Description | Editorial | Solutions (332) | Submissions

920. Number of Music Playlists

Hard | 1.7K | 147

Companies

Your music player contains `n` different songs. You want to listen to `goal` songs (not necessarily different) during your trip. To avoid boredom, you will create a playlist so that:

- Every song is played **at least once**.
- A song can only be played again only if `k` other songs have been played.

Given `n`, `goal`, and `k`, return *the number of possible playlists that you can create.* Since the answer can be very large, return it **modulo** $10^9 + 7$.

Example 1:

```
Input: n = 3, goal = 3, k = 1
Output: 6
Explanation: There are 6 possible playlists: [1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], and [3, 2, 1].
```

Example 2:

```
Input: n = 2, goal = 3, k = 0
Output: 6
Explanation: There are 6 possible playlists: [1, 1, 2], [1, 2, 1], [2, 1, 1], [2, 2, 1], [2, 1, 2], and [1, 2, 2].
```

Example 3:

Java | Auto

```java
class Solution {
    public int numMusicPlaylists(int n, int goal, int k) {
        final int MOD = (int)1e9 + 7;
        long[][] dp = new long[2][n+1];
        dp[0][0] = 1;

        for (int i = 1; i <= goal; i++) {
            dp[i%2][0] = 0;
            for (int j = 1; j <= Math.min(i, n); j++) {
                dp[i%2][j] = dp[(i - 1)%2][j - 1] * (n - (j - 1)) % MOD;
                if (j > k)
                    dp[i%2][j] = (dp[i%2][j] + dp[(i - 1)%2][j] * (j - k)) % MOD;
            }
        }

        return (int)dp[goal%2][n];
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

n =
3

goal =
3

k =
1

Console | Run | Submit

Description | Editorial | Solutions (794) | Submissions

848. Shifting Letters

Medium 1.3K 119

Companies

You are given a string `s` of lowercase English letters and an integer array `shifts` of the same length.

Call the `shift()` of a letter, the next letter in the alphabet, (wrapping around so that `'z'` becomes `'a'`).

- For example, `shift('a') = 'b'`, `shift('t') = 'u'`, and `shift('z') = 'a'`.

Now for each `shifts[i] = x`, we want to shift the first `i + 1` letters of `s`, `x` times.

Return *the final string after all such shifts to s are applied.*

Example 1:

```
Input: s = "abc", shifts = [3,5,9]
Output: "rpl"
Explanation: We start with "abc".
After shifting the first 1 letters of s by 3, we have "dbc".
After shifting the first 2 letters of s by 5, we have "igc".
After shifting the first 3 letters of s by 9, we have "rpl", the answer.
```

Example 2:

```
Input: s = "aaa", shifts = [1,2,3]
```

Java | Auto

```java
class Solution {
    public String shiftingLetters(String s, int[] shifts) {
        char arr[]=s.toCharArray();
        int total_shifts=0;
        int i = s.length() -1;
        while(i>=0){
            total_shifts += shifts[i]%26;
            arr[i]=(char)((arr[i] - 'a' + total_shifts)%26 + 'a');
            i--;
        }
        return String.valueOf(arr);
    }
}
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

s =
"abc"

shifts =
[3,5,9]

Output

"rpl"

Console | Run | Submit