🔢 Next Greater Element III – Full Documentation

1. **Problem Statement**

   Given a positive integer n, find the smallest integer that:

   - Has exactly the same digits as n.
   - Is greater than n.
   - Fits in a 32-bit signed integer.

   If no such number exists, return -1.

   🔹 Example 1:

   Input: n = 12
   Output: 21

   🔹 Example 2:

   Input: n = 21
   Output: -1

2. **Intuition**

   This is a classic "next permutation" problem:

   Find the next number that is greater than the current one by rearranging its digits.

   We seek the next lexicographical permutation of the digits in n.

3. **Key Observations**

- If digits are sorted in descending order (like 321), no greater permutation exists.
- To form the next permutation:
    - Find a pivot: the first digit from the right which is smaller than the digit next to it.

    - Find the smallest digit on the right side of the pivot that is greater than the pivot.
    - Swap the pivot with this digit.
    - Reverse the digits to the right of the pivot.

4. Approach

- Convert the number into a list of digits.
- Traverse from right to left to find the first decreasing digit.
- If not found, return -1.
- Find the smallest digit greater than this pivot on its right.
- Swap and reverse the sublist.
- Convert back to integer and return if within 32-bit limit.

5. Edge Cases

- Input has only one digit → return -1.
- All digits are in descending order → return -1.
- Result exceeds $2^{31} - 1$ → return -1.
- Duplicates exist → still valid if reordering forms a greater number.

6. Complexity Analysis

□ Time Complexity:

- O(n) where n is the number of digits.

💾 Space Complexity:

- O(n) for storing digits as a list.

7. Alternative Approaches

- Brute Force: Generate all permutations, sort and find next → Inefficient: O(n!) time.
- Using built-in next_permutation (C++): Fast but not available in all languages.
- Heap-based methods: Overkill for this problem.

8. Algorithm

- Convert number to list of digits.
- Find the first index i from the end such that digits[i] < digits[i+1].
- If no such index exists, return -1.
- Find index j such that digits[j] > digits[i], starting from the end.
- Swap digits[i] and digits[j].
- Reverse the sublist from i+1 to the end.
- Convert the list back to integer and check if it fits in 32-bit.

9. Test Cases

Input	Output	Description
12	21	Next permutation exists.
21	-1	No greater permutation.
1234	1243	Smallest next permutation.
4321	-1	Already highest permutation.
1999999999	-1	Result exceeds 32-bit limit.

10. Final Thoughts

This problem blends math, logic, and algorithm design. It's a great case of:

- Understanding permutations.
- Applying optimal O(n) solution.
- Handling constraints like integer bounds.