

# 状压 DP



## 简介

状压 DP 是动态规划的一种，通过将状态集合转化为整数记录在 DP 状态中来实现状态转移的目的。

为了达到更低的时间复杂度，通常需要寻找更低状态数的状态。大部分题目中会利用二元状态，用  $n$  位二进制数表示  $n$  个独立二元状态的情况。

使用状态压缩通常涉及位运算，关于基础位运算详见 [位运算](#) 页面。

## 例题 1

### 「SCOI2005」互不侵犯

在  $N \times N$  的棋盘里面放  $K$  个国王 ( $1 \leq N \leq 9, 1 \leq K \leq N \times N$ )，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，以及左上右下右上左下八个方向上附近的各一个格子，共 8 个格子。

## 解释

设  $f(i, j, l)$  表示前  $i$  行，第  $i$  行的状态为  $j$ ，且棋盘上已经放置  $l$  个国王时的合法方案数。

对于编号为  $j$  的状态，我们用二进制整数  $sit(j)$  表示国王的放置情况， $sit(j)$  的某个二进制位为 0 表示对应位置不放国王，为 1 表示在对应位置上放置国王；用  $sta(j)$  表示该状态的国王个数，即二进制数  $sit(j)$  中 1 的个数。例如，如下图所示的状态可用二进制数 100101 来表示（棋盘左边对应二进制低位），则有  $sit(j) = 100101_{(2)} = 37, sta(j) = 3$ 。



设当前行的状态为  $j$ ，上一行的状态为  $x$ ，可以得到下面的状态转移方程：

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

设上一行的状态编号为  $x$ ，在保证当前行和上一行不冲突的前提下，枚举所有可能的  $x$  进行转移，转移方程：

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

实现

#### 参考代码

```
1  #include <algorithm>
2  #include <iostream>
3  using namespace std;
4  long long sta[2005], sit[2005], f[15][2005][105];
5  int n, k, cnt;
6
7  void dfs(int x, int num, int cur) {
8      if (cur >= n) { // 有新的合法状态
9          sit[++cnt] = x;
10         sta[cnt] = num;
11         return;
12     }
13     dfs(x, num, cur + 1); // cur位置不放国王
14     dfs(x + (1 << cur), num + 1,
15         cur + 2); // cur位置放国王，与它相邻的位置不能再放国王
16 }
17
18 bool compatible(int j, int x) {
19     if (sit[j] & sit[x]) return false;
20     if ((sit[j] << 1) & sit[x]) return false;
21     if (sit[j] & (sit[x] << 1)) return false;
22     return true;
23 }
24
25 int main() {
26     cin >> n >> k;
27     dfs(0, 0, 0); // 先预处理一行的所有合法状态
28     for (int j = 1; j <= cnt; j++) f[1][j][sta[j]] = 1;
29     for (int i = 2; i <= n; i++)
30         for (int j = 1; j <= cnt; j++)
31             for (int x = 1; x <= cnt; x++) {
32                 if (!compatible(j, x)) continue; // 排除不合法转移
33                 for (int l = sta[j]; l <= k; l++) f[i][j][l] += f[i-1]
34 [x][l - sta[j]];
35             }
36     long long ans = 0;
37     for (int i = 1; i <= cnt; i++) ans += f[n][i][k]; // 累加答案
38     cout << ans << endl;
39     return 0;
40 }
```

## 例题 2

 [POI2004] PRZ 

有  $n$  个人需要过桥，第  $i$  的人的重量为  $w_i$ ，过桥用时为  $t_i$ 。这些人过桥时会分成若干组，只有在某一组的所有人全部过桥后，其余的组才能过桥。桥最大承重为  $W$ ，问这些人全部过桥的最短时间。

$100 \leq W \leq 400$ ， $1 \leq n \leq 16$ ， $1 \leq t_i \leq 50$ ， $10 \leq w_i \leq 100$ 。

### 解释

我们用  $S$  表示所有人构成集合的一个子集，设  $t(S)$  表示  $S$  中人的最长过桥时间， $w(S)$  表示  $S$  中所有人的总重量， $f(S)$  表示  $S$  中所有人全部过桥的最短时间，则：

$$\begin{cases} f(\emptyset) = 0, \\ f(S) = \min_{T \subseteq S; w(T) \leq W} \{t(T) + f(S \setminus T)\}. \end{cases}$$

需要注意的是这里不能直接枚举集合再判断是否为子集，而应使用 [子集枚举](#)，从而使时间复杂度为  $O(3^n)$ 。

### 实现

## 参考代码

```
1  #include <iostream>
2  #include <limits>
3  #include <vector>
4  using namespace std;
5
6  int main() {
7      ios::sync_with_stdio(false);
8      cin.tie(nullptr);
9      int W, n;
10     cin >> W >> n;
11     const int S = (1 << n) - 1;
12     vector<int> ts(S + 1), ws(S + 1);
13     for (int j = 0, t, w; j < n; ++j) {
14         cin >> t >> w;
15         for (int i = 0; i <= S; ++i)
16             if (i & (1 << j)) {
17                 ts[i] = max(ts[i], t);
18                 ws[i] += w;
19             }
20     }
21     vector<int> dp(S + 1, numeric_limits<int>::max() / 2);
22     for (int i = 0; i <= S; ++i) {
23         if (ws[i] <= W) dp[i] = ts[i];
24         for (int j = i; j; j = i & (j - 1))
25             if (ws[i ^ j] <= W) dp[i] = min(dp[i], dp[j] + ts[i ^ j]);
26     }
27     cout << dp[S] << '\n';
28     return 0;
29 }
```

## 习题

- 「NOI2001」 炮兵阵地
- 「USACO06NOV」 玉米田 Corn Fields
- 「九省联考 2018」 一双木棋

🔧 本页面最近更新：2025/8/8 20:46:23, [更新历史](#)

✎ 发现错误？想一起完善？ [在 GitHub 上编辑此页！](#)

👤 本页面贡献者： [StudyingFather](#), [Ir1d](#), [H-J-Granger](#), [NachtgeistW](#), [countercurrent-time](#), [Enter-tainer](#), [ouuan](#), [Marcythm](#), [sshwy](#), [AngelKitty](#), [CCXXI](#), [cjsoft](#), [diauweb](#), [Early0v0](#), [ezoixx130](#), [GekkaSaori](#), [Henry-ZHR](#), [HeRaNO](#), [Konano](#), [LovelyBuggies](#), [Makkiy](#), [mgt](#), [minghu6](#), [P-Y-Y](#), [PotassiumWings](#), [SamZhangQingChuan](#), [Suyun514](#), [weiyong1024](#), [chieh2lu2](#), [Chrogeek](#),

GavinZhengOI, Gesrua, hsfzLZH1, iamtwz, kenlig, ksyx, kxccc, Link-cute, lychees, Peanut-Tang, REYwmp, shinzanmono, SukkaW, TianKong-y, Tiphereth-A, Xeonacid, YuJunDongGit, zhb2000

© 本页面的全部内容在 **CC BY-SA 4.0** 和 **SATA** 协议之条款下提供，附加条款亦可能应用