556. Next Greater Element III

April 8, 2017 | 13.3K views
Average Rating: 4.21 (12 votes)

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

Approach #1 Brute Force [Time Limit Exceeded]

To solve the given problem, we treat the given number as a string, s. In this approach, we find out every possible permutation of list formed by the elements of the string s formed. We form a list of strings, $list$, containing all the permutations possible. Then, we sort the given $list$ to find out the permutation which is just larger than the given one. But this one will be a very naive approach, since it requires us to find out every possible permutation which will take really long time.

Java

```
public class Solution {
    public String swap(String s, int i0, int i1) {
        if (i0 == i1)
            return s;
        String s1 = s.substring(0, i0);
        String s2 = s.substring(i0 + 1, i1);
        String s3 = s.substring(i1 + 1);
        return s1 + s.charAt(i1) + s2 + s.charAt(i0) + s3;
    }
    ArrayList < String > list = new ArrayList < > ();
    void permute(String a, int l, int r) {
        int i;
        if (l == r)
            list.add(a);
        else {
            for (i = l; i <= r; i++) {
                a = swap(a, l, i);
                permute(a, l + 1, r);
                a = swap(a, l, i);
            }
        }
    }
    public int nextGreaterElement(int n) {
        String s = "" + n;
        permute(s, 0, s.length() - 1);
        Collections.sort(list);
        int i;
        for (i = list.size() - 1; i >= 0; i--) {
```

Complexity Analysis

- Time complexity : $O(n!)$. A total of $n!$ permutations are possible for a number consisting of n digits.
- Space complexity : $O(n!)$. A total of $n!$ permutations are possible for a number consisting of n digits, with each permutation consisting of n digits.

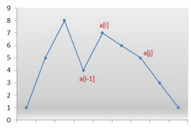
Approach #2 Linear Solution [Accepted]

Algorithm

In this case as well, we consider the given number n as a character array a. First, we observe that for any given sequence that is in descending order, no next larger permutation is possible. For example, no next permutation is possible for the following array: `[9, 5, 4, 3, 1]`

We need to find the first pair of two successive numbers $a[i]$ and $a[i-1]$, from the right, which satisfy $a[i] > a[i-1]$. Now, no rearrangements to the right of $a[i-1]$ can create a larger permutation since that subarray consists of numbers in descending order. Thus, we need to rearrange the numbers to the right of $a[i-1]$ including itself.

Now, what kind of rearrangement will produce the next larger number? We want to create the permutation just larger than the current one. Therefore, we need to replace the number $a[i-1]$ with the number which is just larger than itself among the numbers lying to its right section, say $a[j]$.



We swap the numbers $a[i-1]$ and $a[j]$. We now have the correct number at index $i-1$. But still the current permutation isn't the permutation that we are looking for. We need the smallest permutation that can be formed by using the numbers only to the right of $a[i-1]$. Therefore, we need to place those numbers in ascending order to get their smallest permutation.

But, recall that while scanning the numbers from the right, we simply kept decrementing the index until we found the pair $a[i]$ and $a[i-1]$ where, $a[i] > a[i-1]$. Thus, all numbers to the right of $a[i-1]$ were already sorted in descending order. Furthermore, swapping $a[i-1]$ and $a[j]$ didn't change that order. Therefore, we simply need to reverse the numbers following $a[i-1]$ to get the next smallest lexicographic permutation.

The following animation will make things clearer:

1 / 14

Java

```
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--;
```

Complexity Analysis

- Time complexity : $O(n)$. In worst case, only two scans of the whole array are needed. Here, n refers to the number of digits in the given number.
- Space complexity : $O(n)$. An array a of size n is used, where n is the number of digits in the given number.

Rate this article:

Previous Next

Comments: 10

Sort By

Type comment here... (Markdown is supported)

Preview Post

guestly ★ 78 February 27, 2019 8:45 AM

This explanation is terrible. It explains the "how" but not the "why"...

1. Why do you have to walk from right to left? Because we want the least significant digit that is greater than the current number.
2. Why do you have to find `a[i]` and swap? We're trying to find a digit which is only 1 distance

Read More

37 Share Reply

SHOW 3 REPLIES

a-bi-c ★ 882 April 3, 2019 8:22 PM

This is the Next Permutation problem - https://leetcode.com/problems/next-permutation/
This is the best explanation that I've seen - https://www.youtube.com/results?search_query=back+to+back+next+permutation

11 Share Reply

RogerFederer ★ 857 January 5, 2018 5:48 AM

```
class Solution(object):
    def nextGreaterElement(self, n):
        """
        :type n: int
```

Read More

2 Share Reply

heathensoul ★ 1 July 29, 2019 9:19 AM

Python Solution [Accepted]

(Runtime: 32 ms, faster than 87.50% of Python3 online submissions
Memory Usage: 13.7 MB, less than 5.00% of Python3 online submissions).

Read More

1 Share Reply

jzchen ★ 6 August 8, 2018 12:12 AM

I'm getting a wrong answer with the test case n =230241
My answer is 230421, but the expected one is 230412.
I think both answers are correct, according to the problem description! :(
Can someone fix it?

0 Share Reply

SHOW 2 REPLIES

CAFEBABY ★ 314 April 18, 2017 12:00 AM

while (j >= 0 && a[j] <= a[i]) {
j--;
}
can be changed to:
while (a[j] <= a[i]) {

Read More

0 Share Reply

lwillcrackit ★ 7 July 14, 2020 8:48 AM

```
class Solution {
    public int nextGreaterElement(int n) {
        String s = String.valueOf(n);
```

Read More

0 Share Reply

mrthepratik ★ 17 June 5, 2020 6:48 AM

this is really intuitive explaination of the problem . totally disagree with the other negative comments on this thread , the solution does layout the solving approach really well

0 Share Reply

monstroliang ★ 6 May 17, 2020 3:50 AM

well same solution of next permutation.....

0 Share Reply

sbmm ★ -1 April 15, 2018 2:28 AM

This will not work if the input is negative
-1

SHOW 1 REPLY