ĐẠI HỌC THÁI NGUYÊN
TRƯỜNG ĐẠI HỌC KHOA HỌC
****

VŨ THỊ THANH HẬU

MỘT SỐ ỨNG DỤNG CỦA SỐ HỌC TRONG LÝ THUYẾT MẬT MÃ

LUẬN VĂN THẠC SĨ TOÁN HỌC

Thái Nguyên, năm 2009

ĐẠI HỌC THÁI NGUYÊN
TRƯỜNG ĐẠI HỌC KHOA HỌC
****

VŨ THỊ THANH HẬU

MỘT SỐ ỨNG DỤNG CỦA SỐ HỌC TRONG LÝ THUYẾT MẬT MÃ

Chuyên ngành : Phương pháp toán sơ cấp
Mã số **: 60.46.40**

LUẬN VĂN THẠC SĨ KHOA HỌC TOÁN HỌC

Người hướng dẫn khoa học : GS.TSKH. Hà Huy Khoái

Thái Nguyên, năm 2009

Mục lục

Lời nói đầu 2

1 Một số kiến thức cơ bản **5**
1.1 Thuật toán và độ phức tạp của thuật toán 5
1.1.1 Khái niệm: 5
1.1.2 Độ phức tạp của thuật toán 7
1.2 Phép tính đồng dư và các vấn đề liên quan 10
1.2.1 Số nguyên tố và định lý cơ bản của số học 10
1.2.2 Thuật toán Euclid và mở rộng 11
1.2.3 Phi - hàm Euler 12
1.2.4 Phép tính đồng dư và phương trình đồng dư 13
1.2.5 Định lý Fermat và các mở rộng 14
1.2.6 Tính toán với đồng dư của luỹ thừa bậc lớn 15
1.2.7 Thặng dư bình phương và ký hiệu Legendre 16
1.3 Phân số liên tục 17
1.3.1 Khái niệm. 17
1.3.2 Tính chất 18

2 Một số ứng dụng của số học trong lý thuyết mật mã **21**
2.1 Nguyên tắc chung và một số hệ mã đơn giản 21
2.1.1 Hệ mã Ceasar 21
2.1.2 Hệ Mã Khối 24
2.2 Một số hệ mã mũ thông dụng 26
2.2.1 Hệ mã mũ của Pohligvà Hellman 26
2.2.2 Giao thức trao đổi chìa khoá của Diffie - Hellman 29
2.2.3 Hệ mã ElGamal 30
2.2.4 Hệ mã RSA 33
2.3 Phân tích ra thừa số nguyên tố 35
2.3.1 Phân tích Fermat và mở rộng của nó 35
2.3.2 Phân tích sử dụng liên phân số. 40
2.3.3 Phân tích dùng phương pháp của Pollard 43
Tài liệu tham khảo 46

LỜI CẢM ƠN

Luận văn này được hoàn thành dưới sự hướng dẫn tận tình và nghiêm khắc của GS.TSKH. Hà Huy Khoái. Nhân dịp này, tôi xin bày tỏ lòng kính trọng và biết ơn sâu sắc tới GS.TSKH. Hà Huy Khoái, người Thầy mẫu mực đã giành nhiều thời gian và công sức để hướng dẫn tôi hoàn thành luận văn này.

Tôi xin chân thành cảm ơn Trung tâm Đào tạo sau đại học, Phòng Đại số, Trường Đại học Khoa học - Đại học Thái Nguyên, Sở Giáo dục và Đào tạo tỉnh Quảng Ninh, Trung tâm Hướng nghiệp và Giáo dục thường xuyên tỉnh Quảng Ninh, đã tạo điều kiện thuận lợi để tôi hoàn thành luận văn này.

Nhân dịp này, tôi xin bày tỏ lòng biết ơn tới các Giáo sư, Phó giáo sư, Tiến sĩ của Viện Toán học và Trường Đại học Khoa học - Đại học Thái Nguyên, những người thầy đã tận tình giảng dạy và tạo mọi điều kiện thuận lợi cho tôi hoàn thành khoá học. Tôi cũng xin cảm ơn bạn bè và gia đình đã động viên và giúp đỡ tôi trong suốt quá trình học tập tại Trường Đại học Khoa học - Đại học Thái Nguyên.

LỜI NÓI ĐẦU

Trước những năm 70 của thế kỷ XX, Số học thường được xem là một trong những ngành toán học thuần tuý, chỉ có ý nghĩa lý thuyết. Đối tượng nghiên cứu của Số học là các quy luật trong tập hợp số, giả thuyết lớn tồn tại trong Số học là các giả thuyết số nguyên tố. Thậm chí, có những nhà toán học cho rằng vẻ đẹp của Số học có được nhờ sự xa rời thực tiễn của nó! (theo câu nói của G.H. Hardy, A Mathematician's Apology, 1940).

Ngày nay, thật khó đồng ý với Hardy, khi mà vẻ đẹp của Số học không chỉ thể hiện trong ý nghĩa "thuần tuý" của nó, mà cả trong những ứng dụng bất ngờ vào thực tiễn. Cách đây khoảng 30 năm, khó có thể hình dung được rằng, một số kết quả lý thuyết số trong Số học lại làm nên một cuộc cách mạng trong bảo mật thông tin. Cơ sở của những ứng dụng đó lại chính là Số học thuật toán, lĩnh vực nghiên cứu các thuật toán trong Số học.

Có thể nói, mật mã đã có từ thời cổ đại. Người đầu tiên áp dụng mật mã một cách có hệ thống để đảm bảo bí mật thông tin quân sự là nhà quân sự thiên tài của người La Mã cổ đại, Julius Caesar. Hệ mã cổ nhất là hệ mã Ceasar, thông qua phép mã thay thế mỗi ký tự thay bởi ký tự đứng ngay sau nó 3 vị trí (hoặc k vị trí).

Vào những năm đầu thế kỷ XX hệ mã mới có tính bảo mật cao hơn được xuất hiện với sự ra đời của hệ mã British Playfair năm 1910, đó là mã khối thông qua phép *thay thế* theo *chìa*. Song song với quá trình phát triển của lịch sử và nhu cầu về bảo mật thông tin trên nhiều lĩnh vực đã thúc đẩy các hệ mã mới ra đời có tính bảo mật ngày càng cao, như hệ mã mũ của Pohlig và Hellman (năm 1978), tiếp theo là giao thức trao đổi chìa khoá của Diffie-Hellman, sau nữa là hệ mã ElGamal. Một nét chung của hai hệ mã trên là

cho phép công bố công khai một phần thông tin cho việc lập mã gọi là *mã hoá với khoá công khai*, một mô hình hoàn hảo cho hệ mã kiểu này được công bố bởi Rivest, Shamir và Adleman vào năm 1978, mang tên RSA. Hệ mã RSA vẫn đang là một thách thức lớn đối với những nhà thám mã.

Mục đích của bản luận văn này nhằm trình bày cơ sở của việc áp dụng lý thuyết số vào mật mã, đặc biệt là mã hoá RSA và một số thuật toán phân tích số nguyên đang sử dụng trong thám mã. Luận văn gồm hai chương. Chương I trình bày các kiến thức chuẩn bị phục vụ cho chương sau như các khái niệm về thuật toán, độ phức tạp của thuật toán, các kiến thức về đồng dư và phân số liên tục. Chương II trình bày một số hệ mã đơn giản, hệ mã thông dụng, hệ mã RSA và ứng dụng của số học vào mật mã khoá công khai như phân tích Fecmat, phân tích Fecmat mở rộng, phân tích sử dụng phân số liên tục, phân tích dùng phương pháp của Pollard.

Từ đó viết một số thủ tục phân tích số, thủ tục lập mã và giải mã chạy trên Maple.

Chương 1

Một số kiến thức cơ bản

Trong chương này chúng tôi trình bày một số kiến thức chuẩn bị. Tiết 1.1 nhắc lại các khái niệm về thuật toán và độ phức tạp của thuật toán. Đồng thời để tiện theo dõi, chúng tôi trình bày trong tiết 1.2 một số kiến thức về phép tính đồng dư và các vấn đề liên quan, trong tiết 1.3 là một số kiến thức về phân số liên tục.

1.1 Thuật toán và độ phức tạp của thuật toán

1.1.1 Khái niệm:

Có thể định nghĩa thuật toán theo nhiều cách khác nhau. Trong luận văn này, chúng ta có thể hiểu khái niệm thuật toán theo cách thông thường nhất.
Thuật toán là một qui tắc để với những dữ kiện ban đầu đã cho, tìm được lời giải của bài toán được xét sau một khoảng thời gian hữu hạn.

Để minh họa cách ghi một thuật toán, cũng như tìm hiểu các yêu cầu đặt ra cho thuật toán, ta xét một ví dụ cụ thể sau: Cho n số tự nhiên $X[1], X[2], ..., X[n]$. Tìm m và j sao cho j là số lớn nhất thoả mãn:

$$m = X[j] = \max_{1 \le k \le n} X[k].$$

Bài toán này cũng có nghĩa là tìm cực đại của các số đã cho và tìm chỉ số lớn nhất trong các số đạt cực đại. Vì cần tìm chỉ số lớn nhất trong các số

đạt cực đại, ta xuất phát từ giá trị $X[n]$. Trong bước thứ nhất ta tạm thời xem $m = X[n]$ và $j = n$. Tiếp theo ta so sánh $X[n]$ và $X[n-1]$. Trong trường hợp $n-1 = 0$, tức $n = 1$, thuật toán kết thúc. Nếu $X[n-1] \leq X[n]$, ta chuyển sang so sánh $X[n]$ với $X[n-2]$. Trong trường hợp ngược lại, $X[n-1]$ chính là số cực đại trong hai số đã xét (hai số $X[n]$ và $X[n-1]$). Khi đó ta phải thay đổi $m = X[n-1]$ và $j = n-1$. Với cách làm như trên ta luôn nhận được số cực đại trong những số đã xét, và cũng nhận được chỉ số lớn nhất j trong các chỉ số của các số đạt cực đại đó. Bước tiếp theo đó là so sánh nó với số đứng ngay trước những số đã xét, hoặc kết thúc thuật toán trong trường hợp không còn số nào đứng trước nó.

Bây giờ ta có thể ghi lại thuật toán trên như sau :

Thuật toán tìm cực đại.

M1. (Bước xuất phát). Đặt $j \longleftarrow n, k \longleftarrow n-1, m \longleftarrow X[n]$.

M2. (Đã kiểm tra xong?). Nếu $k = 0$, thuật toán kết thúc.

M3. (So sánh). Nếu $X[k] \leq m$, chuyển sang M5.

M4. (Thay đổi m). Đặt $j \longleftarrow k, m \longleftarrow X[k]$ (ta hiểu m tạm thời đang là cực đại).

M5. (Giảm k). Đặt $k \longleftarrow k-1$, quay về M2.

Trong bảng ghi trên đây, dấu $\longleftarrow$ dùng để chỉ một phép toán quan trọng, đó là phép *thay chỗ*. Trên đây ta đã ghi một thuật toán bằng ngôn ngữ thông thường. Trong trường hợp thuật toán được viết bằng ngôn ngữ làm việc của máy tính, ta có một *chương trình*.

Trong thuật toán, có những số liệu ban đầu được cho trước khi thuật toán bắt đầu làm việc. Ta gọi chúng là *đầu vào (input)*. Chẳng hạn, trong thuật toán tìm cực đại trên, đầu vào chính là các số $X[1], X[2], ..., X[n]$.

Một thuật toán có thể có nhiều *đầu ra (output)*. Trong thuật toán tìm cực đại trên, đầu ra là các số m và j.

Có thể thấy rằng thuật toán tìm cực đại mô tả trên thoả mãn những yêu cầu của một thuật toán nói chung, đó là *tính hữu hạn* và *tính chính xác*.

Tính hữu hạn. Thuật toán cần phải kết thúc sau một số hữu hạn bước. Khi thuật toán ngừng làm việc, ta phải thu được câu trả lời cho vấn đề đặt ra. Thuật toán tìm cực đại mô tả trên đây rõ ràng thoả mãn điều kiện này vì ở mỗi bước ta chuyển được từ việc xét một số sang số đứng trước nó, và số các số cần xét là hữu hạn.

Tính xác định.Ơ mỗi bước, thuật toán cần phải xác định, nghĩa là chỉ rõ việc cần làm. Thuật toán tìm cực đại ở trên chỉ ra rõ ràng những việc cần làm của mỗi bước.

Ngoài những yếu tố kể trên, ta còn phải xét đến tính hiệu quả của thuật toán. Có rất nhiều thuật toán về mặt lý thuyết là kết thúc sau một số hữu hạn bước, tuy nhiên thời gian làm việc đó lại vượt quá khả năng làm việc của chúng ta. Vì thế, ta còn phải chú ý đến cái gọi là *độ phức tạp của thuật toán*. Độ phức tạp của thuật toán có thể đo bằng *không gian*, tức là *dung lượng bộ nhớ* của máy tính cần thiết để thực hiện thuật toán, và đo bằng thời gian, tức là *thời gian máy tính làm việc*. Trong luận văn này, khi nói đến độ phức tạp của thuật toán, ta luôn luôn hiểu là độ phức tạp thời gian.

1.1.2 Độ phức tạp của thuật toán

Dĩ nhiên, thời gian làm việc của máy tính khi chạy một thuật toán nào đó không chỉ phụ thuộc vào thuật toán, mà còn phụ thuộc vào máy tính sử dụng để chạy thuật toán đó. Vì thế, để có một tiêu chuẩn chung, ta sẽ đo độ phức tạp của một thuật toán bằng số các phép tính phải làm khi thực hiện thuật toán. Khi tiến hành cùng một thuật toán, số các phép tính phải làm khi thực hiện còn phụ thuộc vào cỡ của bài toán, tức là phụ thuộc vào độ lớn của đầu vào. Vì thế độ phức tạp của thuật toán sẽ là một hàm số của độ lớn của đầu vào. Trong những ứng dụng thực tiễn, chúng ta không cần biết chính xác hàm này, mà chỉ cần biết *cỡ* của chúng, tức là cần có một ước lượng đủ tốt của chúng.

Khi làm việc, máy tính thường ghi các chữ số bằng những bóng đèn *sáng, tắt*: bóng đèn sáng chỉ số 1, bóng đèn tắt chỉ số 0. Vì thế thuận tiện nhất là dùng hệ đếm cơ số 2, trong đó để biểu diễn một số, ta chỉ cần dùng hai kí hiệu 0 và 1. Một kí hiệu 0 và 1 được gọi là *một bít*. Một số nguyên n biểu diễn bởi k chữ số 1 và chữ số 0 được gọi là số k-bít. Trong mục này và các mục tiếp theo ta sẽ thấy rằng số tự nhiên n sẽ là một số k-bít với $k = [log_2 n]$ (dấu [ ] là kí hiệu phần nguyên của một số).

Độ phức tạp của thuật toán được đo bằng số các phép tính bít.

Phép tính bít là một phép tính lôgic hay số học thực hiện trên các số một bít 0 và 1

Để ước lượng độ phức tạp của thuật toán ta dùng khái niệm bậc O-lớn.

Định nghĩa 1:

Giả sử $f(n)$ và $g(n)$ là hai hàm xác định trên tập hợp các số nguyên dương. Ta nói $f(n)$ có bậc O lớn của $g(n)$ và viết $f(n) = O(g(n))$ hoặc $f = O(g)$, nếu tồn tại một số $C > 0$, sao cho n đủ lớn, các hàm $f(n)$ và $g(n)$ đều dương, đồng thời $f(n) < C.g(n)$.

Ví dụ:

Cho $f(n) = a_i n^i + a_{i-1} n^{i-1} + \ldots + a_1 n + a_0$, trong đó $a_i > 0$. Khi đó $f(n) = O(n^i)$. Chúng ta có thể kiểm tra được rằng:

Nếu $f_1(n) = O(g(n)), f_2(n) = O(g(n))$ thì $f_1(n) + f_2(n) = O(g(n))$;

và nếu $f_1(n) = O(g_1(n)), f_2(n) = O(g_1(n))$ thì $(f_1 f_2)(n) = O(g_1 g_1(n))$.

Hơn nữa nếu tồn tại giới hạn hữu hạn

$$\lim_{n\to\infty} \frac{f(n)}{g(n)}$$

thì f(n) = O(g(n))

Định nghĩa 2:

Một thuật toán được gọi là có độ phức tạp đa thức hoặc có thời gian đa thức, nếu số các phép tính cần thiết khi thực hiện thuật toán không vượt quá $O(log^d n)$, trong đó, n là độ lớn của đầu vào và d là số nguyên dương nào