QUY HOẠCH ĐỘNG CHỮ SỐ

I. ĐẶT VẤN ĐỀ

Có rất nhiều dạng bài toán yêu cầu đếm số lượng các số nguyên k trong phạm vi từ A đến B và thỏa mãn một tính chất cụ thể có thể liên quan đến các chữ số của nó.

Thuật toán đơn giản chung cho các bài toán đó là:

- Nhập A, B
- Dem = 0
- Với mỗi k chạy từ A đến B:
    - Nếu k thoả mãn thì tăng Dem
- Xuất Dem

Độ phức tạp: lớn hơn O(B) do có thể việc kiểm tra sốk thoả mãn hay không cũng tốn thêm thời gian. Nếu B quá lớn (chẳng hạn như tới 10^{18}) thì không thể chạy được trong thời gian cho phép.

Cách khác là ta sẽ sử dụng kĩ thuật quy hoạch động chữ số.

II. Ý TƯỞNG CHÍNH

Gọi `G(x)` là số lượng các số nguyên như vậy trong phạm vi từ 0 đến x, thì đáp án của bải toán là `G(B)-G(A-1)`. Như vậy, ta cần viết hàm G(x).

Giả sử số x có n chữ số. Chẳng hạn, ta có $x= a_{n-1}a_{n-2}\dots a_2a_1a_0$, trong đó a_i ($0 \le i \le n-1$) cho biết chữ số thứ i tính từ bên phải. Chữ số tận cùng bên trái a_n là chữ số có nghĩa đầu tiên. Khi đó, giả sử số $k \le x$ thì k sẽ có dạng $k = t_{n-1}t_{n-2}\dots t_2t_1t_0$, trong đó có các điều kiện ràng buộc sau:

- $0 \le t_{n-1} \le a_{n-1}$
- $0 \le t_{n-2} \le a_{n-2}$nếu $t_{n-1} = a_{n-1}$, ngược lại $0 \le t_{n-2} \le 9$.
- $0 \le t_{n-3} \le a_{n-3}$nếu $t_{n-1} = a_{n-1}$ và$t_{n-2} = a_{n-2}$, ngược lại $0 \le t_{n-3} \le 9$.
- …

Tổng quát: $0 \le t_i \le a_i$ nếu $t_j = a_j$, $\forall j=i+1..n-1$, ngược lại $0 \le t_i \le 9$.

Như vậy, chữ số t_i có thể bị giới hạn ($0 \le t_i \le a_i$) hoặc không bị giới hạn ($0 \le t_i \le 9$).

Điều kiện để t_i bị giới hạn là: $t_{i+1} = a_{i+1}$, $t_{i+2} = a_{i+2}$, …, $t_{n-1} = a_{n-1}$. Hay có thể nói cách khác: t_i bị giới hạn nếu t_{i+1} bị giới hạn và t_{i+1} đạt đến giới hạn (tức là $t_{i+1}=a_{i+1}$).

Hàm chính ở đây là một hàm đệ quy `thu(i,…)`, là hàm quay lui để thử các khả năng của chữ số thứ i (tức là t_i). Với mỗi giá trị của t_i, ta sẽ gọi đệ quy đến `thu(i-1,…)`. Bằng cách gọi `thu(n-1,…)`, ta sẽ sinh ra các số k trong phạm vi từ 0 đến x. Với mỗi số sinh ra, ta kiểm tra nó có thoả mãn tính chất đề bài yêu cầu hay không, nếu có thì tăng kết quả thêm 1.

Với cách này, số trường hợp sinh ra cũng quá lớn. Tuy nhiên, sẽ có nhiều trường hợp 1 hàm cùng tham số giống nhau (gọi là một trạng thái) được gọi nhiều lần. Ta sẽ khắc phục bằng cách dùng một mảng để lưu trạng thái đó. Nếu gặp lại, ta không cần tính lại nữa mà lấy ngay kết quả đã lưu trong mảng (người ta còn gọi đây là kĩ thuật đệ quy có nhớ).

Giá trị của hàm `thu(i,…)` là số lượng số thoả mãn đề bài khi chúng ta đã có các chữ số từ n-1 về i+1, mà các giá trị của các chữ số đó sẽ được đại diện bởi một hoặc nhiều tham số thêm vào (tuỳ thuộc vào từng bài toán).

Mẫu chung cho các hàm như sau:

Khai báo mảng `a[]` để lưu các chữ số của x và biến n là số chữ số.

Khai báo mảng `F[i][…]` để lưu các trạng thái //số chiều tuỳ thuộc vào từng bài toán

Ban đầu, mảng `F[i][…]` sẽ được gán bằng -1 hết, tức là trạng thái đó chưa được tính.

Hàm `thu(i, gh, …)` //Thử các trường hợp cho chữ số thứ i; có giới hạn hay không (gh=true hay false). Các tham số khác tuỳ bài toán.Chữ số thứ i nếu bị giới hạn thì nó chỉ nhận giá trị từ 0..a[i], còn nếu không, nó nhận giá trị từ 0..9.

Hàm `thu(i, gh, …)`

- Nếu i < 0 thì:
    - Nếu số sinh ra thoả mãn điều kiện thì trả về 1;
    - Ngược lại, trả về 0;
- Nếu gh=false và F[i][...] >= 0 thì trả về F[i][...] //*nếu trạng thái này đã được tính trước đó rồi thì lấy kết quả từ mảng lưu kết quả của trạng thái.*
- kq = 0;
- maxc = (gh=true ? a[i] : 9); //*giá trị tối đa mà chữ số thứ i đạt được*

- Với mỗi c chạy từ 0 đến maxc: //*cho chữ số thứ i bằng c*
    - ghm = (gh = true) AND (c = maxc) //*giới hạn của chữ số thứ i-1*
    - kq += `thu(i-1, ghm, …)` //*gọi đệ quy đến chữ số phía sau*
- Nếu gh=false thì F[i][...]=kq; //*lưu kết quả của trang thái để lần sau dùng*
- Trả về kq;

Hàm `G(x)`:

- n=0;
- a[0]=0;
- Trong khi x > 0 thì
    - a[n] = x mod 10; //*tách và lưu chữ số đơn vị của x vào a[n]*
    - x = x div 10; //*xoá chữ số đơn vị của x*
    - n = n + 1;
- Trả về `thu(n-1, true,...)` //*chữ số thứ n-1 luôn bị giới hạn.*

Hàm `main()` //*chương trình chính*

- Nhập A, B
- Cho F[i][...] = -1 hết //fillchar hoặc memset
- Xuất G(B)-G(A-1)

Đánh giá độ phức tạp thời gian:

Thời gian tiêu tốn nhiều cho việc gọi các trạng thái. Do đó, số trạng thái là tích của số khả năng của các tham số của hàm `thu(i,gh,…)`. Trong mỗi trạng thái, có một vòng lặp chạy tối đa 10 lần. Vậy số lần gọi trạng thái tối đa là 10 * tích của số khả năng của các tham số của hàm `thu(i,gh,…)`.