状压 DP

学习状压 DP 之前，请确认你已经完成了 + 动态规划基础 + 部分内容的学习。

（同时建议学习 ==位运算== 部分的内容）

状压 DP 简介

状压 DP 是动态规划的一种，通过将状态压缩为整数来达到优化转移的目的。

例题

> SCOI2005互不侵犯
>
> 在 $N\times N$ 的棋盘里面放 K 个国王，使他们互不攻击，共有多少种摆放方案。国王能攻击到它上下左右，以及左上左下右上右下八个方向上附近的各一个格子，共 8 个格子。

我们用 $f(i,j,l)$ 表示前 i 行，当前状态为 j且已经放置 l 个国王时的方案数。

其中 j 这一维我们用一个二进制整数表示 j 的某个二进制位为 0 代表对应的列不放国王，否则代表对应的列放国王）。

我们需要在刚开始的时候预处理出一行的所有合法状态 $sta(x)$ （排除同一行内两个国王相邻的不合法情况），在转移的时候枚举这些可能状态进行转移。

设当前行的状态为 j上一行的状态为 x可以得到下面的转移方程：

$f(i,j,l)=\sum f(i-1,x,l-sta(x))$

需要注意在转移时排除相邻两行国王互相攻击的不合法情况。

参考代码：

```
#include <algorithm>
#include <iostream>
using namespace std;
long long sta[2005], sit[2005], f[15][2005][105];
int n, k, cnt;
void dfs(int x, int num, int cur) {
  if (cur >= n) {  //有新的合法状态
    sit[++cnt] = x;
    sta[cnt] = num;
    return;
  }
  dfs(x, num, cur + 1);  // cur位置不放国王
  dfs(x + (1 << cur), num + 1,
      cur + 2);  // cur位置放国王，与它相邻的位置不能再放国王
}
```

```
int main() {
  cin >> n >> k;
  dfs(0, 0, 0);  //先预处理一行的所有合法状态
  for (int i = 1; i <= cnt; i++) f[1][i][sta[i]] = 1;
  for (int i = 2; i <= n; i++)
    for (int j = 1; j <= cnt; j++)
      for (int l = 1; l <= cnt; l++) {
        if (sit[j] & sit[l]) continue;
        if ((sit[j] << 1) & sit[l]) continue;
        if (sit[j] & (sit[l] << 1)) continue;
        //排除不合法转移
        for (int p = sta[j]; p <= k; p++) f[i][j][p] += f[i - 1][l][p -
sta[j]];
      }
  long long ans = 0;
  for (int i = 1; i <= cnt; i++) ans += f[n][i][k];  //累加答案
  cout << ans << endl;
  return 0;
}
```

习题

NOI2001 炮兵阵地

USACO06NOV 玉米田 Corn Fields

AHOI2009 中国象棋

九省联考 2018 一双木棋

参考链接

OI Wiki