J - a + 1 Problem (cf1513C)

题目大意

一次操作是将这个数的每一位都加一。求将 n 进行 m 次操作后有几位。

思路

这一道题如果暴力枚举，时间复杂度至少是 O(mt)，过不了这一道题。

其实手模几个样例后会发现，这一次操作之后，原数字的每一位（十进制位）之间没有联系，如：1912，所有 1 变为 2，所有 2 变为 3，所有 9 变为 1 和 0

于是考虑状态递推，只需维护**当前数字存在多少 0 到 9 的数量**即可。

诶，为什么会 TLE3 呢?

> **温馨提示**
>
> 注意测试组数 t 很大哦，并且 n 与 m 也很大，那我该怎么样处理达到即使输入 $2e5$ 个 $n = 1e9, m = 2e5$ 范围的数据也不会 TLE 呢?

如果每读入一个样例就跑一次 dp，那岂不是重复了

于是在 “外面” 预处理出每一个数字（0-9）从第 1 轮到第 $2e5$ 轮演变来的数字数量，每次读入一个数字按找每一个十进制位去统计答案即可

复杂度为 $O(2e6)$，因为预处理和查询都是 $10 * 2e5$

参考代码如下：

```cpp
// Created by qyy

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

#define PII pair<int, int>
#define endl "\n"

const ll inf = 0x3f3f3f3f3f3f3f3f;
const int N = 2e5 + 20;
const int mod = 1e9 + 7;

int n, m;
ll dp[N][11];
```

```
ll cnt[N];

void solve() {
    cin >> n >> m;
    int x = n;
    ll ans = 0;
    while(x){
        int tmp = x % 10;
        ans = (ans + cnt[m + tmp]) % mod;
        x /= 10;
    }

    cout << ans << endl;
}

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    dp[0][0] = 1;
    for(int i = 1; i <= 200017; i++){
        for(int j = 2; j <= 9; j++){
            dp[i][j] = dp[i - 1][j - 1];
        }
        dp[i][1] = (dp[i - 1][9] + dp[i - 1][0]) % mod;
        dp[i][0] = (dp[i - 1][9]) % mod;
        for(int j = 0; j < 10; j++){
            cnt[i] = (dp[i][j] + cnt[i]) % mod;
        }
    }
    int t = 1;
    cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```