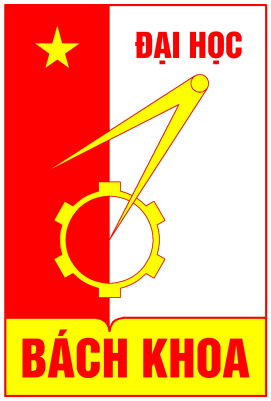
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
_____*******_____



BÁO CÁO
MÔN HỌC
PROJECT I

ĐỀ TÀI :Tính toán số nguyên lớn
Tạo một thư viên tính toán số nguyên lớn trên một modular

Sinh viên thực hiện:

Họ và Tên	MSSV	Mã Lớp
Đặng Quang Trung	20134145	83132

Giáo viên hướng dẫn: TS Trần Vĩnh Đức

Hà Nội 11–2015

Mục lục

Lời mở đầu

Khoa học máy tính là ngành nghiên cứu các cơ sở lý thuyết về thông tin và tính toán cùng sự thực hiện và ứng dụng của chúng trong các hệ thống máy tính. Khoa học máy tính gồm nhiều ngành hẹp; một số ngành tập trung vào các ứng dụng thực tiễn cụ thể chẳng hạn như đồ họa máy tính, trong khi một số ngành khác lại tập trung nghiên cứu đến tính chất cơ bản của các bài toán tính toán như lý thuyết độ phức tạp tính toán). Ngoài ra còn có những ngành khác nghiên cứu các vấn đề trong việc thực thi các phương pháp tính toán. Ví dụ, ngành lý thuyết ngôn ngữ lập trình nghiên cứu những phương thức mô tả cách tính toán khác nhau, trong khi ngành lập trình nghiên cứu cách sử dụng các ngôn ngữ lập trình và các hệ thống phức tạp, và ngành tương tác người-máy tập trung vào những thách thức trong việc làm cho máy tính và công việc tính toán hữu ích, và dễ sử dụng đối với mọi người dùng.

Môn học Prọejct I là cung cấp cho sinh viên rất nhiều kiên thức và khả năng tự học hỏi,tìm hiểu,tìm kiếm tài liệu để giải quyết các vấn đề của mình....

Hiện nay, tính toán số nguyên lớn có vai trò quan trọng trong thực tiễn đời sống hàng ngày nhất là trong bảo mật thông tin vì vậy em lựa chọn đề tài này để tìm hiểu. Đây là một đề tài hay.

Lời cảm ơn

Nhờ có sự tận tình hướng dẫn và chỉ bảo tận tình của thầy giáo bộ môn: TS Trần Vĩnh Đức,em đã hoàn thành đề tài này. Mặc dù đã rất cố gắng để hoàn thành tốt nhưng trong quá trình làm việc, sai sót là điều không thể tránh khỏi. Em mong nhận được nhận xét và phê bình của quý thầy cô để bài làm được hoàn thiện hơn đồng thời bổ sung những kiến thức còn thiếu sót cho em.

Em xin chân thành cảm ơn!

Chương 1

Mở Đầu

1.1 Giới thiệu thư viện GMP

GMP là một thư viện số học chính xác miễn phí. Nó có thể tính toán trên số nguyên,hợp số và số thực.Không có giới hạn thực tế chính xác, ngoại trừ những hàm ý bởi có sẵn bộ nhớ trong máy GMP chạy trên (giới hạn chiều toán hạng là $2^{32} - 1$ bit trên các máy 32-bit và 2^{37} bit trên máy 64-bit)

GMP có một tập hợp phong phú các chức năng, và các chức năng có thường xuyên giao diện. Giao diện cơ bản là cho C nhưng wrappers tồn tại cho các ngôn ngữ khác như Ada, C++, C#, OCaml, Perl, PHP, Python và R. Trong quá khứ, các Kaffe Java máy ảo được sử dụng để hỗ trợ Java GMP được xây dựng trong số học chính xác tùy ý. Tính năng này đã được gỡ bỏ từ bản phát hành gần đây, gây ra các cuộc biểu tình từ những người cho rằng họ đã sử dụng Kaffe chỉ duy nhất cho lợi ích tốc độ dành bởi GMP. Như một kết quả, hỗ trợ GMP đã được thêm vào GNU Classpath.

Các ứng dụng mục tiêu chính của GMP là mật mã ứng dụng và nghiên cứu, ứng dụng bảo mật Internet, và các hệ thống đại số máy tính.

GMP nhằm mục đích là nhanh hơn so với bất kỳ khác bignum thư viện cho tất cả các kích cỡ toán hạng. Một số yếu tố quan trọng trong việc này là:

- Sử dụng đầy đủ các từ như là kiểu số học cơ bản.
- Sử dụng khác nhau các thuật toán khác nhau cho toán hạng kích cỡ; các thuật toán được nhanh hơn cho số lượng rất lớn thường chậm hơn với số lượng nhỏ.
- Cao tối ưu hóa ngôn ngữ lắp ráp mã cho là quan trọng nhất vòng bên trong, chuyên ngành khác nhau cho bộ vi xử lý.

1.1.1 Cài đặt

Hướng dẫn cài đặt với ở link:**https://gmplib.org/manual/Installing-GMP.html**

1.2 Tổng quan về số học modular

Trong toán học, số học modular là một hệ thống số học cho số nguyên, nơi mà con số "quấn quanh" khi đạt đến một giá trị nhất định, các mô đun. Các phương pháp tiếp cận hiện đại để số học modular được phát triển bởi Carl Friedrich Gauss trong cuốn sách của ông Disquisitiones Arithmeticae, xuất bản năm 1801.

Một sử dụng quen thuộc của số học modular là trong đồng hồ 12 giờ, trong đó số ngày được chia thành hai giai đoạn 12 giờ. Nếu thời gian là 7:00 giờ, sau đó 8 giờ sau đó nó sẽ được 03:00. Ngoài ra thông thường sẽ cho thấy rằng thời gian sau sẽ có $7 + 8 = 15$, nhưng đây không phải là câu trả lời vì thời gian đồng hồ "kết thúc tốt đẹp xung quanh" mỗi 12 giờ; trong thời gian 12 giờ, không có "15 giờ". Tương tự như vậy, nếu đồng hồ bắt đầu vào lúc 12:00 (trưa) và 21 giờ trôi qua, rồi thời gian sẽ là 09:00 ngày hôm sau, chứ không phải là 33:00. Vì số lượng giờ bắt đầu lại sau khi nó đạt đến 12, đây là số học modulo 12. Theo định nghĩa dưới đây, 12 là đồng dạng không chỉ đến 12 bản thân, mà còn để 0, vì vậy thời gian gọi là "12:00" cũng có thể được gọi là "00:00", vì 12 là đồng dư với 0 modular 12.

Ở đây:

- N là môt số nguyên dương
- p là số nguyên tố

Kí hiệu: $Z_N = \{0, 1, 2, 3, \ldots, N-1\}$
Chúng ta có thể cộng,trừ,nhân và chia trên modular N.

Ví dụ: Cho N = 12

$9 + 8 = 5$ in Z_{12}

$5 * 7 = 11$ in Z_{12}

$5 - 7 = 7$ in Z_{12}

Chương 2

Tạo thư viện tính toán số học

2.1 Tổng quan

Thư viện được viết bằng C và C++.Có sử dụng một số các thư viện và các cấu trúc dữ liệu có sẵn.
Định nghĩa kiểu dữ liệu mới là ZZ như sau:

```
typedef struct ZZ {
        long long number = 0;
};
```

Tạo một đối tượng số nguyên lớn BigInt:

- Chứa kiểu dữ liệu là vector<ZZ> dùng để biểu diễn số nguyên lớn.Số nguyên lớn được biểu diễn từ trái qua phải theo thứ tự hàng đơn vị,hàng chục ,hàng trăm,.... Để tiện cho việc cài đặt tính toán. Hệ số được chọn ở đây là 10^9.
- Chứa kiểu char(sign) dùng để xác định là số âm hay dương, nếu là âm thì có sign = '-' còn dương thì sign = '0'.
- Lớp đối tượng kiểu BigInt chứa các phương thức tính toán như: cộng(add),trừ(sub),nhân(mul),chia(div),chia dư(mod),UCLN(gcd_extend) ,....

```
class BigInt {
        private:
                vector<ZZ> zz;
                char sign;
        public:
        // method
}
```

2.2 Cài đặt các phép toán trong thư viện

2.2.1 Phép Cộng

Thuật toán được cài đặt dựa trên ý tưởng tính toán trên giấy tờ được dạy ở các cấp học dưới.Giả sử chúng ta thực hiện phép toán trên hê cơ số thập

phân(10):

- Phép cộng được thực hiện từ trái qua phải
- Kiểm tra mỗi lần cộng nếu >= hê cơ số(10) thì lấy phần dư và nhớ 1 để cộng vào phân sau
- Thời gian chạy và độ phức tạp: $O(n)$

Mã giả:
function add
Input 2 số kiểu bigInt a,b.
Output 1 số kiểu bigInt.

```
1  i = 0;
2  carry = 0;
3  BigInt e;
4
5  while i < a.size && i < b.size
6         s = (a[i] + b[i] + carry)mod 10
7         carry = s div 10
8         e[i] = s;
9         i = i+1
10 while i < a.size
11        s = (a[i] + 0 + carry)mod 10
12        carry = s div 10
13        e[i] = s;
14        i = i+1
15 while i < b.size
16        s = (b[i] + 0 + carry)mod 10
17        carry = s div 10
18        e[i] = s;
19        i = i+1
20 if carry then e[i] = carry;
21
22 return e
```

2.2.2 Phép trừ

Thuật toán được cài đặt dựa trên ý tưởng tính toán trên giấy tờ được dạy ở các cấp học dưới.Giả sử chúng ta thực hiện phép toán trên hê cơ số thập phân(10):

- Phép trừ được thực hiện từ trái qua phải.
- Mỗi lần trừ ta cộng thêm hệ cơ số(10) và trừ đi phần.
- Lấy phần vừa tính chia dư cho hê số(10) lấy phần dư là kết quả và lấy 1 - phần dư để lấy phần nhớ.
- Thời gian chạy và độ phức tạp O(n)

Mã giả:
function sub
Input 2 số kiểu bigInt a,b.(a > b)
Output 1 số kiểu bigInt.

```
1  i = 0
2  carry = 0
3  BigInt e
4
5  while i < a.size && i < b.size
6          s = (10 + a[i] - b[i] -carry) mod 10
7          carry = 1 - s
8          e[i] = s
9          i = i+1
10 while i < a.size // if a.size > b.size
11         s = (10 + a[i] - 0 - carry) mod 10
12         carry = 1 - s
13         e[i] = s
14         i = i + 1
15 while e[--i] == 0 && i > 0 // delete 0 in top of e
16         e[i] = null
17 return e
```

2.2.3 Phép nhân

Sử dụng thuật toán Karatsuba. Bài toán đặt ra:

$$x = x_{n-1}x_{n-2}\dots x_1x_0$$
$$y = y_{n-1}y_{n-2}\dots y_1y_0$$

là 2 số nguyên không âm có n chư số thâp phân.Cần tính

$$z = z_{2n-1}y_{2n-2}\dots y_1y_0$$

Biểu diễn với 2n chữ số thập phân của xy. Ta có:

$$x = x_{n-1} * 10^{n-1} + x_{n-2} * 10^{n-2} + \dots + x_1 * 10^1 + x_0 * 10^0$$
$$y = y_{n-1} * 10^{n-1} + y_{n-2} * 10^{n-2} + \dots + y_1 * 10^1 + y_0 * 10^0$$

Vì thế

$$\begin{aligned} z &= z_{2n-1} * 10^{2n-1} + z_{2n-2} * 10^{2n-2} + \dots + z_1 * 10^1 + z_0 * 10^0 \\ &= (x_{n-1} * 10^{n-1} + x_{n-2} * 10^{n-2} + \dots + x_1 * 10^1 + x_0 * 10^0) \\ &\quad *(y_{n-1} * 10^{n-1} + y_{n-2} * 10^{n-2} + \dots + y_1 * 10^1 + y_0 * 10^0) \end{aligned}$$

Đặt

$$a = x_{n-1}x_{n-2} + \dots + x_{n/2+1} + x_{n/2}$$
$$b = x_{n/2-1}x_{n/2-2} + \dots + x_1 + x_0$$
$$c = y_{n-1}y_{n-2} + \dots + y_{n/2+1} + y_{n/2}$$
$$d = y_{n/2-1}x_{n/2-2} + \dots + x_1 + x_0$$

Karatsuba đã phát hiện cách thực hiên việc nhân 2 số nguyên có n chữ số đòi hỏi 3 phép nhân các số có n/2 chữ số sau đây:
Đặt: U = a*c, V =b*d, W =(a+b)*(c+d)
Khi đó ta có:

$$Z = x * y = U * 10^n + (W - U - V) * 10^{n/2} + V$$

Thời gian chạy và độ phức tạp : $O(n \log n)$ Mã giả:

```
function Karatsuba(x,y,n)
begin
        if(n =1) then return x[0]*y[0];
        else
        begin
                a = x[n-1]...x[n/2];
                b = x[n/2-1]...x[0];
                c = y[n-1]...y[n/2];
                d = y[n/2-1]...y[0];
                U = Karatsuba(a,c,n/2);
                V = Karatsuba(b,d,n/2);
                W = Karatsuba(a+b,c+d,n/2);
                return U*10^n + (W-U-V)*10^(n/2) + V;
        end
end
```

2.2.4 Phép chia

Input Cho 2 số nguyên không âm A và B,sao cho $A < \beta^n$ và $\beta^n/2 \leq B < \beta$. n phải là 1 số chẵn.
Output Thương $\lfloor A/B \rfloor$ và phần dư A mod B

1. $A = A_3\beta^{3n/2}+A_2\beta^n+A_1\beta^{n/2}+A_0$ và $B = B_1\beta^{n/2}+B_0$,với $0 \leq A_i < \beta^{n/2}$, $0 \leq B_i < \beta^{n/2}$.

2. Tính nửa cao Q_1 của thương $Q_1 = \frac{A_3\beta^n+A_2\beta^{n/2}+A_1}{B}$ với nhớ R_1

3. Tính nửa thấp Q_0 của thương $Q_0 = \frac{R_1\beta^{n/2}+A_4}{B}$ với nhớ R_0.

4. Return thương $Q = Q_1\beta^{n/2} + Q_0$ và nhớ R = R_0.

2.2.5 Tìm Ước chung lớn nhất

Thuật toán tìm ước chung lớn nhất ở đây được cài đặt là thuật toán Euclidean mở rộng.
Thời gian chạy và độ phức tạp: $O(n)$
Ta có đoạn mã giả sau:

```
function extended_gcd(a,b)
        s = 0;          old_s = 1;
        t = 0;          old_t = 0;
        r = b;          old_r = a;
        while r != 0
                quotient = old_r div r
                (old_r, r) = (r, old_r - quotient*r)
                (old_s, s) = (s, old_s - quotient*s)
                (old_t, t) = (t, old_t - quotient*t)
        output "greatest common divisor:", old_r
        output "quotient by the gcd:", (t, s)
```

2.3 Cài đặt các phép toán trên modular

Giả sử cho một modular P(là một số nguyên dương) khi đó ta có kí hiệu: $Z_P = \{0, 1, 2, 3, ..., P-1\}$.
Tính toán số học trên modular P là các kết quả nằm trong modular P hay in $Z_P = \{0, 1, 2, 3, ..., P-1\}$.
Một số phép toán cơ bản hay được sử dụng khi tính toán như cộng, trừ, nhân ,chia ...
Giả sử cho 2 số nguyên lớn a,b và modular P (với $a, b \in Z_P$)khi đó:
Phép cộng: thực hiện phép tính (a+b) mod P.
Phép trừ: thực hiện phép tính (a - b) nếu (a - b) $>$ thì (a - b) mod P ngược lại (a - b + P).
Phép nhân: thực hiện phép tính (a*b) mod P.
Phép chia: thực hiện phép tinhd (a/b) mod P.

2.3.1 Tìm nghịch đảo trên modular

Ý tưởng dựa trên thuật toán Eculidean mở rộng để tìm kiếm nghịch đảo trên modular số học.
Thời gian chạy và độ phức tạp: $O(n)$
Mã giả:

```
1  function inverse(a, n)
2      t := 0;     newt := 1;
3      r := n;     newr := a;
4      while newr != 0
5          quotient := r div newr
6          (t, newt) := (newt, t - quotient * newt)
7          (r, newr) := (newr, r - quotient * newr)
8      if r > 1 then return "a is not invertible"
9      if t < 0 then t := t + n
10     return t
```

2.3.2 Phép tính lấy mũ nhanh

Ý tưởng sử dụng neo đệ quy để tính.Ta có:

$$x^n = \begin{cases} x^{n/2} * x^{n/2} & \text{nếu n là chẵn} \\ x^{(n-1)/2} * x^{(n-1)} * x & \text{nếu n là lẻ} \end{cases}$$

Thời gian chạy và độ phức tạp: $O(\log n)$
Mã giả:

```
function modExp(base,exp,n)
        if(exp = 0) return 1;
        if(exp = 1) return base mod n;
        if(exp % 2 = 0)
                ans = modExp(base,exp/2,n);
                return (ans*ans) mod n;
        else
                ans = modExp(base,(exp-1)/2,n);
                return (ans*ans*base) mod n;
```

2.3.3 Tài liệu tham khảo

- Sile bài giảng phân tích thiết kế thuật toán by Nguyễn Đức Nghĩa
- Bài giảng Toán Chuyên đề by Trần Vĩnh Đức
- INTRODUCTION TO ALGORITHMS by T H O M A S H.CORMEN ,C H A R L E S E.LEISERSON ,R O N A L D L .RIVEST ,CLIFFORD STEIN.
- www.algorithmist.com
- https://asdfcoding.wordpress.com/
- https://www.wikipedia.org/