leetcode 1814

📜 Problem Statement

Ek array `nums` diya gaya hai jisme non-negative integers hain.

`rev(x)` = integer `x` ka reverse (leading zeros remove ho jaate hain).

A pair `(i, j)` **nice** hai agar:

1. `0 <= i < j < nums.length`
2. `nums[i] + rev(nums[j]) == nums[j] + rev(nums[i])`

Return karo **nice pairs ka total count**, modulo 109+710^9 + 7.

🔍 Key Observation

Given condition ko rearrange karo:

```
nums[i] + rev(nums[j]) == nums[j] + rev(nums[i])
```

→

```
nums[i] - rev(nums[i]) == nums[j] - rev(nums[j])
```

Matlab, **sirf yeh difference** `diff = nums[k] - rev(nums[k])` match hona chahiye for a nice pair.

💡 Approach

1. Ek `unordered_map` rakho jo har `diff` ki frequency store kare.
2. Har number ke liye:
   - `diff` calculate karo.
   - Agar yeh diff pehle aa chuka hai, iska matlab hai ki **current element pehle wale sab same-diff elements ke saath nice pair banayega**.
   - Count me frequency add karo.
   - Frequency update karo.
3. Har bar modulo 109+710^9 + 7 lo taaki overflow na ho.

💻 Code

```
class Solution {
public:
    int rev(int n) {
        int rev = 0;
        while (n != 0) {
```

```
        rev = rev * 10 + n % 10;
        n /= 10;
    }
    return rev;
}

int countNicePairs(vector<int>& nums) {
    unordered_map<int, int> mp;
    int count = 0;
    const int MOD = 1000000007;

    for (int i = 0; i < nums.size(); i++) {
        int diff = nums[i] - rev(nums[i]);

        // Agar yeh diff pehle aa chuka hai
        if (mp.find(diff) != mp.end()) {
            count = count % MOD;              // overflow prevent
            count += mp[diff];                // naye pairs add
            mp[diff]++;                       // frequency badhao
        }
        else {
            mp[diff]++;                       // pehli baar aaya
        }
    }
    return count % MOD;
}
};
```

🧱 Dry Run Example

Input:

```
nums = [42, 11, 1, 97]
```

i	nums[i]	rev(nums[i])	diff	mp before	count before	Action
0	42	24	18	{}	0	mp[18] = 1
1	11	11	0	{18:1}	0	mp[0] = 1
2	1	1	0	{18:1, 0:1}	0	count += 1 → 1, mp[0] = 2
3	97	79	18	{18:1, 0:2}	1	count += 1 → 2, mp[18] = 2

✅ Final Answer: **2**

📦 Why `% 1000000007`?

- Max elements = 10510^5

- Max pairs ≈ n(n−1)2\frac{n(n-1)}{2} → ~5×1095 \times 10^9
- `int` me overflow ho sakta hai
- Problem statement me explicitly bola hai result `% (10^9 + 7)` return karna hai.
