状压DP简介

主讲人：邓哲也

状压DP

状态压缩 DP，顾名思义就是要把状态压缩起来。

比如对于一个 8 * 8 的棋盘，每个位置上可以放一个棋子。

对于某一行我们在第 2 个位置和第 6 个位置放了棋子。

如果用 8 维的状态来表示：f[0][1][0][0][0][1][0][0]，

非常的不便。并且如果棋盘大小变成 9 * 9，那么数组还需

要重新定义。

状压DP

因此我们就有了把一行的状态压缩成一个数字的做法。

一般来说，转化为二进制（每个位置都只有放/不放棋子两种选择）。如果每个位置可以有 3 种状态，那么就用三进制。

对于第 2 个位置和第 6 个位置放了棋子的状态，可以转化为 $2^{2-1} + 2^{6-1} = 2 + 32 = 34$

这样只需要一个大小为 2^8 的一维数组就可以存下所有状态。

这就是状态压缩。

状压DP例题

现在有一个 $n \times m$ 的方格棋盘，和无限的 1×2 的骨牌。

问有多少种方法可以用骨牌铺满棋盘。

$1 \leqslant n, m \leqslant 11$

样例输入：

```
1 2
1 3
1 4
2 2
2 3
2 4
2 11
4 11
0 0
```

样例输出：

```
1
0
1
2
3
5
144
51205
```

状压DP例题

首先如果 $n \times m$ 是奇数就一定拼不出来。

使用状态压缩。

把每一行都压缩成一个二进制。

如果第 i 个位置上被放了骨牌，那么二进制状态中的第 i 位就是 1，否则是 0。

用 f[i][s] 来表示现在填到了第 i 行，状态是 s 的方案数。

答案就是 f[n][(1 << m) - 1]

怎么转移？

状压DP例题

考虑决策——骨牌的放法：横着 或者 竖着。

如果横着：

需要两个连续的空位，并且上一行的这两个位置也得已经被覆盖。

如果竖着：

（a）上一行对应的位置是空的，我们把那个空填上。

（b）上一行对应的位置是被覆盖的，那么我们把这一行的位置设为空，表示下一行的对应位置必须竖放，填上这块空白。

状压DP例题

运用搜索找出所有的可行的转移状态：pre -> now

```
void dfs(int i, int now, int pre){
	if (l > m) return;
	if (l == m){
		++ tot;
		from[tot] = pre;
		to[tot] = now;
		return;
	}
	dfs(i + 2, now << 2 | 3, pre << 2 | 3);
	dfs(i + 1, now << 1, (pre << 1) | 1);
	dfs(i + 1, now << 1 | 1, pre << 1);
}
```

状压DP例题

```
int f[12][1 << 11];

scanf("%d%d", &n, &m);
if (n < m) swap(n, m);
tot = 0;
dfs(0, 0, 0);
memset(f, 0, sizeof(f));
f[0][(1 << m) - 1] = 1;        // 边界条件
for(int i = 0;i < n;i ++)
      for(int j = 1;j <= tot;j ++)
            f[i + 1][to[j]] += f[i][from[j]];
printf("%lld\n", f[n][(1 << m) - 1]);
```

下节课再见