I - 矩阵连乘

题目大意：计算A^n

利用同一矩阵相乘的结合律：

$A^{(a_1+a_2+...+a_n)} = A^{a_1}A^{a_2}...A^{a_n}$

$A^{(k_02^0+k_12^1+...+k_n2^n)} = A^{k_02^0}A^{k_12^1}...A^{k_n2^n}$

可以使用快速幂进行计算。

代码：

```
#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

const ll mod = 1e9 + 7;

int t, n;
ll a[128][128], b[128][128], tmp[128][128];

//设置单位阵
void setI(ll a[128][128]) {
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (i == j) {
                a[i][j] = 1;
            } else {
                a[i][j] = 0;
            }
        }
    }
}
//将b矩阵复制到a矩阵中
void mov(ll a[128][128], ll b[128][128]) {
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            a[i][j] = b[i][j];
        }
    }
}
//计算a*b，结果储存在tmp矩阵中
void mul(ll a[128][128], ll b[128][128]) {
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            tmp[i][j] = 0;
            for (int k = 0; k < n; ++k) {
                tmp[i][j] += a[i][k] * b[k][j];
                tmp[i][j] %= mod;
            }
        }
```

```
    }
}

void binpow(ll p) {
    // 先将b置为单位阵
    setI(b);
    while (p) {
        if (p & 1) {
            // 计算A*B，结果储存在TMP中
            mul(b, a);
            // 将TMP的值复制给B矩阵
            mov(b, tmp);
            //相当于b = b * a
        }
        // 相当于a = a * a
        mul(a, a);
        mov(a, tmp);
        p >>= 1;
    }
}
int main() {
    cin.tie(NULL);
    ios::sync_with_stdio(false);
    cin >> t;
    while (t--) {
        cin >> n;
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                cin >> a[i][j];
            }
        }
        binpow(n);
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                cout << b[i][j] << " ";
            }
            cout << endl;
        }
    }
    return 0;
}
```