BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM HÀ NỘI

KỲ THI TUYỂN SINH LỚP 10
TRƯỜNG THPT CHUYÊN ĐẠI HỌC SƯ PHẠM NĂM 2024

ĐỀ CHÍNH THỨC

ĐÁP ÁN VÀ THANG ĐIỂM
MÔN THI: TIN HỌC

Câu	Nội dung	Điểm thành phần	Điểm
1	**Subtask 1: $n \leq 10^4$** Duyệt mọi giá trị m từ 1 đến n: Với mỗi giá trị m, tìm $GCD(n,m)$ và tính giá trị $k = GCD(n,m) + m$. Đáp số là giá trị lớn nhất trong các giá trị k tìm được.	**1.8 điểm**	**3.0 điểm**
	Subtask 2: $n \leq 10^{14}$ Nhận thấy, nếu $m < n - 1$ và $GCD(m,n) = 1$ thì đáp số luôn nhỏ hơn n. Từ đó suy ra nếu n nguyên tố thì $m = n - 2$ chính là đáp số cần tìm. Với n không nguyên tố, nên chọn số m sao cho $GCD(m,n) > 1$. Giả sử n có các ước $x_1, x_2, \ldots, x_k$ được liệt kê tăng dần. Vì $GCD(m,n)$ phải là ước của n nên ta sẽ thử lần lượt các giá trị của $x_1, x_2, \ldots, x_k$ để tìm m. Lần lượt duyệt qua các ước của n, nếu $GCD(m,n) = x_i$. Khi đó, m sẽ là bội của x_i lớn nhất nhỏ hơn n. Vậy $m = n - x_i$. Do m cần tìm phải đạt giá trị lớn nhất nên $m = n - x_1$ với x_1 là ước nhỏ nhất của n. Để tìm ước số nhỏ nhất của n, lần lượt xét các giá trị i từ 2 đến $\sqrt{n}$ và tìm số i đầu tiên là ước của n. Khi đó, đáp số cần tìm là $n - i$. Độ phức tạp: $O(\sqrt{n})$.	**1.2 điểm**	
2	**Subtask 1: $n \leq 10$** Duyệt lần lượt các số i từ 1 đến n và kiểm tra xem có số i có nằm trong xâu chứa chữ số bị hỏng không để đếm.	**0.5 điểm**	**2.5 điểm**
	Subtask 2: $n \leq 10^5$ Duyệt tất cả các số i từ 1 $\rightarrow n$, Với mỗi số i, tách các chữ số và kiểm tra xem có chữ số nào của i nằm trong xâu chứa các chữ số đã bị hỏng hay không? Nếu không có, tăng đáp số lên 1 đơn vị.	**0.75 điểm**	
	Subtask 3: $n \leq 10^7$ Vì số lượng các số cần kiểm tra lớn nên ta chuẩn bị trước mảng đánh dấu gồm 10 phần tử $d_0, d_1, \ldots d_9$, với $d_i = 0/1$ nếu chữ số i tương ứng là bị hỏng/không bị hỏng. Sau đó, duyệt các số i, lần lượt từ 1 đến n và kiểm tra i có chứa chữ số bị hỏng hay không bằng cách kiểm tra giá trị d[.] tương ứng với từng chữ số của i.	**1.25 điểm**	

	Độ phức tạp tính toán: $O(n \times$ (số chữ số của n)).		
3	**Subtask 1: $n \leq 5000$** Gọi dãy s có $n+1$ phần tử với $s_i = a_1 + a_2 + \cdots + a_i$, với $s_0 = 0$. Khi đó, $s_i = s_{i-1} + a_i$. Nhận xét, độ dài tối đa cần tìm là n. Vậy thử xét k lần lượt từ 1 tới n. Với mỗi độ dài k, xuất phát từ a_1 và kiểm tra có tồn tại tất cả các đoạn con độ dài k có tổng lớn hơn hoặc bằng m không bằng công thức $s_i - s_{i-k} \geq m$ với mọi $i \in [k, n]$. Giá trị k đầu tiên thỏa mãn điều kiện trên chính là đáp số cần tìm. Độ phức tạp: $O(n^2)$	**1.25 điểm**	**2.5 điểm**
	Subtask 2: $n \leq 10^6$ Cải tiến thuật toán từ subtask 1 bằng nhận xét: Nếu ta thử với một độ dài x và kiểm tra thấy: tồn tại ít nhất một đoạn con có độ dài x có tổng nhỏ hơn m thì đáp số không thể nhỏ hơn x. Vậy ta sử dụng phương pháp tìm kiếm nhị phân giá trị k trên đoạn $[1, n]$ để tìm đáp số. Độ phức tạp: $O(n * \log_2 n)$	**1.25 điểm**	
4	**Subtask 1: $T \leq 10, a \leq 10^6$** Với mỗi bộ dữ liệu đầu vào, đọc vào số a, lần lượt xét các giá trị i từ $a \rightarrow 0$: với mỗi số i đó, thực hiện tách các chữ số và kiểm tra các chữ số của i có tăng nghiêm ngặt theo đề bài yêu cầu. Số đầu tiên thỏa mãn chính là đáp s	**0.8 điểm**	**2.0 điểm**
	Subtask 2: $a \leq 10^6$ Vì số bộ dữ liệu lớn, ta không thể xử lý trực tiếp như subtask 1 mà với mỗi a từ $0 \rightarrow a$ tính trước giá trị $f[a] = x$ là giá trị chỉ gồm các chữ số tăng nghiêm ngặt từ trái qua phải. Xử lý trước để tính mảng f. Sau đó, tiến hành đọc dữ liệu vào, với mỗi số a đọc vào, in ra giá trị $f[a]$ tương ứng.	**0.6 điểm**	
	Subtask 3: $T \leq 100$ Số lượng những số tự nhiên có biểu diễn thập phân tăng ngặt không nhiều, chỉ có $2^9 = 512$ số như vậy, số nhỏ nhất là 0, số lớn nhất là 123456789. Có thể liệt kê những số có biểu diễn thập phân tăng ngặt theo cách: duyệt mọi tập con của tập {1,2,3,4,5,6,7,8,9}: Tập $\emptyset$ ứng với số 0, các tập $\neq \emptyset$ chỉ cần mang các phần tử của nó viết ra theo thứ tự tăng dần là được một biểu diễn thập phân tăng ngặt. Tuy nhiên ta có thể liệt kê những số có biểu diễn thập phân tăng ngặt bằng cách thuận tiện hơn: Khởi tạo hàng đợi Q để chứa các số có biểu diễn thập phân tăng ngặt. Ban đầu Q chỉ gồm một số 0, lần lượt rút một phần tử x ra khỏi Q, nối vào sau biểu diễn thập phân của x các chữ số d lớn hơn chữ số hàng đơn vị của x để được số y rồi đẩy y vào Q. Thuật toán kết thúc khi hàng đợi rỗng và các số lấy ra	**0.4 điểm**	

khỏi Q ở mỗi bước chính là các số có biểu diễn thập phân tăng ngặt, hơn nữa những số này cũng đã được liệt kê theo thứ tự tăng dần.

```
vector<int> xlist;

void Prepare()
{
    queue<int> Q;
    Q.push(0); //Hàng đợi ban đầu chỉ có số 0
    while (!Q.empty())
    {
        int x = Q.front(); Q.pop(); //Lấy x khỏi hàng đợi
        xlist.push_back(x); //Lưu x vào xlist
        for (int d = x % 10 + 1; d <= 9; ++d) //Xét chữ số d > chữ số đơn vị của x
            Q.push(x * 10 + d); //Nối d vào đuôi x được số x * 10 + d, đẩy vào Q
    }
}
```

Sau đó, với mỗi dữ liệu a, ta tìm chỉ số lớn nhất $\leq a$ xuất hiện trong danh sách $xlist$ bằng thuật toán tìm kiếm tuần tự:

```
i = 0;
while(xlist[i] <= a) i++;
cout << xlist[i - 1] << '\n';
```

Subtask 4: $T \leq 10^6, 0 \leq a \leq 10^9$

Thực hiện việc sinh các số có các chữ số tăng nghiêm ngặt như subtask 3. Ta có nhận xét, các số trong dãy $xlist$ theo cả hai cách sinh đều đã được sắp xếp tăng dần.

Vậy nên, với mỗi dữ liệu a, ta chỉ cần tìm số lớn nhất $\leq a$ đã lưu trong danh sách $xlist$ bằng thuật toán tìm kiếm nhị phân:

```
cout << *(upper_bound(xlist.begin(), xlist.end(), a) - 1) << '\n';
```

Một cách khác là thao tác trực tiếp trên dãy chữ số của a bằng thuật toán sinh, tuy nhiên cách này đòi hỏi kỹ thuật cao hơn, cần cài đặt cẩn thận để tránh nhầm lẫn.

0.2 điểm

HẾT