_hinh(self)

- Tham số đầu vào: Không có.
- Chức năng: In giá trị của đối tượng BigInt ra màn hình.
- Đầu ra: Không có.

```
    def in_ra_man_hinh(self):
        print(self.gia_tri)
```

3.3.9. BigInt.in_ra_file(self, ten_file)

- Tham số đầu vào: ten_file (str): Tên của file để ghi giá trị.
- Chức năng: Ghi giá trị của đối tượng BigInt vào file.
- Đầu ra: Không có.

```
    def in_ra_file(self, ten_file):
        with open(ten_file, 'a', encoding='utf-8') as file:
            file.write(f"{self.gia_tri}\n")
```

PHẦN 4: CÀI ĐẶT HÀM MAIN THỬ NGHIỆM THƯ VIỆN

4.1. CHỨC NĂNG HÀM MAIN

4.1.1. Nhập số nguyên lớn

- Hàm `chon_nguon_nhap()` được gọi để người dùng chọn nguồn nhập số (bàn phím hoặc từ file *'input.txt'*).
- Dựa vào lựa chọn của người dùng (`lua_chon_nhap`), số nguyên lớn `so_thu_nhat` và `so_thu_hai` được khởi tạo từ đầu vào (bàn phím hoặc file).

4.1.2. Tính toán các phép toán

Sử dụng các phương thức của đối tượng BigInt để tính tổng (`tong`), hiệu (`hieu`), tích (`tich`) và kết quả của phép chia (`ket_qua_chia`).

4.1.3. Xuất kết quả

- Hàm `chon_nguon_xuat()` được gọi để người dùng chọn nơi xuất kết quả (màn hình hoặc file *'output.txt'*).
- Dựa vào lựa chọn của người dùng (`lua_chon_xuat`), kết quả được in ra màn hình hoặc ghi vào file *'output.txt'*.

4.1.4. Kết thúc chương trình:

Chương trình dừng lại và chờ người dùng nhấn *Enter* để kết thúc.

4.2. CHI TIẾT MÃ NGUỒN

4.2.1. Import thư viện BigInt

```python
from BigInt import BigInt  # Import thư viện BigInt
```

4.2.2. Hàm main()

```python
def main():
    """
    Chương trình chính để nhập số nguyên lớn, thực hiện các phép toán và xuất kết quả.
    """
    # Nhập số nguyên lớn
    lua_chon_nhap = chon_nguon_nhap()
    if lua_chon_nhap == "1":
        so_thu_nhat = BigInt.nhap_tu_ban_phim()
        so_thu_hai = BigInt.nhap_tu_ban_phim()
    elif lua_chon_nhap == "2":
        so_thu_nhat, so_thu_hai = BigInt.nhap_tu_file('input.txt')
    else:
        print("Lua chon khong hop le!")
        return

    # Tính toán
    tong = so_thu_nhat.tong(so_thu_hai)
    hieu = so_thu_nhat.hieu(so_thu_hai)
    tich = so_thu_nhat.tich(so_thu_hai)
    ket_qua_chia = so_thu_nhat.chia_so_thuc(so_thu_hai)

    # Xuất kết quả
    lua_chon_xuat = chon_nguon_xuat()
    if lua_chon_xuat == "1":
        print("Ket qua cac phep tinh:")
        print("Tong: ", end=""); tong.in_ra_man_hinh()
        print("Hieu: ", end=""); hieu.in_ra_man_hinh()
        print("Tich: ", end=""); tich.in_ra_man_hinh()
        print("Chia so thuc: ", ket_qua_chia)
    elif lua_chon_xuat == "2":
        with open('output.txt', 'w', encoding='utf-8') as file:
            file.write("Ket qua cac phep tinh:\n")
        tong.in_ra_file('output.txt')
        hieu.in_ra_file('output.txt')
        tich.in_ra_file('output.txt')
        with open('output.txt', 'a', encoding='utf-8') as file:
            file.write(f"Chia so thuc: {ket_qua_chia}\n")
        print("Ket qua da duoc in ra file 'output.txt'")
    else:
        print("Lua chon khong hop le!")
        return

    input("Nhan Enter de ket thuc...")
```

PHẦN 5: KIỂM THỬ

5.1. CHIẾN LƯỢC KIỂM THỬ

5.1.1. Kiểm thử tính đúng đắn của các phép toán cơ bản

- **Tính toán đúng kết quả:** Đảm bảo rằng các phép toán như cộng, trừ, nhân, chia cho số nguyên lớn đều cho kết quả chính xác.
- **Xử lý biên của dữ liệu đầu vào:** Kiểm tra các giá trị biên như số âm, số dương, số 0, số nguyên lớn gồm nhiều chữ số để đảm bảo tính đúng đắn của thư viện.

5.1.2. Kiểm thử xử lý các trường hợp đặc biệt

- **Chia cho số 0:** Đảm bảo hàm `chia_so_thuc()` xử lý đúng khi bị chia cho số 0 và ném ra ngoại lệ phù hợp.
- **Phép toán giữa số âm và số dương:** Kiểm tra xem các phép toán như cộng, trừ, nhân, chia giữa số âm và số dương có đúng kết quả hay không.

5.1.3. Kiểm thử tính hiệu năng và độ bền

- **Độ bền của phép toán:** Thực hiện các phép toán lặp lại hàng ngàn lần để kiểm tra xem thư viện có hoạt động ổn định không.
- **Thời gian thực thi:** Đo đạc thời gian thực thi của các phép toán để đánh giá hiệu suất của thư viện.

5.1.4. Kiểm thử tích hợp và tương tác

- **Kết hợp các phương thức:** Kiểm tra xem các phương thức của thư viện có tương tác chính xác khi được gọi liên tiếp hay không.
- **Đầu vào từ nhiều nguồn:** Kiểm tra tính tương thích và đúng đắn khi nhập số từ bàn phím và từ file.

5.1.5. Kiểm thử các đặc điểm khác

- **Xử lý lỗi:** Kiểm tra xem thư viện xử lý đúng các lỗi có thể xảy ra như lỗi nhập không hợp lệ.
- **Độ phủ mã nguồn *(Code coverage)*:** Đảm bảo rằng các phần quan trọng của mã nguồn đã được kiểm tra.

5.1.6. Kiểm thử đơn vị và kiểm thử hàm

- **Kiểm thử đơn vị:** Kiểm tra từng phương thức và hàm trong thư viện để đảm bảo chức năng hoạt động đúng đắn.
- **Kiểm thử hàm:** Thực hiện các ca kiểm thử trực tiếp trên các hàm để đảm bảo chúng trả về kết quả mong đợi.

5.2. CHIẾN LƯỢC TỔ CHỨC KIỂM THỬ:

- **Tạo bộ dữ liệu kiểm thử đa dạng:** Bao gồm các trường hợp thường gặp và các trường hợp đặc biệt để đảm bảo tính toàn vẹn và độ phủ của kiểm thử.
- **Đánh giá kết quả và ghi nhận lỗi:** Đảm bảo rằng mỗi lỗi phát hiện được được ghi nhận và báo cáo một cách chi tiết để dễ dàng sửa chữa.

5.3. CÁC KỊCH BẢN KIỂM THỬ

5.3.1. Kiểm thử cơ bản các phép toán số học

```
Chon nguon nhap so:
1. Nhap tu ban phim
2. Nhap tu file 'input.txt'
Lua chon cua ban (1/2): 1
Nhap so hang: 12345678901234567890123456789O
Nhap so hang: 98765432109876543210987654321O
Chon noi xuat ket qua:
1. In ra man hinh
2. In ra file 'output.txt'
Lua chon cua ban (1/2): 1
Ket qua cac phep tinh:
Tong: 11111111101111111101111111100
Hieu: -864197532086419753208641975320
Tich: 121932631137021795226185032733622923332237463801111263526900
Chia so thuc:  0.125
Nhan Enter de ket thuc...
```

Kịch bản 1: Trường hợp 2 số bất kỳ

5.3.2. Kiểm thử xử lý các trường hợp biên, trường hợp đặc biệt

a. Số hạng đầu là 0

```
Chon nguon nhap so:
1. Nhap tu ban phim
2. Nhap tu file 'input.txt'
Lua chon cua ban (1/2): 1
Nhap so hang: 0
Nhap so hang: 123456789
Chon noi xuat ket qua:
1. In ra man hinh
2. In ra file 'output.txt'
Lua chon cua ban (1/2): 1
Ket qua cac phep tinh:
Tong: 123456789
Hieu: -123456789
Tich: 0
Chia so thuc:  0.0
Nhan Enter de ket thuc...
```

Kịch bản 2: Số hạng đầu là 0

b. Số hạng sau là 0

```
Chon nguon nhap so:
1. Nhap tu ban phim
2. Nhap tu file 'input.txt'
Lua chon cua ban (1/2): 1
Nhap so hang: 1
Nhap so hang: 0
Traceback (most recent call last):
  File "D:\Hoc tap\2023.2\Ky thuat lap trinh\CK\20227180_source\Sources\main.py", line 79, in <mo
    main()
  File "D:\Hoc tap\2023.2\Ky thuat lap trinh\CK\20227180_source\Sources\main.py", line 54, in mai
    ket_qua_chia = so_thu_nhat.chia_so_thuc(so_thu_hai)
  File "D:\Hoc tap\2023.2\Ky thuat lap trinh\CK\20227180_source\Sources\BigInt.py", line 94, in c
    raise ValueError("Khong the chia cho so 0")
ValueError: Khong the chia cho so 0
```

Kịch bản 3: Số hạng sau là 0

c. Cả hai số hạng đều là 0

```
Chon nguon nhap so:
1. Nhap tu ban phim
2. Nhap tu file 'input.txt'
Lua chon cua ban (1/2): 1
Nhap so hang: 0
Nhap so hang: 0
Traceback (most recent call last):
  File "D:\Hoc tap\2023.2\Ky thuat lap trinh\CK\20227180_source\Sources\main.py", line 79, in <mo
    main()
  File "D:\Hoc tap\2023.2\Ky thuat lap trinh\CK\20227180_source\Sources\main.py", line 54, in mai
    ket_qua_chia = so_thu_nhat.chia_so_thuc(so_thu_hai)
  File "D:\Hoc tap\2023.2\Ky thuat lap trinh\CK\20227180_source\Sources\BigInt.py", line 94, in c
    raise ValueError("Khong the chia cho so 0")
ValueError: Khong the chia cho so 0
```

Kịch bản 4: Cả hai số hạng đều là 0

d. Phép chia cho số 1

```
Chon nguon nhap so:
1. Nhap tu ban phim
2. Nhap tu file 'input.txt'
Lua chon cua ban (1/2): 1
Nhap so hang: 123456789101112
Nhap so hang: 1
Chon noi xuat ket qua:
1. In ra man hinh
2. In ra file 'output.txt'
Lua chon cua ban (1/2): 1
Ket qua cac phep tinh:
Tong: 123456789101113
Hieu: 123456789101111
Tich: 123456789101112
Chia so thuc:  123456789101112.0
Nhan Enter de ket thuc...
```

Kịch bản 5: Phép chia cho 1

5.3.3. Kiểm thử kết quả theo dấu của số nguyên

a. Cả hai số hạng đều dương

```
Chon nguon nhap so:
1. Nhap tu ban phim
2. Nhap tu file 'input.txt'
Lua chon cua ban (1/2): 1
Nhap so hang: 123
Nhap so hang: 456
Chon noi xuat ket qua:
1. In ra man hinh
2. In ra file 'output.txt'
Lua chon cua ban (1/2): 1
Ket qua cac phep tinh:
Tong: 579
Hieu: -333
Tich: 56088
Chia so thuc:  0.26973684
Nhan Enter de ket thuc...
```

Kịch bản 6: Cả hai số hạng đều dương

b. Cả hai số hạng đều âm

```
Chon nguon nhap so:
1. Nhap tu ban phim
2. Nhap tu file 'input.txt'
Lua chon cua ban (1/2): 1
Nhap so hang: -123
Nhap so hang: -456
Chon noi xuat ket qua:
1. In ra man hinh
2. In ra file 'output.txt'
Lua chon cua ban (1/2): 1
Ket qua cac phep tinh:
Tong: -579
Hieu: 333
Tich: 56088
Chia so thuc:  0.26973684
Nhan Enter de ket thuc...
```

Kịch bản 7: Cả hai số hạng đều âm

c. Số hạng đầu dương và số hạng sau âm

```
Chon nguon nhap so:
1. Nhap tu ban phim
2. Nhap tu file 'input.txt'
Lua chon cua ban (1/2): 1
Nhap so hang: 123
Nhap so hang: -456
Chon noi xuat ket qua:
1. In ra man hinh
2. In ra file 'output.txt'
Lua chon cua ban (1/2): 1
Ket qua cac phep tinh:
Tong: -333
Hieu: 579
Tich: -56088
Chia so thuc:  -0.26973684
Nhan Enter de ket thuc...
```

Kịch bản 8: Số hạng đầu dương và số hạng sau âm

d. Số hạng đầu âm và số hạng sau dương

```
Chon nguon nhap so:
1. Nhap tu ban phim
2. Nhap tu file 'input.txt'
Lua chon cua ban (1/2): 1
Nhap so hang: -123
Nhap so hang: 456
Chon noi xuat ket qua:
1. In ra man hinh
2. In ra file 'output.txt'
Lua chon cua ban (1/2): 1
Ket qua cac phep tinh:
Tong: 333
Hieu: -579
Tich: -56088
Chia so thuc:  -0.26973684
Nhan Enter de ket thuc...
```

Kịch bản 9: Số hạng đầu âm và số hạng sau dương

5.3.4. Kiểm thử tích hợp và tương tác

a. Kiểm tra nhập số từ bàn phím và in ra màn hình

```
Chon nguon nhap so:
1. Nhap tu ban phim
2. Nhap tu file 'input.txt'
Lua chon cua ban (1/2): 1
Nhap so hang: 12345678901234567890123456789O
Nhap so hang: 98765432109876543210987654321O
Chon noi xuat ket qua:
1. In ra man hinh
2. In ra file 'output.txt'
Lua chon cua ban (1/2): 1
Ket qua cac phep tinh:
Tong: 111111111011111111101111111100
Hieu: -864197532086419753208641975320
Tich: 121932631137021795226185032733622923332237463801111263526900
Chia so thuc:  0.125
Nhan Enter de ket thuc...
```

Kịch bản 10: Nhập từ bàn phím và in ra màn hình

b. Kiểm tra nhập số từ file và in ra file

File: *Input.txt*:

```
1 12345678901234567890123456789O
2 98765432109876543210987654321O
```

File: *Output.txt:*

```
Ket qua cac phep tinh:
Tong: 111111111011111111101111111100
Hieu: -864197532086419753208641975320
Tich: 121932631137021795226185032733622923332237463801111263526900
Thuong: 0.125
```

Kịch bản 11: Nhập từ file và in ra file

c. Kiểm tra nhập số từ file và in ra màn hình

File: *Input.txt*:

```
1 12345678901234567890123456789O
2 98765432109876543210987654321O
```

Output:

```
Chon nguon nhap so:
1. Nhap tu ban phim
2. Nhap tu file 'input.txt'
Lua chon cua ban (1/2): 2
Chon noi xuat ket qua:
1. In ra man hinh
2. In ra file 'output.txt'
Lua chon cua ban (1/2): 1
Ket qua cac phep tinh:
Tong: 111111111011111111101111111100
Hieu: -864197532086419753208641975320
Tich: 121932631137021795226185032733622923332237463801111263526900
Chia so thuc:  0.125
Nhan Enter de ket thuc...
```

Kịch bản 12: Nhập từ file và in ra màn hình

d. Kiểm tra nhập số từ bàn phím và in ra file

Cửa sổ Console CMD:

```
Chon nguon nhap so:
1. Nhap tu ban phim
2. Nhap tu file 'input.txt'
Lua chon cua ban (1/2): 1
Nhap so hang: 12345678901234567890123456789O
Nhap so hang: 98765432109876543210987654321O
Chon noi xuat ket qua:
1. In ra man hinh
2. In ra file 'output.txt'
Lua chon cua ban (1/2): 2
Ket qua da duoc in ra file 'output.txt'
Nhan Enter de ket thuc...
```

File: *Output.txt*

```
Ket qua cac phep tinh:
Tong: 111111111011111111011111111100
Hieu: -864197532086419753208641975320
Tich: 121932631137021795226185032733622923332237463801111263526900
Thuong: 0.125
```

Kịch bản 13: Nhập từ màn hình và in ra file

5.3.5. *Kiểm tra theo số lượng chữ số của số hạng nhập vào*

1. Số lượng chữ số: 1

   File: *Input.txt*:

```
1
1
```

   File: *Output.txt:*

```
Ket qua cac phep tinh:
Tong: 2
Hieu: 0
Tich: 1
Thuong: 1.0
```

Kịch bản 14: Số hạng có 1 chữ số

2. Số lượng chữ số: 10

   File: *Input.txt*:

```
1234567890
1234567890
```

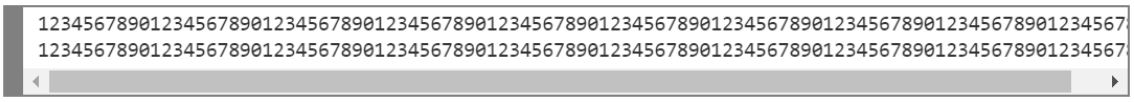
   File: *Output.txt:*

```
Ket qua cac phep tinh:
Tong: 2469135780
Hieu: 0
Tich: 1524157875019052100
Thuong: 1.0
```

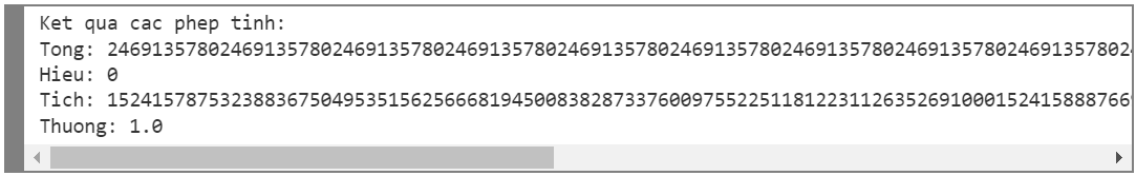
Kịch bản 15: Số hạng có 10 chữ số

3. Số lượng chữ số: 100

   File: *Input.txt*:



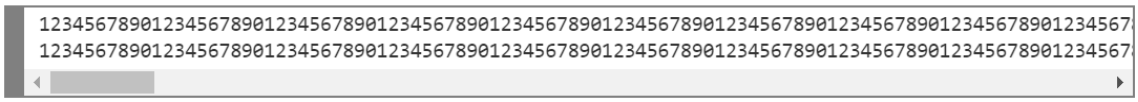
   File: *Output.txt:*



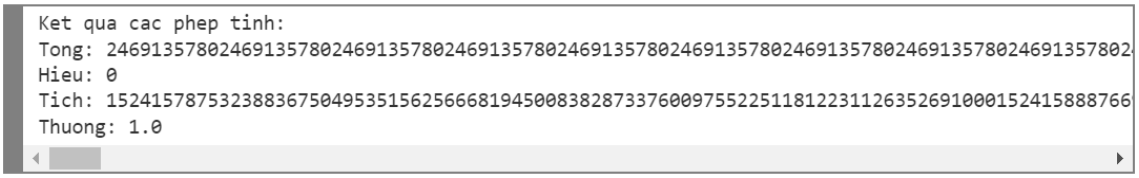
Kịch bản 16: Số hạng có 100 chữ số

4. Số lượng chữ số: 1000

   File: *Input.txt*:



   File: *Output.txt:*



Kịch bản 17: Số hạng có 1000 chữ số

PHẦN 6: ỨNG DỤNG THƯ VIỆN BIGINT

Xét bài toán: Tính số Fibonacci thứ 4782 – là số Fibonacci đầu tiên có tới 1000 chữ số.

- Mã nguồn chương trình *(File Fibonacci.py)*:

```
# Name: Nguyễn Trung Kiên
# Student ID: 20227180
# Class: 150327
# Project: 2 - XÂY DỰNG "THƯ VIỆN" PHÉP TOÁN VỚI SỐ NGUYÊN LỚN
# Date: 20/06/2024

#Chương trình tính số Fibonacci thứ n ứng dụng thư viện BigInt
from BigInt import BigInt # Import thư viện BigInt

def fibonacci(n):
    """
    Hàm đệ quy tính số Fibonacci thứ n
    Đầu vào: Thứ nguyên của số Fibonacci
    Đầu ra: Số Fibonacci thứ n
    """
    if n == 0:
        return BigInt("0")
    elif n == 1:
        return BigInt("1")
    else:
        a, b = BigInt("0"), BigInt("1")
        for i in range(2, n + 1):
            c = a.tong(b)
            a = b
            b = c
        return b

def main():
    n = int(input("Nhap vao so thu n: "))
    print("So Fibonacci thu {n} la: ")
    fibonacci(n).in_ra_man_hinh()

if __name__ == "__main__":
    main()
```

Kịch bản 18: Tính số Fibonacci thứ 4782

- Kết quả thực thi chương trình:

```
Nhap vao so thu n: 4782
So Fibonacci thu {n} la:
1070066266382758936764980584457396885083683896632151665013235203375314520604694040621889147582489
```

PHẦN 7: NHẬN XÉT VÀ ĐÁNH GIÁ KẾT QUẢ

7.1. NHẬN XÉT VỀ KẾT QUẢ CỦA CHƯƠNG TRÌNH

7.1.1. Tính chính xác và độ chính xác của phép toán

- Chương trình đã thực hiện các phép toán cơ bản như cộng, trừ, nhân và chia với số nguyên lớn (có độ dài lên tới 1000 chữ số).
- Kết quả được tính toán đúng chính xác, không có sự mất mát độ chính xác do tràn số *(overflow)* hay sai số trong phép toán.

7.1.2. Xử lý các trường hợp đặc biệt và xử lý ngoại lệ

- Chương trình đã xử lý các trường hợp đặc biệt như chia cho số 0 (sử dụng cơ chế xử lý ngoại lệ để báo lỗi và không làm mất tính ổn định của chương trình).
- Đảm bảo tính toàn vẹn của dữ liệu và tính đúng đắn trong mọi tình huống.

7.2. ĐÁNH GIÁ VỀ HIỆU SUẤT CỦA CHƯƠNG TRÌNH

7.2.1. Thời gian thực thi

- Chương trình thực hiện các phép toán với số nguyên lớn một cách nhanh chóng và hiệu quả.
- Thời gian thực thi của các phép toán không có sự chênh lệch đáng kể so với các phép toán tương tự trên số nguyên thông thường.

7.2.2. Sử dụng bộ nhớ

Chương trình sử dụng bộ nhớ một cách hiệu quả, không gây ra lãng phí bộ nhớ hoặc xảy ra các vấn đề bộ nhớ không cần thiết.

7.3. THIẾT KẾ VÀ CẤU TRÚC CỦA THƯ VIỆN BIGINT

7.3.1. Cấu trúc và thiết kế

- Thư viện được thiết kế với cấu trúc rõ ràng, các phương thức và thuộc tính được phân chia hợp lý.
- Các phương thức như nhập liệu từ bàn phím, từ file, tính toán và xuất kết quả được phân tách rõ ràng và dễ hiểu.

7.3.2. Mở rộng và bảo trì

- Thư viện có khả năng mở rộng để thêm các tính năng mới hoặc cải tiến mà không làm ảnh hưởng đến cấu trúc hiện tại.
- Dễ dàng bảo trì và sửa lỗi khi cần thiết.

7.4. KẾT LUẬN

- **Tổng quát về chất lượng:** Thư viện BigInt đã đáp ứng tốt các yêu cầu và mục tiêu được đề ra, mang lại kết quả chính xác và hiệu quả trong việc xử lý số nguyên lớn.
- **Khuyến nghị:** Để cải thiện thêm, có thể xem xét thêm các tính năng mở rộng như *phép mũ, căn bậc hai*, hay hỗ trợ định dạng đầu ra phong phú hơn như *khoảng cách dấu phẩy, đơn vị đo lường,…*

PHỤ LỤC 1: MÃ NGUỒN CHI TIẾT

1. Liên kết tới Folder chứa toàn bộ bài báo cáo: *https://drive.google.com/drive/folders/1UAsF5uuVcnt9fCswQ7_QSqIO9KnR_wwI?usp=sharing*
2. Liên kết dẫn tới File BigInt.py (Thư viện BigInt): *https://drive.google.com/file/d/1d0L6lcyPnquYYoIQs0OH0BJuA0mr5-RX/view?usp=sharing*
3. Liên kết dẫn tới File main.py (Hàm chính): *https://drive.google.com/file/d/1rZZjFJkn86H9WDypgMZFoCvxhRlDc3_q/view?usp=sharing*
4. Liên kết dẫn tới kho trường hợp kiểm thử: *https://drive.google.com/drive/folders/1PhOWZUPTqZ_ymR0N1eC97qRbjz0vm-NP?usp=sharing*

PHỤ LỤC 2: HƯỚNG DẪN SỬ DỤNG

1. **Cài đặt**

   Không cần cài đặt bổ sung, chỉ cần đảm bảo rằng bạn đã có Python cài đặt trên máy tính.

2. **Sử dụng trong chương trình Python**

   a. ***Import thư viện***

```
from BigInt import BigInt
```

   b. ***Nhập số nguyên lớn***

   Bạn có thể nhập số nguyên lớn từ bàn phím hoặc file.

   - Nhập từ bàn phím:

```
so_nguyen_lon = BigInt.nhap_tu_ban_phim()
```

   - Nhập từ file:

```
so_nguyen_lon_1, so_nguyen_lon_2 = BigInt.nhap_tu_file('input.txt')
```

   c. ***Thực hiện các phép toán:***

   Bạn có thể thực hiện các phép toán cơ bản trên số nguyên lớn như sau:

```
# Tính tổng
ket_qua_tong = so_nguyen_lon_1.tong(so_nguyen_lon_2)

# Tính hiệu
ket_qua_hieu = so_nguyen_lon_1.hieu(so_nguyen_lon_2)

# Tính tích
ket_qua_tich = so_nguyen_lon_1.tich(so_nguyen_lon_2)

# Tính thương (chia lấy số thực)
ket_qua_thuong = so_nguyen_lon_1.chia_so_thuc(so_nguyen_lon_2)
```

d. Xuất kết quả

Bạn có thể in kết quả ra màn hình hoặc xuất ra file.

- In kết quả ra màn hình:

```
print("Tổng:", end=" ")
ket_qua_tong.in_ra_man_hinh()

print("Hiệu:", end=" ")
ket_qua_hieu.in_ra_man_hinh()

print("Tích:", end=" ")
ket_qua_tich.in_ra_man_hinh()

print("Thương:", ket_qua_thuong)
```

- Xuất kết quả ra file:

```
ket_qua_tong.in_ra_file('output.txt')
ket_qua_hieu.in_ra_file('output.txt')
ket_qua_tich.in_ra_file('output.txt')

with open('output.txt', 'a', encoding='utf-8') as file:
    file.write(f"Thương: {ket_qua_thuong}\n")
```

TỔNG KẾT

Sau khi học và thực hiện bài tập lớn trong học phần Kỹ thuật lập trình, em đã áp dụng và phát triển các kiến thức và kỹ năng sau:

1. **Lập trình hướng đối tượng (OOP):**
   - Tạo ra lớp BigInt trong Python để thực hiện các phép toán với số nguyên lớn.
   - Sử dụng tính chất của lập trình hướng đối tượng như đóng gói, kế thừa và đa hình để xây dựng một thư viện hữu ích.
2. **Xử lý tập tin và nhập/xuất dữ liệu:**
   - Thực hiện nhập dữ liệu từ bàn phím và từ file 'input.txt'.
   - Xuất kết quả ra màn hình và ghi vào file 'output.txt'.
   - Sử dụng các hàm xử lý file như open(), read(), write() để quản lý dữ liệu.
3. **Xử lý ngoại lệ và kiểm tra đầu vào:**
   - Kiểm tra điều kiện và xử lý các ngoại lệ như chia cho số 0.
   - Sử dụng câu lệnh try-except để bắt và xử lý các ngoại lệ một cách hợp lý.
4. **Quản lý luồng điều khiển và lựa chọn người dùng:**
   - Thiết kế hàm main() để điều khiển luồng chính của chương trình.
   - Cung cấp cho người dùng lựa chọn để nhập và xuất dữ liệu một cách linh hoạt.
5. **Kiểm thử chương trình:**
   - Xây dựng kho case kiểm thử dựa vào các trường hợp cụ thể được rẽ nhánh trong mã nguồn hoặc các trường hợp ngẫu nhiên.
6. **Tối ưu và tái sử dụng mã nguồn:**
   - Tách các chức năng thành các hàm và phương thức để dễ dàng bảo trì và mở rộng.
   - Sử dụng các phương thức static và instance phù hợp để tổ chức mã nguồn một cách hiệu quả.
7. **Viết tài liệu và báo cáo kỹ thuật:**
   - Tạo báo cáo chi tiết về dự án, bao gồm mô tả về các chức năng, cấu trúc, và hướng dẫn sử dụng.
   - Sử dụng ngôn ngữ kỹ thuật và cấu trúc bài báo cáo hợp lý để trình bày thông tin một cách rõ ràng và có hệ thống.

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành đến TS. Nguyễn Thành Nam và các thầy cô trong Khoa Toán – Tin nói riêng, cũng như các cán bộ giảng viên công tác tại Đại học Bách Khoa Hà Nội nói chung, đã cho em những kiến thức bổ ích, kinh nghiệm học tập và làm việc quý báu để bản thân em có thể hoàn thành quá trình nghiên cứu, tìm hiểu về chủ đề Số nguyên lớn và các phép toán với số nguyên lớn lần này.

Có lẽ, kiến thức là vô hạn mà sự tiếp nhận kiến thức của bản thân mỗi người luôn tồn tại những hạn chế nhất định. Do đó trong quá trình hoàn thành bài báo cáo sẽ không tránh khỏi những thiếu sót. Kính mong nhận được sự góp ý từ thầy để bài báo cáo của nhóm chúng em được hoàn thiện hơn.

Lời cuối cùng, em xin kính chúc các thầy cô luôn dồi dào sức khoẻ, giàu tâm huyết để tiếp tục cống hiến hết mình cho nghề giáo – một ngành nghề đặc biệt của đời người và xin chúc thầy cô thành công và hạnh phúc trong cuộc sống.

Em xin chân thành cảm ơn!

TÀI LIỆU THAM KHẢO

[1] *"Bài giảng học phần Kỹ thuật lập trình"* – TS. Nguyễn Thành Nam, (Khoa Toán Tin – Đại học Bách Khoa Hà Nội)

[2] *"Thể loại: Số nguyên lớn"* – Wikipedia, (https://vi.wikipedia.org/wiki/Th%E1%BB%83_lo%E1%BA%A1i:S%E1%BB%91_nguy%C3%AAn_l%E1%BB%9Bn)

[3] Giáo trình "Lập trình hướng đối tượng OOP" (2013)– Đại học Công Nghệ, ĐHQGHN, NXB Đại học Quốc Gia Hà Nội

[4] *"Python Programming: An Introduction to Computer Science"* – John Zelle (2004, Franklin, Beedle & Associates Inc.)

[5] *"Fluent Python: Clear, Concise, and Effective Programming"* – Luciano Ramalho (2015, O'Reilly Media)

[6] *"Python Cookbook: Recipes for Mastering Python 3"* – David Beazley, Brian K. Jones (2013, O'Reilly Media)

[7] Carl Marnewick and Lessing Labuschagne (2005), *"A conceptual model for enterprise resource planning (ERP), Information Management & Computer Security"*, vol.13 No. 2

[8] Carlos Ferran, Ricardo Salim, *"Enterprise resource planning for Global Economiies"*, Informatin science reference, IGI Global, 2008

[9] James A O'Brien, George M.Marakas(2011*), "Management Information System"*

[10] "Automate the Boring Stuff with Python: Practical Programming for Total Beginners" – Al Sweigart (2015, No Starch Press)

[11] "Head First Python: A Brain-Friendly Guide" – Paul Barry (2016, O'Reilly Media)

ଓ HẾT ∞