Xử lý số nguyên lớn

Posted by basicalgorithm on December 6, 2009

Xử lý số nguyên lớn là một kỹ năng không thể thiếu của một thí sinh tham gia kỳ thi HSGQG. Bài toán thường liên quan tới việc cộng/trừ/nhân với các số nguyên có nhiều (khoảng vài trăm, vài nghìn) chữ số. Bài viết này xin được cung cấp cho các bạn cách thực hiện các phép toán với số nguyên lớn.

1/ Ý tưởng: Chúng ta sẽ thực hiện các phép toán này như cách làm mà hồi cấp 1 đã được học. Đó là thực hiện các phép toán lần lượt từ phải qua trái và sử dụng thêm một biến "nhớ".

2/ Khai báo: Các phép toán sẽ được thực hiện trên mảng, do đó ta cần xây dựng mảng 1 chiều có kích thước là số chữ số tối đa của bài toán. Mỗi phần tử của mảng sẽ là 1 chữ số. Ngoài ra, cần khai báo biến `base` là hệ cơ số mà chúng ta dùng khi thực hiện các phép toán. Trong ví dụ này, tôi đặt `base = 10` để thỏa mãn mỗi phần tử chỉ chứa 1 chữ số.

```
Const
  maxn = 100; // Số chữ số tối đa
  base = 10; // Hệ cơ số khi sử dụng
Type
  BigNum = array[0..maxn] of LongInt; // Kiểu số nguyên lớn
```

3/ Phép cộng:

```
Function Plus(x , y : BigNum) : BigNum; // Kết quả là số nguyên lớn
var
  i , nho : LongInt;
begin
  Fillchar(Plus , SizeOf(Plus) , 0); // Khởi gán Plus = 0
  nho := 0; // Khởi gán nho = 0
  For i := maxn downto 1 do // Cộng lần lượt từng chữ số từ phải qua trái
    begin
      Plus[i] := x[i] + y[i] + nho; // Công thức cộng như cấp 1
      nho := Plus[i] div base; // Tính lại biến nho cho lần cộng tiếp theo
      Plus[i] := Plus[i] mod base; // Đảm bảo mỗi phần tử chỉ lưu một chữ số
    end;
end;
```

Độ phức tạp: `O(N)`

4/ Phép trừ:

Một điều đáng lưu ý khi thực hiện phép trừ đó là biến "nhớ" chỉ nhận `1` trong `2` giá trị: `0` hoặc `1`. Ngoài ra, để đảm bảo kết quả đúng, bạn cần chú ý khi lấy `X - Y` thì `X >= Y` (có thể viết 1 hàm kiểm tra trước khi thực hiện phép trừ).

```
Function Minus(x , y : BigNum) : BigNum;
var
  i , nho : LongInt;
begin
  Fillchar(Minus , SizeOf(Minus) , 0); // Khởi gán Minus = 0
  nho := 0;
```

```
    For i := maxn downto 1 do
      begin
        Minus[i] := x[i] + base - y[i] - nho; // Cộng thêm base để đảm bảo Minus[i] >= 0
        if x[i] < y[i] + nho then nho := 1 else nho := 0; // Tính lại biến nho
        Minus[i] := Minus[i] mod base; // Đảm bảo mỗi phần tử chỉ chứa 1 chữ số
      end;
  end;
```

Độ phức tạp: `O(N)`

5/ Phép nhân:

Chúng ta vẫn thực hiện cách làm như cấp 1: Lấy từng chữ số của thừa số thứ hai nhân với thừa số thứ nhất, được bao nhiêu cộng vào kết quả. Chú ý sau mỗi lần nhân 1 chữ số của thừa số thứ hai với thừa số thứ nhất, ta cần lùi kết quả đó sang trái 1 chữ số.

Ngoài ra, số chữ số tối đa của kết quả sẽ là *tổng số chữ số của 2 thừa số*. Vì thế, ta cần đảm bảo độ lớn cho kết quả để tránh bị `RangeCheck`. Tốt nhất bạn hãy khai báo `maxn` lớn bằng 2 lần độ dài tối đa của 2 thừa số.

```
Function Multi(x , y : BigNum) : BigNum;
var
  i , j , nho : LongInt;
  Temp : BigNum;
begin
  Fillchar(Multi , SizeOf(Multi) , 0); // Khởi gán Multi = 0
  nho := 0;
  count := -1; // Số chữ số phải lùi vào trước khi nhân là -1
  For i := maxn downto 1 do
    begin
      Inc(count); // Tăng số lượng chữ số cần lùi vào ở lượt nhân thứ i
      Fillchar(Temp , SizeOf(Temp) , 0); // Mảng nhân chữ số thứ i của y với x
      For j := maxn downto (maxn div 2 + 2) do
        begin
          Temp[j-count] := y[i]*x[j] + nho;
          nho := Temp[j-count] div base;
          Temp[j-count] := Temp[j-count] mod base;
        end;
      Minus := Plus(Minus , Temp);
    end;
end;
```

Độ phức tạp: `O(N^2)`

6/ Cải tiến:

Khi thực hiện phép cộng/trừ, ta nhận thấy mỗi phần tử của kết quả thuộc kiểu `LongInt`, có thể lưu đc 9 chữ số. Do đó, ta có thể đặt `base = 10^9` nhằm giảm đi số lượng phép tính. Số lượng phần tử tối đa của mảng `BigNum` cũng giảm đi ~9 lần.

Tuy nhiên, khi in ra, do mỗi phần tử lưu 9 chữ số nên ta cần in ra đủ cả 9 chữ số này (tính cả các chữ số 0 vô nghĩa ở đầu).

```
Procedure Print;
var
```

```
        i , j : LongInt;
        s : string;
      begin
        For  i := 1 to maxn do
          if Res[i] <> 0 then break; // Tìm phần tử khác 0 đầu tiên của kết quả
        write(Res[i]);
        For j := i + 1 to maxn do
          begin
            str(Res[j] , s); // Chuyển Res[j] sang xâu s
            while length(s) < 9 do s := '0' + s; // Thêm các chữ số 0 vào đầu để đảm bảo đủ 9 chữ số
            write(s);
          end;
      end;
```