基础动态规划——常见模型与技巧

4182_543_731

2024/07

Table of Contents

局部限制与状态

一些众所周知的 DP 问题：

每个物品有重量 w_i，最多选一次，求总和不超过 C 的方案数。

局部限制与状态

一些众所周知的 DP 问题：

每个物品有重量 w_i，最多选一次，求总和不超过 C 的方案数。
记 $dp_{i,j}$ 表示前 i 个物品选了重量和为 j 的方案数，则

$$dp_{i,j} = dp_{i-1,j} + dp_{i-1,j-w_i}$$

n 个元素排成一列，不能同时选相邻的两个元素，求最大总和

局部限制与状态

一些众所周知的 DP 问题：

每个物品有重量 w_i，最多选一次，求总和不超过 C 的方案数。
记 $dp_{i,j}$ 表示前 i 个物品选了重量和为 j 的方案数，则

$$dp_{i,j} = dp_{i-1,j} + dp_{i-1,j-w_i}$$

n 个元素排成一列，不能同时选相邻的两个元素，求最大总和
记 $dp_{i,0/1}$ 表示前 i 个物品，第 i 个是否可以选的最大总和，则

$$\begin{cases} dp_{i,1} = \max(dp_{i-1,1}, dp_{i-1,0} + v_i) \\ dp_{i,0} = dp_{i-1,1} \end{cases}$$

局部限制与状态

什么是 DP？这极其难以回答，但我们可以尝试总结一些共性。

- 我们可以在问题中用简单的信息描述一个局部的子问题（**状态**）。例如，在背包问题中，由于问题只考虑总和，在处理了前若干个物品后，我们只需要关心总重量，而不需要考虑具体选了哪些物品。
- 问题的限制具有一定的局部性，使得我们可以用简单的**转移**将子问题连接起来，进而得到原问题的答案。例如，在链上独立集问题中，限制只和相邻两个元素有关，因此转移时只需要再考虑下一个元素是否被选。

很多 DP 问题的关键便在于，通过分析原问题，找到合适的**状态**，然后通过合适的**转移**得到所有子问题的结果。

局部限制与状态

状态设计需要注意的点：在子问题中，当前记录的状态是不是足够处理问题的限制?

转移设计需要注意的点：这样的转移是否不重不漏?（计数）/是否找到了最优解?（最优化）

我们将在接下来的若干模型和例子中体会这一点。

最最常见的 DP 模型。

最常见的状态设计是前缀：记 $dp_{i,\dots}$ 表示考虑了序列前 i 个元素，然后 ...序列上的很多限制是相邻的，因此我们通常可以只记录很少的状态以解决问题。

Tree Planting(Easy)

有 n 个位置，每个位置有 a_i 种方式选 0，b_i 种方式选 1。要求：

- 相邻两个位置不能同时选 1。
- 距离为 k 的两个位置不能同时选 1。

求合法方案数。$n \le 300, k \le 16$

序列 DP

最最常见的 DP 模型。

最常见的状态设计是前缀：记 $dp_{i,\dots}$ 表示考虑了序列前 i 个元素，然后 ...序列上的很多限制是相邻的，因此我们通常可以只记录很少的状态以解决问题。

另一种常见的可能性是区间：记 $dp_{l,r}$ 表示考虑区间 $[l, r]$ 内的状态，...有些时候，我们不能直接按照序列顺序考虑问题。但序列进行分裂后还是区间，因此按照其它角度通常会得到区间 DP。

相信大家都会，这里就不赘述了。我们将在综合练习中进一步考虑。

Table of Contents

状态压缩

在一些情况下，DP 的状态会比较复杂，但我们实现程序的时候一般只能用数来描述状态。这种时候，我们通常可以通过一些方式把状态映射到一个数。

这更多的是一种思想：我们可以把很复杂的东西作为一个状态，而不一定总是 $dp_{i,j,k}$。

最经典的，如果状态是 $\{1, 2, \cdots, n\}$ 的一个子集，它可以被描述为一个 n 位 01 串，其中第 i 位表示 i 是否在集合中。这又可以被看成一个二进制数。通过简单的位运算，我们可以快速支持一些简单操作：求集合的交，判掉一个元素是否在集合内，求集合大小（预处理）...

在 n 不大的情况下，我们完全可以把集合作为一个状态。

子集 DP 的第一种常见转移：加入下一个元素。

[CCO2015] Artskjid

给一张有向图，求 1 到 n 的最长简单路。$n \leq 18$

简单路径要求不经过重复点，因此我们需要记录之前走过的点。记 $dp_{i,S}$ 表示当前路径走到 i，前面经过的点集合为 S 时的最长路径。转移时枚举下一个点 j，判断 $j \notin S$ 以及存在边，然后转移过去。复杂度 $O(n^2 2^n)$。

子集 DP

子集 DP 的第二种常见转移：加入下一个子集。

经典例题 2

给定 $\{1, \cdots, n\}$ 中的一些子集是好的，求有多少种将 $\{1, 2, \cdots, n\}$ 划分为若干个好的子集的方案。$n \le 15$

~~熟练的选手应该知道这东西可以~~ $\cancel{O(2^n * poly(n))}$~~，但我找不出简单的好例子了~~

考虑将子集一个一个加进去。记 dp_S 表示当前加进去的部分为 S 的方案数，枚举下一个好的子集 T 转移。但这样显然会算重，类似的事情在子集 DP 中经常出现。

一种常见的去重方式是，每次考虑不在 S 中的最小元素 x，找到包含 x 的子集 T。这样就不重复了。类似的技巧（包含最小元素）也在各种子集 DP 中经常出现。

子集 DP 的第二种常见转移：加入下一个子集。

经典例题 2

给定 $\{1, \cdots, n\}$ 中的一些子集是好的，求有多少种将 $\{1, 2, \cdots, n\}$ 划分为若干个好的子集的方案。$n \le 15$

这样的复杂度是啥？我们需要枚举 S, T，但显然正确的转移中 $S \cap T = \emptyset$。熟知这只有 3^n 种可能，但直接枚举 S, T 是 4^n。

我们需要一种方式，快速枚举 $\bar{S}$ 的子集。使用位运算技巧：

```
for(int i=s;;i=(i-1)&s)
```

当然，对于一个一般的 01 串，我们也可以用相同的位运算技巧来处理：

Tree Planting(Easy)

有 n 个位置，每个位置有 a_i 种方式选 0，b_i 种方式选 1。要求：

- 相邻两个位置不能同时选 1。
- 距离为 k 的两个位置不能同时选 1。

求合法方案数。$n \leq 300, k \leq 16$

顺序 DP，直接记前 k 个位置的情况，复杂度 $O(n2^k)$。

一个常见的情况是二维问题：有一个宽度较小的矩形，在限制只和矩形相邻位置有关的情况下，我们可以一行一行考虑。

[USACO06NOV] Corn Fields

有一个 $n \times m$ 的网格，有些格子上面可以放棋子。你不能在四相邻的格子中同时放格子，求合法方案数。$n, m \leq 12$

显然是局部限制。记 $dp_{i,S}$ 表示有多少种填前 i 行的方式使得第 i 行状态为 S。位运算容易处理：判断是否放在能放的格子上，是否行内 S 存在相邻不合法情况，相邻两行 S, T 是否存在不合法。复杂度 $O(n4^m)$（枚举下一行转移）。

改成八相邻：

状压 DP

[NOI2001] 炮兵阵地

有一个 $n \times m$ 的网格，有些格子上面可以放棋子。两个放的棋子不能距离不超过 2 且同行或同列。求最多能放多少棋子。$n \le 100, m \le 10$

这里限制会与相邻三行相关，因此考虑记录上两行的状态，然后枚举下一行的状态，大量位运算判定合法性。复杂度 $O(n8^m)$...

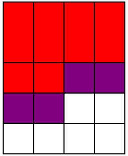
但每一行相邻两个棋子距离至少需要是 2。可以发现 $m = 10$ 时行方案数不超过 60。代数一下复杂度约为 $O(n3.148^m)$

[USACO06NOV] Corn Fields 加强版

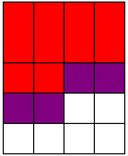
有一个 $n \times m$ 的网格，有些格子上面可以放棋子。你不能在四相邻的格子中同时放格子，求合法方案数。$n, m \leq 18$

上面的做法并不是最好的：我们每次直接枚举了一行，但这里的限制在行上也是局部的！如果我们在一行内也依次枚举，我们可能可以得到更好的做法。

考虑依次填每一行，每一行内从前往后。如果我们填到某个位置，此时需要记录哪些状态？



插头 DP / 轮廓线 DP



设 $dp_{i,j,S}$ 表示填到了第 i 行第 j 列，当前轮廓线（紫色部分）状态为 S 时前面最多放多少个棋子。转移时枚举下一个位置填啥，判定轮廓线上的限制并更新轮廓线。复杂度 $O(nm2^m)$。

位运算技巧的进阶考验。

当然如果注意到一行相邻两个不能都是 1，直接枚举行是 $O(n1.618^{2m})$ 的。但轮廓线也可以这样搞变成 $O(nm1.618^m)$。

Rikka with K-Match(Easy)

有一个 $n \times m$ 的网格图，边有边权，求最大权匹配的权值。

$n \le 5 \times 10^4, m \le 6$

如果枚举行，还需要枚举两行间的匹配情况，这太复杂了。

还是可以轮廓线：记录轮廓线上的位置有没有匹配。加入下一个位置时只需要枚举它是否和左侧或者上面的匹配。复杂度 $O(nm2^m)$。

如果想知道原版是啥，请参考 Div1 课件。

Tree Planting(Medium)

有 n 个位置，每个位置有 a_i 种方式选 0，b_i 种方式选 1。要求：

- 相邻两个位置不能同时选 1。
- 距离为 k 的两个位置不能同时选 1。

求合法方案数。$n \le 150$

k 小的时候大家都会了。

Tree Planting(Medium)

有 n 个位置，每个位置有 a_i 种方式选 0，b_i 种方式选 1。要求：

- 相邻两个位置不能同时选 1。
- 距离为 k 的两个位置不能同时选 1。

求合法方案数。$n \le 150$

k 小的时候大家都会了。

1	2	$\cdots$	k
$k+1$	$k+2$	$\cdots$	$2k$
$2k+1$	$2k+2$	$\cdots$	$\cdots$

插头 DP / 轮廓线 DP

1	2	$\cdots$	k
$k+1$	$k+2$	$\cdots$	$2k$
$2k+1$	$2k+2$	$\cdots$	$\cdots$

这样几乎就是二维上相邻的限制，唯一区别是第一列和最后一列之间有关，这是类似环形 DP 的形式。那么考虑破开环，枚举第一列的情况，再做轮廓线 DP。复杂度 $O(n4^{n/k})$。

另一侧复杂度 $O(n2^k)$，平衡可得 $O(n2^{\sqrt{2n}})$。

1	2	$\cdots$	k
$k+1$	$k+2$	$\cdots$	$2k$
$2k+1$	$2k+2$	$\cdots$	$\cdots$

这样几乎就是二维上相邻的限制，唯一区别是第一列和最后一列之间有关，这是类似环形 DP 的形式。那么考虑破开环，枚举第一列的情况，再做轮廓线 DP。复杂度 $O(n4^{n/k})$。

另一侧复杂度 $O(n2^k)$，平衡可得 $O(n2^{\sqrt{2n}})$。

[HDU 多校] Tree Planting

$n \le 300$

注意到还是相邻两个位置不能同时选。可以把 2 变成 $\frac{1}{2}(1+\sqrt{5}) \approx 1.618$。

状压 DP

对于更一般的复杂状态，我们也可以把它强行表示下来。这时通常不能表示得很简单，但多数时候这里不会卡时间，可以手动维护映射/强行 map。

[NOI2007] 生成树计数

n 个点，两个点 i,j 间有边当且仅当 $|i-j| \le k$，求生成树数量。$n \le 10^{15}, k \le 5$。

状压 DP

对于更一般的复杂状态，我们也可以把它强行表示下来。这时通常不能表示得很简单，但多数时候这里不会卡时间，可以手动维护映射/强行 map。

[NOI2007] 生成树计数

n 个点，两个点 i, j 间有边当且仅当 $|i-j| \le k$，求生成树数量。$n \le 10^{15}, k \le 5$。

顺序考虑所有点，往生成树里面加边，因为连边的局部性，在只考虑 $[1, i]$ 之间边的选择情况下，合法方案也必然满足所有点和 $[i-k+1, i]$ 中的点连通，因为只有这些点和后面的点有边。

记 $dp_{i,S}$ 表示考虑了前 i 个点之间的边，当前所有点和最后 k 个点连通且构成森林，且最后 k 个点之间连通性为 S 的方案数。转移枚举 $i+1$ 和前面点的连边情况。

但现在 $n = 10^{15}$。注意到 dp_i 到 dp_{i+1} 是一个线性方程，可以矩阵快速幂。复杂度 $O(Bell(k)^3 \log n)$。

[HDU 多校] Yinyang

给一个 $n \times m$ 的网格，把每个格子染成黑白之一，满足如下条件：

- 黑色格子四连通
- 白色格子四连通
- 任意一个 2×2 的连续子矩阵不同色

现在给定部分颜色，求合法地染剩下部分的方案数。$nm \leq 100$

状压 DP

[HDU 多校] Yinyang

给一个 $n \times m$ 的网格，把每个格子染成黑白之一，满足如下条件：

- 黑色格子四连通
- 白色格子四连通
- 任意一个 2×2 的连续子矩阵不同色

现在给定部分颜色，求合法地染剩下部分的方案数。$nm \leq 100$

类似之前某个题，选择 n, m 较小的一侧做轮廓线 DP。因为限制 3，我们需要额外多维护一位的值。因为限制 $1, 2$，我们需要再维护边界上每一段之间的连通性。然后硬写。

限制 3 使得不会有一个连通块在轮廓线 DP 到一半的时候就消失。常数卡得好就能过。

可以再观察一些性质，比如如果轮廓线上依次有黑白黑白四个连通块，那就不可能合法，然后多剪枝。

Table of Contents

$$\begin{array}{r} 1\quad 0\quad 1 \\ +\quad {}_1\,1_1\;\; 1\quad 0_1\; 1 \\ \hline 1\quad 0\quad 0\quad 1\quad 0 \end{array}$$

如图所示，有很多操作都可以是按位做的：加减法，比较，位运算，...~~当然还有问题要求你考虑数位~~

这些操作在按位时都有着很好的局部性：加减法只需要从后往前，记录后面进位了多少；比较最常见的方式是从大到小一位一位比；位运算自然只考虑这一位。

在这种时候，我们可以一位一位地填数，然后记录考虑了这些位后上面这些问题需要的状态。常见的状态设计即为：填了高/低 i 位，当前进位多少/比较关系如何/...

数位 DP

从高到低还是从低到高？根据问题考虑。一般的关键点在于加减法和比较在两种顺序下的表现。对于比较：

- 在从低到高的情况下，比较需要记录两个后缀的大小关系。如果限制是 $a \le b$，那么只需要记录 a 的后缀是否 $\le b$ 的后缀。
- 在从高到低的情况下，前缀比出了就可以直接确定关系。假设限制是 $a \le b$，则如果前缀比出了 $a > b$ 就可以直接跳过，只需要记录前缀是已经比出来 $a < b$，还是不确定。

看似效率没有差别，但从高到低有一个好处：如果已经比出来了，就不会变回去，因此下面的转移只有三种，而上面的有四种。在有多个数的情况下，结合状态压缩技巧可能可以做到类似 3^m 和 4^m 的区别。通常我们会从高往低做。

数位 DP

[SCOI2009] windy 数

求 $[l,r]$ 中有多少整数 x 满足：x 的十进制表示下相邻两数位差不小于 2。

$r \le 2\times 10^9$

为了简便，对于 $[l,r]$ 的限制（在时间限制问题不严重的情况下）我们通常容斥为 $\le r$ 的减去 $\le l-1$ 的。考虑从高往低填。因为额外限制是相邻数位差，因此填了高位后只需要再记录填的最后一个数位（还需要记录比较需要的前缀大小关系）

因此记 $dp_{i,v,0/1}$ 表示填了高位到第 i 位，第 i 位填的是 v，且已经填的前缀和上界 r 是相等还是小于。然后对 $l-1$ 再做一遍。

数位 DP

「PA 2020」Liczba Potyczkowa

求 $[l, r]$ 中有多少整数 x 满足其十进制表示中不存在 0，且 x 被其十进制表示中每种出现过的数整除。

$l, r \leq 10^{18}$

数位 DP

「PA 2020」Liczba Potyczkowa

求 $[l, r]$ 中有多少整数 x 满足其十进制表示中不存在 0，且 x 被其十进制表示中每种出现过的数整除。

$l, r \leq 10^{18}$

从高到低填，状压记录十进制表示中每种数位是否出现（2^9）。注意到 $[1, 9]$ 的最小公倍数是 2520，因此我们还需要记录前缀模 2520 的结果，这好像可以通过。

还可以注意到可能的最小公倍数只有 48 种，以及最后一位之前加的都是 10 的倍数等等。

数位 DP

从高到低还是从低到高？根据问题考虑。一般的关键点在于加减法和比较在两种顺序下的表现。对于加法：

- 在从低到高的情况下，加减法是非常简单的：状态只需要记录向前进位了多少，转移也很直接。
- 从高到低的加减法则稍微复杂一点：状态可以记作“如果后面进位了 k，则前面这样的方案数”。转移时枚举下一位情况，算出下一位又应该进位多少。

效率上其实不存在差别，前者稍微直观。

[CF 1710C] XOR Triangle

求有多少组非负整数 a, b, c 满足：

- $a, b, c \le n$
- 记 $x = a \oplus b, y = b \oplus c, z = c \oplus a$，则 $x + y > z, y + z > x, z + x > y$

$n \le 2^{2\times 10^5}$，输入为二进制

[CF 1710C] XOR Triangle

求有多少组非负整数 a, b, c 满足：

- $a, b, c \leq n$
- 记 $x = a \oplus b, y = b \oplus c, z = c \oplus a$，则 $x + y > z, y + z > x, z + x > y$

$n \leq 2^{2\times10^5}$，输入为二进制

进行一个直接的数位 DP：这里又有进位又有比较。考虑从小到大填，记录当前后缀 a, b, c 和 n 的大小关系，$x + y, z$ 的关系及 $x + y$ 的进位，以此类推。复杂度线性，常数 2^{12}。

[CF 1710C] XOR Triangle

求有多少组非负整数 a, b, c 满足：

- $a, b, c \le n$
- 记 $x = a \oplus b, y = b \oplus c, z = c \oplus a$，则 $x + y > z, y + z > x, z + x > y$

$n \le 2^{2\times 10^5}$，输入为二进制

进行一个直接的数位 DP：这里又有进位又有比较。考虑从小到大填，记录当前后缀 a, b, c 和 n 的大小关系，$x + y, z$ 的关系及 $x + y$ 的进位，以此类推。复杂度线性，常数 2^{12}。

再观察一下。$x \oplus y \oplus z = 0, z = x \oplus y$。那么不合法情况一定是 $x \wedge y = 0$ 或者另外两种。状态数变为 2^6，常数变为 2^9。可以通过。

在直接开始数位 DP 前，还是应该再观察一下性质。

Lotus(Easy)

求有多少组非负整数 a_i 满足 $\sum a_i 2^i \le 2^n$。

$n \le 500$

数位 DP

Lotus(Easy)

求有多少组非负整数 a_i 满足 $\sum a_i 2^i \le 2^n$。

$n \le 500$

乘法不直接满足局部性质。但这里乘的是 2^i，所以二进制下可以直接看成加法。相当于最低位有一个数，第二位有两个数，以此类推。

直接从低往高，记 $f_{i,j}$ 表示填了后 i 位，当前加起来向上一位一共进位了 j。枚举这一位填了几个 1：

$$f_{i,j}\binom{i+1}{k} \to f_{i+1,\lceil\frac{j+k}{2}\rceil}$$

复杂度 $O(n^3)$

练习题：[NOIP2021] 数列

「THUPC 2021」游戏

考虑到您可能不会博弈，问题相当于：求有多少组非负整数 $a_1, \cdots, a_m$ 满足

- $\sum a_i = n$
- $\bigoplus a_i \neq 0$
- $a_i \leq l_i$

$m \leq 10, n \leq 10^{18}$

「THUPC 2021」游戏

考虑到您可能不会博弈，问题相当于：求有多少组非负整数 $a_1, \cdots, a_m$ 满足

- $\sum a_i = n$
- $\oplus a_i \neq 0$
- $a_i \leq l_i$

$m \leq 10, n \leq 10^{18}$

先考虑从低到高，记录进位多少，每个数的大小关系。转移枚举这一位怎么填。复杂度 $O(4^m m \log n)$。这显然不能通过。

然后有多种方式可以通过。

考虑从高往低做，每个数的前缀可能还和上限相同，也可能已经脱离限制。但后一种情况下就可以任意填了！这里只需要求和，因此脱离限制的位置只需要枚举总共填了多少个 1，然后枚举子集。复杂度 $O(3^m m^2 \log n)$。这里假设我们可以通过位运算 $O(1)$ 更新状态。

然后有多种方式可以通过。

考虑从高往低做，每个数的前缀可能还和上限相同，也可能已经脱离限制。但后一种情况下就可以任意填了！这里只需要求和，因此脱离限制的位置只需要枚举总共填了多少个 1，然后枚举子集。复杂度 $O(3^m m^2 \log n)$。这里假设我们可以通过位运算 $O(1)$ 更新状态。

注意到按位填 n 个数很像一个二维的东西。因为只需要记录和，我们还可以直接轮廓线处理“枚举这一位怎么填”的部分！这样直接把从低到高优化到了 $O(2^m m^2 \log n)$。

~~实测记忆化搜索在这种情况下薄纱填表 DP~~

数位 DP

当然更一般的情况下，从高到低确有其优势：

分形图

求有多少组非负整数 $a_1, \cdots, a_m$ 满足

- $a_i \in [l_i, r_i]$
- 对于每一位，这 m 个数在这一位上的取值必须取某些组合（组合数量不超过 2^m）

$m \leq 11, V \leq 2^{60}$

这下就没法轮廓线了，因为每一位上不能再局部分开。直接容斥 $[l, r]$ 加上从低到高的复杂度高达 $O(8^m \log V)$。

考虑从高到低做，同时把两个限制放到一起。每一个数可能贴着下界限制，可能贴着上界限制，可能没有限制。枚举前两种的取值，第三种任意。相当于问一些位任意的情况下有多少方案，这可以 $O(4^m)$ 预处理搜出来。复杂度 $O(5^m \log V)$。

Table of Contents

树形 DP

树也是一个极其常见的 DP 模型。它有着最自然的子问题形式：以一点 u 为根的子树。以 u 为根的子树与剩余部分相邻的点只有根 u，因此我们通常只需要记录一些与子树根 u 有关的状态，就可以维护足够的信息。

没有上司的舞会

树上每个点有点权，求最大权独立集。$n \le 10^6$

考虑填了一个子树后我们需要记住什么状态。因为子树内只有根 u 和外面相连，因此只需要记录 u 的状态。

记 $dp_{u,0/1}$ 表示填了 u 的子树，u 是否被选择时的最优答案，则显然

$$\begin{cases} dp_{u,1} = w_u + \sum_{v \in son_u} dp_{v,0} \\ dp_{u,0} = \sum_{v \in son_u} \max(dp_{v,0}, dp_{v,1}) \end{cases}$$

树形 DP

树形 DP 的状态设计要点：考虑了整个子树之后，在上面的问题中要记录多少信息?
树形 DP 的转移：先合并所有子树的信息，然后考虑根向上的边。

树形 DP

树形 DP 的状态设计要点：考虑了整个子树之后，在上面的问题中要记录多少信息？
树形 DP 的转移：先合并所有子树的信息，然后考虑根向上的边。

小练习

给一棵 n 个点的树，删掉一些边，然后在剩余每个连通块内选一个点，求方案数。
$n \leq 10^6$

树形 DP

树形 DP 的状态设计要点：考虑了整个子树之后，在上面的问题中要记录多少信息?

树形 DP 的转移：先合并所有子树的信息，然后考虑根向上的边。

小练习

给一棵 n 个点的树，删掉一些边，然后在剩余每个连通块内选一个点，求方案数。

$n \leq 10^6$

处理了一个子树内的选点/删边后，根所在的连通块之外都完全确定了，因此它们必须合法。

$dp_{u,0/1}$ 表示处理了 u 的子树，u 所在的连通块有没有选点，剩余部分合法的方案数。

树形 DP 的状态设计要点：考虑了整个子树之后，在上面的问题中要记录多少信息？

树形 DP 的转移：先合并所有子树的信息，然后考虑根向上的边。

小练习

给一棵 n 个点的树，删掉一些边，然后在剩余每个连通块内选一个点，求方案数。

$n \leq 10^6$

处理了一个子树内的选点/删边后，根所在的连通块之外都完全确定了，因此它们必须合法。

$dp_{u,0/1}$ 表示处理了 u 的子树，u 所在的连通块有没有选点，剩余部分合法的方案数。

首先 $dp_{u,0} = dp_{u,1} = 1$（是否选 u）。对于每个儿子 v，$dp_{v,1}$ 可以断边，$dp_{v,0}$ 必须保留。

$$\begin{cases} dp_{u,1} \leftarrow dp_{u,1}(dp_{v,0} + dp_{v,1}) + dp_{u,0}dp_{v,1} \\ dp_{u,0} \leftarrow dp_{u,0}(dp_{v,0} + dp_{v,1}) \end{cases}$$

树形背包

[51nod 1353] 树

给一棵 n 个点的树，删掉一些边，使得剩余每个连通块大小不超过 k。求方案数。$n \le 5000$

[51nod 1353] 树

给一棵 n 个点的树，删掉一些边，使得剩余每个连通块大小不超过 k。求方案数。$n \leq 5000$

显然需要记的状态是子树根所在连通块的大小。记 $dp_{u,j}$ 表示处理了 u 的子树，子树内 u 所在连通块大小为 j 的方案数。

那么向上相当于可以把 j 变成 0。合并 u 和一个新的子树直接是合并两个背包。

但这样复杂度是啥？直接数 for 循环数量看起来是 $O(n^3)$ 的。但是

树形背包

[51nod 1353] 树

给一棵 n 个点的树，删掉一些边，使得剩余每个连通块大小不超过 k。求方案数。$n \le 5000$

显然需要记的状态是子树根所在连通块的大小。记 $dp_{u,j}$ 表示处理了 u 的子树，子树内 u 所在连通块大小为 j 的方案数。

那么向上相当于可以把 j 变成 0。合并 u 和一个新的子树直接是合并两个背包。

但这样复杂度是啥？直接数 for 循环数量看起来是 $O(n^3)$ 的。但是

点权为 1 时，树形背包复杂度为 $O(n^2)$

树形背包

点权为 1 时，树形背包复杂度为 $O(n^2)$

简单证明：每次我们合并两个连通块（u 合并了部分子树和下一个 v 的子树)。设合并的两边大小为 a, b，则背包合法复杂度 $O(ab)$。

注意到 $(a+b)^2 = a^2 + b^2 + 2ab$ 即可归纳证明总复杂度为 $O(n^2)$。

直观解释：$O(ab)$ 可以看成两边各选一个点 x, y。而一对 x, y 只会被统计一次：在 x, y 的 LCA 处。

树形背包

树 v2

给一棵 n 个点的树，删掉一些边，使得剩余每个连通块大小不超过 k。求方案数。
$n \leq 10^5, k \leq 300$

背包的复杂度看起来是 $O(nk^2)$。但实际上可以证明是 $O(nk)$。
数学证明：归纳证复杂度不超过 $O(nk - \frac{1}{2}k^2)$。
组合证明：合并代价是 $\min(a, k)\min(b, k)$。分类讨论：

- 对于 $a, b \leq k$ 的合并，它们合出来每块大小 $O(k)$（在被合并掉之前），每一块根据之前结论 $O(k^2)$，总复杂度 $O(nk)$
- 对于一个 $\leq k$，一个 $> k$ 的合并，一个 $\leq k$ 的只会被这样合并一次，因此 $O(nk)$。
- 对于都 $> k$ 的，这样的合并只有 $O(n/k)$ 次，因此还是 $O(nk)$。

换根 DP

Maximum White Subtree(Easy)

给一棵树，点权为 ± 1。求包含根的连通块点权和的最大值。$n \le 10^6$

注意到如果一个连通块包含根，那么任意一个子树内包含连通块的部分是包含子树根的一个连通块，这是一个子问题。

记 dp_u 表示 u 子树内包含 u 的连通块最大权值和。首先需要选 u，对于 u 的每个儿子 v，我们可以直接跳过 v 子树，也可以选一个 v 子树内包含 v 的连通块。

$$dp_u = v_u + \sum_{v \in son_u} \max(dp_v, 0)$$

换根 DP

Maximum White Subtree(Easy)

给一棵树，点权为 ± 1。求包含根的连通块点权和的最大值。$n \le 10^6$

注意到如果一个连通块包含根，那么任意一个子树内包含连通块的部分是包含子树根的一个连通块，这是一个子问题。

记 dp_u 表示 u 子树内包含 u 的连通块最大权值和。首先需要选 u，对于 u 的每个儿子 v，我们可以直接跳过 v 子树，也可以选一个 v 子树内包含 v 的连通块。

$$dp_u = v_u + \sum_{v \in son_u} \max(dp_v, 0)$$

众所周知，常见树形 DP 可以求出每个点子树内问题的答案，但是

换根 DP

[CF 1324F] Maximum White Subtree

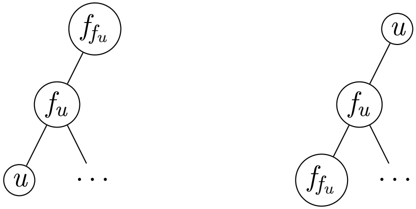
给一棵树，点权为 ± 1。对每个点 u，求包含 u 的连通块点权和最大值。$n \le 10^6$

刚才的 DP 对每个 u 只求了考虑 u 子树内的答案，怎么把外面的部分也考虑进来？

如果以 u 为根，外面的东西也是一个子树。考虑类似地设 f_u 表示以 u 为根，u 原先父亲所在子树的答案。

如何转移？考虑以 u 为根，u 的父亲 fa_u 有哪些子树。一部分是 fa_u 除去 u 外原先的其它子树，另一个是 fa_u 的父亲子树。

[CF 1324F] Maximum White Subtree



那么我们需要合并两部分的信息：fa_u 的其它子树和 fa_u 的父亲。那么

$$f_u = v_{fa_u} + \max(f_{fa_u}, 0) + \sum_{v \in son_{fa_u}, v \neq u} \max(dp_v, 0)$$

换根 DP 的常见实现：先 dfs 一次从下往上求出 dp，再 dfs 一次从上往下求出 f，然后合并得到答案。

[CF 1324F] Maximum White Subtree

$$f_u = v_{fa_u} + \max(f_{fa_u}, 0) + \sum_{v \in son_{fa_u}, v \neq u} \max(dp_v, 0)$$

怎么实现？这里是直接求和，可以考虑先求出 $\sum_{v \in son_{fa_u}} \max(dp_v, 0)$，然后对于每个 u 减去 $\max(dp_u, 0)$。这是最常见的写法。

但有的信息是不可减的！例：$\max$，模合数的乘法。此时常见处理方式如下：将儿子排成一列，那么删掉一个元素相当于剩下的一个前缀和一个后缀，因此预处理前缀和和后缀和然后合并。

复杂度 $O(n)$

树形 DP

不一定所有树形 DP 都是从下往上的。比如上面换根 DP 的例子，比如

某个题的某一步

给一棵有根树，点有点权 v_i，对于每个叶子 u，记 $path_u$ 表示 u 到根的路径，求出 $\prod_{x \notin path_u} v_x$，答案模**合数** 10^9+2022。

不一定所有树形 DP 都是从下往上的。比如上面换根 DP 的例子，比如

某个题的某一步

给一棵有根树，点有点权 v_i，对于每个叶子 u，记 $path_u$ 表示 u 到根的路径，求出 $\prod_{x \notin path_u} v_x$，答案模**合数** 10^9+2022。

考虑从上往下，每次往一个子树走的时候，把其它子树的权值全部乘进去。因此先从下往上求出每个点子树内所有点权之积，然后记 f_u 表示 u 子树外，且不在 u 到根路径上的点权乘积。从 f_{fa_u} 转移到 f_u 就是把其它子树权值全部乘进去。前后缀和的方式。

在更多的时候，状态设计不是直接的：树的性质非常好，因此我们通常可以推出很多性质，然后再根据这些性质来设计状态。推性质可以非常、非常、非常难。

[CF 1032F] Vasya and Maximum Matching

给一棵树，你可以任意删一些边。求有多少种删边方式使得最大匹配唯一。$n \leq 3 \times 10^5$

在更多的时候，状态设计不是直接的：树的性质非常好，因此我们通常可以推出很多性质，然后再根据这些性质来设计状态。推性质可以非常、非常、非常难。

[CF 1032F] Vasya and Maximum Matching

给一棵树，你可以任意删一些边。求有多少种删边方式使得最大匹配唯一。$n \leq 3 \times 10^5$

如果一个点没有被匹配，那它可以去抢旁边匹配点的匹配，这样必然不唯一。那么合法的一个必要条件是，每个 ≥ 2 大小的连通块都存在完美匹配。

树形 DP

在更多的时候，状态设计不是直接的：树的性质非常好，因此我们通常可以推出很多性质，然后再根据这些性质来设计状态。推性质可以非常、非常、非常难。

[CF 1032F] Vasya and Maximum Matching

给一棵树，你可以任意删一些边。求有多少种删边方式使得最大匹配唯一。$n \leq 3 \times 10^5$

如果一个点没有被匹配，那它可以去抢旁边匹配点的匹配，这样必然不唯一。那么合法的一个必要条件是，每个 ≥ 2 大小的连通块都存在完美匹配。

注意到这是一棵树，树的完美匹配显然唯一：每次选一个叶子和上面匹配。那么充分必要条件就是这个。

[CF 1032F] Vasya and Maximum Matching

合法条件是，对于剩下的每个大小 ≥ 2 的连通块，如果我们贪心的选叶子和上面匹配，能得到完美匹配。

考虑子树需要记录什么信息。根所在的连通块有三种状态：

1. 当前连通块大小为 1。
2. 当前连通块大小 ≥ 2，且当前点没有被使用（即必须向上匹配）
3. 当前连通块大小 ≥ 2，且已经被使用。

考虑向上转移。情况 $1,3$ 可以切边，情况 1 如果不切就变成情况 2。

1. 情况 1 要求子树全部切边。
2. 情况 2 要求存在子树不切边，且所有不切的子树都是情况 3。
3. 情况 3 要求存在一个子树不切边切是情况 $1,2$，剩余子树要么切要么是情况 3。

组合意义 (Intro)

一道题

给一棵 n 个点的树，你有 2^{n-1} 种方式删掉一些边，对所有方式求和最后得到的每个连通块大小乘积。

$n \le 10^6$

显然可以直接 DP 记录根所在连通块大小，但显然是 $O(n^2)$。

组合意义 (Intro)

一道题

给一棵 n 个点的树，你有 2^{n-1} 种方式删掉一些边，对所有方式求和最后得到的每个连通块大小乘积。

$n \le 10^6$

显然可以直接 DP 记录根所在连通块大小，但显然是 $O(n^2)$。

给一棵 n 个点的树，删掉一些边，然后在剩余每个连通块内选一个点，求方案数。

$n \le 10^6$

考虑两个问题删边后的方案数，可以发现后者正好表示了连通块大小乘积。

如果您想进一步了解，可以参考 div1 课件。

Table of Contents

综合练习

你已经学会了基础的 DP 模型，让我们来尝试一点基础的应用。

hint 1：在很多时候，我们需要找到正确的角度去分析问题、设计状态。

「PA 2019」Muzyka pop

给一个长度为 n 的序列 v 和 m，你需要找到一个长度为 n 的整数序列 a 使得 $0 \le a_1 \le a_2 \le \cdots \le a_n \le m$，最大化 $\sum_{i=1}^{n} popcount(a_i) * v_i$，权值可能为负数。

$n \le 200, m \le 10^{18}$

按序列顺序做？显然不可行。

从数位的角度，考虑从高往低位填。因为最高位不同可以直接决定比较结果，最高位一定是 $00\cdots011\cdots1$。此时中间位置的限制就断开了，两边的填数变成两个独立的区间。注意到 popcount 可以分到每一位上，这就完全独立了。

那么我们可以在数位 DP 上区间 DP。

按序列顺序做？显然不可行。

从数位的角度，考虑从高往低位填。因为最高位不同可以直接决定比较结果，最高位一定是 $00\cdots011\cdots1$。此时中间位置的限制就断开了，两边的填数变成两个独立的区间。注意到 popcount 可以分到每一位上，这就完全独立了。

那么我们可以在数位 DP 上区间 DP。记 $dp_{k,l,r,0/1}$ 表示考虑区间 $[l,r]$，还需要填后 k 位，当前前缀是否还和上界 m 相同。不考虑上界的话，转移形如：

$$dp_{k,l,r} = \max_{t\in[l-1,r]} dp_{k-1,l,t} + dp_{k-1,t+1,r} + \sum_{i=t+1}^{r} v_i$$

上界容易处理，也可以加一个元素 $a_{n+1} = m$。

复杂度 $O(n^3 \log m)$

[hdu 6710] Kaguya

有一个二分图，两侧分别有 n, m 个点，中间每条边有 $1/2$ 概率存在。

求左边一个点到右边一个点的期望距离。(如果不连通，定义距离为 0)

多组数据，$T, n, m \leq 30$

最短路应该按照什么顺序 DP?

[hdu 6710] Kaguya

最短路应该按照什么顺序 DP？最短路是从小往大的走，因此考虑按照最短路从小到大 DP。

考虑如果确定了左边点 1 到所有点的距离 d，那么图需要满足什么条件？

[hdu 6710] Kaguya

最短路应该按照什么顺序 DP？最短路是从小往大的走，因此考虑按照最短路从小到大 DP。

考虑如果确定了左边点 1 到所有点的距离 d，那么图需要满足什么条件？

一个距离为 $d \geq 1$ 的点必须连向至少一个距离为 $d-1$ 的点（最短路的转移），不能连向距离 $< d-1$ 的点。

因为图是二分图，一定是左边距离偶数右边距离奇数，所以不存在相同 d 之间连边。

那么考虑按照距离，一层一层做。只会有相邻两层的边，因此只需要记录上一层有多少点。

[hdu 6710] Kaguya

最直接地，设 $f_{d,n,m,k}$ 表示当前考虑到距离 d，左边访问了 n 个点，右边访问了 m 个点，上一个距离这一层有 k 个点（可以根据 d 判定在哪一侧）。

转移枚举下一层有 t 个点。假设当前层在左侧，下一层在右侧。那么是 $f_{d,n,m,k}$ 转移到 $f_{d+1,n,m+t,t}$。考虑系数，根据之前的分析，这一层只连到上一层，且每个点必须连到上一层。那么系数是 $(2^{-(n-k)}(1-2^{-k}))^t$。

统计答案的时候按照 $f_{2t+1,n,m,k}$ 统计，认为第二个点在这 k 个里面（搜到了就不需要考虑剩下的部分了）。

复杂度 $O(Tn^2m^2(n+m))$，可以倒着做去掉 t 这一维，但这个已经可以通过了。

给一张 n 个点的图，选择一棵生成树，再选择一个根 u。记一条边的深度为它到根经过的其它边数量加一，最小化

$$\sum_{e \in \text{Spanning Tree}} w_e * dep_e$$

$n \leq 12$

[NOIP2017] 宝藏

如何对生成树进行 DP？最直接的想法还是从下往上 DP 子树。但现在子树不是固定的，因此我们可以子集（状压）DP：状态记录根 u 以及根的子树点集 S。

但代价和深度有关，这还会影响决策。因此考虑把深度也放进来：记 $dp_{d,u,S}$ 表示 u 为根，子树点集为 S，u 处深度（u 下面边的深度）为 d 时的最小代价。

转移考虑 u 新接出去一个子树 v，那么

$$dp_{d,u,S} \leftarrow dp_{d,u,S \setminus T} + dp_{d+1,v,S} + d * e_{u,v}$$

复杂度 $O(n^3 3^n)$，可能能过。

[NOIP2017] 宝藏

如何对生成树进行 DP？另一个方向（自上而下）的想法是，和之前的最短路一样，按照深度分层，然后每一层需要从上一层的某个点处连过来。

直观上看，我们的状态需要记录深度 d，点集 S，当前层的点集 T。然后我们扩展下一层 U，对于 U 里面的每个点，它需要向 T 里面的某个点连边，然后找到最小代价。这三个集合已经是 $O(4^n)$ 的复杂度。

[NOIP2017] 宝藏

如何对生成树进行 DP？另一个方向（自上而下）的想法是，和之前的最短路一样，按照深度分层，然后每一层需要从上一层的某个点处连过来。

直观上看，我们的状态需要记录深度 d，点集 S，当前层的点集 T。然后我们扩展下一层 U，对于 U 里面的每个点，它需要向 T 里面的某个点连边，然后找到最小代价。这三个集合已经是 $O(4^n)$ 的复杂度。

但我们真的需要这样吗？放大一个点的深度只会变差。考虑让 U 也可以从 S 连过来，这样转移只会把边的深度考虑大，因此不影响最优解（正确的解里面 U 已经提前加进去了）。

那么状态是 $dp_{d,S}$。预处理 $v_{u,S}$ 表示 S 中点到 u 的最小边权，然后

$$dp_{d,S} + \sum_{u \in T} d * v_{u,S} \to dp_{d+1,S+T}$$

复杂度 $O(n3^n)$

[CERC2014] Outer space invaders

在任意时刻，你可以花费 k 的代价进行一次强度为 k 的攻击。

有 n 个外星人，第 i 个外星人要求你在时间段 $[l_i, r_i]$ 内至少进行一次强度不小于 c_i 的攻击。

求最小总代价。

$n \le 300$

[CERC2014] Outer space invaders

按时间顺序做？不可以，因为我们可能解决一个 c 很小但很紧急的，但把一个 c 很大而不紧急的放到之后。这样每个时刻还没解决的外星人可以是任意集合。

什么角度可以划分问题？

[CERC2014] Outer space invaders

按时间顺序做？不可以，因为我们可能解决一个 c 很小但很紧急的，但把一个 c 很大而不紧急的放到之后。这样每个时刻还没解决的外星人可以是任意集合。

什么角度可以划分问题？如果我们做了一次最大 c 的攻击，那么所有 c 更小（且经过攻击时间点）的外星人都被解决了。因此接下来只需要考虑攻击时间点切开的两个区间。

[CERC2014] Outer space invaders

按时间顺序做？不可以，因为我们可能解决一个 c 很小但很紧急的，但把一个 c 很大而不紧急的放到之后。这样每个时刻还没解决的外星人可以是任意集合。

什么角度可以划分问题？如果我们做了一次最大 c 的攻击，那么所有 c 更小（且经过攻击时间点）的外星人都被解决了。因此接下来只需要考虑攻击时间点切开的两个区间。

记 $dp_{l,r}$ 表示只需要考虑完全被 $[l,r]$ 时间区间包含的外星人的最小代价。转移时找到这个区间内一个 c 最大的 (l_i,r_i,c_i)，枚举在哪个时间点攻击它，然后直接变为

$$dp_{l,r}=\max_{t\in[l_i,r_i]} c_i+dp_{l,t-1}+dp_{t+1,r}$$

时间端点可以离散化，复杂度 $O(n^3)$

[CERC2014] Parades

给一棵 n 个点的树，保证每个点度数不超过 d。

给 m 条路径，选出尽量多的路径，满足选出的路径不使用相同的边（可以使用相同的点）。求最多能选多少路径。

$n \leq 1000, d \leq 10$

[CERC2014] Parades

考虑一个 u 的子树需要记录什么状态。子树内考虑完后，只需要看选了哪些走到外面去的路径。因为 u 到父亲的边只能用一次，所以这样的路径最多有一条。到外面的路径可能有 n^2 条，但在子树内，我们只关心这一个端点是啥，所以只有 n 个状态。

记 dp_u 表示 u 子树内，不选向外的路径时的最优答案；记 $f_{u,v}$ 表示 u 子树内，想要选一条 v 到外面的路径（不统计这条向外的路径）...

[CERC2014] Parades

考虑一个 u 的子树需要记录什么状态。子树内考虑完后，只需要看选了哪些走到外面去的路径。因为 u 到父亲的边只能用一次，所以这样的路径最多有一条。到外面的路径可能有 n^2 条，但在子树内，我们只关心这一个端点是啥，所以只有 n 个状态。

记 dp_u 表示 u 子树内，不选向外的路径时的最优答案；记 $f_{u,v}$ 表示 u 子树内，想要选一条 v 到外面的路径（不统计这条向外的路径）...

但有必要吗？显然 $f_{u,v} \le dp_u$。如果 $f_{u,v} \le dp_u - 1$，那我们不如放弃这条路径，然后用 dp_u 的方案。因此只需要记录一个 $0/1$ 的值表示是否 $f_{u,v} = dp_u$。

现在考虑向上转移。首先考虑算 dp_u。如果不选经过 u 的路径，那就是 $\sum_{v\in son_u} dp_v$。但我们还可以选一些 u 子树内经过 u 的路径。

考虑选了一条路径 (a,b)，且 a 属于子树 v_i，b 属于子树 v_j。首先根据上一页的分析，如果 $f_{v_i,a} < dp_{v_i}$ 或者 $f_{v_j,b} < dp_{v_b}$，那不如不选这条路径。否则，我们相当于占了 v_i, v_j 两个子树，然后多选一条路径。

那么相当于有若干组 (v_i, v_j)，我们需要选出尽量多的组，使得它们两两不交。这是个一般图匹配，但这里 $d=10$，因此可以状压 DP：dp_S 表示 S 集合的最大匹配，每次加一条边。复杂度 $O(d^2 2^d)$

[CERC2014] Parades

现在考虑向上转移。首先考虑算 dp_u。如果不选经过 u 的路径，那就是 $\sum_{v\in son_u} dp_v$。但我们还可以选一些 u 子树内经过 u 的路径。

考虑选了一条路径 (a, b)，且 a 属于子树 v_i，b 属于子树 v_j。首先根据上一页的分析，如果 $f_{v_i,a} < dp_{v_i}$ 或者 $f_{v_j,b} < dp_{v_b}$，那不如不选这条路径。否则，我们相当于占了 v_i, v_j 两个子树，然后多选一条路径。

那么相当于有若干组 (v_i, v_j)，我们需要选出尽量多的组，使得它们两两不交。这是个一般图匹配，但这里 $d = 10$，因此可以状压 DP：dp_S 表示 S 集合的最大匹配，每次加一条边。复杂度 $O(d^2 2^d)$

然后考虑算 $f_{u,a}$。记 a 在 v_i 子树中。这相当于首先要有 $f_{v_i,a}$，然后在上面 v_i 子树还得保留下来。相当于 $dp_{S\setminus\{v_i\}}$。

注意到度数总和为 n，因此复杂度 $O(n^2 + nd2^d)$。

~~如果抄个 Tutte 可以~~ $\cancel{O(nd^2)}$

「PA 2019」Desant

给一个 n 阶排列，对于每个 $k = 1, 2, \cdots, n$，解决如下问题：

你需要选一个长度为 k 的子序列，最小化逆序对数量。求最小的逆序对数量和达成这个值的方案数。

$n \leq 40, 6s$

「PA 2019」Desant

按照序列顺序考虑。在处理了前 i 个数后，我们需要记录前面选了哪些数，以及前面的逆序对数——但这和暴力没有任何区别。

在需要根据后面选的数继续计算逆序对的情况下，可不可能少记录一点状态?

「PA 2019」Desant

按照序列顺序考虑。在处理了前 i 个数后，我们需要记录前面选了哪些数，以及前面的逆序对数——但这和暴力没有任何区别。

在需要根据后面选的数继续计算逆序对的情况下，可不可能少记录一点状态?

如果前面连续的 $i, i+1, i+2$ 都出现了，那么在计算后面的逆序对贡献时，这三个元素里面出现两个的情况贡献都是一样的，只需要记录出现多少个。

具体而言，在考虑了前 i 个后，我们把所有数从小到大排成一列，然后前面出现过的写 1，没有出现的写 0。那么对于每一段 1，我们只需要关心里面选了多少个（$0, 1, \cdots, l$）。

这样状态数是多少？每一段（结合后面那个 0）长度为 $l+1$，有 $l+1$ 种状态。状态数是把 $n+1$ 拆成若干个数乘起来。熟知其为 $O(3^{n/3})$。

「PA 2019」Desant

然后就是实现问题。如果每次转移 $O(n)$ 地更新段的状态，那复杂度大概是 $O(n^2 3^{m/3})$，这能过 6s 的时限。

但更精细的技巧可以卡到更好：首先状态可以用一个混合进制来表示，然后更新是局部的：合并两个段，加一个段，或者更改一个段的长度。然后大量进制操作可以做到 $O(1)$ 转移。

挑战：在 loj 上卡进 0.5s。

Thanks!

如果您是这些 DP 模型的初学者，那您更应该多加练习相关题目（这里讲到的题目只是很少的例子）以更加熟练。