**★★★★☆**

**題組：Problem Set Archive** **with Online Judge**

**題號：11472: Beautiful Numbers**

**解題者：賴柏翰**

**解題日期：2025年5月1日**

**題意：**

一個 N 進位的數字，如果能夠符合
（1）0到N-1每一個數字都用過
（2）每個相鄰的數字皆差1
則此為Beautiful Numbers

給出N以及位數M，請求出在N進位下，M位數以內的數字，有多少Beautiful Numbers.

**題意範例：**

若N=2、M=4，則有01、101、1010這三個Beautiful Numbers。

若N=3、M=3，則有012這一個Beautiful Numbers。

**解法：**

使用陣列 arr[bit\_used][length][last\_num]表示狀態​
bit\_used:使用的數字。如：使用了1、3、4，即是11010​
length:數字長度​
last\_num:最後一位數字​

建立陣列後，若想求N=2 M=4的Beautiful numbers，將 [00011][length][last\_num]的結果全數相加，即是Beautiful numbers數量

**解法範例：**

先初始化，將陣列中所有表示個位數的值設為1​
arr[00010][1][1]=1​
arr[00100][1][2]=1​
arr[01000][1][3]=1​
arr[10000][1][4]=1​
​
以1為例1-12,1-10​
程式動作為​
arr[00110][2][2]+=arr[00010][1][1];​
arr[00110][2][0]+=arr[00010][1][1];​
​
DP function 為​
if (last\_num-1 >= 0) arr[bit\_used | (1 << (last\_num - 1))][length + 1][last\_num - 1] += arr[bit\_used][length][last\_num];​
 if (last\_num+1 < 10) arr[bit\_used | (1 << (last\_num + 1))][length + 1][last\_num + 1] += arr[bit\_used][length][last\_num];

接上數字前確認是合法的最後一位數是合法的(0<=num<=9)

填完陣列後，若想求N=2 M=4，將陣列

arr[00011][4][1] //length>=2

arr[00011][4][0]

arr[00011][3][1]

arr[00011][3][0]

arr[00011][2][1]

arr[00011][2][0]

相加，即是答案

**討論：**

(1) 直接暴力窮取每個數都用過的Beautiful Number會TLE

(2)三維陣列不需要完全填完，bit\_used非連續的1不需考慮，如:10101 100010

**程式：**

#include <bits/stdc++.h> using namespace std;

const int MOD = 1e9 + 7; const int MAX\_N = 10; const int MAX\_M = 100; const int BITMASK\_COUNT = 55;

long long BN[1 << MAX\_N][MAX\_M + 1][MAX\_N]; int valid\_bitmasks[BITMASK\_COUNT];

void generate\_valid\_bitmasks() { int index = 0; for (int len = 1; len <= MAX\_N; ++len) { int start = (1 << len) - 1; for (int shift = 0; shift <= MAX\_N - len; ++shift) { valid\_bitmasks[index++] = start << shift; } } }

int main() { ios::sync\_with\_stdio(false); cin.tie(nullptr);

generate\_valid\_bitmasks();

int T;
cin >> T;

while (T--) {
 int N, M;
 cin >> N >> M;

 memset(BN, 0, sizeof(BN));

 for (int i = 1; i < N; ++i) {
 BN[1 << i][1][i] = 1;
 }

 for (int len = 1; len < M; ++len) {
 for (int mask : valid\_bitmasks) {
 for (int last = 0; last < N; ++last) {
 if (!(mask & (1 << last))) continue;
 long long curr = BN[mask][len][last];
 if (curr == 0) continue;

 if (last > 0) {
 int next\_digit = last - 1;
 int next\_mask = mask | (1 << next\_digit);
 BN[next\_mask][len + 1][next\_digit] = (BN[next\_mask][len + 1][next\_digit] + curr) % MOD;
 }
 if (last < N - 1) {
 int next\_digit = last + 1;
 int next\_mask = mask | (1 << next\_digit);
 BN[next\_mask][len + 1][next\_digit] = (BN[next\_mask][len + 1][next\_digit] + curr) % MOD;
 }
 }
 }
 }

 int full\_mask = (1 << N) - 1;
 long long result = 0;
 for (int len = 1; len <= M; ++len) {
 for (int last = 0; last < N; ++last) {
 result = (result + BN[full\_mask][len][last]) % MOD;
 }
 }
 cout << result << '\n';
}

return 0;

}