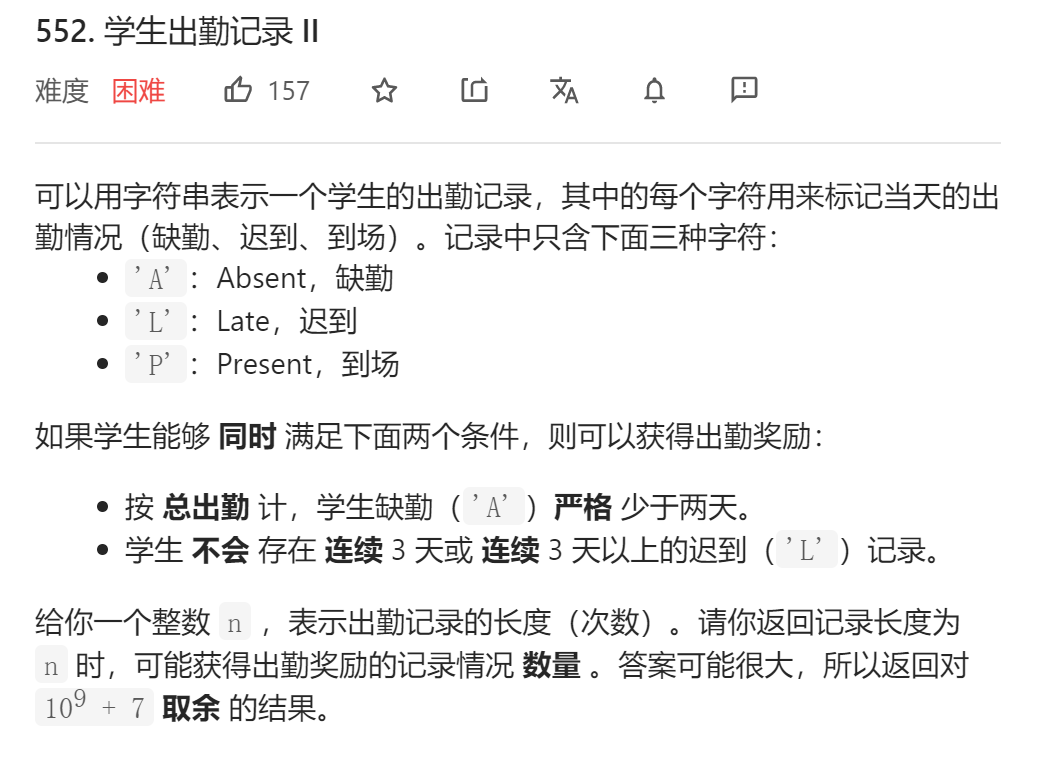
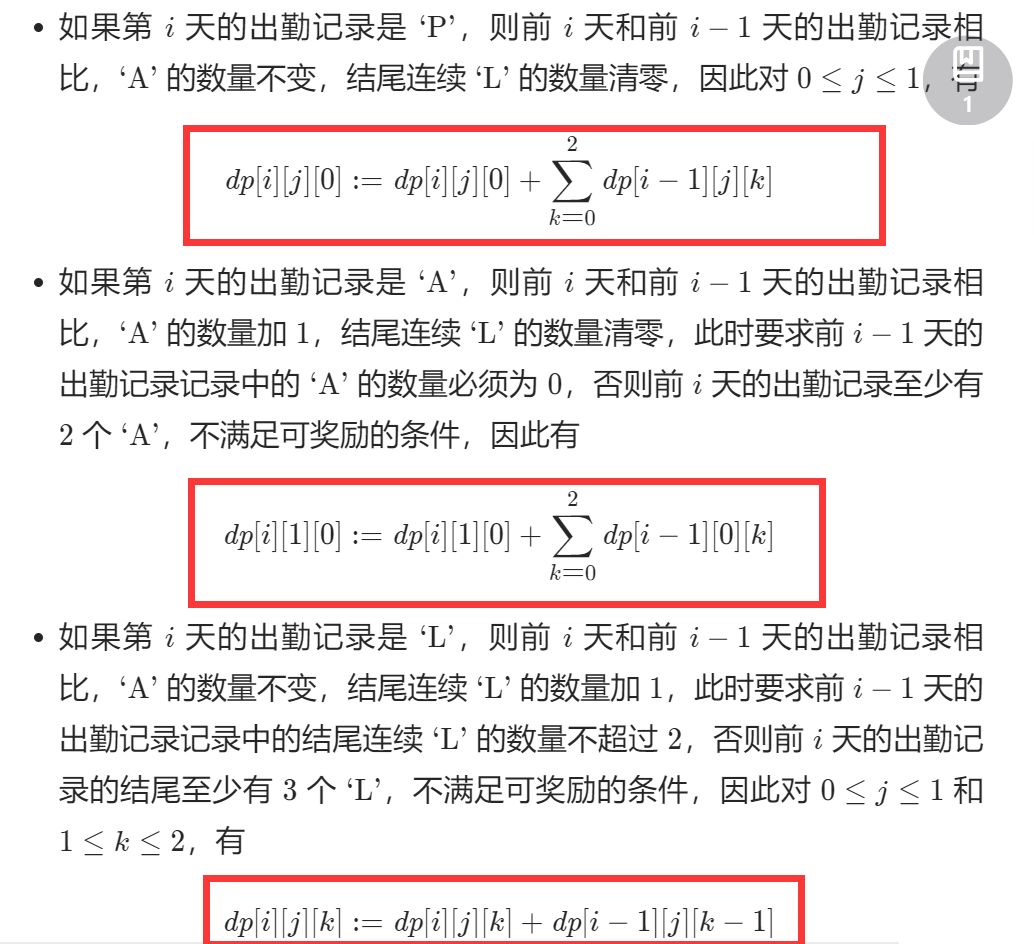
# 552【动态规划加速】【降低维度】【矩阵快速幂】

## 题目描述：

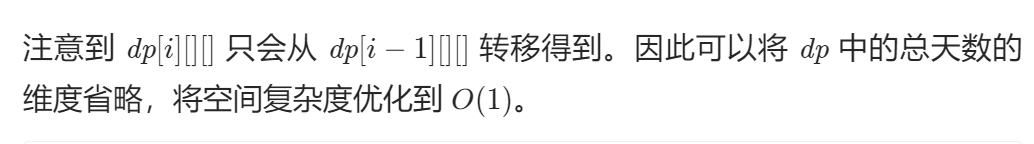


## 初始动态规划：dp[i] [j] [k]



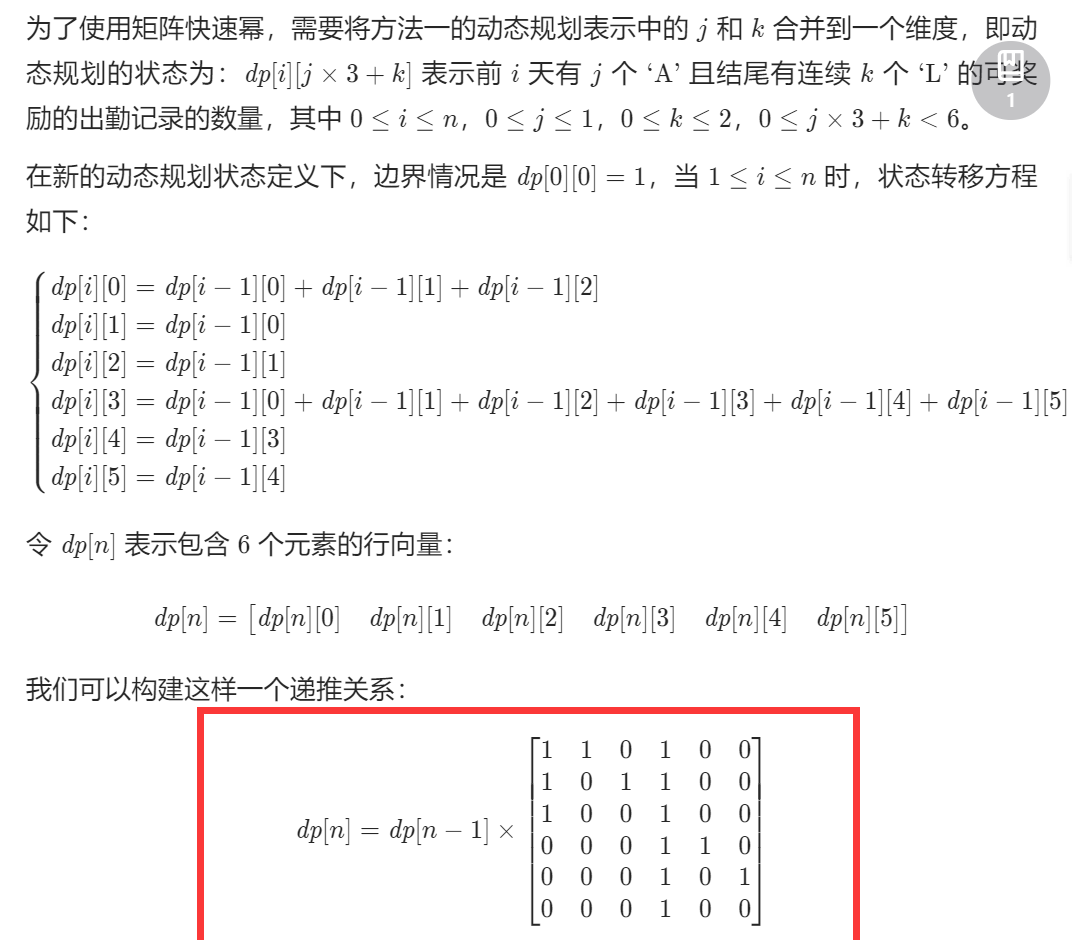
class Solution {  
public:  
 int checkRecord(int n) {  
 //动态规划  
 //dp[i][j][k]表示当字符串长度为i时，缺勤次数为j  
 //连续迟到天数为k的条件下的复合条件的数量  
 vector<vector<vector<int>>>dp(n+1,  
 vector<vector<int>>(2,vector<int>(3,0)));  
 //初始化  
 dp[0][0][0]=1;  
 int mod=1e9+7;  
 //状态转移  
 for(int i=1;i<=n;i++){  
 for(int j=0;j<2;j++){  
 for(int k=0;k<3;k++){  
 if(k==0){  
 dp[i][j][k]+=dp[i-1][j][0];  
 dp[i][j][k]%=mod;  
 dp[i][j][k]+=dp[i-1][j][1];  
 dp[i][j][k]%=mod;  
 dp[i][j][k]+=dp[i-1][j][2];  
 dp[i][j][k]%=mod;  
 if(j==1){  
 dp[i][j][k]+=dp[i-1][j-1][0];  
 dp[i][j][k]%=mod;  
 dp[i][j][k]+=dp[i-1][j-1][1];  
 dp[i][j][k]%=mod;  
 dp[i][j][k]+=dp[i-1][j-1][2];  
 dp[i][j][k]%=mod;  
 }  
 }else if(k>=1){  
 dp[i][j][k]+=dp[i-1][j][k-1];  
 dp[i][j][k]%=mod;  
 }   
 }  
 }  
 }  
 int res=0;  
 for(int j=0;j<2;j++){  
 for(int k=0;k<3;k++){  
 res+=dp[n][j][k];  
 res%=mod;  
 }  
 }  
 return res;  
 }  
};

## 动态规划：初步压缩



class Solution {  
public:  
 static constexpr int MOD = 1'000'000'007;  
  
 int checkRecord(int n) {  
 int dp[2][3]; // A 的数量，结尾连续 L 的数量  
 memset(dp, 0, sizeof(dp));  
 dp[0][0] = 1;  
 for (int i = 1; i <= n; i++) {  
 int dpNew[2][3]; // A 的数量，结尾连续 L 的数量  
 memset(dpNew, 0, sizeof(dpNew));  
 // 以 P 结尾的数量  
 for (int j = 0; j <= 1; j++) {  
 for (int k = 0; k <= 2; k++) {  
 dpNew[j][0] = (dpNew[j][0] + dp[j][k]) % MOD;  
 }  
 }  
 // 以 A 结尾的数量  
 for (int k = 0; k <= 2; k++) {  
 dpNew[1][0] = (dpNew[1][0] + dp[0][k]) % MOD;  
 }  
 // 以 L 结尾的数量  
 for (int j = 0; j <= 1; j++) {  
 for (int k = 1; k <= 2; k++) {  
 dpNew[j][k] = (dpNew[j][k] + dp[j][k - 1]) % MOD;  
 }  
 }  
 memcpy(dp, dpNew, sizeof(dp));  
 }  
 int sum = 0;  
 for (int j = 0; j <= 1; j++) {  
 for (int k = 0; k <= 2; k++) {  
 sum = (sum + dp[j][k]) % MOD;  
 }  
 }  
 return sum;  
 }  
};

## 动态规划：矩阵快速幂



class Solution {  
public:  
 static constexpr int MOD = 1'000'000'007;  
  
 vector<vector<long>> pow(vector<vector<long>> mat, int n) {  
 vector<vector<long>> ret = {{1, 0, 0, 0, 0, 0}};  
 while (n > 0) {  
 if ((n & 1) == 1) {  
 ret = multiply(ret, mat);  
 }  
 n >>= 1;  
 mat = multiply(mat, mat);  
 }  
 return ret;  
 }  
  
 vector<vector<long>> multiply(vector<vector<long>> a, vector<vector<long>> b) {  
 int rows = a.size(), columns = b[0].size(), temp = b.size();  
 vector<vector<long>> c(rows, vector<long>(columns));  
 for (int i = 0; i < rows; i++) {  
 for (int j = 0; j < columns; j++) {  
 for (int k = 0; k < temp; k++) {  
 c[i][j] += a[i][k] \* b[k][j];  
 c[i][j] %= MOD;  
 }  
 }  
 }  
 return c;  
 }  
  
 int checkRecord(int n) {  
 vector<vector<long>> mat = {{1, 1, 0, 1, 0, 0}, {1, 0, 1, 1, 0, 0}, {1, 0, 0, 1, 0, 0}, {0, 0, 0, 1, 1, 0}, {0, 0, 0, 1, 0, 1}, {0, 0, 0, 1, 0, 0}};  
 vector<vector<long>> res = pow(mat, n);  
 long sum = accumulate(res[0].begin(), res[0].end(), 0ll);  
 return (int)(sum % MOD);  
 }  
};  
  
作者：LeetCode-Solution  
链接：https://leetcode-cn.com/problems/student-attendance-record-ii/solution/xue-sheng-chu-qin-ji-lu-ii-by-leetcode-s-kdlm/  
来源：力扣（LeetCode）  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。