动态规划选讲

蒋炎岩 (jyy@nju.edu.cn)

南京大学 计算机科学与技术系 计算机软件研究所

个人简介

蒋炎岩

- 南京大学计算机科学与技术系 临时工
- 南京大学ICPC集训队教练
- 研究方向：软件分析/测试/合成

- JSOI老队员，ICPC World Finalist (2009)
- JSOI/相关竞赛命题 (2009至今)
- JSOI技术组 (2013至今)

欢迎报考南京大学！

南大特色的系统主线课程 (拔尖计划)

- 硬 ⛢ 核编程
- 计算机系统基础 —— x86全系统模拟器
- 操作系统 —— 多处理器操作系统

中国最好的理论计算机科学研究组

- 理论研究：近似算法、复杂性、量子计算……
- 你们可能见过(但读不下去)的STOC/FOCS/SODA论文
- (每年都举办Theory Day)
  - 欢迎大家来现场体验

2019年南京大学一流学科专题营

点我查看详情；报名地址：http://acm.nju.edu.cn/camp

- 门槛：NOIP提高组一等奖；限100人，评特一二三等奖

本讲概述

> 做DP题的体验从来都不太好吧
>
> - 简单(套路)的都会做
> - 困难的做不出

试着分析一下DP题应该怎么解

热身

$O(n \log n)$ LIS

让我们先把你学过的套路都忘掉

- 回到你第一次(比如小学五年级)学LIS的时候
- LIS都不会做，$O(n \log n)$ LIS怎么理解？？？

但那个时候我会暴搜啊！！！

```
def is_increasing(s):
  return all(s[i] < s[i+1] for i in range(len(s)-1))
T = [ [] ]
for a in A:
  T = T + [(seq + [a]) for seq in T]
LIS = max(len(seq) for seq in T if is_increasing(seq))
```

优化搜索空间

考虑`[1, 4, 2, 3, 5]`:

```
#1: [], [1]
#2: [],  [1]        (1)
    [4], [1, 4]     (4)
#3: [],  [1],       (1)
    [4], [1, 4],    (4)
    [2], [1, 2]     (2)
#4: [], [1], [4], [1, 4], [2], [1, 2],
    [3], [1, 3], [2, 3], [1, 2, 3]   (3)
#5: [], [1], [4], [1, 4], [2], [1, 2],
    [3], [1, 3], [2, 3], [1, 2, 3],
    [5], [1, 5], [4, 5], [1, 4, 5], [2, 5], [1, 2, 5], [3, 5], [1, 3, 5], [2, 3, 5], [1, 2, 3, 5]
```

对LIS来说有太多东西不必要：总是从`T`里找一个序列，在后面插入

- 同样长度的，保留数字最小的那个就行
- 以同一个数字结尾的，保留最长的一个就行

人人都会 $O(n\log n)$ LIS

但你有没有在看算法之前就想出来?

- 这是非常与众不同的
  - 这才是DP的本源

在搜索过程中:

- $\ell = 0$，最小结尾是 $-\infty$
- $\ell = 1$，最小结尾是1
- $\ell = 2$，最小结尾是4
- $\ell = 3$，最小结尾是9
- $\ell = 4$，最小结尾是10 (一定是递增的)
  - 因此如果下一个数字是5，会发生什么?

Tree Editing Distance

给两棵有根树，可以插入/删除节点， 问最少做多少操作，可以把一棵树变成另一棵 $(n \leq 500)$

Tree Editing Distance

给两棵有根树，可以插入/删除节点， 问最少做多少操作，可以把一棵树变成另一棵 ($n \leq 500$)

- 简单：$dp_{i,j}$ 表示 $T_i \to T_j$ 的编辑距离
- 转移的时候，做个最小权匹配就好了 (这做毛线啊)

Tree Editing Distance

给两棵有根树，可以插入/删除节点， 问最少做多少操作，可以把一棵树变成另一棵 $(n \le 500)$

- 简单：$dp_{i,j}$ 表示 $T_i \to T_j$ 的编辑距离
- 转移的时候，做个最小权匹配就好了 (这做毛线啊)

分支定界

- 如果$LB(dp_{i,j}) + LB(others) \ge cur$，那么 $dp_{i,j}$ 在本轮不用计算
  - 比如：$dp_{i,j} \ge \left| |T_i| - |T_j| \right|$
  - 其实我们在搜索啊——先找那些有可能匹配的试试，迅速得到 $dp_{i,j}$ 的上下界
  - 最后还可以耍赖皮：提前终止

来自NOI的DP题

这些题大家都做过……

但为什么每年到了NOI现场就做不出……呢?

NOI2015 寿司晚宴

把 $2, 3, 4, \dots n$ 分给两个人 (有些数字可以不选)

- 问有多少方案，两个人分到数字两两互质
- $n \leq 500$

看过题解，你就觉得这题简单了

(当然也是这几年最简单的题之一了)

- 按顺序把数字分给两个人之一
- $X = \{4, 6, 8\}$ 和 $X = \{2, 3\}$ 对后续搜索是等价的
- 只需要记住质因子就可以了
  - $f_{\{2,5\},\{3\}}$ 表示分别拥有因子 $2, 5$ 和 3 的方案数

但质数还是挺多的……

一些分析

对于 $x = p_1 \cdot p_2 \dots \cdot p_k$

- 如果有大质数，比如 2×131
- 你会发现2有没有，之前已经决定了
    - 你需要知道的是131有没有
    - 于是可以把131, 262, 393放在一起考虑

一个新的dp

- 已知A, B分别拥有一些小因子(因此有些数字不能放入)
- $f_{i,0/1,0/1}$ 表示放完第 i 个数字后，分别是否拥有大因子的方案数
    - 简单类型，你都不需要思考
- 大因子的问题是独立的：(131, 262, 393; 149, 298, 447)

你记住了啥？

只需要记住不超过 $\sqrt{n}$ 的因子，这题就搞定了

- 怎么推出这个性质的？
  - 换句话说，是怎么保证你在NOI赛场上遇到类似难度的题，你能想得出来？

你记住了啥？

只需要记住不超过 $\sqrt{n}$ 的因子，这题就搞定了

- 怎么推出这个性质的?
  - 换句话说，是怎么保证你在NOI赛场上遇到类似难度的题，你能想得出来?

核心性质：已知2, 3因子归属的前提下：

- $f(\{131, 262, 149, 298\}) = f(\{131, 262\}) \times f(\{149, 298\})$
- 找到了divide-and-conquer的入手点
  - NOI系列的题，基本最可靠的办法就是从观察特例出发……

NOI2016 国王饮水记

有若干水箱，每次可以选择若干水箱执行“均分”操作

- 问如何“均分” k次，使水箱1的水量最大
- $n \leq 8000$，保留3000位小数 (???)

猜结论吧

(不妨假设水箱1的水量是最小的)

暴力搜索，把所有最优解都打印出来，先把不可做变成做

- 所有的水箱都会用上
    - 不用上的用上能提高平均值
- 一定是用排序以后连续的一段
    - 交换论证 (套路)

有这两个性质，这肯定是个DP题了

- 不确定对不对？没事啊，对拍就完了

套路DP？

$$f_{i,j} = \max_{1 \le k < i} \frac{f_{k,j-1} + S_i - S_k}{i - k + 1}$$

- 我靠，看起来就是个套路DP啊

套路DP？

$$f_{i,j}=\max_{1\le k<i}\frac{f_{k,j-1}+S_i-S_k}{i-k+1}$$

- 我靠，看起来就是个套路DP啊

$$f'_i=\max_{1\le j\le i}\frac{S_i-(S_j-f_j)}{i-(j-1)}$$

- 这个式子有点熟悉啊：$\max_{(x,y)\in f}\frac{S_i-y}{i-x}$
- 哦斜率，还有决策单调性……
- 直觉：误差没那么大吧，弄个double试试？→ 90+分，我靠

Hmmm….

当然了，就算用上三分/决策单调性，还是TLE的

- 不过每一轮DP都算出了很多的值
- 如何利用已经算出的值呢?
  - 标准答案：算若干轮，这样每个点都有一些潜在的最优决策；剩下不够 k 的用1来填！(看起来好像骗分……)

关于决策：从抛硬币开始

抛100次独立的硬币(1/2)，你预期观察到多少次正面朝上?

- 大约50次？51/52都合理，那30合理吗?

关于决策：从抛硬币开始

抛100次独立的硬币(1/2)，你预期观察到多少次正面朝上?

- 大约50次？51/52都合理，那30合理吗?

随机变量$X_i = \sum_{1\le i\le n} x_i$； $\mu = E[X] = \sum_{1\le i\le n} p_i$

- $Pr[X < (1-\delta)\mu] < e^{-\delta^2\mu/2}, 0 < \delta < 1$
- $Pr[X > (1+\delta)\mu] < e^{-\delta^2\mu/3}, 0 < \delta < 1$

关于决策：从抛硬币开始

抛100次独立的硬币(1/2)，你预期观察到多少次正面朝上?

- 大约50次？51/52都合理，那30合理吗?

随机变量$X_i = \sum_{1\le i\le n} x_i$；$\mu = E[X] = \sum_{1\le i\le n} p_i$

- $Pr[X < (1-\delta)\mu] < e^{-\delta^2\mu/2}, 0 < \delta < 1$
- $Pr[X > (1+\delta)\mu] < e^{-\delta^2\mu/3}, 0 < \delta < 1$

说人话：独立随机变量集中在$O(\sqrt{E[x]})$

- “正态分布”(中心极限定理) ——生活中的很多例子

随机有什么用呢？

考虑两个01串的LCS，已经证明了下界大约是 $O(n^2)$ 的

- 大家都会做啦，$f_{i,j} = ...$

但既然随机，最优解几乎总是诞生在对角线附近啊!

- 根据Chernoff Bound……
- 我们有一个 $O(n^{1.5})$ 的算法，几乎总是正确

随机有什么用呢？

考虑两个01串的LCS，已经证明了下界大约是 $O(n^2)$ 的

- 大家都会做啦，$f_{i,j} = ...$

但既然随机，最优解几乎总是诞生在对角线附近啊！

- 根据Chernoff Bound……
- 我们有一个 $O(n^{1.5})$ 的算法，几乎总是正确

我会告诉你NOI2010以前的题目都可以轻松卡到100分?

- 所以后来NOI都不太敢出普通的优化DP了
- 太容易被卡过去了
- 都是现在这种技巧题、计数，没法投机取巧……

如果不随机呢？

优化问题的结构：最优解的决策通常是有规律的

- 诞生在之前最优决策的附近
- 诞生在“贪心最优解”的附近
- 诞生在假设随机下的附近

如果不随机呢？

优化问题的结构：最优解的决策通常是有规律的

- 诞生在之前最优决策的附近
- 诞生在“贪心最优解”的附近
- 诞生在假设随机下的附近

屡试不爽的骗分手段：

- 强行假设存在决策单调性，减少决策数量
  - 其实很多DP是不满足四边形不等式的，但假设有就好了啊
- 在时间允许的范围里探索其他合理的决策点
  - 建议大家试试：用最短的代码水过NOI系列赛的DP题

NOI2017 泳池

给一个 $n \times m$ 的网格，每个格子独立等概率可用/不可用

- 问底边与最下边重合的最大可用矩形大小恰好为 k 的概率

互联网上的解答

状态表示和转移方法从天而降

考虑 DP：令 $f_{n,m}$ 表示底边长为 n ，且已知每一行安全高度都不小于 m 时的概率。

$$f_{n,m} = \begin{cases} 0 & nm >= k \\ 1 & n = 0 \\ p^n f_{n,m+1} + \sum_{i=0}^{n-1} p^i f_{i,m+1}(1-p) f_{n-i-1,m} & \text{otherwise} \end{cases}$$

- 精妙啊！
- 一年以后：咦我已经不记得这个题怎么做的了……

开心地尝试一下

DP = 搜索

- 从左到右枚举所有可能的情况
  - (失败，做题最讨厌的时刻......)

开心地尝试一下

DP = 搜索

- 从左到右枚举所有可能的情况
  - (失败，做题最讨厌的时刻……)
- 从下到上枚举所有可能的情况
  - 能想到这么试试，就是成功的第一步了
- 不过还是有点麻烦
  - $f_{i,j}$ 表示到第 i 层，有 j 个安全格子的方案数
  - 不能轻易地合并搜索空间：都是3个安全格子，$1+1+1$和3对上层的影响是不同的

分析

从下到上、从左到右一个一个格子看

- 如果在第一行放一个障碍，这一列就绝对不可能再用了
  - 又是特殊情况！

第一行

- 要么全是空的——上面的最大矩形全部都加一行
  - 编程了高度减一、宽度不变的子问题
- 要么至少有一个障碍——考虑第一个障碍的位置（套路）
  - 变成了两个高度不变、宽度变小的独立子问题

设计动态规划

这样构造搜索空间，如果考虑最大安全的矩形面积

- 我们并不关心每一行上有多少安全/不安全的格子
- 真正关心的是当前层底下有多少层是安全的

可以用额外的一个dp维度表示；但也可以用 $dp_{m,n}$ 表示已经有 m 层安全、宽度是 n、且最大矩形不超过 k 的方案总数

- 看起来有点诡异
- 之后就更套路了……
    - n 这么大， 肯定是个线性递推吧……

思考

搜索空间有不同的构造方法

- 从左到右很容易写出一个形式相当优美的公式(陷阱)
- 但换一种搜索方法，才得到正确的思路

解题需要“BFS”

- 多尝试多种分析方法，不要对一种办法死磕太久

NOI2018 冒泡排序

有一个程序，输入一个排列`p[1 ... n]`:

```
for (i = 1; i ⩽ n; i++)
  for (j = 1; j ⩽ n - 1; j++)
    if (p[j] > p[j + 1])
      交换p[j], p[j + 1]
```

可以证明交换次数的下界：$1/2\sum_{i=1}^{n}|i-p_i|$

- 给一个排列 $(n \leq 10^5)$，问比这个排列字典序大的排列中，交换次数恰好等于下界的数量

分析

这个题分成很多个部分……

- 非DP的部分：什么样排列的逆序对数 = $1/2\sum_{i=1}^{n}|i-p_i|$？？？
- 反正3 2 1不行，因为该死的2
  - 我没办法在保持中间数字不变的情况下，交换两个数
- 但3 4 1 2没这个问题
  - 4为了回到正确的位置上，会把1, 2的位置都拉近

然后就有了那个充分必要条件

- 你问我是怎么想到的……？ emmm……一半是猜的
- 考场上：想不到 = gg (但做习惯cf的这应该不是问题)
  - 但也可以不择手段打表观察

分析 (cont'd)

序列中不存在长度超过2的下降子序列

- 进入套路计数时刻
- 但如果我忽然忘记套路了怎么办?

搜索啊!

- 就生成全排列呗

搜索 (cont'd)

`[6, 2, 3]` => 下一个数可以填什么呢?

- < 3的肯定是不行的，不然就构成长度为3的下降子序列了
- 但是4, 5也不行，因为1还没填!
- 所以什么数也不能填，搜索到此为止

其实`[6, 2]`的时候就已经失败了

- `[6, 1, 2, 3, 4, 5]`是6开头唯一可行的
- `[7, 13, 19]`之后，只有1 (最小值)和> 19 (大值)的能填
  - 好像有一些DP的潜质了……

搜索 → DP

尝试合并状态

- `[1, 5, 7]` - `[2, 3, 4, 6 | 8, 9]`
- `[4, 7, 1]` - `[2, 3, 5, 6 | 8, 9]`之后可填方案和上面一一对应
- 所以我们其实只关心剩下的数字里，有多少比现在最大的数字大

DP就有啦

- $f_{i,j}$ 表示排列生成了 i 个数字，其中最大数字为 j 的方案数
  - $f_{i,j} \to f_{i+1,j}$, 如果剩下的小数字还够用 $(j - i > 1)$
  - $f_{i,j} \to f_{i+1,k}$, 如果剩下的大数字还够用 $(k > j)$
  - 连续的一段求和，可以用部分和优化

处理给定的排列

(又是套路，看来不会套路寸步难行……)

- 不，假装我不会套路，枚举全排列

```
2 3 1 4 // 我们求 > 它的排列
        //   2    3    1  [>4] (不存在的)
2 3 4 x //   2    3  [>1]   x
2 4 x x //   2  [>3]   x    x
3 x x x // [>2]   x    x    x
4 x x x //
```

然后我们的DP就在带x的排列里继续

- 第一位可以枚举
- 分每一种形如`6 8 1 [>3] x x x x x`来处理
    - 等价于计算`S(6 8 1 x x x x x x)`-`S(6 8 1 2 x x x x x)`

终于……

`S(6 8 1 x x x x x x)` $= f_{i,j} / (i \cdot \binom{j-1}{i-1} \cdot (i-1)!)$

- 按顺序每填一个`x`，要么填大数；要么填小数 (和之前DP一样！)
- 终于有80分了，勉强一下也许有84分……
  - 然后，我们发现求解只需要 $O(n)$ 个 $f_{i,j}$

终于……

`S(6 8 1 x x x x x x)` $= f_{i,j} / (i \cdot \binom{j-1}{i-1} \cdot (i-1)!)$

- 按顺序每填一个`x`，要么填大数；要么填小数（和之前DP一样！）
- 终于有80分了，勉强一下也许有84分……
  - 然后，我们发现求解只需要 $O(n)$ 个 $f_{i,j}$

- 然后你需要把DP方程展开观看
- 需要一些直觉
  - 选大小和走格子/组合数的关系

小结

分析：NOI DP题的特点

反正每年都会有的；但一定是脱离(传统)套路的

- 首先，你得会任何流行的套路，否则 = gg
- 在套路的基础上，需要相对比较复杂的分析
  - 得分取决于你分析出多少问题中的性质

年份题目	用到的性质
2015 寿司晚宴	可枚举的核心 → 状态压缩
2016 国王饮水记	推性质 → 区间DP → 凸包/单调性 → 决策稀疏性
2017 泳池	神奇的计数 → 套路线性递推
2018 冒泡排序	推性质 → 神奇的计数 → 处理字典序 → 推性质
2019？？？？	反正感觉不会容易

谈谈训练：DP题的误区

天上掉下了 $f_{i,j}$ 的表示

- 解题报告常有——考虑XXX，于是应该用YYY表示状态
- 但我压根不知道要先考虑XXX啊——下一次新的题还是不会做

谈谈训练：DP题的误区

天上掉下了 $f_{i,j}$ 的表示

- 解题报告常有——考虑XXX，于是应该用YYY表示状态
- 但我压根不知道要先考虑XXX啊——下一次新的题还是不会做

回归本质：两个视角

- 分治中的子问题划分 (特别适用于计数)
- 搜索空间的合并 (特别适用于优化)

线索本身就很难找……(做过的题/套路就起作用了)

谈谈训练：更多的思考

有没有可能写一个程序

- 输入一个OI题 (集合题/数论题)
- 输出一个解 (数学证明)?

谈谈训练：更多的思考

有没有可能写一个程序

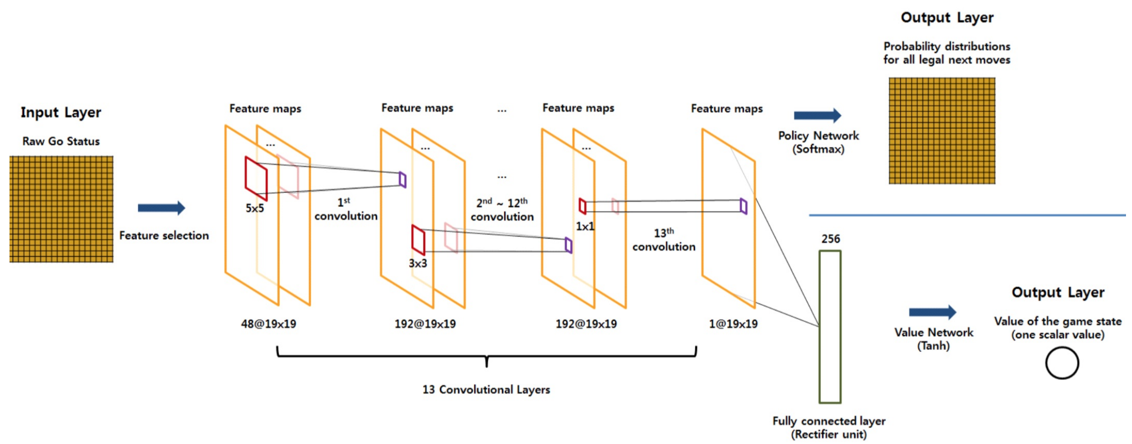
- 输入一个OI题 (集合题/数论题)
- 输出一个解 (数学证明)?

原则上很简单

- 从最短的证明开始搜索；直到找到第一个合法的证明为止
- 但实际很困难：搜索的空间太大了

减少搜索空间：AlphaGo

卷积神经网络、增强学习……



说人话

- 设一个很复杂的函数$f(A,X)$ ($X_{i,j} \in \{0,1\}$是棋盘)
- 然后根据大量的棋局，求出合适的A，使得 $f(A,X)$ 输出可能有用的下子位置

AlphaGo到底是什么？

搜索 + 剪枝：探索搜索问题的“结构”

- 试想：如果对方这么下，我怎么下呢……就这样一直想到终盘
- 每次都和自己下一盘(带有随机性)的快棋
- 看看下在哪里胜率最大，就下在哪里

我们也是AlphaGo

- 把问题写成数学形式
- 根据我们的经验，试一试公式的展开、变形、改写……
- 直到得到正确的解

训练提示

对所有OI选手都奏效

- 放弃一切套路
- 静下心来刷CF，尽量别看题解
- 注重恢复解题的过程；如果没想到，试图去补你的搜索

训练提示

对所有OI选手都奏效

- 放弃一切套路
- 静下心来刷CF，尽量别看题解
- 注重恢复解题的过程；如果没想到，试图去补你的搜索

对NOI特别凑效的秘诀

- 反正要对拍的，先写暴力搜索
- 观察最优解/计数的结构
  - 打印出所有的(最优)解、决策的位置……
  - 疯狂猜想 + 挖掘可能的性质
  - 用好time limit里的每一次计算

终于结束了啊啊啊啊

过气老选手做这些题已经很吃力了啊喂

Happy Hacking and 欢迎报考南京大学

轻松一下

Counting Graphs

n 个不同的(带编号)的点，能组成多少不同的：

- 无向连通图?
- 有向强连通图?

该死的计数……

Blackjack

有若干张牌，牌上有点数，一开始经过洗牌后，每一种排列都等概率出现

- A先抽牌，并且告诉B一个数字 a
- B开始抽牌，一张一张抽
  - 如果抽到点数和超过 b 就输掉游戏
  - 否则，如果抽到点数和超过 a 就赢得游戏
  - 抽完最后一张牌也输掉游戏

求A赢得游戏的概率

说人话：求一个数组所有排列中，存在前缀和满足$a < S < b$的数量 $(n, a, b \leq 100)$

排列组合

给定给一个字符串`aaabbbbcdde`

- 问其所有的排列中，相邻字母不同排列的数量
- (JSOI2019 “神经网络”的主问题)