

若干层楼，若干个鸡蛋，让你算出最少的尝试次数，找到鸡蛋恰好摔不碎的那层楼。

分析：

1. 其状态分别为鸡蛋数 k 和楼层数 N ;
2. 选择就是从哪个楼层扔鸡蛋。

分析递归条件： 当在第 i 层向下扔鸡蛋有如下两种情况：

- 如果鸡蛋没碎，则继续上面楼层搜索 $dp(k, N-i)$, 此时鸡蛋数未变；
- 如果鸡蛋碎了，则向下层开始搜索 $dp(k-1, i-1)$, 此时鸡蛋已经碎了。

递归解法：

```
int dp(int k, int N) {
    if(k == 1) {
        return N;
    }
    if(N == 0) {
        return 0;
    }

    int res = INT_MAX;
    for(int i = 1; i < N; ++i){
        res = std::min(res, std::max(dp(k-1, i-1), dp(k, N-i))+1)
    }

    return res;
}
```

备忘录优化：

```
#include <climits>
#include <vector>

int minegg(int k, int n, std::vector<std::vector<int>>> memo) {
    if (k == 1) {
        return n;
    }
    if (n == 0) {
        return 0;
    }

    if (memo[k][n] != 0) {
        return memo[k][n];
    }
}
```

```
int res = INT_MAX;
for (int i = 1; i <= n; i++) {
    res = std::min(
        res,
        1 + std::max(minegg(k - 1, i - 1, memo), minegg(k, n - i, memo)));
}

memo[k][n] = res;

return memo[k][n];
}

int superEggDrop(int k, int n) {
    std::vector<std::vector<int>> memo =
        std::vector<std::vector<int>>(k + 1, std::vector<int>(n + 1, 0));
    return minegg(k, n, memo);
}
```