{% note info %} **摘要** Title: 291. 蒙德里安的梦想 Tag: 连通性状态压缩dp、dp Memory Limit: 64 MB Time Limit: 1500 ms {% endnote %}

Powered by:**NEFU AB-IN**

Link

@TOC

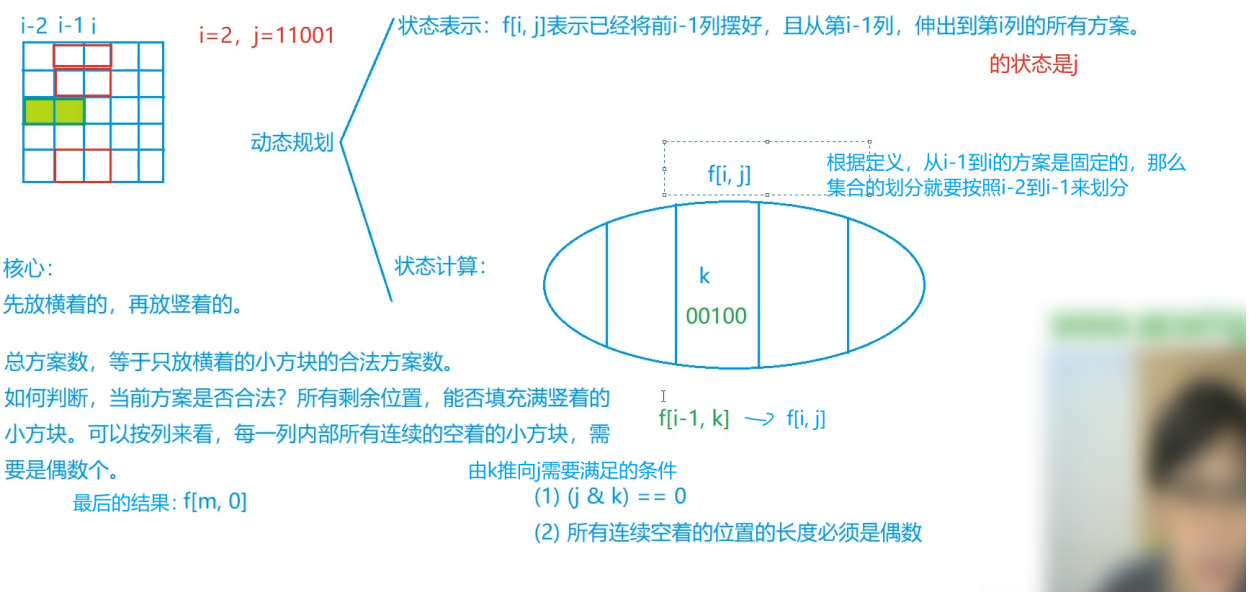
291. 蒙德里安的梦想

- 题意

> 求把 N×M 的棋盘分割成若干个 1×2 的的长方形，有多少种方案。 例如当 N=2，M=4 时，共有 5 种方案。当 N=2，M=3 时，共有 3 种方案。

- 思路

一篇写的很不错的题解，link



注释基本在代码中

- 代码

```python
'''
Author: NEFU AB-IN
Date: 2022-03-13 16:54:54
FilePath: \ACM\Acwing\291.py
LastEditTime: 2022-03-13 17:17:23
'''
N = 13  #列
M = 1 << N  #列的可能的种类
dp = [[0] * M for _ in range(N)]
```

```
st = [0] * M #
state = [[] for _ in range(M)]

while True:
    try:
        n, m = map(int, input().split())
        if n == 0 and m == 0:
            break
        # 子问题1，问n的所有二进制串，标记连续的0长度不为奇数的串
        for i in range(1 << n):  #预处理每一列的合法情况
            cnt = 0
            flag = 1
            for j in range(n):
                if i >> j & 1:
                    if cnt & 1:
                        flag = 0
                        break
                else:
                    cnt += 1
            if cnt & 1: #考虑最后一个串
                flag = 0
            st[i] = flag

        # 子问题2，将n的所有二进制串一一比较，看有多少k能满足j
        for j in range(1 << n):
            state[j] = []
            for k in range(1 << n):
                if j & k == 0 and st[j | k]:  # 保证j,k不起冲突，也要保证k突进来的，对j没有影响
                    state[j].append(k)

        dp = [[0] * M for _ in range(N)]
        dp[0][0] = 1 #即这里第0列只有竖着摆这1种状态。

        for i in range(1, m + 1):
            for j in range(1 << n):
                for k in state[j]:
                    dp[i][j] += dp[i - 1][k]

        print(dp[m][0])
    except:
        break
```