1842. Next Palindrome Using Same Digits

Leetcode Link

Problem Description

You are given a numeric string `num`, representing a very large palindrome. Your goal is to return the smallest palindrome larger than `num` that can be created by rearranging its digits. If no such palindrome exists, return an empty string "".

A palindrome is a number that reads the same backward as forward.

Example 1:

Input: num = "1221"
Output: "2112"
Explanation: The next palindrome larger than "1221" is "2112".

Example 2:

Input: num = "32123"
Output: ""
Explanation: No palindromes larger than "32123" can be made by rearranging the digits.

Example 3:

Input: num = "45544554"
Output: "54455445"
Explanation: The next palindrome larger than "45544554" is "54455445".

Constraints:

- 1 <= num.length <= 105
- num is a palindrome.

Approach

To solve this problem, we can use the following algorithm:

1. Create a vector `A` containing the first half of the characters in the input string `num`.
2. Apply the "next permutation" function on the vector `A`. This function will rearrange the digits in the vector to create the next greater permutation.
3. If there is no next greater permutation, return an empty string.
4. Reconstruct the palindrome from the modified vector `A`.
5. Return the reconstructed palindrome.

Walkthrough

Let's walk through an example with the input `num = "1221"`.

1. Create the vector `A` containing the first half of the characters in `num`: `[1, 2]`.
2. Apply the "next permutation" function on `A`, getting the new vector `[2, 1]`.
3. Since there is a next greater permutation, we can proceed to step 4.
4. Reconstruct the palindrome from the modified `A`: The result is `"2112"`.
5. Return the reconstructed palindrome: `"2112"`.

Solution in Python

```python
from itertools import permutations

class Solution:
    def nextPalindrome(self, num: str) -> str:
        n = len(num)
        A = [int(x) for x in num[:n // 2]]

        next_permutation = None
        for perm in sorted(permutations(A)):
            if tuple(A) < perm:
                next_permutation = perm
                break

        if not next_permutation:
            return ""

        s = "".join(str(x) for x in next_permutation)
        if n % 2 == 1:
            return s + num[n // 2] + s[::-1]
        return s + s[::-1]
```

Solution in Java

```java
import java.util.Arrays;

class Solution {
    public String nextPalindrome(String num) {
        int n = num.length();
        int[] A = new int[n / 2];

        for (int i = 0; i < A.length; ++i) {
            A[i] = num.charAt(i) - '0';
        }

        int[] nextPermutation = nextPermutation(A);
        if (nextPermutation == null) {
            return "";
        }

        StringBuilder sb = new StringBuilder();
        for (int a : nextPermutation) {
            sb.append(a);
        }

        if (n % 2 == 1) {
            return sb.toString() + num.charAt(n / 2) + sb.reverse().toString();
        }
        return sb.toString() + sb.reverse().toString();
    }

    private int[] nextPermutation(int[] nums) {
        int n = nums.length;
        int i = n - 2;
        while (i >= 0 && nums[i] >= nums[i + 1]) {
            --i;
        }

        if (i < 0) {
            return null;
        }

        int j = n - 1;
        while (j > i && nums[j] <= nums[i]) {
            --j;
        }

        // swap nums[i] and nums[j]
        int temp = nums[i];
        nums[i] = nums[j];
        nums[j] = temp;

        // reverse nums[i + 1 .. n - 1]
        int l = i + 1, r = n - 1;
        while (l < r) {
            temp = nums[l];
            nums[l] = nums[r];
            nums[r] = temp;
            ++l;
            --r;
        }
        return nums;
    }
}
```

Solution in JavaScript

```javascript
var nextPermutation = function(nums) {
    let i = nums.length - 2;
    while (i >= 0 && nums[i] >= nums[i + 1]) {
        i--;
    }
    if (i < 0) return null;

    let j = nums.length - 1;
    while (j > i && nums[j] <= nums[i]) {
        j--;
    }
    [nums[i], nums[j]] = [nums[j], nums[i]];
    nums.slice(i + 1).reverse().forEach((x, idx) => nums[idx + i + 1] = x);
    return nums;
};

var nextPalindrome = function(num) {
    let n = num.length, A = [];
    for (let i = 0; i < n / 2; ++i) {
        A[i] = parseInt(num[i]);
    }
    let nextPermutation = nextPermutation(A);
    if (nextPermutation === null) return "";
    let s = nextPermutation.join("");
    if (n % 2) return s + num[n / 2] + s.split("").reverse().join("");
    return s + s.split("").reverse().join("");
};
```

Solution in C++

```cpp
#include <algorithm>
#include <string>
#include <vector>

class Solution {
public:
    std::string nextPalindrome(std::string num) {
        int n = num.size();
        std::vector<int> A(n / 2);

        for (int i = 0; i < A.size(); ++i)
            A[i] = num[i] - '0';

        if (!std::next_permutation(A.begin(), A.end()))
            return "";

        std::string s;

        for (const int a : A)
            s += '0' + a;

        if (n % 2)
            return s + num[n / 2] + std::string(s.rbegin(), s.rend());
        return s + std::string(s.rbegin(), s.rend());
    }
};
```

Solution in C#

~~~csharp
using System;

public class Solution {
    public string NextPalindrome(string num) {
        int n = num.Length;
        int[] A = new int[n / 2];

        for (int i = 0; i < A.Length; ++i) {
            A[i] = num[i] - '0';
        }

        if (!NextPermutation(A)) {
            return "";
        }

        string s = "";

        foreach (int a in A) {
            s += (char)('0' + a);
        }

        if (n % 2 == 1) {
            return s + num[n / 2] + new string(s.ToCharArray().Reverse().ToArray());
        }
        return s + new string(s.ToCharArray().Reverse().ToArray());
    }

    private bool NextPermutation(int[] nums) {
        int n = nums.Length;
        int i = n - 2;

        while (i >= 0 && nums[i] >= nums[i + 1]) {
            --i;
        }

        if (i < 0) {
            return false;
        }

        int j = n - 1;

        while (j > i && nums[j] <= nums[i]) {
            --j;
        }

        Swap(ref nums[i], ref nums[j]);
        Array.Reverse(nums, i + 1, n - i - 1);
        return true;
    }

    private static void Swap(ref int a, ref int b) {
        int temp = a;
        a = b;
        b = temp;
    }
}
```## Conclusion

In conclusion, we present the solutions to find the next greater palindrome of a given numeric string `num` in Python, JavaScript,
~~~

Got a question? Ask the Teaching Assistant anything you don't understand.