DP

kczno1

2020.2.13.

背包DP

01背包问题：有 n 个物品，每个物品有重量 w_i 和价值 v_i ，求在总重量不超过 W 的情况下，能选出的物品的最大价值和。
$f_{i,j}$ 表示前 i 个物品，选出的物品重量总和为 j ，能得到的最大价值和。
$f_{i,j} = \max\{f_{i-1,j-w_i} + v_i, f_{i-1,j}\}$ 。
时间复杂度为 $O(nW)$ 。
若倒序枚举第二维，第一维可省略不记。
$f_{i,j}$ 表示前 i 个物品，选出的物品价值总和为 j ，所需要的最小重量和。时间复杂度 $O(n \cdot ans)$ 。

背包DP

完全背包问题：有 n 种物品，每种物品有重量 w_i 和价值 v_i ，且每种物品数量无限，求在总重量不超过 W 的情况下，能选出的物品的最大价值和。

$f_{i,j}$ 表示前 i 种物品，选出的物品重量总和为 j ，能得到的最大价值和。

$f_{i,j} = \max\{f_{i,j-w_i} + v_i, f_{i-1,j}\}$ 。

时间复杂度为 $O(nW)$ 。

若顺序枚举第二维，第一维可省略不记。

$f_{i,j}$ 表示前 i 种物品，选出的物品价值总和为 j ，所需要的最小重量和。时间复杂度 $O(n \cdot ans)$ 。

背包DP

多重背包问题：有 n 种物品，每种物品有重量 w_i 和价值 v_i ，且每种物品数量为 c_i，求在总重量不超过 W 的情况下，能选出的物品的最大价值和。
$f_{i,j}$ 表示前 i 种物品，选出的物品重量总和为 j ，能得到的最大价值和。
$f_{i,j} = \max\{f_{i,j-k\cdot w_i} + k \cdot v_i | 0 \le k \le c_i\}$ 。
时间复杂度为 $O(\sum c_i W)$ 。
对于每个 i ，在转移的时候，将 j 按照 $j \bmod w_i$ 分类，相当于每次从一个区间转移，使用单调队列优化，时间复杂度为 $O(nW)$ 。

背包DP

树形依赖背包问题：有 n 个物品，每个物品有重量 w_i 和价值 v_i 。
且有 $n-1$ 条依赖关系，第 i 条关系形如第 u_i 个物品如果要选，就必须同时选择第 v_i 个物品。其中 $u_i \to v_i$ 形成一棵内向树。
求在总重量不超过 W 的情况下，能选出的物品的最大价值和。
$f_{i,j}$ 表示考虑 DFS 序为 $i \ldots n$ 的物品及他们之间的依赖关系，选出的物品重量总和为 j ，能得到的最大价值和。
设 DFS 序 $= i$ 的点为 x ，x 的子树大小为 $size_x$ ，则
$f_{i,j} = \max\{f_{i+1,j-w_x} + v_x, f_{i+size_x,j}\}$ 。

题目

Easy:
金明的预算方案
Medium:
HDU1561 The more, The Better
HNOI2007 梦幻岛宝珠

树形DP

顾名思义，树型动态规划就是在“树”的数据结构上的动态规划。
一般以每棵子树为子结构，在父亲节点合并。
有时巧妙利用 BFS 或 DFS 序，或将多叉树转化成二叉树，可以优化或简化问题。
有时可以用树分治，以及树上的数据结构来优化。
有时需要在虚树上 DP 。

树形DP

顾名思义，树型动态规划就是在“树”的数据结构上的动态规划。
一般以每棵子树为子结构，在父亲节点合并。有时需要“换根”DP 。
有时巧妙利用 BFS 或 DFS 序，或将多叉树转化成二叉树，可以优化问题，或得到好的解决方法。
有时可以用树分治，以及树上的数据结构来优化。
有时需要在虚树上 DP 。
可以扩展到各类树形数据结构上，比如 Trie 上的 DP、AC 自动机上的 DP、后缀自动机上的 DP 等。
可以扩展到基环外向树、仙人掌等。

树形DP

树的中心问题：给出一棵边带权的树，求一个点，使得此点到树中其他结点的最远距离最近。
记 f_i 为以 1 为根，点 i 的子树里的点到 i 最远距离。
$f_i = \max\{f_son + w(i, son)\}$ 。
记 g_i 为所有点到 i 最远距离。
$g_i = \max\{f_i, g_{fa_i} + w(i, fa_i)\}$ 。
这样有个问题就是如果 f_{fa_i} 是从 i 那里转移得到的，就可能出错。
f, g 都记一个最大值和次大值，保证这两个值必须由不同的结点转移得到，问题就解决了。

题目

Medium:
ZJOI2018 骑士
HNOI2007 梦幻岛宝珠
JSOI2007 文本生成器
Hard:
SHOI2008 cactus仙人掌图
HNOI2014 世界树
NOI2014 购票
运算数

数位DP

一般的数位 DP 是统计某个区间满足某个性质的数的个数。
例题：windy数
不含前导零且相邻两个数字之差至少为 2 的正整数被称为 *windy* 数。求 $[A, B]$ 内共有多少个 *windy* 数。
$1 \leq A \leq B \leq 10^9$ 。

数位DP

首先 $[A,B]=[1,B]-[1,A-1]$，所以下面只考虑询问为 $[1,n]$ 的情况。
对于一个 $<n$ 的数，它从高到低一定会出现某一位，前面的位它跟 n 一样，这一位它 $<n$。
所以我们只需枚举它第一次 $<n$ 的那一位，以及那一位是多少，则之后的每一位就都是 $[0,9]$ 任取。
可以用 $dp_{i,x}$ 表示第 $i+1$ 位为 x，第 1 位到第 i 位在 $[0,9]$ 任取的 *windy* 数的个数。
枚举第 i 位转移，预处理时间复杂度为 $O(\log^3)$，单次询问复杂度为 $O(\log^2)$。

数位DP

另一种方法是直接对 n 进行 DP 。
$dp_{i,j,0/1}$ 表示确定了 $\geq i$ 的数位，第 i 位的值为 j ，目前是 $< n$ 还是 $= n$ 的方案数。
枚举第 $i+1$ 位的值转移，单次询问时间复杂度为 $O(\log^3)$ 。
这两种方法都需要注意前导 0 的影响。

题目

Medium:
Beautiful numbers
不要62
B-number
Xorequ
SCOI2013 数数
Hard:
Counting Hexagons
SDOI2013 淘金
操作序列计数

状压DP

传统的状压 DP 一般是以一个集合内的元素的信息为状态，状态总数为指数级别的动态规划；一般维数较多，但每维的值域非常小。
状压 DP 还包括插头 DP ， DP 套 DP。

题目

Medium:
NOI2001 炮兵阵地
NOIP2016 愤怒的小鸟
NOIP2017 宝藏
SDOI2009 学校食堂
Hard:
NOI2015 寿司晚宴

插头DP

《基于连通性状态压缩的动态规划问题》陈丹琦

题目

Medium:
HNOI2007 神奇游乐园
WC2008 游览计划
Formula 1
SCOI2011 地板
Hard:
SCOI2012 喵星人的入侵

DP套DP

DP 套 DP 就是通过一个 DP 来计算使得另一个 DP 的输出为特定值的输入的数量。
外层 DP 的状态是子 DP 的所有状态的值。

Hero meet devil

给你一个只由 ACGT 组成的字符串 $S(|S| \leq 15)$，对于每个 $i \in [0, |S|]$，求有多少个只由 ACGT 组成的长度为 $m(m \leq 1000)$ 的字符串 T，使得 $LCS(S, T) = i$。
LCS 就是最长公共子序列。

Hero meet devil

给你一个只由 ACGT 组成的字符串 $S(|S| \leq 15)$，对于每个 $i \in [0, |S|]$，求有多少个只由 ACGT 组成的长度为 $m(m \leq 1000)$ 的字符串 T，使得 $LCS(S, T) = i$。
LCS 就是最长公共子序列。

首先考虑给你 S, T，怎么求 $LCS(S, T)$。
可以令 $f_{i,j}$ 表示 $LCS(T_{1\ldots i}, S_{1\ldots j})$。
那么我们用 $g_{i,f'}$ 表示确定了 $T_{1\ldots i}$，且 $f_{i,j} = f'_j$ 的方案数。
注意到 $f_{i,j} - f_{i,j-1} \in \{0, 1\}$，故 f' 最多只有 $2^{|S|}$ 种。
时间复杂度为 $O(m2^{|S|})$。

题目

Medium:
Square
LCS Again
Jigsaw Puzzle

决策单调性

对于规模为 n 的问题，若一个转移方程其状态数为 $O(n^x)$，每个状态转移数为 $O(n^y)$，则称这个问题是 xD/yD 的。
对于 1D/1D 模型 $f_i = \min_{j=1}^{i-1}\{g_j + w(j,i)\}$。
令 $k(i)$ 表示 f_i 最小的最优决策点，则决策单调性指 $\forall i \le j, k(i) \le k(j)$。
如果给定 g，且 f 满足决策单调性，我们可以用分治将这个模型优化至 $O(n \log n)$。
即每次要求 $f_{l\ldots r}$ 时，先求 $k(mid)$ 和 f_{mid}，再递归两边，递归时利用 $k(mid)$ 来限制转移范围。

决策单调性

```
void DP(int l, int r, int k_l, int k_r) {
  int mid = (l + r) / 2, k = k_l;
  // 求状态f[mid]的最优决策点
  for (int i = k_l; i <= min(k_r, mid - 1); ++i)
    if (w(i, mid) < w(k, mid)) i = k;
  f[mid] = w(k, mid);
  // 根据决策单调性得出左右两部分的决策区间，递归处理
  if (l < mid) DP(l, mid - 1, k_l, k);
  if (r > mid) DP(mid + 1, r, k, k_r);
}
```

四边形不等式

现在考虑 g_i 依赖 f_i 的情况，比如 $g_i = f_i$ 。
那么分治做法就不成立了。
如果 $\forall a \leq b < c \leq d, w(a,b) + w(b,d) \leq w(a,d) + w(b,c)$ ，则称函数 w 满足四边形不等式，易证此时 f 必定满足决策单调性。
当 w 满足四边形不等式时，我们依旧可以将这个模型优化至 $O(n \log n)$ 。

四边形不等式

如果函数 w 满足四边形不等式，那么对于 i 和 $i+1$，一定存在 j，$\leq j$ 时用 i 转移优于 $i+1$，$>j$ 时 $i+1$ 优于 i。
所以对于 $\forall j \in [1,i]$，一定存在一个连续区间 j 是 $[1,i]$ 中的最优决策，且这个区间(如果非空)随着 j 增加是单调上升的。
所以我们可以从小到大枚举 i，用一个队列来维护当前所有可能的决策点，同时队列每个元素记录哪一个区间它是最优值，
每次如果队头区间的右端点已经 $<i$，就删除队头；
每次插入 i 这个新决策的时候就从后往前扫描队列的每个元素，对于每一个老决策，如果区间的起点处还是新决策更优，则删除老决策，继续扫描下一个；否则在这个老决策的区间中二分出转折点，将新决策插入队列尾部，结束操作。

题目

Medium:
HNOI2008 玩具装箱
USACO 土地购买
POI2011 Lightning Conductor
IOI2000 邮局
IOI2014 Holiday

斜率优化

$f(n) = \min_{i=1}^{n-1}\{a_n x(i) + b_n y(i)\}$ ，其中 $x(i), y(i)$ 依赖于 i 和 $f(i)$ 。

斜率优化

$f(n)=\min_{i=1}^{n-1}\{a_n x(i)+b_n y(i)\}$，其中 $x(i),y(i)$ 依赖于 i 和 $f(i)$。

以 $x(i)$ 为横轴，$y(i)$ 为纵轴建立平面直角坐标系，这样每个 $(x(i),y(i))$ 就可以看成坐标系上的一个点。

我们目标是最小化 $P=ax+by$，设 $b>0$，则 $y=-\frac{a}{b}x+\frac{P}{b}$，我们的任务就是让这条直线的纵截距最小。

可以想象有一条直线自负无穷向上平移，所碰到的第一个点就是最优决策点。

所以可以发现一个重要性质：最优决策点必然在平面点集的下凸包上。

斜率优化

首先考虑比较简单的情况：询问直线的斜率和点的横坐标同时满足单调性。
由于斜率变化是单调的，所以最优决策点必然在下凸包上单调移动。
只需要用一个单调队列维护下凸包即可。
时间复杂度 $O(n)$ 。

斜率优化

现在考虑一般的情况。
对于下凸壳，相邻两点的斜率必定是单调上升的。
对于询问的斜率 k ，最优决策点必定满足这样的性质：该点两侧的边的斜率 k_1, k_2 满足 $k_1 \leq k \leq k_2$ 。
所以我们可以通过二分来确定最优决策点。
插入点的时候，首先根据横坐标二分到相应的位置，然后对两侧分别删掉一些点维护凸性(也有可能是把自己删除)，用平衡树维护即可。
时间复杂度为 $O(n \log n)$ 。
另一种方法是使用 CDQ 分治。将每个决策看作一次查询和插入，这样就变成了上一页的情况。时间复杂度为 $O(n \log n)$ 。

题目

Medium:
HNOI2008 玩具装箱
NOI2005 瑰丽华尔兹
APIO2010 特别行动队
Hard:
NOI2007 货币兑换

WQS二分

王钦石在《浅析一类二分方法》中首次介绍了这种二分方法，把一类看似与二分毫不相关的限制个数的最优化问题通过二分转化为不限制个数、可以用简单高效的算法解决的问题，从而高效的解决限制个数的问题。

例1

给定一个正整数序列 $a_{1\ldots n}$，要求将 a 分成若干段，记每段的和为 S_i，求最小的 $\sum S_i^2$。

例1

给定一个正整数序列 $a_{1\ldots n}$，要求将 a 分成若干段，记每段的和为 S_i，求最小的 $\sum S_i^2$。

显然，把每个数分为一段答案最优。

例2

给定一个正整数序列 $a_{1\ldots n}$，要求将 a 分成若干段，记每段的和为 S_i，求最小的 $\sum(AS_i^2+BS_i+C)(A,B,C\in\mathbb{N}^*)$。

例2

给定一个正整数序列 $a_{1\ldots n}$，要求将 a 分成若干段，记每段的和为 S_i，求最小的 $\sum(AS_i^2+BS_i+C)(A,B,C\in\mathbb{N}^*)$。

记 $f(i)$ 为前 i 个数的最优答案，枚举最后一段的长度进行转移，时间复杂度为 $O(n^2)$。
斜率优化，复杂度为 $O(n)$。

例3

给定一个正整数序列 $a_{1\ldots n}$，要求将 a 分成若干段，记每段的和为 S_i，求最小的 $\sum(AS_i^3+BS_i^2+CS_i+D)(A,B,C,D\in\mathbb{N}^*)$。

例3

给定一个正整数序列 $a_{1\ldots n}$，要求将 a 分成若干段，记每段的和为 S_i，求最小的 $\sum(AS_i^3+BS_i^2+CS_i+D)(A,B,C,D\in\mathbb{N}^*)$。

暴力同例2，$O(n^2)$。
由决策单调性可优化至 $O(n\log n)$。

例4

给定一个正整数序列 $a_{1\ldots n}$，要求将 a 分成不超过 k 段，记每段的和为 S_i，求最小的 $\sum(S_i^2)$。

例4

给定一个正整数序列 $a_{1\ldots n}$，要求将 a 分成不超过 k 段，记每段的和为 S_i，求最小的 $\sum(S_i^2)$。

记 $f(i,j)$ 为将前 i 个数分为 j 段的最优答案，枚举最后一段的起始点转移，时间复杂度为 $O(n^2k)$。
对于固定的 j，同例 2 一样可以斜率优化，时间复杂度为 $O(nk)$。
也可以利用决策单调性优化。记 $p(i,j)$ 为 $f(i,j)$ 的最优决策中最小的一个，显然 $p(i-1,j) \le p(i,j) \le p(i,j-1)$。时间复杂度为 $O(n^2)$。

例4

记 $ans(i)$ 为将该序列分成 i 段的最小平方和，显然 $ans(i-1)-ans(i) \geq ans(i)-ans(i+1)$ 。
对于某个 C ，求最小的 $\sum(S_i^2+C)$ 。如果最优方案段数刚好为 k ，那么此答案减去 $C \times k$ 就是原问题的答案。
显然，对于一个 C ，最优段数 $\in [\max\{i|ans(i)-ans(i-1)>C\}, \min\{i|ans(i)-ans(i+1)<C\}]$ 。
所以 C 越大，最优段数越小。
所以只需二分 C ，求出最优段数最小值 $\leq k$ 的最小的 C ，那么此时的最优答案减去 $C \times k$ 就是原问题的答案。
时间复杂度为 $O(n \log \sum a_i)$ 。

题目

Medium:

tree

IOI2016 aliens

Hard:

九省联考2018 林克卡特树

yanQval 的生成树