动态规划选讲

-----abc1057510554

自我介绍

内容

- 序列DP
- *背包DP
- 区间DP
- *状态压缩DP
- *树形DP
- *期望DP/概率DP
- 数位DP
- *计数DP

背包DP

- 通常指这样一类问题

- 你有一个大小为m的背包，有n个物品，每个物品有价值和大小，你可以选择一些物品装进背包，求背包能装下的物品的最大价值。

- 一般设$f[i][j]$表示取i件物品且背包容量为j时能装下的最大价值。

Example 1

- (CSP-J 2019 纪念品)小伟知道未来T天N种纪念品每天的价格。某个纪念品的价格是指购买一个该纪念品所需的金币数量，以及卖出一个该纪念品换回的金币数量。
- 小伟每天可以选择花费一定金币购买纪念品，或卖出一些纪念品以换回金币，每天可以进行任意多次买入和卖出操作
- T天之后，小伟的超能力消失。因此他一定会在第T天卖出所有纪念品换回金币。
- 小伟现在有M枚金币，他想要在超能力消失后拥有尽可能多的金币。
- $T \le 100, N \le 100, M \le 1000,$ 保证金币最大值不会超过10000

Example 1

- 如果我们保存每个纪念品买入和卖出的时间，那状态会非常复杂，但容易发现第i天买入第j天卖出，等价于第i天买入第$i+1$天卖出，第$i+1$天再买入第$i+2$天卖出……直到第$j-1$天买入第j天卖出，不妨考虑对任意一个买入的物品，都将它在下一天卖出。对答案无影响。

Example 1

- 如果我们保存每个纪念品买入和卖出的时间，那状态会非常复杂，但容易发现第i天买入第j天卖出，等价于第i天买入第$i+1$天卖出，第$i+1$天再买入第$i+2$天卖出……直到第$j-1$天买入第j天卖出，不妨考虑对任意一个买入的物品，都将它在下一天卖出。对答案无影响。
- 容易观察出这应该是个背包模型，但是我们熟悉的背包中的物品有两维，一维是重量，一维是价值，但这题中间我们只知道价值，所以该怎么转化呢？

Example 1

- 如果我们保存每个纪念品买入和卖出的时间，那状态会非常复杂，但容易发现第i天买入第j天卖出，等价于第i天买入第$i+1$天卖出，第$i+1$天再买入第$i+2$天卖出……直到第$j-1$天买入第j天卖出，不妨考虑对任意一个买入的物品，都将它在下一天卖出。对答案无影响。
- 容易观察出这应该是个背包模型，但是我们熟悉的背包中的物品有两维，一维是重量，一维是价值，但这题中间我们只知道价值，所以该怎么转化呢？
- 对每一天考虑。将本日的买入费用抽象为背包问题中的重量，明日的卖出费用抽象为背包问题中的价值，问题转化为做T次完全背包，时间复杂度$O(T*n*\max(ans))$

Example 2

- (JOISC2014 挂件) JOI君有N个装在手机上的挂饰，编号为$1 \dots N$。 JOI君可以将其中的一些装在手机上。
- JOI君的挂饰有一些与众不同——其中的一些挂饰附有可以挂其他挂件的挂钩。每个挂件要么直接挂在手机上，要么挂在其他挂件的挂钩上。直接挂在手机上的挂件最多有1个。
- 此外，每个挂件有一个安装时会获得的喜悦值，用一个整数来表示。如果JOI君很讨厌某个挂饰，那么这个挂饰的喜悦值就是一个负数。
- JOI君想要最大化所有挂饰的喜悦值之和。注意不必要将所有的挂钩都挂上挂饰，而且一个都不挂也是可以的。
- $1 \le N \le 2000$

Example 2

- 与一般的背包有什么不同？

Example 2

- 与一般的背包有什么不同？
- 物品的价值可以是负数
- 背包的容量一定是1
- 且物品的重量可以是负数，可以是正数，也可以是0

Example 2

- 与一般的背包有什么不同？
- 物品的价值可以是负数→初始化dp数组为-INF
- 背包的容量一定是1→ $\mathrm{dp}[0][1] = 0$
- 且物品的重量可以是负数，可以是正数，也可以是0→排序

状压DP

- 指DP的状态包含一维S，表示某些元素是否选取
- 时间复杂度往往是指数级别
- 数据的规模一般很小。

一道简单的例题(UOJ370)

- 吉老师有n个滑稽果，第i个滑稽果的大小为a_i。
- 他现在想把它们构成一棵任意形态的有根树，每个点的滑稽度为它的大小和它父亲的滑稽度的 and ，其中 and 表示按位与运算，例如 $2\ and\ 3 = 2$，$1\ and\ 0 = 0$，$1\ and\ 1 = 1$。特别地，根的滑稽度等于他的大小。
- 为了世界的和平，他希望能最小化这棵树上所有滑稽果的滑稽度之和。请问你能帮他解决这个问题吗？
- $1 \le n \le 10^5, 1 \le a_i \le 2 * 10^5$

一道简单的例题(UOJ370)

- 容易想到答案一定是一条链。
- 对于二进制的某一位b，如果所有a_i都有这一位，这一位一定是从头到尾一直存在，直接在答案中加上$n * 2^b$，再将b这一位在所有a_i中删除。

一道简单的例题(UOJ370)

- 容易想到答案一定是一条链。
- 对于二进制的某一位b，如果所有a_i都有这一位，这一位一定是从头到尾一直存在，直接在答案中加上$n * 2^b$，再将b这一位在所有a_i中删除。
- 令$S = a_1 \cup a_2 \ldots \cup a_n$，我们的目标是在转移过程中让S变为0
- 对于$f[S_0]$，选取a_i，有$f[S_0] + S_0 \cap a_i \xrightarrow{min} f[S_0 - S_0 \cap a_i]$
- 时间复杂度$O(na)$

一道简单的例题(UOJ370)

- 令$S = a_1 \cup a_2 \ldots \cup a_n$，我们的目标是在转移过程中让S变为0
- 对于$f[S_0]$，选取a_i，有$f[S_0] + S_0 \cap a_i \xrightarrow{min} f[S_0 - S_0 \cap a_i]$
- 时间复杂度$O(na)$
- 考虑优化
- 上述算法之所以慢，是因为很多不可能用来转移的a_i被考虑了，且很多种a_i的选择实际上转移到了同一状态。
- 可以从枚举a_i转到枚举S_0的子集S_1，只要S_1是a_i的子集，这个转移就是合法的。
- 时间复杂度$O(a_i^{\log_2 3})$(小\tips，预处理S1的时候要优化)

NOIP2017 宝藏

- 给定一个 n 个点 m 条边的图(可以有重边)，选取一个有根生成树，代价为

$$\sum_{v_i \in V} dep_{v_i} * val_{v_i}$$

- 也就是，每个点的深度乘以它与它父亲连边的边权
- 求最小代价
- $1 \le n \le 12, 0 \le m \le 10^3$，不会爆int
- 对70%的数据，$n \le 8$

NOIP2017宝藏

- Noip规律之：正解一般是60%-80%级别部分分的优化
- 先思考$n \leq 8$的情况，首先不难想到可以先将重边去除，因为在有重边的条件下，选边权尽量小的边得到的代价肯定会尽量小。
- 由于数据范围很小，不难想到搜索或者是状态压缩dp。

- 可以直接搜索吗？

NOIP2017宝藏

- Noip规律之：正解一般是60%-80%级别部分分的优化
- 先思考$n \leq 8$的情况，首先不难想到可以先将重边去除，因为在有重边的条件下，选边权尽量小的边得到的代价肯定会尽量小。
- 由于数据范围很小，不难想到搜索或者是状态压缩dp。

- 可以直接搜索吗？
- 显然是不行的。因为搜索一定是对边搜索，虽然$n \leq 8$，但是去掉重边后m最大可以到$C_n^2 = 28$，复杂度是$\mathrm{O}(2^{\mathrm{n}*\mathrm{n}})$级别的，这是不能接受的。

NOIP2017宝藏

- Noip规律之：正解一般是60%-80%级别部分分的优化
- 先思考$n \le 8$的情况，首先不难想到可以先将重边去除，因为在有重边的条件下，选边权尽量小的边得到的代价肯定会尽量小。
- 由于数据范围很小，不难想到搜索或者是状态压缩dp。
- 那么，考虑状态压缩
- 令$f[S][k][i]$表示考虑集合S，其中i为根，且假设i在最终所得的树中的深度为k所得到的最小代价。
- $f[S][k][i] = \min(f[S - S_0][k][i] + f[S_0][k+1][j] + (k+1) * w(i,j))$
- $w(i,j)$为边权，不存在边时为INF 时间复杂度$O(3^n * n^3)$

NOIP2017宝藏

- 令$f[S][k][i]$表示考虑集合S，其中i为根，且假设i在最终所得的树中的深度为k所得到的最小代价。
- $f[S][k][i] = \min(f[S - S_0][k][i] + f[S_0][k+1][j] + (k+1) * w(i,j))$
- $w(i,j)$为边权，不存在边时为INF

- 考虑优化
- 可以发现每次都在DP方程中重复计算$f[S_0][k+1][j] + w(i,j)$的最小值，因此可以令$g[S_0][i][k+1] = \min(f[S_0][k+1][j] + (k+1) * w(i,j))$
- 时间复杂度$O(n^2 * 3^n + n^3 * 2^n)$，可以通过此题

NOIP2017宝藏

- 还有别的做法吗？

NOIP2017宝藏

- 还有别的做法吗？
- 不妨直接模拟在每一层加哪些点。
- 令$f[i][S]$表示当前层数为i，已经选定S这些点的最小代价。
- $f[i][S] = \min(f[i-1][S_0] + minval[S][S_0])$
- 时间复杂度$O(4^n + n * 3^n)$

PKUWC2018 随机算法

- 我们知道，求任意图的最大独立集是一类NP完全问题，目前还没有准确的多项式算法，但是有许多多项式复杂度的近似算法。
- 例如，小 C 常用的一种算法是：
- 对于一个 n 个点的无向图，先等概率随机一个 $1 \ldots n$的排列 $p[1 \ldots n]$。
- 维护答案集合 S ，一开始 S 为空集，之后按照 $i = 1 \ldots n$ 的顺序，检查 $\{p[i]\} \cup S$是否是一个独立集，如果是的话就令 $S = \{p[i]\} \cup S$。
- 最后得到一个独立集 S 作为答案。
- 小 C 现在想知道，对于给定简单图，这个算法的正确率。
- $1 \leq n \leq 20$

PKUWC2018 随机算法

- 一个简单的暴力
- 令$f[n][S]$表示当前还有S这些点没选，这些点构成独立集大小为n。
- 正向的考虑选择了哪个点，并把与这个点有连边的所有点在集合内进行删除，令找到的新状态为$f[n-1][P]$。把P中的结点与S中不在P中的点进行标号拼接。即
- $f[n][S] += f[n-1][P] * \binom{|S|-1}{|P|} * (|S-P|-1)!$
- 时间复杂度$O(n^2 * 2^n)$。

PKUWC2018 随机算法

- 一个简单的优化
- 在进行上述DP的过程中，我们多算了独立集大小不是n的情况的方案数。因此可以针对这一点做出优化。

PKUWC2018 随机算法

- 一个简单的优化
- 在进行上述DP的过程中，我们多算了独立集大小不是n的情况的方案数。因此可以针对这一点做出优化。
- 令$g[S]$表示S对应的最大独立集大小
- 对于$f[S]$的转移$f[P]$,只当$g[P] = g[S] - 1$时转移
- 时间复杂度$O(n * 2^n)$

计数DP

- Q:什么是计数DP哇？
- A:就是前面那些DP式子中的min，max换成+号
- Q:那计数DP和递推有什么区别哇？
- A:没有区别(有一定的区别，比如dp会输入很多信息，递推只会输入一个n)

例题1

- 给定一个数列a[1..n]，求有多少个区间(l,r)，满足
- $\sum_{i=l}^{r}\sum_{j=i+1}^{r} a_i * b_j$能被3整除
- $1 \le n \le 500000$

例题1

- 令$f[i][x][y]$，表示以i为右端点的区间中，有x_0个模3余1的数$(x = x_0\%3)$，有y_0个模3余2的数$(y = y_0\%3)$。
- 容易转移
- 时间复杂度$O(9n)$
- Q：为什么不多加一维保存模3余0

一道遥远的例题2

- n个盒子m个球，球相同，盒子相同，盒子可以为空，有多少种放法。
- $1 \leq m \leq 1000, 1 \leq n \leq m$
- ~~相信大家都会做(雾)~~
- 球盒问题一共有八道，有兴趣的同学可以思考一下八道题分别怎么做

一道遥远的例题2

- 由于盒子相同，不妨设
- 第1个盒子的球数≥第二个盒子的球数≥…≥第n个盒子
- 那我们先不考虑dp方程怎么写，先考虑要你放球的话，你怎么确保上述式子在你放球的过程恒成立？

一道遥远的例题2

- 由于盒子相同，不妨设
- 第1个盒子的球数≥第二个盒子的球数≥…≥第n个盒子
- 那我们先不考虑dp方程怎么写，先考虑要你放球的话，你怎么确保上述式子在你放球的过程恒成立？
- 一个简单的想法是，我们可以选择一个盒子的前缀，往这个前缀的所有盒子里都放一个球，下一次选择一个更短的前缀，再往这个前缀的所有盒子里都放一个球，这样就可以保证盒子中的球数是不上升的。

一道遥远的例题2

- 由于盒子相同，不妨设
- 第1个盒子的球数≥第二个盒子的球数≥…≥第n个盒子
- 怎么用dp模拟这个过程？

一道遥远的例题2

- 由于盒子相同，不妨设
- 第1个盒子的球数≥第二个盒子的球数≥…≥第n个盒子
- 怎么用dp模拟这个过程？
- 考虑$f[i][j]$表示i个盒子存j个球的方案数
- 对于每个状态，有两个选择，一是对所有的盒子放入一个球，另一个是新增一个盒子
- 可得dp方程$f[i][j] = f[i-1][j] + f[i][j-i]$

一道全新的例题3

- 小H是一个热爱跑步的孩子，她给自己制定了跑步计划。小H计划跑n米，其中第i分钟跑x_i(x_i是正整数)米。
- 随着时间的增加，小H会越来越累，所以小H的计划必须满足对于任意的$i(i > 1)$都满足$x_i <= x_i - 1$。
- 现在小H想知道她有多少种方式安排自己计划，答案对p取模(p不一定是质数)。
- $1 \leq n \leq 100000$
- ~~相信大家也都会做(雾)~~

一道全新的例题3

- 一个很简单的想法是采用远古例题1的方法，求$f[n][n]$。但显然会TLE+MLE。

一道全新的例题3

- 一个很简单的想法是采用远古例题1的方法，求$f[n][n]$。但显然会TLE+MLE。
- 考虑分块。设置一个值B，那么xi中大于等于B的数不会超过$\frac{n}{B}$个。对这一部分数，可以枚举"盒子数"，再在每个盒子里铺上B个球，跑一遍例题1中的方法。时间复杂度$O(n * \frac{n}{B})$.

一道全新的例题3

- 一个很简单的想法是采用远古例题1的方法，求$f[n][n]$。但显然会TLE+MLE。
- 考虑分块。设置一个值B，那么xi中大于等于B的数不会超过$\frac{n}{B}$个。对这一部分数，可以枚举"盒子数"，再在每个盒子里铺上B个球，跑一遍例题1中的方法。时间复杂度$O(n*\frac{n}{B})$.
- 考虑小于B的数，可以像做完全背包一样做一遍计数dp(设有B种物品，第i种物品的重量为i，背包能装重量为n的物品，问有多少种存放方法)。时间复杂度是$O(n*B)$的
- 因此在$\mathrm{B}=\sqrt{n}$时时间复杂度取最小值。时间复杂度$O(n^{1.5})$

一道全新的例题3

- 多说一句，如果大家会五边形数定理的话，这道题可以在$O(nlogn)$的时间求解(前提是你会任意模数NTT)，并且可以求解多组询问的情况，但那涉及多项式相关知识，和本次讲课的主题无关，有兴趣的可以去了解一下。
- https://baike.baidu.com/item/%E4%BA%94%E8%BE%B9%E5%BD%A2%E6%95%B0/9459853

数论分块

- 对于一个数n，$\lfloor\frac{n}{i}\rfloor$的值和$\left\lceil\frac{n}{i}\right\rceil$的值都是$O(\sqrt{n})$级别
- 证明。

例题4

- 给定一个$r*c$的矩阵，矩阵的每个格子有一些正整数。如果从左上角走到右下角，且每次只能向右或向下走到相邻格子，那么使得路径上所有数的乘积不小于 n 的路径有多少条？
- $1 \le r,c \le 300, 1 \le n \le 10^6$

例题4

- 朴素的想法就是$f[i][j][k]$表示走到格子(i,j)时，乘积为k的方案数。时间复杂度$O(rcn)$
- 这种做法没有优化的空间

例题4

- 另一种朴素的想法就是$f[i][j][k']$表示走到格子(i,j)时，还需乘k'才等于n的方案数。这里可以注意到$k'=\lceil\frac{n}{k}\rceil$.时间复杂度$O(rcn)$.
- 这种转移的可行性建立于$\left\lceil\frac{\left\lceil\frac{n}{a}\right\rceil}{b}\right\rceil=\lceil\frac{n}{ab}\rceil$
- 再注意到k'的取值只有$\sqrt{n}$种，因此可以将时间复杂度优化到$O(rc\sqrt{n})$

斯特林数

- Q1：n个不同的元素构成m个圆排列的方案数为多少
- A1：令$s[n,m] = s[n-1,m-1] + (n-1) * s[n-1,m]$
- $s[n,m]$为第一类无符号斯特林数
- Q2：$x * (x+1) * \cdots * (x+n-1)$的$x^m$项系数为多少
- A2：令$s[n,m] = s[n-1,m-1] + (n-1) * s[n-1,m]$
- 这体现了第一类无符号斯特林数和升幂函数的关系
- Q3： $x * (x-1) * \cdots * (x-n+1)$的$x^m$项系数为多少
- A3：令$s'[n,m] = s'[n-1,m-1] - (n-1) * s'[n-1,m]$
- $s'[n,m]$为第一类有符号斯特林数，可以证明$|s'[n,m]| = s[n,m]$

斯特林数

- Q4：n个不同的球，m个相同的盒子，盒子不允许为空，有多少种放法
- A4：令$s\{n,m\} = s\{n-1,m-1\} + \mathrm{m} * s\{n-1,m\}$
- S{n,m}为第二类斯特林数
- 或者容斥原理，有
- $S\{n,m\} = \frac{1}{m!}\sum_{k=0}^{m}(-1)^k C_m^k (m-k)^n$
- 将上述式子反演，可得
- $n^k = \sum_{i=0}^{k} i! * C_n^i * S\{k,i\}$

树形DP

- 把DP放在树上，把叶子设成初状态，树上的边为转移途径，自叶子到根转移的一类dp。
- 由于二叉树在转移时比较方便，在过去大家又懒得写邻接表的年代，常常将多叉树转二叉树进行转移。
- 现在可以用邻接表或vector转移。

JZPTREE

- 给出n个结点的一棵树，对每一个点x求所有点到x的距离的k次方之和。
- 即$\sum_{i=1}^{n}\sum_{j=i+1}^{n} dis(i,j)^k$
- $1 \le n \le 50000, 1 \le k \le 500$

- 为了大家熟悉树形DP，可以先思考k=1怎么做

JZPTREE

- 对$k=1$，设1号点为根。
- 令$sum[x]$表示x的子树中所有的点到x点的距离之和，$sz[x]$表示x的子树的点数。
- $sz[x]=\sum_y sz[y]+1$ y是x的儿子结点
- $sum[x]=\sum_y sum[y]+sz[y]$ y是x的儿子结点
- 第一次dfs $ans+=sum[x]$
- 第二次dfs 不断维护$sum[1]$的值，向某个子树x走的时候，删除子树的值$sum[x]$，同时$sum[1]$的值加上$(sz[1]-sz[x])$
- 上述做法在$k>1$的情况下受到怎样的限制？

JZPTREE

- 利用斯特林数的反演公式，可以得到$dis(x,y)^k = \sum_{i=0}^{k} i! * C_{dis(x,y)}^{i} * S\{k,i\}$
- 重新考虑$sum[x]$数组
- $sum[x] = \sum_{y\in x} dis(x,y)^k = \sum_{y\in x} \sum_{i=0}^{k} i! * C_{dis(x,y)}^{i} * S\{k,i\}$

$$= \sum_{i=0}^{k} i! * S\{k,i\} \sum_{y\in x} C_{dis(x,y)}^{i}$$

dp[x,i]= $\sum_{y\in x} C_{dis(x,y)}^{i} = \sum_{z\in son[x]} \sum_{t\in z} C_{dis(y,z)+1}^{i} = \sum_{z\in son[x]} dp[z,i] + dp[z,i-1]$

处理dp[x,i]只需做k次树形dp。求ans和前面一样

HNOI2018 道路

- 给出一个有n个叶子结点的二叉树。对每个非叶子结点可以选择一条边翻新。对每个叶子结点，有三个属性a_i, b_i, c_i，若叶子结点到根经过x条未翻新的左边，y条未翻新的右边，则产生代价$a_i(b_i + x)(c_i + y)$。求一个翻新方案，使代价最小。
- $n \leq 20000$,树高不超过40

HNOI2018 道路

- 既然树高已经限制在40了，那也就是未翻新的L边和R边数量不超过40.不妨设$f[i][L][R]$表示根节点到i点有L条未翻新的左边，R条未翻新的右边。自叶子到根，每次选择左边或者右边翻新进行转移，并更新最小值。
- 时间复杂度$O(nh^2)$