

# 2015天津大学ACM暑期集训队培训讲座

## 状压DP 插头DP 数位DP

---

CS2013 艾宇杰

# 状态压缩DP

## β 状态压缩动态规划：

动态规划的状态有时候比较复杂，不容易表示，需要用一些编码技术，把状态压缩，用简单的方式进行表示。

β 一个典型的做法：利用二进制来表示一个集合，每一位分别代表是否选取对于元素。这样就可以把集合压缩为一个整数，用一个整数来表示集合的子集。

β 因此状态往往是指数级别的，时空复杂度也会是指数级别的，所以能处理的数据范围较小

# 位运算

- β 状态压缩时，由于使用二进制来表示一个集合，所以经常会用到位运算
- β & 按位与， $a \& b$ 就是把a和b的二进制数的每一位相与得到的数，如 $001 \& 011 = 001$
- β | 按位或，同上， $001 | 011 = 011$
- β ^ 异或，相异为1 相同为0， $001 \wedge 011 = 010$
- β ~ 按位非， $\sim 001 = 110$
- β << 左移  $11 < < 1 = 110$ ,  $101 < < 3 = 101000$
- β >> 右移  $11 > > 1 = 1$ ,  $101 > > 3 = 0$  保持符号位

# 位运算的优势

- β 由于位运算是在二进制下进行的，所以他要比普通加减乘除更快
- β 往往代码更精简
- β 左移一位即乘以2，右移一位即整除2
- β  $\&1$  相当于模2
- β xor判断数出现的奇偶次数(toj4119)
- β 树状数组lowbit():  $x \& (-x)$
- β 统计x的二进制表示中1的个数:  

```
while(x){ x &= x-1; cnt++;}
```



# 一些关于集合的运算

- β 用二进制表示集合 $\{0, 1, \dots, n-1\}$ :
- β 空集:  $0$
- β 只含元素 $i$ 的子集 $\{i\}$ :  $1 \ll i$
- β 全集:  $(1 \ll n) - 1$
- β 判断第 $i$ 个元素是否属于集合 $S$ : `if (S >> i & 1)`
- β 向 $S$ 中加入元素 $i$ : `S | 1 << i`
- β 从 $S$ 中去除元素 $i$ : `S & ~(1 << i)`
- β 求并集 $S \cup T$ : `S | T`
- β 求交集 $S \cap T$ : `S & T`
- β 枚举所有子集 `for (int S = 0; S < 1 << n; S++) {}`
- β 枚举某个集合 $S$ 的子集: `for (int i = S; i; i = (i-1) & S)`
- β 也可以从 $0$ 开始: `for (int i = 0; i != S; i = (i-S) & S)`

# 注意事项

使用位运算时  
要注意优先级

优先顺序：  
高  
↓  
低

按位与、按位  
或、异或的优  
级比等于低，  
使用时必须加  
括号

优先级	符号	描述	运算顺序
3	! ~	逻辑非，按位非	从右到左
5	* / %	乘，除，模	从左到右
6	+ -	加，减	从左到右
7	<< >>	左移，右移	从左到右
8	< <= > >=	大于(等于),小于(等于)	从左到右
9	== !=	等于，不等于	从左到右
10	&	按位与	从左到右
11	^	异或	从左到右
12		按位或	从左到右
13	&&	逻辑与	从左到右
14	//	逻辑或	从左到右
15	=	赋值符	从右到左

# 经典问题：旅行商问题

- β  $n$ 个点带权有向图，求边权值和最小的汉密尔顿回路（除了起点，走过每个点恰好一次）
- β  $n \leq 16$

# 经典问题：旅行商问题

- β  $n$ 个点带权有向图，求边权值和最小的汉密尔顿回路（除了起点，走过每个点恰好一次）
- β  $n \leq 16$  (重要条件,状态压缩的标志)
- β 思路：
  - β 起点无关：回路，所以起点任意
  - β 路径无关：不关心之前怎么走的，只关心走过哪些点，现在在哪个点



# 经典问题：旅行商问题

- β  $n$ 个点带权有向图，求边权值和最小的汉密尔顿回路（除了起点，走过每个点恰好一次）
- β  $n \leq 16$  (重要条件,状态压缩的标志)
- β 状态：
  - β 定义 $dp[i][j]$ 为在访问了集合 $j$ 中的点的情况下最后在 $i$ 点停留的最小边权和。
  - β 集合用一个数 $j$ 来表示， $j$ 的2进制形式如0100101代表访问过了1，4，6这几个点。

# 经典问题：旅行商问题

- β 状态：
- β 定义  $dp[i][j]$  为在访问了集合  $j$  中的点的情况下最后在  $i$  点停留的最小边权和。
- β 转移：
- β 枚举不在  $j$  中的点  $k$ ，若有边  $(i, k)$
- β  $dp[k][j \cup \{k\}] =$
- β  $\min(dp[k][j \cup \{k\}], dp[i][j] + w[i][k])$
- β 初始化：  $dp[0][0] = 0$ , others = inf
- β 目标：  $dp[0][(1 \ll n) - 1]$

# 棋盘模型：炮兵阵地

- β 在 $N \times M$ 的网格地图上部署炮兵部队。
- β 一个 $N \times M$ 的地图由 $N$ 行 $M$ 列组成
- β 地图的每一格可能是山地(用“H”表示)，也可能是平原(用“P”表示)。
- β 在每一格平原地形上最多可以布置一支炮兵部队，山地上不能够部署炮兵部队。
- β ( $N \leq 100, M \leq 10$ )

# 棋盘模型：炮兵阵地

- β 一支炮兵部队在地图上的攻击范围如图中黑色区域所示：沿横向左右各两格，沿纵向上下各两格。图上其它白色网格均攻击不到。从图上可见炮兵的攻击范围不受地形的影响。

P♞	P♞	H♞	P♞	H♞	H♞	P♞	P♞	♞
P♞	H♞	P♞	H♞	P♞	H♞	P♞	P♞	♞
P♞	P♞	P♞	H♞	H♞	H♞	P♞	H♞	♞
H♞	P♞	H♞	P♞	P♞	P♞	P♞	H♞	♞
H♞	P♞	P♞	P♞	P♞	H♞	P♞	H♞	♞
H♞	P♞	P♞	H♞	P♞	H♞	H♞	P♞	♞
H♞	H♞	H♞	P♞	P♞	P♞	P♞	H♞	♞



# 棋盘模型：炮兵阵地

- β 现在，将军们规划如何部署炮兵部队，在防止误伤的前提下（保证任何两支炮兵部队之间不能互相攻击，即任何一支炮兵部队都不在其他支炮兵部队的攻击范围内），在整个地图区域内最多能够摆放多少我军的炮兵部队。
- β  $(N \leq 100, M \leq 10)$

# 解题思路

- β 首先注意到 $n$ 虽然很大， $m$ 却最多只有10
- β 启发我们将每行进行状态压缩
- β 1代表放炮兵，0代表不放

# 解题思路

- β 首先注意到 $n$ 虽然很大， $m$ 却最多只有10
- β 启发我们将每行进行状态压缩
- β 1代表放炮兵，0代表不放
- β 假设我们在第 $i$ 行，前 $i-1$ 行都放好了，由于炮兵能往上两格，那么我们需要关心 $i-1$ 行和 $i-2$ 行是如何放的

# 解题思路

β 状态：

β  $dp[i][j][k]$ 代表第*i*行，*i*行状态为*j*，*i*-1行状态为*k*的最多炮兵数量

β 转移：

β 枚举当前行的状态，判断是否能从前两行转移过来（自身合法、和地形是否冲突、和前两行是否冲突）



# 解题思路

β 空间复杂度为

$$n * 2^m * 2^m = 100 * 2^{10} * 2^{10}$$

β 存不下，跑得慢，复杂度太高！

β 考虑优化：

β 注意到炮兵横向也有很大的攻击范围，所以一行至多能放3个炮兵

β 无用状态很多！

# 解题思路

- β 写一个小程序来测试，发现一行的状态至多只有60种左右
- β 于是状态数只有约 $100 \times 60 \times 60$ 个
- β 状态转移方程至多为60次
- β 注：有时写一个小程序来测试，寻找规律或者找出某个数的上限等是很有用的

# 具体实现

---

- β 小程序怎么写，dfs?
- β 转移怎么写，将整数解码到数组再循环判断?

# 位运算的威力

β 小程序的正确姿势（转移也类似，通过&判断冲突）：

```
β void init()
β {
β     size = 0;
β     int maxn = 1 << m;
β     for (int i = 0; i < maxn; i++){
β         if (!(i&(i<<2)) && !(i&(i<<1))){//是否合法
β             st[size] = i;
β             ct[size] = 0;
β             int tmp = i;
β             while(tmp) tmp = tmp&(tmp-1), ct[size]++;//统计1个数
β             size++;
β         }
β     }
β }
```



# 习题

---

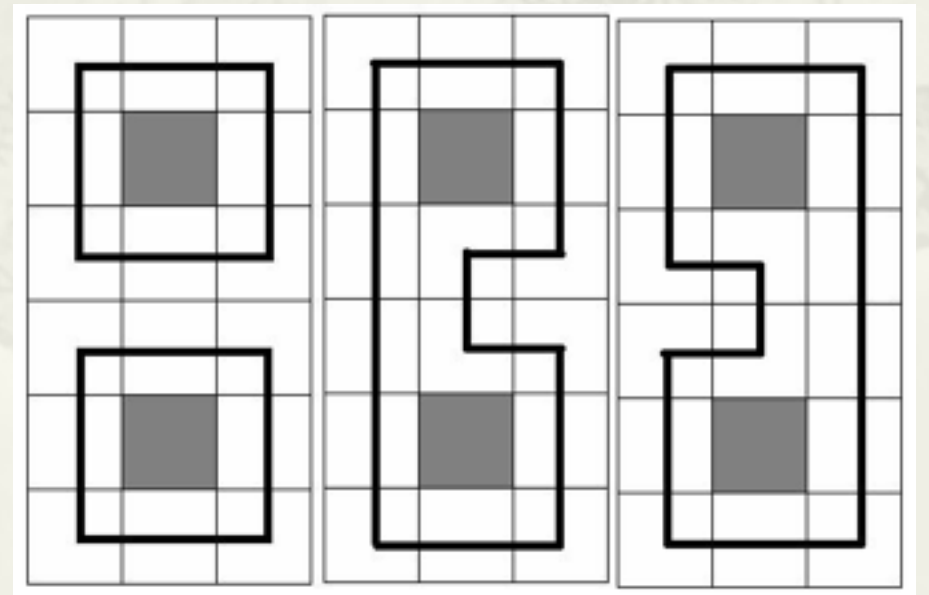
- β HDU 3001 Travelling (读题, 可走两次)
- β POJ 1185 炮兵阵地
- β POJ 3254 Corn Fields
- β POJ 3311 Hie with the Pie
- β POJ 2411 Mondriaan's Dream

# 插头DP

- β 插头dp全称为基于连通性状态压缩的动态规划问题
- β 状态中需要记录元素的连通情况（有的像例1，不记录连通情况，也被统称为插头dp）
- β 通常为在一个矩形范围内(棋盘)的状态压缩dp问题（也不全是）
- β 插头dp常用于处理在矩形范围内绘制各种各样的图形的问题

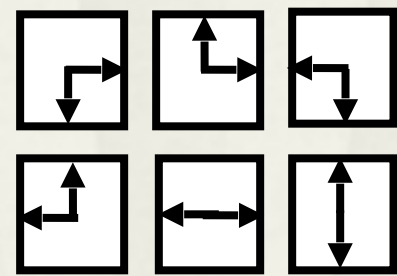
# Eat the Trees

- 屠夫作为DOTA里的一个前期英雄，在后期的作用并不强大，于是他的队友派给他一个小任务，吃树~
- 在一个 $m*n$ 的地图中，屠夫要吃掉所有的树。
- 屠夫通过走一个封闭回路来吃掉路上所有的树。
- 有些地方是没有树的，曾经被吃过或者没有树的地方是不可以路过的
- 屠夫可以走许多条回路
- 问屠夫一共有多少种吃法？
- $1 \leq n, m \leq 11$



# Eat the Trees

- β 解题思路：
- β 所有非障碍格子经过且经过一次，我们发现每格只有6种情况：



- β 还是考虑当前行，如果之前行都放好了，那么我们只关心上一行的状态。
- β 而上一行水平方向的状态我们也不关心，只需记录是否有往下走。0代表无下箭头，1代表有下箭头。



# Eat the Trees

- β 状态:
- β  $dp[i][j]$  表示到了第  $i$  行,  $j$  存放第  $i$  行各格是否有往下的箭头, 的方案数
- β 转移:
- β 对于每个  $dp[i][j]$ , 枚举所有  $dp[i-1][j']$  ?
- β 虽然上一行只要考虑有没有往下, 但当前行还要考虑左右, 障碍的问题, 有  $O(6^m)$

# Eat the Trees

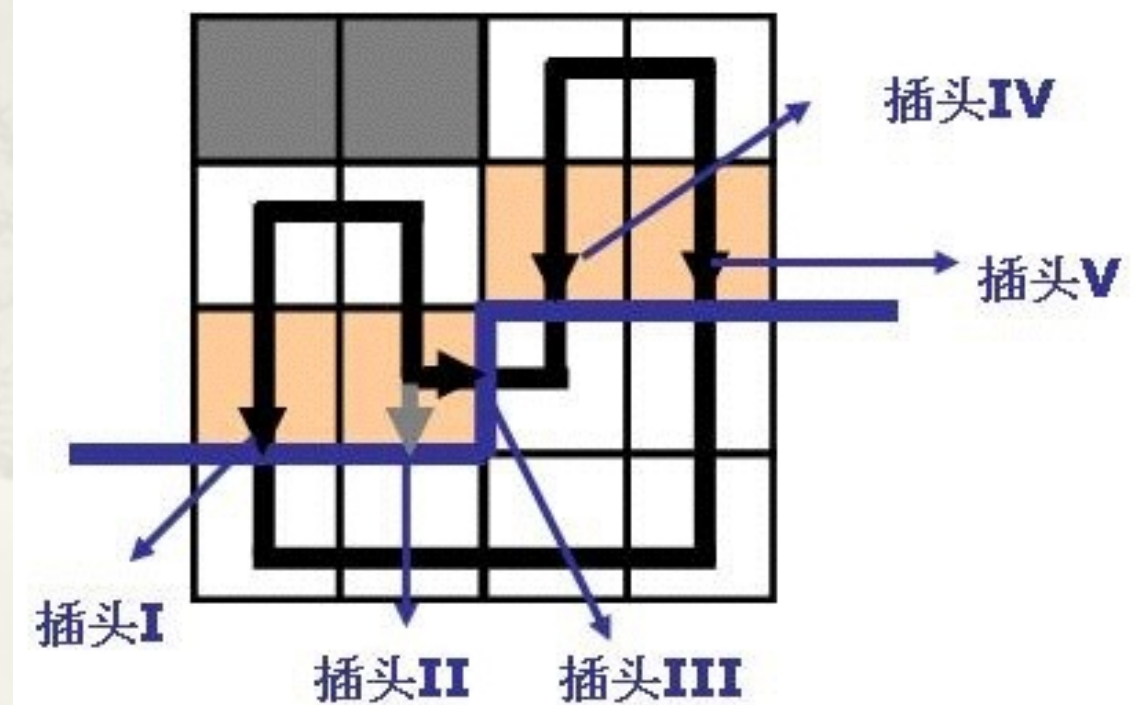
- β 状态：
- β  $dp[i][j]$  表示到了第  $i$  行， $j$  存放第  $i$  行各格是否有往下的箭头，的方案数
- β 转移：
- β 我们发现确定了每行的第一格，以及上一行的情况，之后该行每格最多两种情况，都可以递推得到，即  $O(n * 2^m * 2^m)$
- β 对于每个  $dp[i][j]$ ，递推所能转移到的所有状态即可

# Eat the Trees

启发：  
通过左边和上边知道当前格的状态，尝试按格转移

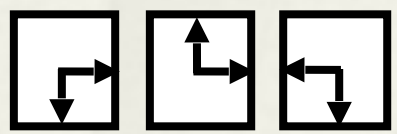
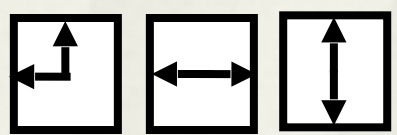
# Eat the Trees

- 考虑  $dp[i][j][k]$ ，表示在  $(i,j)$  位置，上半行状态为  $k$  时的方案数
- 1 代表有连接点，0 代表没有
- $k$  的长度为  $j$  的最大值 + 1
- 如右图， $k$  为 10111





# Eat the Trees

- β 这样 $dp[i][j][k]$ 可由 $dp[i][j-1]$ 推来 ( $j=0$ 时由 $dp[i-1][m]$ 推来)  

- β 每个节点只有6种情况  

- β 每个点只用考虑自己上方和左方有没有插头即可。如果有，就接上。没有，就不接
- β 复杂度  $O(nm \cdot 2^{m+1})$ ，比按行转移优秀
- β 因为可以走多条回路，实际上不需要记录连通性，是广义上的插头dp

转向另一个ppt。。

---



# 习题

---

- β HDU 1693 Eat the Trees
- β URAL 1519 Formula 1

# 数位DP

- β 解决和整数有关的计数问题。
- β 问题通常是询问一段区间 $[L, R]$ 内的数满足某种性质的有多少。
- β dp的本意是带有决策的，不过现在在很多递推和计数的做法都被统称为dp了



# BOMB

β 题意：问从1到N，有多少个数用十进制表示，含有49这个子串。 $(1 \leq N \leq 2^{63}-1)$

# BOMB

- β 解题思路：
- β 数很大，按位考虑
- β 首先通过递推，我们可以预处理得到dp数组
- β  $dp[i][0]$ 表示长度为i的数串，含49的个数。
- β  $dp[i][1]$ 表示长度为i的数串，不含49，且第一位为9的个数。
- β  $dp[i][2]$ 表示长度为i的数串，不含49，且第一位不为9的个数。
- β 以上 $dp[][]$ 均包含有前缀0的情况

# BOMB

- β 解题思路：
- β 再考虑对于给定的N，怎么求解。
- β 关键在于不重不漏，数位dp问题一般有如下方法：
- β 从高位往下枚举。比当前枚举位更高的位我们都取最大值（即n对应位的值）。当前位最大值为x，我们就算填 $0..x-1$ 的方案。
- β 当前到了第i位，有49的无非几种情况：
- β 1.高位已有49
- β 2.当前位取4，下一位为9
- β 3.低位有49

# BOMB

β 解题思路：

β 1.高位已有49

β 2.当前位取4，下一位为9

β 3.低位有49

β 情况1：如果n的第i位和第i-1位组成了49，那么之后都是含49的，不必再枚举，加上就可以。

β 情况2：如果n的当前位大于4，我们可以填4，下一位填9(且之后不填49，否则和情况3重复)。

β 情况3：  $x * dp[i][0]$ ，即当前位填0..x-1乘以之后位含49的方案。



# BOMB

β 感觉很繁琐。。换一种方式：

```
β ll dfs(int pos, bool has49, bool is4, bool bound){  
β     if (pos == 0) return has49;  
β     int lim = bound? d[pos]: 9;  
β     ll ret = 0;  
β     for (int i = 0; i <= lim; i++){  
β         ret += dfs(pos-1, has49||(is4&& i==9), i==4, bound&& i==lim);  
β     }  
β     return ret;  
β }
```

β 可以看出这就是枚举==。。

# BOMB

- β 但是!
- β 注意到bound为false时，函数求解结果是固定的：
- β 大量重复的子问题!
- β 记忆化搜索!

# BOMB

```
℞ ll dfs(int pos, bool has49, bool is4, bool bound){
℞     if (pos == 0) return has49;
℞     int st = 0;
℞     if (has49) st = 2;
℞     else if (is4) st = 1;
℞     if (!bound && dp[pos][st] != -1) return dp[pos][st]; //记忆化搜索
℞     int lim = bound? d[pos]: 9;
℞     ll ret = 0;
℞     for (int i = 0; i <= lim; i++){
℞         ret += dfs(pos-1, has49||(is4&&i==9), i==4, bound&&i==lim);
℞     }
℞     if (!bound) dp[pos][st] = ret;
℞     return ret;
℞ }
```

# BOMB

- β 实际上记忆化的dp[]数组和前面递推用的dp[]数组是同一个东西
- β 记忆化搜索 vs 从高位向低位递推
- β 个人感觉记忆化搜索更好想好写一些
- β 根据具体题目选择用哪个实现



# 不要62

β 题意：问 $[L, R]$ 中，有多少个数用十进制表示，不含4，不含62。

# 不要62

- β 思路：
- β  $\text{Solve}(x)$ 统计 $[1, x]$ 的合法数个数，
- β 答案即 $\text{Solve}(R) - \text{Solve}(L-1)$
- β  $\text{Solve}()$ 的实现类似上题

# XHXJ's LIS

β 题意：问 $[L, R]$ 中，有多少个数满足：将其看成一个序列，LIS长度为 $K$ （严格递增）。

# XHXJ's LIS

- β 解题思路：
- β 先考虑LIS怎么做。想到 $O(n \log n)$ 解法。
- β 但怎么存储dp[]数组呢？ $O(10^{10})$ ？
- β 虽然可以用long long压缩，但是无法记忆化，空间复杂度太高



# XHXJ's LIS

- β 解题思路：
- β 先考虑LIS怎么做。想到 $O(n\log n)$ 解法。
- β  $dp[]$ 是单调递增的，且元素只在0-9，题目又是严格递增，只要记录每个数是否出现。
- β 即 $O(2^{10})$
- β 不太好用递推实现
- β 记忆化搜索分分钟就写好了

# 习题

---

- β HDU 3555 BOMB
- β HDU 2089 不要62
- β HYSBZ 1026 Windy数
- β POJ 3252 Round Numbers
- β HDU 4352 XHXJ's LIS