



# 问求算法设计与实践

## 07

### 动态规划

康望程



# DP 状态设计

- 回顾0/1背包问题
- 有 $N$ 件重 $W_i$ 值 $V_i$ 的物品，装进容量 $W$ 的包
- 要求价值最大

# 0/1 背包问题1

- 我们用 $F[i]$ 表示前 $i$ 个物品能取得的最大价值
- $F[i]$ 从 $F[j]$ 转移来
- 但我们如何知道第 $i$ 件物品能放下？
- 状态设计与当前背包容量要有关
- 才能完整表示一个状态
- 注意，我们是考虑到不能转移才考虑状态设计，而有时候转移看上去也没有问题，而状态设计还是过于简化

# 0/1 背包问题2

- 我们用 $F[i]$ 表示每个物品的选取情况
- 即 $i$ 为二进制压缩的数
- 每次考虑一个没放过的物品能否放入
- 当前背包容量可以有已经选取的物品得知
- 复杂度为 $O(2^N * N)$
- 但是这是一个正确的算法
- 存在冗余

# 0/1 背包问题3

- 考虑去除冗余，思考问题的本质
- 我们记录了每个物品是否选取，从此来获知当前背包容量
- 我们可以直接在状态中表示背包容量
- 可以一个一个考虑物品
- $F[i][j]$ 表示前 $i$ 个物品，当前容量为 $j$ 的最优值

# 小结

- $F[i]$
- 过于简化，本质不同的状态被放进同一节点
- $F[i]$ 状态压缩
- 过于冗余，存在可以合并的状态(如两种选取方案的当前背包容量相等)
- $F[i][j]$
- 物品的放入只与当前背包容量有关



# NOIP 2011 P2 乌龟棋

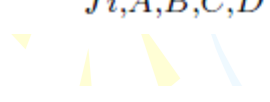
- N个格子，每个格子有一个分数
  - M张4种卡片，分别可前进1，2，3，4格
  - 总分数为各经过格子分数和
  - 不同的卡片使用顺序的分数不一样
  - 要求能获得的最大分数
  - $N \leq 350, M \leq 120$
- 
- 同一种卡片完全一样，所以没必要分开




## 分析

算法一：枚举当前使用的卡片进行搜索。总时间复杂度 $O(M!)$ ,预计得分：30~40分。

算法二：可以发现每到达一个格子的状态与之前使用了每种卡片多少张是有关系的。所以我们需要在DP的状态表示中加入4中卡片的使用量，即用 $f_{i,A,B,C,D}$ 表示到达地 $i$ 格，四种卡片各用 $A,B,C,D$ 张，所获得的最大分数。状态转移方程为：


$$f_{i,A,B,C,D} = \max\{f_{i-1,A-1,B,C,D}, f_{i-2,A,B-1,C,D}, f_{i-3,A,B,C-1,D}, f_{i-4,A,B,C,D-1}\} + sum_i$$

其中转移要保证当前卡片不等于0。另外，这样子做会超空间，由于每一个状态至于前面4个状态有关系，所以任何时刻我们只要保存5个状态即可，处理时可以用 $i \bmod 4$ 来代替下标 $i$ 。时间复杂度为 $O(N * m_A * m_B * m_C * m_D)$ 。预计得分：50。








# 分析

算法三：由于直接DP会超时，所以必须优化，而由于此题状态转移已经是 $O(1)$ 的了，所以只能优化状态数。

这里有两种优化方法，其思想均是降维。一种是将第5维 $D$ 去掉，因为知道了当前所在格，三种卡片的用量，可以计算出第4种卡片用掉的数量。时间复杂度 $O(N * m_A * m_B * m_C)$ 。

另一种是将第一维去掉，与上同理，知道四种卡片用量，可计算出当前所在格。时间复杂度 $O(m_A * m_B * m_C * m_D)$

从本题数据范围来看，第二种优化较之第一种更好，但无论哪一种优化，预计得分均为100分。





# “车”放置问题

- 有一个 $N \times M$ 的空白盘
- 在其中放任意数量的“车”
- 要求“车”两两不攻击(不同行同列)
- 求方案数  $\text{MOD } P$

# 分析

- 按照一般状态压缩思路
- $F[i][j]$ 表示前 $i$ 行，每一列是否有车的状态为 $j$ 的方案数量
- 状态转移为当前行放或不放
- 如果放，枚举每一可以放的列进行转移
- 时间复杂度 $O(M * N * 2^M)$
- 如果 $N, M \leq 1000$ 。。。

# 分析

- 效率低下是因为存在冗余
- 考虑 $F[3][(10101)_2]$ 和 $F[3][(00111)_2]$
- 本质是一样的
- 我们只需知道还有多少列可以放，不需要知道是哪些列
- $F[i][j]$ 表示前 $i$ 行还有 $j$ 行可以放
- $F[i+1][j] = F[i][j] + F[i][j+1] * (j+1)$
- 可见对问题透彻的分析是得出最有效规划方式的前提

# 滑雪问题

- 有一个滑雪场，我们用 $N \times N$ 的矩阵来表示他各处的高度
- Oliver想要从一个点开始滑雪，每次滑向周围四个格子中的一个，滑尽可能长的距离
- 当然，一条滑雪路径的高度必须是单调下降的
- 请给出最长的一条滑雪路径
- $N \leq 1000$

# 分析

- $F[i][j] = \max(F[i-1][j], F[i+1][j], F[i][j-1], F[i][j+1]) + 1$
- 记忆化搜索！避免重复计算同一状态
- 复杂度  $O(N^2 * 4)$

# 分析

- 将 $N^2$ 个数从小到大排序
- 从前往后依次求F值
- 记录每个点的4个邻居在新序列的下标
- 每次对于当前点，看四个下标是否在自己前面，如果是则尝试更新
- $O(N^2 \log N + N^2 * 4)$