

FOI2022 算法夏令营

基础班 day7

2022.07.22

福建师大附中
邹品聪



经典问题

- 有 n 种物品，每种物品有一个，且有一个价值 v_i 和一个重量 w_i ，而你有一个背包，至多能容纳重量之和为 W 的物品，求所选物品的价值和最大值？
- $n \leq 1000; v_i \leq 10^6; w_i, W \leq 10000$
- 1s & 512 MB
- 01 背包问题



动态规划 (DP)

- 动态规划 (Dynamic Programming, DP) 是通过把原问题分解为相对简单的子问题的方式求解复杂问题的方法
- 动态规划常常适用于有重叠子问题和最优子结构性质的问题（在OI中也用于解决计数问题）
- 动态规划背后的基本思想非常简单。大致上，若要解一个给定问题，我们需要解其不同部分（即子问题），再根据子问题的解以得出原问题的解。
- 有时需要进行优化（优化转移过程或优化状态设计）
- 需保证状态**无后效性**
- 注意边界条件

PART 01

背包 DP

经典 dp

F012022 首 day7 课件



回到之前的问题

- 很明显，有个很暴力的方法是枚举每个物品选与不选，但时间就是 $O(2^n)$ ，显然不可取。
- 注意到如果已经考虑完了前 i 个物品选与不选，且当前选择物品重量和为 j ，则当前选择物品价值和越大越好
- 即设 $f_{i,j}$ 表示考虑完了前 i 个物品，当前选择物品重量和为 j ，当前选择物品价值和的最大值
- 那么现在考虑第 i 个物品选择情况，枚举 j ，如果选择，则为 $f_{i-1,j-w_i}$ 转移而来，再加上选择产生的收益 v_i ；如果不选择，则为 $f_{i-1,j}$ 转移而来，不产生任何收益；两者取最大值



回到之前的问题

- 状态转移方程为 $f_{i,j} = \text{Max}\{f_{i-1,j-w_i} + v_i, f_{i,j}\}$
- 最后答案为 $\text{Max}\{f_{i,0}, f_{i,1}, \dots, f_{i,W}\}$
- 时间复杂度: $O(n \times W)$



更进一步

- 有 n 种物品，每种物品有无限个，且有一个价值 v_i 和一个重量 w_i ，而你有一个背包，至多能容纳重量之和为 W 的物品，求所选物品的价值和最大值？
- $n \leq 1000; v_i \leq 10^6; w_i, W \leq 10000$
- 1s & 512 MB
- 完全背包问题

FOI2022竞赛样题





更进一步

- 考虑继续使用之前的想法
- $f_{i,j}$ 表示考虑完了前 i 个物品，当前选择物品重量和为 j ，当前选择物品价值和的最大值
- 枚举 i, j, k ，其中 k 表示第 i 个物品选了 k 个，那么转移为 $\text{Max}\{f_{i-1, j-k \times w_i} + k \times v_i\}$
- 枚举的 k 至多到 W ，最坏时间复杂度 $O(n \times W \times W)$ ，不是很能接受



更进一步

- 有什么地方可以优化的？
- 注意到当从 $f_{i-1,j-2 \times w_i}$ 转移到 $f_{i,j}$ 和从 $f_{i-1,j-2 \times w_i}$ 转移到 $f_{i,j-w_i}$ 只差了一个 v_i ，可以把这个过程改为由 $f_{i-1,j-2 \times w_i}$ 先转移到 $f_{i,j-w_i}$ ，再转移到 $f_{i,j}$ 。
- 状态设计改为 $f_{i,j}$ 表示考虑了前 i 个物品，其中选了的重量和不超过 j 的收益最大和。
- 则 $f_{i,j} = \text{Max}\{f_{i-1,j}, f_{i,j-w_i} + v_i\}$ 。
- 最终答案为 $f_{n,W}$ ，时间复杂度 $O(n \times W)$ 。



深度思考

- 如果是 $n \leq 100; w_i \leq 10^7, W \leq 10^9; v_i \leq 100$ 该怎么做？
- 注意到 v 很小，考虑与 v 有关的状态设计
- $f_{i,j}$ 表示考虑了前 i 个物品，当前选择物品价值和为 j 的重量最小值
- $$f_{i,j} = \min\{f_{i-1,j}, f_{i-1,j-v_i} + w_i\}$$
- 时间复杂度 $O(n \times \sum v_i)$
- 注意背包问题转移方程一般只跟上一个有关，可使用滚动数组优化



二维（高维）背包

- 有 n 种物品，每种物品有一个，且有一个价值 v_i ，一个重量 w_i 和体积 g_i ，而你有一个背包，至多能容纳重量之和为 W 且体积和不超过 V 的物品，求所选物品的价值和最大值？
- $n, v_i, w_i, g_i, V, W \leq 100$
- 直接多开一维记录当前选择体积即可



分组背包

- 有 n 种物品，每种物品都在某个组中，每组至多选一个，且有一个价值 v_i 和一个重量 w_i ，而你有一个背包，至多能容纳重量之和为 W 的物品，求所选物品的价值和最大值？
- 状态设计由考虑完前 i 个物品改为考虑完前 i 组，再在每组中枚举重量 j 与选择哪个物品



多重背包

- 有 n 种物品，每种物品有 s_i 个，且有一个价值 v_i 和一个重量 w_i ，而你有一个背包，至多能容纳重量之和为 W 的物品，求所选物品的价值和最大值？
- 时间复杂度要求 $O(n \times W \times W)/O(n \log n \times W)$

FOI2022首日讲解





多重背包

- 很显然与完全背包的暴力做法一样，暴力枚举本次选几个，最坏时间复杂度 $O(n \times W \times W)$ 。
- 考虑对这个枚举过程优化，对于每种物品，可以将他二进制拆成价值和重量分别为 $v_i \times 1, w_i \times 1; v_i \times 2, w_i \times 2; v_i \times 4, w_i \times 4; v_i \times 8, w_i \times 8\dots$ 的若干个只能选一次的物品。这样拆分在原方案选了 x 个的物品相当于选了 x 的二进制拆分的那些物品。
- 每种物品最多拆出 $\log n$ 个物品，时间复杂度 $O(n \log n \times W)$
- 注意要多拆一个倍数为 s_i 减去其二进制下最高位的物品。
- 其实完全背包也可以使用这个方法拆成 01 背包做



求方案数&具体方案

- 方案数：多开一个数组记方案数，每次转移的时候，如果比现答案更劣，不更新；与现答案相同，方案数 +1 ；比现答案更优，方案数清为 1
- 具体方案：多开一个数组记录如何转移的，每次转移并且更新的时候，将该数组更新为这次转移来的位置，最后从答案的位置由后往前访问该数组直到初始状态，再正序输出方案
- 如果要求方案中字典序最小则与求方案数的过程类似



[USACO2. 2]集合 Subset Sums

- 有一个包含元素 $1 - n$ 的集合，求划分方案数，使得将原集合恰好划分为两个子集，使得这两个子集的元素和相等
- 原题 $n \leq 39$; 加强 $n \leq 300$



[USACO2. 2]集合 Subset Sums

- 暴力枚举子集为 $O(2^n)$
- 注意到划分出的子集和一定为 $(1 + n) * n/4$ ，当我们知道前一半的元素选了哪些作为第一个子集，那么后一半的元素中被选为第二个子集的元素和就固定了。优化枚举方式，先只枚举后一半的元素，记录不同的元素和对应选取方案数。再枚举前一半元素，对答案贡献为对应后一半元素和的方案个数。时间复杂度为 $O(\frac{n}{2})$
- 那么一般地，我们记 $f_{i,j}$ 表示考虑前 i 个元素选取情况，对应元素和为 j 的方案数。对于第 i 个元素，如果选择则从 $f_{i-1,j-i}$ 转移而来；不选则从 $f_{i-1,j}$ 转移而来
- 元素和量级为 n^2 ，时间复杂度为 $O(n^3)$



[NOIP2006 提高组] 金明的预算方案

- 有 m 个物品，每个物品有价格 v_i 和重要度 p_i ，被选取最终产生收益为 $v_i * w_i$ 。每个物品可为附件或主件。附件会从属于一个主件，选取该附件必须也选取其从属的主件。每个主件会有不超过两个附件，附件不会再有附件。
- 你能选取不超过 n 元的物品，求选取物品的价格*重要度之和最大值
- $1 \leq n \leq 3.2 * 10^4$, $1 \leq m \leq 60$, $0 \leq v_i \leq 10^4$, $1 \leq p_i \leq 5$, 答案不超过 $2 * 10^5$



[NOIP2006 提高组] 金明的预算方案

- 我们将每个主件和从属他的附件拎出来单独考虑。
- 有四种选取情况：一、一个都不选；二、只选取了该主件；三、选取了该主件和其中一个附件（要求有至少一个附件）；四、选取了该主件和两个附件（要求有两个附件）
- 因此设 $f_{i,j}$ 表示考虑完前 i 个主件与他的附件的选取情况，其中价格和为 j ，那么对下一个主件，安装上面的选取分类依次转移
- 时间复杂度 $O(nm)$



背包问题——总结

- 一般来说，背包问题的形式一般是有若干可选物品，有价值和重量，有选取重量的限制（背包），最终要最优化价值
- 背包问题的状态设计也一般是依次考虑，对于当前考虑物品分别选与不选分别转移，dp 值为在某相同条件下的最优化
- 因为背包问题一般都是由前 $i-1$ 个物品转移到前 i 个物品，因此一般可用滚动数组优化空间



PART 02

区间 DP

经典 dp × 2

F012022 直播 day7 课件



石子合并问题

- 有 n 堆石子，排成一行，每堆石子 a_i 个，每次可以选择相邻两堆合并，合并代价为两堆石子数之和，合并后会形成一个新的数量为两堆石子和的石子堆，求将所有石子合并为一整堆的代价和最小值。
- $n \leq 100; a_i \leq 100$

FOI2022直系





石子合并问题

- 观察每次的合并操作，我们是将相邻的两堆合并，且这相邻的两堆也一定是原先连续的一段。
- 因此若要将第 $l \sim r$ 堆石子合并为一堆，可以枚举断点 $k (l \leq k \leq r)$ 使得最后一次合并操作是由 $l \sim k$ 合并出的石子堆与 $k \sim r$ 合并出的石子堆合并。
- 设 $f_{i,j}$ 表示将 $i \sim j$ 的石子堆合并为一堆产生的代价，转移方程为 $f_{i,j} = \min\{f_{i,k} + f_{k,j} + sum_{l,r}\} (i \leq k \leq j)$ 。
- 最后答案为 $f_{1,n}$ ，时间复杂度 $O(n^3)$ 。
- 本题可用四边形不等式优化到 $O(n^2)$ ，这里不作深入介绍。



括号匹配

- 有一个长度为 n 仅有 “(” 、 “)” 、 “[” 和 “]” 组成的字符串，求其中最长的子序列使得该序列是个完美的括号匹配
- 一个字符串 A 被视为完美的括号匹配当且仅当它的机构是 (B) 、 $\{B\}$ 或者 BC ，其中 B 、 C 均为完美的括号匹配或者空串
- $n \leq 100$

F012021/day1/课件





括号匹配

- 思考题目给的定义方式，考虑子串 $l - r$ 中的最长完美匹配的子序列中最外一层的构成
- 如果是 (A) 这样的，就从子串 $l + 1 - r + 1$ 转移；如果是 AB 这样的，就从子串 $l - k$ 和子串 $k + 1 - r$
- 设 $f_{i,j}$ 表示子串 $i - j$ 中的最长完美匹配的子序列长度
- $f_{i,j} = \text{Max}\{f_{i+1,j-1} + 2 \ (i < j \&& s_i, s_j \text{ 分别是 '()' 或 '[]'}), f_{i,k} + f_{k+1,j} \ (i \leq k < j)\}$
- 时间复杂度 $O(n^3)$



回文字序列数

- 有一个长度为 n 的字符串，求其中回文字序列数
- 这里选取位置不同的子序列即视为不同子序列
- $n \leq 1000$



回文字序列数

- 设 $f_{i,j}$ 表示子串 $i - j$ 中的回文字序列数
- 考虑其中包含位置 i 和 j 的子序列，那么必须 $s_i = s_j$ 也就是能构成新的一个回文字序列的最外层，其对答案贡献是子串 $i + 1 - j - 1$ 能构成的回文字序列数也就是 $f_{i+1,j-1}$
- 考虑位置 i 和 j 至多只包含一个的子序列，那么由 $f_{i,j-1}$ 和 $f_{i+1,j}$ 继承而来
- 注意如果位置 i 和 j 都不包含，那么在 $f_{i,j-1}$ 和 $f_{i+1,j}$ 会各算一次，因此要减去 $f_{i+1,j-1}$
- 时间复杂度 $O(n^2)$



凸多边形的划分

- 给定一个具有 n 个顶点的凸多边形，将顶点从 1 至 n 标号，每个顶点的权值都是一个正整数。将这个凸多边形划分成 $n - 2$ 个互不相交的三角形，试求这些三角形顶点的权值乘积和至少为多少。
- $n \leq 50$, 点权 $\leq 10^9$

FOI2022省赛





凸多边形的划分

- 观察我们一次三角形划分，当我们任意连出一个三角形时，会发现会分出至多两个多边形，且这两个多边形的所有顶点在原多边形上连续
- 因此我们考虑最后一次划分出的三角形，那么接下来就是在连续一段顶点上进行多边形划分的子问题
- 设 $f_{i,j}$ 表示从点 $i - j$ 连续若干个点构成的多边形内进行划分的最小价值
- 转移时，考虑 $j - i$ 这条边形成的三角形，枚举第三个点 k 那么就划分为点 $i - k$ 和 $k - j$ 形成的多边形划分的代价加上这次划分出三角形新增的代价。
- 时间复杂度 $O(n^3)$



凸多边形的划分

- 思考实现细节
- 为什么要选 $j - i$ 这条边，不能是其他边？
- 虽然说选择任意一个三角形划分出的多边形在现多边形上的顶点是连续的，但在原始的多边形是不连续的。选择其它边可能会出现新划分的多边形是 $\{k, k + 1, \dots, j, i\}$ 这样的不连续情况，不方便表示。
- 本题数据范围较大，记得使用高精度





关路灯

- 有一个数轴，分列着 n 盏开着的路灯，第 i 盏开着的路灯每秒消耗 $a_i J$ 的电力。初始你位于第 c 个路灯的位置，每秒你可以向左或向右走一个单位长度，走到某个路灯上可以关掉它。
- 求最终电力消耗的最小值。
- $n \leq 50$



关路灯

- 原题数据范围好小，暴力松松说不一定能过？
- 观察我们关灯的过程，发现我们中间过程一定会关掉包含 c 的一段区间，不可能特意不关某盏灯去关下面一盏。
- 并且当我们刚关好某区间的灯时，一定会站在最左边或最右边的灯上（也就是当前最后关的灯是最左或最右的灯）。
- $f_{i,j,0/1}$ 表示关掉了区间 $i - j$ 的所有灯，并且现在站在最左边/最右边的位置的电功消耗最小值。
- 那么 $f_{i,j,0}$ 的转移即为 $f_{i+1,j,0}$ 或 $f_{i+1,j,1}$ 加上走到指定位置的时间 \times 这段时间的消耗功率； $f_{i,j,1}$ 同理。时间复杂度 $O(n^2)$ 。



区间 dp

- 区间 dp 的问题一般有很明显的区间，操作的影响也是段连续的区间
- 在转移中，一般考虑完成这段区间操作的最后一次操作（或是在操作顺序不重要的情况下调整为方便表达的最后一次操作），分割为子问题进行转移





PART 03

环形 DP

除了破坏成链还有啥说法吗？



环形 DP

有时候遇到题目会有成环的情况，一般的解法是直接暴力破坏成链，将一整段复制到后面然后做 DP 即可。

FOI2022夏





石子合并问题

- 石子合并，但是是环
- 有 n 堆石子，排成一个环，每堆石子 a_i 个，每次可以选择相邻两堆合并（第一堆和最后一堆视为相邻），合并代价为两堆石子数之和，合并后会形成一个新的数量为两堆石子和的石子堆，求将所有石子合并为一整堆的代价和最小值
- $n \leq 100; a_i \leq 100$





石子合并问题

- 将 $1 - n$ 堆的石子复制放到位置 $n + 1 \sim 2n$ 。
- 按照原来的状态设计和转移进行 dp。
- 答案为 $\text{Min}\{f_{i,i+n}\}$ 。
- 为什么可以这样做？
- 在任意 $n - 1$ 次合并操作的方案中，都会存在有某两个相邻位置的石堆不会沿着这里合并；也就是说沿着这两堆石堆切开，把环切成链，该方案依然合法。
- 而 $f_{i,i+n}$ 正好就是第 $i - 1$ 和 i 堆石子切开。





[USACO05JAN] Naptime G

- 贝茜是一只非常缺觉的奶牛. 她的一天被平均分割成 N 段 ($3 \leq N \leq 3830$) , 但是她要用其中的 B 段时间睡觉。每段时间都有一个效用值 $U_i(0 \leq U_i \leq 2 \times 10^5)$ 只有当她睡觉的时候，才会发挥效用。
- 贝茜想使所有睡觉效用的总和最大。不幸的是，每一段睡眠的第一个时间阶段都是“入睡”阶段，而且不记入效用值。
- 时间阶段是不断循环的圆（一天一天是循环的嘛），假如贝茜在时间 N 和时间 1 睡觉，那么她将得到时间 1 的效用值。



[USACO05JAN] Naptime G

- 按照 dp 套路，也是考虑完前 $i - 1$ 个时段，考虑第 i 个时段选择与否。
- 注意到第 i 个时段的贡献还跟上一个时间是否选择有关，因此我们多开一维记录上一维是否选择。
- 记 $f_{i,j,0/1}$ 表示考虑完前 i 个时段选择情况，选了 j 个时段，其中第 i 个选择情况为不选/选了。
- 那么转移为： $f_{i,j,0} = \text{Max}\{f_{i-1,j,0}, f_{i-1,j,1}\}$
- $f_{i,j,1} = \text{Max}\{f_{i-1,j-1,0}, f_{i-1,j-1,1} + U_i\}$
- 对于环形的情况，如果第 n 个时段和第 1 个时段都被选择，那么要重新加上第一个时段的贡献。多开一维记录第一个时段是否选择或者直接 dp 两次分别强制选择和强制不选第一个时段即可。

PART 04

树形 DP

应用广泛的 dp 降下祂的恩惠



没有上司的舞会

- 给 n 个点的树，每个点有点权，求树上权值和最大独立集。
- $n \leq 6 * 10^3$

FOI2022夏令营day7作业





没有上司的舞会

- $f_{i,0/1}$ 表示选择完以 i 为根的子树，其中结点 i 不选/选择。
- 那么显然 $f_{i,0} = \sum_{j \in son_i} \text{Max}\{f_{j,0}, f_{j,1}\}$
- $f_{i,1} = val_i + \sum_{j \in son_i} f_{j,0}$
- 时间复杂度 $O(n)$



没有上司的舞会加强版

- 给 n 个点的树，每个点有点权，求树上权值和最大独立集，且集合大小不超过 m
- $n, m \leq 1 * 10^3$



没有上司的舞会加强版

- 常规地，我们考虑多开一维记录已选集合大小。
- $f_{i,j,0/1}$ 表示结点 i 的子树中，选了 j 个，其中结点 i 不选/选。
- $f_{i,j,0} = \text{Max} \left\{ \sum_{x \in son_i, \sum a_x=j} \text{Max}\{f_{x,a_x,0}, f_{x,a_x,1}\} \right\}$
- $f_{i,j,1} = \text{Max} \left\{ \sum_{x \in son_i, \sum a_x=j} f_{x,a_x,0} \right\} + val_i$
- 在具体实现中，我们通常会一一枚举 i 的所有儿子结点，并把他们一一计算贡献。
- 但这样下来，状态设计已经是 $O(n^2)$ 了，再加上枚举子树选结点个数 $O(n)$ 复杂度似乎变成了 $O(n^3)$ ？



没有上司的舞会加强版

- 我们记 i 的儿子节点依次为 $\{s_1, s_2, s_3, \dots, s_k\}$, 以 x 为根的子树大小为 siz_x 。
- 当我们计算 s_j 的贡献时, 我们会枚举到上界到 siz_{s_j} 一维表示在子树 s_j 中选择的节点数, 并且枚举 $\sum_{1 \leq x < j} siz_x$ 的大小从前面已统计的 $j - 1$ 棵子树转移。
- 这样算的当次的复杂度为 $siz_{s_j} \times \sum_{1 \leq x < j} siz_x$, 他的大小等于在子树 j 与前 $j - 1$ 棵子树中各选一个点的方案数。
- 对树上的任意两点, 都会在他们的 lca 处被计算一次, 因此总时间复杂度为树中任意选两点的方案数, 为 $O(n^2)$ 。
- 转移更新的时候尽量倒序更新, 否则会提前更新导致出错。



二叉苹果树

- 有一棵二叉树，边有边权，如果有分叉，一定是分两叉，即没有只有一个儿子的节点。这棵树共 n 个节点，标号 1 至 n ，树根编号一定为 1。
- 要剪去若干条边，使得原图变成包含结点 1 的只有 q 条边的一棵树，求最终边权和最大值
- $n \leq 100, v_i \leq 30000$





二叉苹果树

- 保留 q 条边相当于保留 $q + 1$ 个点，从儿子节点 x 的转移相当于以 x 为根的子树中保留 j 个点的子问题
- $f_{i,j}$ 表示子树 i 中保留 j 个点的以 i 为根的树的最大边权和
- 若当前位于点 x ，要计算 y 来的转移，那么当前的转移为 $f_{x,i} = \text{Max}\{g_{x,i}, g_{x,j} + f_{y,i-j} + v_{i,j}\}$ ($g_{x,i}$ 表示本次更新前的 $f_{x,i}$)
- 根据与上一道题类似的复杂度分析，时间复杂度为 $O(n^2)$



[HAOI2015] 树上染色

- n 个点的树，边有边权，要将其中 k 个点染成黑色，其余 $n - k$ 个点染成白色，一种染色方案的价值为所有同色点两两距离和之和，最大化这个价值。
- 树上两点的距离为他们之间简单路径的长度。
- $n \leq 2000$

FOI2022直击现场





[HAOI2015] 树上染色

- 我们将最终贡献拆成每一条边分开考虑，也就是对于 $x \rightarrow y$ 这条边，会有多少个黑点对和白点对之间的简单路径会经过这条边。此时 f_x 也只表示子树 x 中的所有边会产生价值。
- 我们发现这个数字会是 $\text{num} = j \times (k - j) + (siz_y - j) \times (n - k - siz_y - j)$ ；即在子树 y 中同色点和不在子树 y 中的同色点之间的路径才会经过这条边。
- 因此枚举当前已计算的前若干个子树中有 i 个黑点，再枚举子树 y 中有 j 个黑点， $f_{x,i+j} = \max\{g_{x,i+j}, f_{y,j} + g_{x,i} + \text{num} \times val_{x \rightarrow y}\}$ 。



[HAOI2015] 树上染色

- 套路地，我们又设 $f_{i,j}$ 表示子树 i 中选择了 j 个点染成黑色的
价值最大化。
- 对于转移，当我们在考虑 $x \rightarrow y$ 这条边的转移时，倘若子树 y
中选取了 j 个点作为黑点，那么一共有 $k - j$ 个黑点不在子树
 y 内；有 $siz_y - j$ 个白点，共有 $n - k - siz_y - j$ 个白点不在子
树 y 内。

PART 05

数位 DP

有趣但延展性不高的 dp



数位 dp

数位 dp 一般是将数字问题的每一位数位看作单独一维从高到低 dp 计算，dp 过程需要注意前导 0 和前面每一位都恰好卡到最大的情况。

卡到上界的情况一般多开一维 0/1 统计或者分开计算。

有时需要将低位数补足前导 0 至所有数位数一样方便处理，有时需对前导 0 特殊处理，依题目而定。





数字游戏

- 某人命名了一种不降数，这种数字必须满足从左到右各位数字成小于等于的关系，如 123, 456。现在大家决定玩一个游戏，指定一个整数闭区间 $[a, b]$ ，问这个区间内有多少个不降数。
- $1 \leq a \leq b \leq 2^{31} - 1$



数字游戏

- 区间 $[a, b]$ 的个数即为区间 $[1, b]$ 的个数减去区间 $[1, a - 1]$ 的个数，考虑区间 $[1, x]$ 的个数怎么算。
- $f_{i,j}$ 表示考虑完第 $n \sim i$ 位的数字情况，其中第 i 位填数字 j 的合法方案数。
- 当第 i 位填 j 时，第 $i + 1$ 位可以填 $0 - j$ ，因此显然 $f_{i,j} = \sum_{0 \leq k \leq j} f_{i+1,k}$ 。



数字游戏

- 还要再计算一下第 $n \sim i + 1$ 位正好填入了 x 的第 $n \sim i + 1$ 位上的数字，第 i 位填的数字小于 x_i ，且都是单调不递减的情况。
- 也就是 $f_{i,j}++$ If $\forall_{i \leq k < n} x_k \leq x_{k+1} \& \& x_{i+1} \leq j < x_i$ 。
- 注意最终全是 0 的情况。



数字游戏

- 为什么要这么做？为什么要从高到低考虑？
- 当从高到低考虑时，当某位填入了比限制数字同一位上更小的数字后，接下来的数位随意填 $0 \sim 9$ 都会比限制数字小，比较好处理。
- 如果一直填和限制数字相同的数字时，会一直有不能超过 x_i 的限制，因此特殊化处理。



Amount of Degrees

- 求给定区间 $[X, Y]$ 中满足下列条件的整数个数：这个数恰好等于 K 个互不相等的 B 的整数次幂之和。例如，设 $X = 15, Y = 20, K = 2, B = 2$ ，则有且仅有下列三个数满足题意：

$$17 = 2^4 + 2^0$$

$$18 = 2^4 + 2^1$$

$$20 = 2^4 + 2^2$$

- $1 \leq X \leq Y \leq 2^{31} - 1, 1 \leq K \leq 20, 2 \leq B \leq 10$



Amount of Degrees

- 观察题目的条件，发现是在 B 进制下共有 K 个 1，其余都是 0 的数满足要求。
- 那么直接在 B 进制下进行数位 dp，同样考虑前缀相减的方法。对于某个数 x ，设 $f_{i,j}$ 表示从高到低考虑了第 $n \sim i$ 位，其中共有 j 个位置填 1，其余填 0 的合法方案数。
- 显然 $f_{i,j} = f_{i+1,j} + f_{i+1,j-1}$ 也就是这一位填 0/1



Amount of Degrees

- 对于恰好卡到 x 上界的情况，再分情况讨论：
 - 如果 x_{n-i+1} 出现了大于 1 的数，不进行任何操作（因为已经填了不合法的数）
 - 如果 x_{n-i+1} 的所有数字都为 0/1，设 1 的个数为 c
 - 如果 $x_i > 0$ ，则 $f_{i,c}$ 加1（填 0）
 - 如果 $x_i > 1$ ，则 $f_{i,c+1}$ 加1（填 1）
- x 本身的合法性再单独处理



数位 dp

- 求 $1 \sim n$ 中满足含有子串 “13” 且能被 13 整除的数的个数。
- $n \leq 10^{100}$

F012022夏
day7





数位 dp

- $f_{i,j,0/1/2}$ 表示从高到低考虑到第 i 位，其中前面的位模 13 余 j ，并且第 \diamond 位不为 1 / 第 \diamond 位为 1 / $\geq \diamond$ 的位存在子串 “13”，且不卡到上界的方案数
- 我们不妨设 $f_{i+1,j,l}$ 会转移到 $f_{i,j',l'}$ ，且第 i 位填入了数字 d ，则：
 - 若 $l' = 2$ 则 $l = 2$ 或 $l = 1 \& d = 2$
 - 若 $l' = 1$ 则 $l \neq 2 \& d = 1$
 - 若 $l' = 0$ 则 $l \neq 2 \& d \neq 1$
- $j' = (j \times 10 + d) \bmod 13$



数位 dp

- 对于前面都卡到上界且本次卡不到上界的情况，转移与上面的类似，需要多考虑一个 d 必须小于 x_i
- 时间复杂度 $O(\log_{10} n \times 13 \times 3 \times 2 \times 10)$ 。

PART 06

状压 DP

很好想但是实现复杂的 dp



旅行商问题

- 给定一张带权无向完全图，求路径长度最小的哈密顿回路。
- 哈密顿回路为经过图中所有点恰好一次的一条回路（起点即为终点）。
- $n \leq 15$



旅行商问题

- 任意选一个点作为哈密顿回路起点，考虑依次进行哈密顿回路的过程。令 $f_{S,i}$ 表示已经考虑好了 S 集合中的点的哈密顿路径（即集合 S 中的点已经被走过了），点 i 是该哈密顿路径中最后一个点，该哈密顿路径长度最小为多少。
- 一般使用二进制来表示集合 S 。即若点 $i \in S$ ，二进制第 i 位为 1，否则为 0。将二进制转成十进制后存储在状态中。
- $f_{S,i} = \text{Min}_{j \neq i, j \in S} \{f_{S-2^i, j} + \text{cost}(j, i)\}$ ，最终答案需要加上走回起点的花费。
- 时间复杂度 $O(2^n n^2)$ 。



状压 dp

给定一张无向图，将每个点染上一种颜色使得相邻点颜色不相同，求最少需要几种颜色

F012022夏day7作业





状压 dp

- 令 f_S 表示对 \diamond 集合内的点染色最小需要的颜色数。
预处理出 g_S 表示 \diamond 集合是否是一个独立集。
- $$f_S = \min_{S' \subseteq S, g_{S'} = 1} \{f_{S-S'} + 1\}$$
- 时间复杂度为枚举子集复杂度 $O(3^n)$
- 可以使用子集卷积 FST 优化至 $O(n^2 \times 2^n)$

谢谢大家

F012022夏令营day7课堂