✅ String Processing / Counting / Automata

📌 Problem Statement:

Count the number of strings of length `n` over the alphabet `{a, b, c}` that **do not contain the substring "ab"**.

Return the count modulo `10^9 + 7`.

📥 Example:

```
Input:
n = 2

Output:
8

Explanation:
All length-2 strings from {a,b,c} = 9.
The only invalid one is "ab".
So, 9 - 1 = 8.
```

🔁 Brute Force:

- Generate all strings of length `n`.
- Check each if it contains "ab".
- Count valid strings.
- Time complexity: O(3^n), impractical for large `n`.

🚀 Optimal Approach (DP with Automata):

- Use DP to count strings without "ab".
- Keep track of last character to avoid "ab".
- States:
  - `dp[i][lastChar]`: number of valid strings of length `i` ending with `lastChar`.
- Transitions based on lastChar and new character.

✅ Java Code:

```java
public class CountStringsNoAB {
    static final int MOD = 1_000_000_007;
```

```java
    public static int countStrings(int n) {
        // Map characters to indices: a=0, b=1, c=2
        // dp[i][last] = number of strings length i ending with lastChar
        long[][] dp = new long[n + 1][3];

        // Base case: strings of length 1
        for (int ch = 0; ch < 3; ch++) {
            dp[1][ch] = 1;
        }

        for (int i = 2; i <= n; i++) {
            for (int last = 0; last < 3; last++) {
                for (int curr = 0; curr < 3; curr++) {
                    // Avoid substring "ab" = last='a'(0), curr='b'(1)
                    if (last == 0 && curr == 1) continue;
                    dp[i][curr] = (dp[i][curr] + dp[i - 1][last]) % MOD;
                }
            }
        }

        long result = 0;
        for (int ch = 0; ch < 3; ch++) {
            result = (result + dp[n][ch]) % MOD;
        }
        return (int) result;
    }

    public static void main(String[] args) {
        int n = 2;
        System.out.println(countStrings(n)); // Output: 8
    }
}
```

✅ **Output:**

```
8
```