动态规划专题复习

雅礼中学 罗煜楚

写在前面

- 由于个人水平有限，课件可能有一些错误，欢迎大家指出与批评
- 由于这堂课的目的是较为系统地复习，配合少量例题，所以比较简单，大家可以根据自己实际水平，选择性地听课，下一节课将有欧阳思琦大神给大家带来更有难度更加精彩的dp杂题选讲
- 祝愿大家在今后考试中都能发挥水平，实现自己的OI目标与人生理想

目录

- 1. DP的类型
  - Dp的定义
  - 数位Dp
  - 概率Dp
  - 树形Dp
  - 状态压缩DP
  - Dp套Dp
- 2. Dp的优化
  - Dp形式优化
  - 决策单调性优化
  - 斜率优化

Chapter 1: Dp的类型

1.Dp定义
2.数位Dp
3.概率Dp
4.树形Dp
5.状态压缩DP
6.Dp套Dp

Dp的定义

- 动态规划在查找有很多重叠子问题的情况的最优解时有效。它将问题重新组合成子问题。为了避免多次解决这些子问题，它们的结果都逐渐被计算并被保存，从简单的问题直到整个问题都被解决。因此，动态规划保存递归时的结果，因而不会在解决同样的问题时花费时间。
- 动态规划只能应用于有最优子结构的问题。最优子结构的意思是局部最优解能决定全局最优解（对有些问题这个要求并不能完全满足，故有时需要引入一定的近似）。简单地说，问题能够分解成子问题来解决。

Dp的定义

- Dp的一般形式是表示出能代表出至于后续或答案有关的当前的状态，并在这些状态之中进行转移，而Dp的优劣就用状态数以及转移的时间复杂度来评定了，好的Dp应该是深挖了题目或数据的性质，从而达到时间与空间上的良好效果。

Bzoj 2298

一次考试共有n个人参加，第i个人说：“有ai个人分数比我高，bi个人分数比我低。”可以有相同分数，问最多有多少人在讲真话

$n \leq 100000$

Bzoj 2298

转换模型，一个人说的话如果为真说明从第ai+1个人到n-bi个人的分数都是一样的，如果我们把他看作一段线段，那么在这n条线段中，相交且不重合的线段一定不能同时为真话，所以即为求问题就转换成选出尽可能多的不向交或完全重合的线段，但是对于某条线段完全重合的个数要小于等于线段长。将线段按某左端点排序，令f[i]为到了i这个坐标以及以前所能选的做多线段f[i] = f[i - 1] + min(线段个数，线段长度)

数位Dp

- 数位Dp常用来统计或查找一个区间满足条件的数，然后按数位顺序Dp，一般需要仔细分情况讨论，常见处理如将区间拆为[1~R]-[1~L-1]，记忆化，预处理等

Sdoi2013

一个二维坐标。X轴，Y轴坐标范围均为1..N。初始的时候，所有的整数坐标点上均有一块金子，共N*N块。

一阵风吹过，初始在(i，j)坐标处的金子会变到(f(i)，f(j))坐标处。其中f(x)表示x各位数字的乘积(去除前导零)，例如f(99)=81，f(12)=2，f(10)=0。如果金子变化后的坐标不在1..N的范围内，我们认为这块金子已经被删除。同时可以发现，对于变化之后的游戏局面，某些坐标上的金子数量可能不止一块，而另外一些坐标上可能已经没有金子。这次变化之后，游戏将不会再对金子的位置和数量进行改变

求出风吹过之后金块数量前K大的坐标的金块数量和

答案可能很大，输出对10^9+7取模之后的答案。

$N <= 10^{12}, K <= 100000$

Sdoi2013

我们发现实际的f可能的取值很少，且取值的质因子只有2,3,5,7几种于是我们可以用f(i,S,j)表示从高到低到了第i位，j表示已经考虑过得位置是否小于N，S为当前乘积是否是0以及不是0时2,3,5,7因子的次数，最后得到答案后用优先队列即可统计答案

概率Dp

- 概率Dp是一类求事件概率或期望的Dp的总称，对于求概率问题，有时利用补集转化，或者将其转化为计数问题，而对于求期望则大多利用期望的线性性来解决问题

Hnoi 2015

玩家有一套卡牌，共 n张。游戏时，玩家将 n 张卡牌排列成某种顺序，排列后 将卡牌按从前往后依次编号为 1 ~ n。本题中，顺序已经确定，即为输入的顺序。

每张卡牌都有一个技能。第 i 张卡牌的技能发动概率为 pi，如果成功发动，则会对敌方造成di点伤害。也只有通过发动技能，卡牌才能对敌方造成伤害。基于现实因素以及小K非洲血统的考虑，pi不会为 0，也不会为 1，即 0 < pi < 1。

一局游戏一共有 r 轮。在每一轮中，系统将从第一张卡牌开始，按照顺序依次考虑每张卡牌。在一轮中，对于依次考虑的每一张卡牌：

1 如果这张卡牌在这一局游戏中已经发动过技能，则

1.1 如果这张卡牌不是最后一张，则跳过之（考虑下一张卡牌）；

1.2 否则（是最后一张），结束这一轮游戏。

2 否则（这张卡牌在这一局游戏中没有发动过技能），设这张卡牌为第 i 张

2.1将其以 pi的概率发动技能。

2.2如果技能发动，则对敌方造成 di点伤害，并结束这一轮。

2.3如果这张卡牌已经是最后一张（即 i 等于n），则结束这一轮；否则考虑下一张卡牌。

请帮助小 K 求出这一套卡牌在一局游戏中能造成的伤害的期望值。

Hnoi 2015

玩家有一套卡牌，共 n张。游戏时，玩家将 n 张卡牌排列成某种顺序，排列后 将卡牌按从前往后依次编号为 1～n。本题中，顺序已经确定，即为输入的顺序。

每张卡牌都有一个技能。第 i 张卡牌的技能发动概率为 pi，如果成功发动，则会对敌方造成di点伤害。也只有通过发动技能，卡牌才能对敌方造成伤害。基于现实因素以及小K非洲血统的考虑，pi不会为 0，也不会为 1，即 0＜pi＜1。

一局游戏一共有 r 轮。在每一轮中，系统将从第一张卡牌开始，按照顺序依次考虑每张卡牌。在一轮中，对于依次考虑的每一张卡牌：

1 如果这张卡牌在这一局游戏中已经发动过技能，则

 1.1 如果这张卡牌不是最后一张，则跳过之（考虑下一张卡牌）；

 1.2 否则（是最后一张），结束这一轮游戏。

2 否则（这张卡牌在这一局游戏中没有发动过技能），设这张卡牌为第 i 张

 2.1将其以 pi的概率发动技能。

 2.2如果技能发动，则对敌方造成 di点伤害，并结束这一轮。

 2.3如果这张卡牌已经是最后一张（即 i 等于n），则结束这一轮；否则考虑下一张卡牌。

请帮助小 K 求出这一套卡牌在一局游戏中能造成的伤害的期望值。

Hnoi 2015

这道题的模型转化十分巧妙，并不直接求解答案，Dp[i][j]表示第i张卡牌有j次发动技能的机会，显然有dp方程

$$Dp[i][j] += Dp[i-1][j]*(1-P_{i-1})^{\wedge j}$$

$$Dp[i][j] += Dp[i-1][j+1]*(1-(1-P_{i-1})^{\wedge (j+1)})$$

求出Dp之后这样我们就求出了每张卡牌发动的概率，分别算期望即可。

BitToggler

给你一个长度为n(n<=20)的01串，以及一个指针，初始时指针在第x0个字符上。每回合随机一个1到n中的数j，如果指针之前在i上，就花费|j-i|的时间把指针从i移动到j上，并且把01串的第j位取反。不停这样随机，直到01串变成全0或者全1为止，问到终止前期望花费的时间是多少？

SRM641

BitToggler

我们发现不能直接将整个状态压起来。但是可以发现产生代价的移动只有n^2种，由于期望的线性性，我们只要对于每一种代价求出他对期望的贡献即可，那么对于一种移动i->j，我们只需关心i与j的状态，其他位置的颜色数目以及当前指针的位置，这样状态就可以被压缩起来了，状态只有大概$n*3*2$种，列方程求解即可。复杂度n^3

树形Dp

- 树形Dp是指基于树的结构的动态规划，基础的有求树的直径重心，树上最大权独立集，树形依赖背包，虚树上的Dp等等。
  - 树的直径：Dp记录子树内的最长路
  - 树的重心：Dp记录子树大小
  - 树的独立集：Dp记录子树的根是否选择
  - ……

树形Dp

- 树形依赖背包：可以在Dfs序上Dp(即每次选择是否跳过子树)，或通过将父节点的Dp值传入孩子Dp
- 虚树：在原树上只保留需要的点与他们的Lca的树称为虚树，建树方法为：将点按Dfs序排序，一次将点加入，用一个栈维护树上的已加入的点和他们Lca的右链，每次加入一个点，与栈顶比较，便可以建出虚树。

bzoj3611

给出一n个点棵树，树上边的长度都为1，每次询问k个点，在k个点之间有C(k,2)条路径，请求出

1. 这些路径的代价和
2. 这些路径中代价最小的是多少
3. 这些路径中代价最大的是多少

$n \leq 1000000$ $\sum k \leq 2 \times n$

bzoj3611

有着询问的和小于的限制的题是虚树的一个显著的特征，建出虚树，然后直接在上面树形dp，f(i)为以i为根的子树内路径的代价和，size(i)为i的子树中询问点的个数，ma(i)为i子树内的到i的最长路，mi(i)为最短

$$f(i) = f(son(\mathrm{i})) + size(i) \times (k - size(son(i)))$$
$$\mathrm{ma}(i) = \max\{\mathrm{ma}(son[i]) + 1\}$$

路径和答案是f[root]，对于每个点，用它的最大与次大的儿子更新答案，如果他是被询问的点，则还可以用它每个儿子更新答案

状态压缩Dp

- 基于状态压缩的Dp是由于状态用单个简单的变量直接存储存在空间的浪费，而采用压缩的状态的动态规划，像插头Dp，斯坦纳树，等问题都可以用状压Dp解决。
  - 插头Dp：维护当前已决策和未决策的一条Z字形的轮廓线的插头状态，用括号序列配对插头，每次只需分情况讨论即可，但是这类Dp的显著特点就是情况繁多，使用时须细心。

状态压缩Dp

- 斯坦纳树：斯坦纳树是解决一类要求关键点联通的问题，由于比较好写，在某些情况可以代替插头Dp，甚至解决其不能解决的问题，通过两部分Dp，一个是根据已选关键点集合大小的顺序Dp，另一部分则是基于Spfa的非关键点的扩张Dp，具体可以表示为
  - f[i][S]：当前在第i个点，已覆盖的关键点集合状态为S
  - f[i][S] = f[i][s] + f[i][S xor s]
  - f[i][S] = f[j][S] + dist[j][i]

Tourism

给定一个n个点，m条边的无向图，在第i 个点建立旅游站点的费用为Ci。在这张图中，任意两点间不存在节点数超过10 的简单路径。

请找到一种费用最小的建立旅游站点的方案，使得每个点要么建立了旅游站点，要么与它有边直接相连的点里至少有一个点建立了旅游站点。

$2 \leq n \leq 20000; 0 \leq m \leq 25000$

Tourism

对于一个连通块，取一个点进行dfs，得到一棵dfs 搜索树，则这棵树的深度不超过10，且所有非树边都是连向祖先对于每个点x，设S 为三进制状态，S 第i 位表示根到x 路径上深度为i 的点的状态：

0：选了

1：没选，且没满足

2：没选，且已满足

设f(i, S) 为考虑根到x 路径上深度为i的点时这些点的状态

为S时的最小费用，然后按DFS 序进行DP即可。

复杂度(m*n)3^10

Bzoj 3205

给定一张h*w地图，一些地方有障碍物，一些地方有转向器，有k<=9个机器人，第i个机器人编号区间初始为[i, i]，现在可以向四个方向推机器人，机器人遇到障碍才会停，遇到转向器会转向，停在同一个格子的两个编号区间相邻的机器人可以合并，合并后的机器人编号变为一段区间，求最少推多少次可以全部合并

$h \le 500\ w \le 500\ k \le 9$

Bzoj 3205

令f[l][r][x][y]表示在点(x,y)将编号在[l,r]区间内的机器人全部合并的最小推动次数

推动：

f[l][r][x][y]=min{f[l][r][from_x][from_y]+1} ((from_x,from_y)->(x,y))

合并：

f[l][r][x][y]=min(f[l][k][x][y]+f[k+1][r][x][y]) (l<=k<r)

下面正常转移，上面的用spfa即可

Bzoj 2734

一个好的子集被定义为：若x在该子集中，则2x和3x不能在该子集中，一个集合{1, 2, 3. . . . n-1, n}，求好的子集的个数

$n \leq 100000$

Bzoj 2734

我们首先写出一个如下的矩阵

1	3	9	27	81	…
2	6	18	54	162	…
4	12	36	108	324	…
8	24	72	216	648	…
16	48	144	432	1296	…
…					

我们发现不合法的数对都在相邻的位置，而n<=100000，所以行和列都不会很多，可以将行的状态压缩起来，做Dp。

而我们又发现有一些数没有出现，所以再枚举一下之前没有出现的数，作为矩阵的左上角。最后答案为所有的矩阵的答案乘起来。

Dp套Dp

- 一些Dp问题的子判定问题不能简单的解决，而必须用另一个Dp解决，此时就只能使用Dp套Dp的方法，他的意思是，外面的Dp的状态是存的里层Dp各个状态的值，利用里层的状态来判断外层的Dp是否合法，类似的问题有可以胡的麻将方案数，还有Lcs为定值的序列的方案数等等。

StringPath

给你一个n*m(n,m<=8)的矩阵，再给你两个长度为n+m+1的小写字母组成的字符串A,B，问有多少种给矩阵在每个格子里填小写字母的方式，使得从矩阵的左上角到右下角（只能往右和往下），存在两条路径经过的字符分别组成A和B。

SRM 591

StringPath

考虑对于给定的矩阵，Dp是否存在这样的路径时，可用f(i,j)表示到(i,j)是否为这个字符串的前缀，那么对于当前矩形，对于已经填字母到了(i,j)只需记录前2m个dp值即可，状态2^{2m}

Bzoj 3964

给你一个只由AGCT组成的字符串S，对于每个0<=i<=|S|,问有多少个只由AGCT组成的长度为m的字符串T,使得LCS(S,T)=i

|S|<=15,1<=m<=1000

Bzoj 3964

对于LCS的dp过程是这样的

$$dp[i][j]=\max\{dp[i-1][j],dp[i][j-1],dp[i-1][j-1]+1\mid(S[i]==T[j])\}$$

则T到当前第x位的状态，便只与dp[x][j]的值有关，所以dp的状态便只要存下这最多15个值，又由于两个值之间最多相差1，所以可用一个大小为2^15的状态存下，所以转移方程为

$$f[i][trans[S][ch]] += f[i-1][S]\ (ch\in ACGT)$$

trans[S][ch]为预处理的S状态加入字符ch之后的转移

Square

给你一个n*n(n<=8)的棋盘，上面有一些格子必须是黑色，其它可以染黑或者染白，求有多少种染色方法使得棋盘上的最大连续白色子正方形边长为k

2014 Asia AnShan Regional Contest,by WJMZBMR

Square

这道题的解法有很多，这里就讲一种比较简单的做法

首先考虑对于一个确定的矩阵，有一个这样的Dp来求最大子正方形

$$f(i,j)=\min(f(i-1,j-1),f(i,j-1),f(i-1,j))+1 \quad \text{白色}$$

$$=0 \quad \text{黑色}$$

则对于一种染色到(i,j)的方案，答案只与n+1个dp值与已存在的最大值有关，又因为f(i,j-1)>=f(i,j)-1所以状态不多，大概只有几万

Chapter 2: Dp的优化

1.形式优化

2.决策单调性优化

3.斜率优化

形式优化

- 有时在做一个Dp问题时将会遇到时间或者空间复杂度过高的问题，而在Dp的形式上优化便是有效的优化技巧，典型的有预处理，分阶段Dp等
  - 预处理：我们可能发现，在Dp的过程中，出现了重复的运算，浪费了时间，所以我们可以通过Dp前预处理，或者Dp过程总处理出最值，而达到为后面的Dp提供便捷的功能与作用，达到优化的目的
  - 分阶段Dp：在某些Dp中将Dp拆为一个个有特点阶段也许比将整个Dp放在一起更加节省时间与空间，所以对于彼此相对无关的转移，可以分开考虑

决策单调性优化

- Dp问题的转移往往需要大量的时间，如果我们能发现一些性质，找到一些规律来优化Dp决策转移的过程，那么在时间上，我们便能得到很大的优化，常见的有四边形不等式优化，以及一些1D/1D动态规划的优化

决策单调性优化

四边形不等式优化：对于形如以下的Dp

$$f[i][j] = f[i][k-1] + f[k][j] + w[i][j]$$

如果w满足四边形不等式

任意$i \le i' \le j \le j'$，有 $w[i][j]+w[i'][j']<=w[i'][j]+w[i][j']$

任意$i' \le i \le j \le j'$，有 $w[i][j]<=w[i'][j']$

那么也可证明

$$f[i][j]+f[i'][j']<=f[i'][j]+f[i][j']$$

决策单调性优化

而如果得到了这样的式子，则就可以证明f[i][j]的决策一定在f[i][j-1]与f[i-1][j]的决策之间

$$s[i][j-1]<=s[i][j]<=s[i-1][j]$$

四边形不等式的证明网络上早就十分详细，在这里就不再演示，这里提供一篇

http://wenku.baidu.com/link?url=344UHCQdTP9z2dFTCCGB3eBYHnlBeF0IAYdFeLmA_p0QU9nGv3L-6A_yISk4zUKcTMBDrokvx_i-5BHh7H5ZFfjS3hf2j9jHdPCgUXwQjqS

决策单调性优化

- 有关决策单调性的优化还有一系列常见的1D/1D动态规划方程的优化

  $f[x]=\min\{f[i]+w[i][x] \mid i<=x\}$

  - 如果w满足四边形不等式，则可以证明f[x]的决策也一定单调，我们可以用一个栈来维护每一个f[x]的最优决策使得时间复杂度变为nlogn

NOI2009 诗人小G

有一个叫小G的诗人，一首诗包含了若干个句子，对于一些连续的短句，可以将它们用空格隔开并放在一行中，注意一行中可以放的句子数目是没有限制的。小G给每首诗定义了一个行标准长度（行的长度为一行中符号的总个数），他希望排版后每行的长度都和行标准长度相差不远。显然排版时，不应改变原有的句子顺序，并且小G不允许把一个句子分在两行或者更多的行内。在满足上面两个条件的情况下，小G对于排版中的每行定义了一个不协调度，为这行的实际长度与行标准长度差值绝对值的P次方，而一个排版的不协调度为所有行不协调度的总和。

小G最近又作了几首诗，现在请你对这首诗进行排版，使得排版后的诗尽量协调（即不协调度尽量小），并把排版的结果告诉他。

N为诗句个数，L为标准长度，每一句诗长度小于等于30

$n \le 100000 \quad L \le 3000000 \quad P \le 10$

NOI2009 诗人小G

首先可以推得最直接的Dp

$$dp[i]=min(dp[j]+abs(sum[i]-sum[j]-l)^p)$$

我们发现这个方程式1D/1D的，令w[i][j]=abs(sum[i]-sum[j]-l)^p，可以证明他是满足四边形不等式的，证明可以看这里 https://www.byvoid.com/blog/noi-2009-poet

由前面讲到的结论，所以对于任意i<=j，dp[i]的决策一定小于等于dp[j]的决策，于是我们可以用一个栈来维护dp的决策，过程如下

NOI2009 诗人小G

一开始，只有 f(1)的函数值被计算出来，于是所有状态的当前最优决策都是 1。

11111111111111111111111111111111111111111111111111111111111

现在，显然 f(2)的值已经确定了：它的最有决策只能是 1。我们用决策 2 来更新这个决策表。由于决策单调性，我们知道新的决策表只能有这样的形式：

1111111111111111111111111111222222222222222222222222222222

这意味着我们可以使用二分法来查找“转折点”，因为如果在一个点 x 上，如果决策 2 更好，则所有比 x 大的状态都是决策 2 更好；如果 x 上决策 1 更好，则所有比 x 小的状态都是决策 1 更好。

现在决策 1 和决策 2 都已经更新完毕，则 f(3)业已确定，现在用决策 3 来更新所有状态。根据决策单调性，现在的决策表只能有以下 2 种类型：

11111111111111111111111111112222222222222222223333333333

111111111111111111111333333333333333333333333333333333

而这样的决策表示绝对不会出现的：

1111111111333333333333333333222222222222222222222222222，不可能。

那么，我们的更新算法就是：

1、考察决策 2 的区间[b,e]的 b 点上是否决策 3 更优，如果是，则全部抛弃决策 2，将此区间划归决策 3；如果否，则在决策 2 的区间[b,e]中二分查找转折点。

2、如果第 1 问的回答是“是”，则用同样的方法考察决策 1。

决策单调性优化

常见形式：

f[x]=min{g[i] | i<=b[x]}+w[x] 且b[x]单调不降

使用一个单调队列来维护g的序列，从而达到O(n)的时间复杂度

f[n]=min{a[n]*x(i)+b[n]*y(i) | i<=n}

可以将等式两边都除以b[n]，然后用斜率优化

斜率优化

斜率优化Dp是当Dp转移式是形如

$$f[i] = \min\{f[j]+a[i]+b[j]+c[i]*d[j]+e\}$$

的式子，先将与j无关的常数去掉，我们可以这么看

$$f[i] = \min\{c[i]*(d[j]) + (f[j]+b[j])\}$$

这与b=kx+y的形式十分相似即最小化一条直线的截距，于是我们便可以使用维护凸壳来将时间复杂度变得更优，根据数据的特征来选择用平衡树或栈或队列的数据结构，也确定查询使用二分或是具有单调性。

鸣谢

- 感谢汪星明老师所付出的一切
- 感谢各位同学的认真聆听
- 再次祝愿大家在考试中取得好的成绩

谢谢