Số học 1 - Modulo & gcd

Số học 1 - Modulo & gcd

Nguồn: HackerEarth và 1 số bài viết trên Wikipedia

Người dịch: Bùi Việt Dũng

Giới thiệu

Các bài toán trong **lập trình thi đấu (competitive programming)** mà liên quan đến Toán học thường sẽ rơi vào hai mảng là **số học (number theory)** và hình học. Nếu bạn biết nhiều về số học, bạn sẽ có khả năng giải quyết nhiều bài toán khó và một nền tảng tốt để giải quyết nhiều bài toán khác.

Các bài toán trong lập trình thi đấu thường đòi hỏi bạn một cái nhìn sâu sắc, vì vậy chỉ biết một số vấn đề về số học là không đủ. Mọi bài toán đều đều yêu cầu bạn phải biết một lượng kiến thức toán nhất định. Ví dụ, một số bài toán yêu cầu bạn phải giải một hệ nhiều phương trình hay tính xấp xỉ nghiệm của nhiều phương trình khác nhau.

Đồng dư thức (Modulo)

Phép đồng dư thức cho bạn số dư của phép chia số này cho số khác. Dấu của phép đồng dư là $\%$.

Ví dụ:

Ta có hai số 5 và 2, khi đó $5\%2$ bằng 1 do khi chia 5 cho 2, ta được số dư là 1.

Tính chất:
Đồng dư thức có một số tính chất sau:

$$(a+b)\%c = (a\%c + b\%c)\%c$$

$$(a.b)\%c = ((a\%c).(b\%c))\%c$$

Ví dụ:

Giả sử $a = 5, b = 3, c = 2$

Khi đó:

- $(5+3)\%2 = 8\%2 = 0$

và cũng bằng $(5\%2 + 3\%2)\%2 = (1+1)\%2 = 0$.

- $(5.3)\%2 = 15\%2 = 1$

và cũng bằng $((5\%2).(3\%2))\%2 = (1.1)\%2 = 1$.

Ước chung lớn nhất

Ước chung lớn nhất (GCD, viết tắt của từ Greatest Common Divisor) của hai hay nhiều số là số nguyên dương lớn nhất mà là **ước chung (common divisor)** của tất cả các số đó.

Ví dụ: GCD của 6 và 10 là 2 vì 2 là số nguyên dương lớn nhất mà là ước chung của 6 và 10.

Thuật toán "ngây thơ" (Naive Approach)

Ta có thể duyệt tất cả các số từ $min(A, B)$ đến 1 và kiểm tra xem số đang xét có phải là ước của của A và B hay không. Nếu đúng như vậy thì số đang xét sẽ là GCD của A và B.

```
1  int gcd(int A, int B) {
2      for (int i = min(A, B); i > 0; --i)
3          if (A % i == 0 && B % i == 0) {
4              return i;
5          }
6      // không bao giờ chạy đến đây vì khi i = 1 thì cả A và B luôn chia hết cho
7  }
```

Độ phức tạp của thuật toán: $O(min(A, B))$.

Thuật toán Euclid

Thuật toán Euclid dựa trên tính chất sau của ước chung lớn nhất $GCD(A, B) = GCD(B, A\%B)$. Thuật toán sẽ quy nạp cho đến khi $A\%B = 0$.

```
1  int gcd(int A, int B) {
2      if (B == 0) return A;
3      else return gcd(B, A % B);
4  }
```

Ví dụ:

Giả sử $A = 16, B = 10$.

$GCD(16, 10) = GCD(10, 16\%10) = GCD(10, 6)$

$GCD(10, 6) = GCD(6, 10\%6) = GCD(6, 4)$

$GCD(6,4) = GCD(4, 6\%4) = GCD(4,2)$

$GCD(4,2) = GCD(2, 4\%2) = GCD(2,0)$

Vì $B = 0$ nên $GCD(2,0)$ sẽ trả về giá trị 2.

Độ phức tạp của thuật toán: $O(\log max(A,B))$.

Thuật toán Euclid mở rộng (Extended Euclid Algorithm)

Đây là một thuật toán mở rộng của thuật toán Euclid ở trên. $GCD(A,B)$ có một tính chất rất đặc biệt: Nó luôn có thể được biểu diễn dưới dạng phương trình $Ax + By = GCD(A,B)$.

Thuật toán sẽ cho ta biết một cặp giá trị $(x;y)$ thỏa mãn phương trình này và nhờ đó giúp ta tính Modular Multiplicative Inverse. x và y có thể có giá trị bằng không hoặc âm. Chương trình sau đọc hai số A và B và in ra $GCD(A,B)$ cũng như một cặp số $(x;y)$ thỏa mãn phương trình.

```
int d, x, y;
void extendedEuclid(int A, int B) {
    if (B == 0) {
        d = A;
        x = 1;
        y = 0;
    }
    else {
        extendedEuclid(B, A%B);
        int temp = x;
        x = y;
        y = temp - (A/B)*y;
    }
}

int main() {
    extendedEuclid(16, 10);
    cout << "gcd(16, 10) = " << d << endl;
    cout << "x, y: " << x <<  ", " << y << endl;
    return 0;
}
```

Kết quả

```
gcd(16, 10) = 2
x, y: 2, -3
```

Ban đầu, thuật toán Euclid mở rộng sẽ chạy như thuật toán Euclid cho đến khi ta có $GCD(A,B)$ hoặc cho đến khi B bằng 0 và khi đó thuật toán sẽ đặt $x = 1$ và $y = 0$. Vì $B = 0$ và $GCD(A,B)$ là A trong thời điểm hiện tại nên phương trình $Ax + By = 0$ trở thành $A.1 + 0.0 = A$.

Giá trị của các biến d, x, y trong hàm `extendedEuclid()` sẽ lần lượt trở thành:

1. $d = 2, x = 1, y = 0$.
2. $d = 2, x = 0, y = 1 - (4/2).0 = 1$.
3. $d = 2, x = 1, y = 0 - (6/4).1 = -1$.
4. $d = 2, x = -1, y = 1 - (10/6).(-1) = 2$.
5. $d = 2, x = 2, y = -1 - (16/10).2 = -3$

Độ phức tạp của thuật toán: Độ phức tạp của thuật toán Euclid mở rộng là $O(\log max(A, B))$.