Next Greater Element III

Given a positive **32-bit** integer **n**, you need to find the smallest **32-bit** integer which has exactly the same digits existing in the integer **n** and is greater in value than n. If no such positive **32-bit** integer exists, you need to return -1.

Example 1:

Input: 12
Output: 21

Example 2:

Input: 21
Output: -1

Solution 1

This solution is just a java version derived from this post.

At first, lets look at the edge cases -

1. If all digits sorted in descending order, then output is always “Not Possible”. For example, 4321.

2. If all digits are sorted in ascending order, then we need to swap last two digits. For example, 1234.
3. For other cases, we need to process the number from rightmost side (why? because we need to find the smallest of all greater numbers)

Now the main algorithm works in following steps -

I) Traverse the given number from rightmost digit, keep traversing till you find a digit which is smaller than the previously traversed digit. For example, if the input number is “534976”, we stop at 4 because 4 is smaller than next digit 9. If we do not find such a digit, then output is “Not Possible”.

II) Now search the right side of above found digit ‘d’ for the smallest digit greater than ‘d’. For “5**3**4976″, the right side of 4 contains “976”. The smallest digit greater than 4 is **6**.

III) Swap the above found two digits, we get 5**3**6974 in above example.

IV) Now sort all digits from position next to ‘d’ to the end of number. The number that we get after sorting is the output. For above example, we sort digits in bold 536**974**. We get “536**479**” which is the next greater number for input 534976.

```java
public class Solution {
    public int nextGreaterElement(int n) {
        char[] number = (n + "").toCharArray();

        int i, j;
        // I) Start from the right most digit and
        // find the first digit that is
        // smaller than the digit next to it.
        for (i = number.length-1; i > 0; i--)
            if (number[i-1] < number[i])
               break;

        // If no such digit is found, its the edge case 1.
        if (i == 0)
            return -1;

         // II) Find the smallest digit on right side of (i-1)'th
         // digit that is greater than number[i-1]
        int x = number[i-1], smallest = i;
        for (j = i+1; j < number.length; j++)
            if (number[j] > x && number[j] <= number[smallest])
                smallest = j;

        // III) Swap the above found smallest digit with
        // number[i-1]
        char temp = number[i-1];
        number[i-1] = number[smallest];
        number[smallest] = temp;

        // IV) Sort the digits after (i-1) in ascending order
        Arrays.sort(number, i, number.length);

        long val = Long.parseLong(new String(number));
        return (val <= Integer.MAX_VALUE) ? (int) val : -1;
    }
}
```

written by sanketdige268 original link here

Solution 2

```
int nextGreaterElement(int n) {
    auto digits = to_string(n);
    next_permutation(begin(digits), end(digits));
    auto res = stoll(digits);
    return (res > INT_MAX || res <= n) ? -1 : res;
}
```

written by votrubac original link here

Solution 3

```
public class Solution {
    public int nextGreaterElement(int n) {
        char[] a=(""+n).toCharArray();
        int i = a.length - 2;
        while (i >= 0 && a[i + 1] <= a[i]) {
            i--;
        }
        if(i<0)
            return -1;
        int j = a.length - 1;
        while (j >= 0 && a[j] <= a[i]) {
            j--;
        }
        swap(a, i, j);
        reverse(a, i + 1);
        try{
           return Integer.parseInt(new String(a));
        }
        catch(Exception e){
           return -1;
        }
    }
    private void reverse(char[] a, int start) {
        int i = start, j = a.length - 1;
        while (i < j) {
            swap(a, i, j);
            i++;
            j--;
        }
    }
    private void swap(char[] a, int i, int j) {
        char temp = a[i];
        a[i] = a[j];
        a[j] = temp;
    }
}
```

written by vinod23 original link here