Monotone Increasing Digits

Given a non-negative integer `N`, find the largest number that is less than or equal to `N` with monotone increasing digits.

(Recall that an integer has *monotone increasing digits* if and only if each pair of adjacent digits `x` and `y` satisfy `x .)`

Example 1:

```
Input: N = 10
Output: 9
```

Example 2:

```
Input: N = 1234
Output: 1234
```

Example 3:

```
Input: N = 332
Output: 299
```

Note: `N` is an integer in the range `[0, 10^9]`.

Solution 1

The idea is to go from the LSB to MSB and find the last digit, where an inversion happens.
There are 2 cases to consider:

case 1:
In 14267 , we see that inversion happens at 4. In this case, then answer is obtained by reducing 4 to 3, and changing all the following digits to 9.
=> 13999

case 2:
1444267, here eventhough the last inversion happens at the last 4 in 1444, if we reduce it to 3, then that itself breaks the rule. So once we find the last digit where inversion happens, if that digit is repeated, then we have to find the last position of that digit. After that it is same as case1, where we reduce it by 1 and set the remaining digits to 9.
=> 1399999

The steps are:

1. Convert n into num array in reverse order
2. Find the leftmost position that is inverted and if the inverted character repeats itself, find the leftmost repeated digit.
3. Fill the digits after inversion as 9
4. Reduce the digit that caused the inversion by -1
5. Reverse back the num array and convert to int

```
def monotoneIncreasingDigits(self, N):
        if N < 10: return N
        n, inv_index = N, -1
        num = [int(d) for d in str(n)[::-1]]

        for i in range(1, len(num)):
            if num[i] > num[i - 1] or (inv_index != -1 and num[inv_index] == num[i]
):
                inv_index = i

        if inv_index == -1: return N

        for i in range(inv_index): num[i] = 9
        num[inv_index] -= 1

        return int(''.join([ str(i) for i in num[::-1]]))
```

written by johnyrufus16 original link here

Solution 2

```
class Solution {
public:
    int monotoneIncreasingDigits(int N) {
        string n_str = to_string(N);

        int marker = n_str.size();
        for(int i = n_str.size()-1; i > 0; i --) {
            if(n_str[i] < n_str[i-1]) {
                marker = i;
                n_str[i-1] = n_str[i-1]-1;
            }
        }

        for(int i = marker; i < n_str.size(); i ++) n_str[i] = '9';

        return stoi(n_str);
    }
};
```

written by Zee original link here

Solution 3

//translated from the simple and very short c++ solution

```
    public int monotoneIncreasingDigits(int N) {

        if(N<=9)
            return N;
        char[] x = String.valueOf(N).toCharArray();

        int mark = x.length;
        for(int i = x.length-1;i>0;i--){
            if(x[i]<x[i-1]){
                mark = i-1;
                x[i-1]--;
            }
        }
        for(int i = mark+1;i<x.length;i++){
            x[i] = '9';
        }
        return Integer.parseInt(new String(x));
    }
```

written by Bruce original link here

From Leetcoder.