DP

吴清月

2020 年 10 月 1 日

定义

定义

粘自维基：

动态规划（英语：Dynamic programming，简称 DP）是一种在数学、管理科学、计算机科学、经济学和生物信息学中使用的，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，动态规划方法所耗时间往往远少于朴素解法。

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。

应用

应用

用 DP 来求解的问题一般满足如下三个性质：

1. 最优子结构性质，即全局最优解包含的子问题的解也是最优解。
2. 无后效性，即前面的决策不收后面影响。
3. 子问题重叠性质。这也是 DP 可以比爆搜快的原因。

【IOI1994】数字三角形

【IOI1994】数字三角形

有一个由数字组成的三角形，第 i 行有 i 个数，一共有 n 行，你需要找出来一条从顶端到底端的路径，每次可以走向左边一个或者右边一个，你需要找出一条经过的数字权值最大的路径。

$n \leq 1000$

最长上升子序列

最长上升子序列

给定一个长度为 n 的序列 A，求最长上升子序列的长度。
$n \leq 1000$

最长公共子序列

最长公共子序列

给定两个长度分别为 n 和 m 的序列，求它们的最长公共子序列。

$n, m \leq 3000$

背包问题

背包问题

n 种物品，每一个物品有一个价值 v 和一个体积 w，你有一个大小为 m 的背包，你需要装进去最大价值的物品。

$n, m \leq 1000$

背包问题

n 种物品，每一个物品有一个价值 v 和一个体积 w，你有一个大小为 m 的背包，你需要装进去最大价值的物品。

$n, m \leq 1000$

01 背包：每种物品只有一个

完全背包：每种物品有无限多个

多重背包：每种物品有一个数量

【ZJOI2007】时态同步

【ZJOI2007】时态同步

给定一棵边带权的有根树，每次可以让一条边的边权 $+1$，求最少多少次操作可以让所有叶节点的深度相同。

$1 \leq n \leq 500000$

【ZJOI2007】时态同步

【ZJOI2007】时态同步

设 f_i 表示使得 i 这棵子树内所有叶节点深度相同，需要改变的最小次数。顺便记录一下这个深度。

转移直接考虑对父子边进行操作即可。

时间复杂度 $O(n)$。

CodeForces 1324F

CodeForces 1324F

给定一棵树，点有黑白两色，对于每一个点求一个包含这个点的连通子树，使得白点数量-黑点数量尽可能多。

$2 \leq n \leq 200000$

CodeForces 1324F

CodeForces 1324F

设 f_i 表示以 i 为根的树白点数量-黑点数量最多是多少。则可以得到状态转移方程：

$$f_u = c_u + \sum_{(u,v)} \max(0, f_v)$$

CodeForces 1324F

设 f_i 表示以 i 为根的树白点数量-黑点数量最多是多少。则可以得到状态转移方程：

$$f_u = c_u + \sum_{(u,v)} \max(0, f_v)$$

但是这样只能求出一个点向下的子树，为了解决这个问题我们还需要求出一个点向上的子树。

CodeForces 1324F

设 f_i 表示以 i 为根的树白点数量-黑点数量最多是多少。则可以得到状态转移方程：

$$f_u = c_u + \sum_{(u,v)} \max(0, f_v)$$

但是这样只能求出一个点向下的子树，为了解决这个问题我们还需要求出一个点向上的子树。

再设 g_i 表示在 i 节点上方选取包含 i 的连通子树，白点-黑点最多是多少。转移的时候用父节点推出子节点：

$$g_v = c_v + g_u + (f_u - \max(0, f_v) - c_u)$$

CodeForces 1324F

设 f_i 表示以 i 为根的树白点数量-黑点数量最多是多少。则可以得到状态转移方程：

$$f_u = c_u + \sum_{(u,v)} \max(0, f_v)$$

但是这样只能求出一个点向下的子树，为了解决这个问题我们还需要求出一个点向上的子树。

再设 g_i 表示在 i 节点上方选取包含 i 的连通子树，白点-黑点最多是多少。转移的时候用父节点推出子节点：

$$g_v = c_v + g_u + (f_u - \max(0, f_v) - c_u)$$

最后一个点的答案就是 $f_i + g_i - c_i$。时间复杂度 $O(n)$。

树上背包问题

树上背包问题

一棵 n 个节点的树，每一个节点上有一个重量为 w 价值为 v 的物品，要想选子节点必须先选父节点。求容量不超过 m 的情况下可以达到的最大价值。

$n, m \leq 3000$

树上背包问题

树上背包问题

类比背包问题，我们设 $f_{i,j}$ 表示从 i 的子树中选取物品，容量为 j 的方案数。

树上背包问题

类比背包问题，我们设 $f_{i,j}$ 表示从 i 的子树中选取物品，容量为 j 的方案数。

转移的时候对于 (u, v) 这条父子边：

$$f_{u,x} + f_{v,y} \to f_{u,x+y}$$

这个算法看起来是 $O(n^3)$ 的。

树上背包问题

类比背包问题，我们设 $f_{i,j}$ 表示从 i 的子树中选取物品，容量为 j 的方案数。

转移的时候对于 (u,v) 这条父子边：

$$f_{u,x}+f_{v,y}\to f_{u,x+y}$$

这个算法看起来是 $O(n^3)$ 的。

但是注意到，对于任意一个点 u，第二维最多只会到 $size_u$。所以这个算法的总时间复杂度为 $\sum_{f_u=f_v} size_u size_v$。在这个式子中每一个点对都只会贡献一次，所以时间复杂度其实是 $O(n^2)$ 的。

【HAOI2015】树上染色

【HAOI2015】树上染色

给你一棵 n 个节点的树，你需要选出来 k 个节点染成黑色，剩下的染成白色，你的收益是所有黑点两两之间的距离+所有白点两两之间的距离。

求最大收益。

$0 \leq k \leq n \leq 2000$

【HAOI2015】树上染色

【HAOI2015】树上染色

首先我们考虑如何方便地求出一个染色方案的收益。

我们考虑每一条边的贡献。对于一条边，若它连接的两棵子树内黑点数量分别为 a, b，白点数量分别为 c, d，则这条边的贡献就是 $ab + cd$。

【HAOI2015】树上染色

首先我们考虑如何方便地求出一个染色方案的收益。

我们考虑每一条边的贡献。对于一条边，若它连接的两棵子树内黑点数量分别为 a, b，白点数量分别为 c, d，则这条边的贡献就是 $ab + cd$。

所以我们设 $f_{i,j}$ 表示 i 的子树中选取了 j 个黑点的最大收益。转移就是一个树上背包，考虑 (u, v) 这条边的贡献即可。

时间复杂度 $O(n^2)$。

【BZOJ4987】Tree

【BZOJ4987】Tree

给定一棵 n 个点的边带权的树。
找出 k 个点 $A_1, A_2, \ldots, A_k$，使得 $\sum dis(A_i, A_{i+1})$ 最小。
$1 \leq k \leq n \leq 3000$

【BZOJ4987】Tree

【BZOJ4987】Tree

不妨先考虑 $k = n$ 的情况。这时候我们只需要选取一个遍历顺序即可。

对于每一条边，除非它位于 A_1 和 A_k 之间，否则一定至少被经过了两次（进去一次出来一次）。所以此时的答案即为 $2\sum w(e)-$直径长度。

【BZOJ4987】Tree

不妨先考虑 $k = n$ 的情况。这时候我们只需要选取一个遍历顺序即可。

对于每一条边，除非它位于 A_1 和 A_k 之间，否则一定至少被经过了两次（进去一次出来一次）。所以此时的答案即为 $2\sum w(e)-$直径长度。

接下来考虑 $k < n$ 的情况。此时的问题变为选出一个大小为 k 的连通子树，使得 $2\sum w(e)-$直径最小。

【BZOJ4987】Tree

【BZOJ4987】Tree

考虑 DP。我们设 $g(i,j)$ 表示在 i 这棵子树里选出来 j 个点，虚树总边长的最小值。

设 $f_0(i,j)$ 表示在 i 这棵子树里选出来 j 个点，而且这 j 个点的虚树直径有一端是 j，构成的虚树总边长$\times 2-$直径长度的最小值。

设 $f_1(i,j)$表示在 i 这棵子树里选出来 j 个点，而且这 j 个点的虚树直径两端都不是 j，构成的虚树总边长$\times 2-$直径长度的最小值。

【BZOJ4987】Tree

考虑 DP。我们设 $g(i,j)$ 表示在 i 这棵子树里选出来 j 个点，虚树总边长的最小值。

设 $f_0(i,j)$ 表示在 i 这棵子树里选出来 j 个点，而且这 j 个点的虚树直径有一端是 j，构成的虚树总边长$\times 2-$直径长度的最小值。

设 $f_1(i,j)$表示在 i 这棵子树里选出来 j 个点，而且这 j 个点的虚树直径两端都不是 j，构成的虚树总边长$\times 2-$直径长度的最小值。

这些都可以用树上背包来转移。

时间复杂度 $O(n^2)$。

【SCOI2005】互不侵犯

【SCOI2005】互不侵犯

在 $n \times n$ 的棋盘里面放 k 个国王，使他们互不攻击，共有多少种摆放方案。国王能攻击到它周围的八个格子。

$n \leq 7$?

$n \leq 9$?

$n \leq 15$?

【SCOI2005】互不侵犯

【SCOI2005】互不侵犯

一看数据范围这么小，我们考虑状态压缩。

设 $f_{i,j,S}$ 表示当前进行到第 i 行，放了 j 个王，上一行的状态是 S 的方案数。S 是一个 n 位的二进制数，01 分别表示是否放置了王。

【SCOI2005】互不侵犯

一看数据范围这么小，我们考虑状态压缩。

设 $f_{i,j,S}$ 表示当前进行到第 i 行，放了 j 个王，上一行的状态是 S 的方案数。S 是一个 n 位的二进制数，01 分别表示是否放置了王。

转移的时候枚举下一行的状态，判断是否合法进行转移。

这样的时间复杂度是 $O(4^n n^2)$ 的，可以通过 $n \leq 7$。

【SCOI2005】互不侵犯

【SCOI2005】互不侵犯

我们考虑优化这个算法。

首先我们注意到，每次枚举下一行的状态并不优秀。我们可以只枚举上一行状态的补集的子集。

【SCOI2005】互不侵犯

我们考虑优化这个算法。

首先我们注意到，每次枚举下一行的状态并不优秀。我们可以只枚举上一行状态的补集的子集。

这样枚举的时间复杂度是多少呢？

$$T(n)=\sum_{i=0}^{n}\binom{n}{i}2^{n-i}=\sum_{i=0}^{n}1^i2^{n-i}\binom{n}{i}=3^n$$

【SCOI2005】互不侵犯

我们考虑优化这个算法。

首先我们注意到，每次枚举下一行的状态并不优秀。我们可以只枚举上一行状态的补集的子集。

这样枚举的时间复杂度是多少呢？

$$T(n)=\sum_{i=0}^{n}\binom{n}{i}2^{n-i}=\sum_{i=0}^{n}1^i2^{n-i}\binom{n}{i}=3^n$$

于是我们成功把复杂度由 $O(4^nn^2)$ 优化到了 $O(3^nn^2)$，可以过掉 $n\leq 9$。

【SCOI2005】互不侵犯

【SCOI2005】互不侵犯

还可以继续优化!

注意到每一行可行的方案数其实非常有限，因为不可能存在相邻的两个 1。

【SCOI2005】互不侵犯

还可以继续优化!

注意到每一行可行的方案数其实非常有限，因为不可能存在相邻的两个 1。

所以我们可以将所有合法的方案预处理出来，这样枚举的时候就不用遍历所有 2^n 个集合了。

时间复杂度不会算，反正跑的飞快。

【SCOI2005】互不侵犯

【SCOI2005】互不侵犯

另一种优化思路：

注意到每次我们都枚举了一整行的状态。能不能一次枚举一个格子的状态呢？

【SCOI2005】互不侵犯

另一种优化思路：

注意到每次我们都枚举了一整行的状态。能不能一次枚举一个格子的状态呢？

可以！我们设 $f_{i,j,k,S}$ 表示到了第 i 行**第 j 列**，总共放了 k 个王，上一行的状态是 S 的方案数。这样我们只需要枚举 (i,j) 这一个格子的状态就好了。

时间复杂度降到 $O(2^n n^3)$，可以通过 $n \leq 15$，不过不一定比上一个优化更快。

【九省联考2018】一双木棋

【九省联考2018】一双木棋

有一个 $n \times m$ 的棋盘，每一个位置有一个 $A_{i,j}$ 和一个 $B_{i,j}$。

双方一黑一白轮流在上面落子。一个位置能够落子，当且仅当这个位置左上两个位置都已经有了棋子。

先手得分是所有黑棋位置的 $A_{i,j}$ 之和，后手得分是所有白棋位置的 $B_{i,j}$ 之和，双方都希望自己的得分减对方的得分最大。

问最优策略下最后先手得分减后手得分的结果。

$n, m \leq 10$

【九省联考2018】一双木棋

【九省联考2018】一双木棋

我们考虑设 f_s 表示从状态 s 出发，最后先手-后手的得分。

对于转移，我们考虑枚举哪些位置可以落子，假设落子后能够到达的所有状态是 t，那么 $f_s = \max(A_{i,j} + f_t)$（黑棋先）或 $f_s = \min(f_t - B_{i,j})$（白棋先）。

这样我们就有了转移的思路了。

【九省联考2018】一双木棋

我们考虑设 f_s 表示从状态 s 出发，最后先手-后手的得分。

对于转移，我们考虑枚举哪些位置可以落子，假设落子后能够到达的所有状态是 t，那么 $f_s = \max(A_{i,j} + f_t)$（黑棋先）或 $f_s = \min(f_t - B_{i,j})$（白棋先）。

这样我们就有了转移的思路了。

可是要怎么记录状态呢？

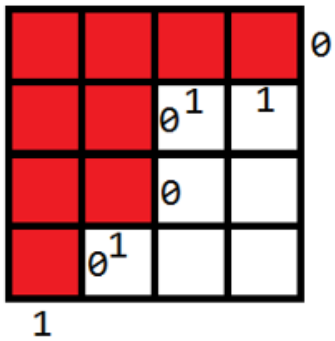
【九省联考2018】一双木棋

【九省联考2018】一双木棋

一个合法方案对应一条从左下走到右上的折线。
在这条折线上，我们用 0 代表向上，1 代表向右。

【九省联考2018】一双木棋

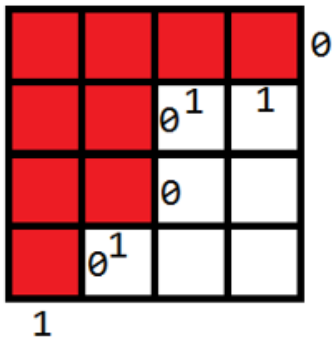
一个合法方案对应一条从左下走到右上的折线。
在这条折线上，我们用 0 代表向上，1 代表向右。
比如下面这种状态：



就对应了 10100110 这样一个二进制串。

【九省联考2018】一双木棋

一个合法方案对应一条从左下走到右上的折线。
在这条折线上，我们用 0 代表向上，1 代表向右。
比如下面这种状态：



就对应了 10100110 这样一个二进制串。

这样对于 $n \times m$ 的网格我们就能用一个长度为 $n+m$ 的二进制串表示。

转移就是将一个 01 变成 10，是 $O(n+m)$ 的，时间复杂度 $O((n+m)2^{n+m})$。

【联合省选2020】信息传递

【联合省选2020】信息传递

一条道路上有 m 个信号站 $1, 2, \ldots, m$，你需要将信息沿着 $S_1 \to S_2 \to \ldots \to S_n$ 的顺序传递。

对于一次传递 $x \to y$：

- 若 $x \le y$，则花费为 $y - x$；
- 若 $x > y$，则花费为 $kx + ky$。

你需要将所有的信号站重排，使得最后的代价之和最小。

$2 \le m \le 23, 2 \le n \le 10^5$

【联合省选2020】信息传递

【联合省选2020】信息传递

注意到我们不关心信息传递的顺序，只关心两个信号站之间进行了多少次传递。

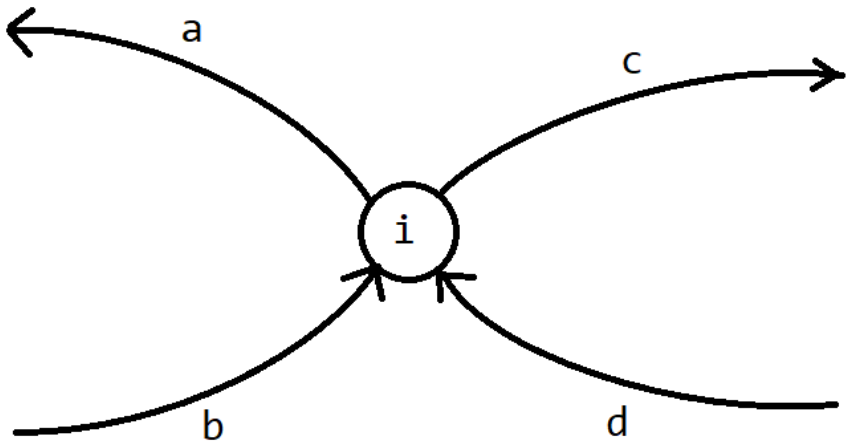
所以先预处理一个 `cnt` 数组，$cnt_{i,j}$ 表示 i 到 j 进行了多少次传递。

【联合省选2020】信息传递

注意到我们不关心信息传递的顺序，只关心两个信号站之间进行了多少次传递。

所以先预处理一个 cnt 数组，$cnt_{i,j}$ 表示 i 到 j 进行了多少次传递。

对于一个位于 i 的信号站，我们希望算出这个位置的系数是多少。假设它向左传递了 a 次，左边向它传递了 b 次，它向右边传递了 c 次，右边向它传递了 d 次，也就是这个样子：



【联合省选2020】信息传递

【联合省选2020】信息传递

接下来开始推式子：

$$a(i-x)+bk(i+x)+ck(i+x)+d(x-i)\to(a+bk+ck-d)i$$

也就是说 i 这个点坐标的系数为 $a+bk+ck-d$。

【联合省选2020】信息传递

接下来开始推式子：

$$a(i-x)+bk(i+x)+ck(i+x)+d(x-i)\to(a+bk+ck-d)i$$

也就是说 i 这个点坐标的系数为 $a+bk+ck-d$。

状压 DP，从左到右依次填充。设 f_S 表示 S 这个集合已经排好了的情况下，代价的最小值。转移的时候枚举下一个信号站是哪一个，然后枚举其它所有点算出对应的 a,b,c,d，进行转移。

【联合省选2020】信息传递

接下来开始推式子：

$$a(i-x)+bk(i+x)+ck(i+x)+d(x-i)\to(a+bk+ck-d)i$$

也就是说 i 这个点坐标的系数为 $a+bk+ck-d$。

状压 DP，从左到右依次填充。设 f_S 表示 S 这个集合已经排好了的情况下，代价的最小值。转移的时候枚举下一个信号站是哪一个，然后枚举其它所有点算出对应的 a,b,c,d，进行转移。

这样的时间复杂度是 $O(2^m m^2)$ 的。

【联合省选2020】信息传递

【联合省选2020】信息传递

考虑优化这个算法。每次枚举其它所有点显然太不优秀了，我们考虑预处理出 $g_{i,s}$，表示 i 左边点集为 s 的情况下 i 这个点的系数。

【联合省选2020】信息传递

考虑优化这个算法。每次枚举其它所有点显然太不优秀了，我们考虑预处理出 $g_{i,s}$，表示 i 左边点集为 s 的情况下 i 这个点的系数。

首先 $g_{i,0} = \mathrm{out}(i)k - \mathrm{in}(i)$，接下来我们考虑一个点 j 由右边挪到左边的时候，坐标的系数会怎样变化。

新的 a, b, c, d 分别为 $a + cnt(i,j), b + cnt(j,i), c - cnt(i,j), d - cnt(j,i)$，因此系数变化为 $cnt(i,j) + cnt(j,i)k - cnt(i,j)k + cnt(j,i)$。得到预处理的状态转移方程：

【联合省选2020】信息传递

考虑优化这个算法。每次枚举其它所有点显然太不优秀了，我们考虑预处理出 $g_{i,s}$，表示 i 左边点集为 s 的情况下 i 这个点的系数。

首先 $g_{i,0} = \text{out}(i)k - \text{in}(i)$，接下来我们考虑一个点 j 由右边挪到左边的时候，坐标的系数会怎样变化。

新的 a, b, c, d 分别为 $a + cnt(i,j), b + cnt(j,i), c - cnt(i,j), d - cnt(j,i)$，因此系数变化为 $cnt(i,j) + cnt(j,i)k - cnt(i,j)k + cnt(j,i)$。得到预处理的状态转移方程：

$$g_{i,s\bigcup\{j\}} = g_{i,s} + cnt(i,j) + cnt(j,i)k - cnt(i,j)k + cnt(j,i)$$

【联合省选2020】信息传递

考虑优化这个算法。每次枚举其它所有点显然太不优秀了，我们考虑预处理出 $g_{i,s}$，表示 i 左边点集为 s 的情况下 i 这个点的系数。

首先 $g_{i,0} = \text{out}(i)k - \text{in}(i)$，接下来我们考虑一个点 j 由右边挪到左边的时候，坐标的系数会怎样变化。

新的 a, b, c, d 分别为 $a + cnt(i,j), b + cnt(j,i), c - cnt(i,j), d - cnt(j,i)$，因此系数变化为 $cnt(i,j) + cnt(j,i)k - cnt(i,j)k + cnt(j,i)$。得到预处理的状态转移方程：

$$g_{i,s\bigcup\{j\}} = g_{i,s} + cnt(i,j) + cnt(j,i)k - cnt(i,j)k + cnt(j,i)$$

总的状态转移方程即为 $f_s = \min_{i \in s} f_{s-\{i\}} + g_{i,s-\{i\}}$。

时间复杂度 $O(m2^m)$

【BZOJ1026】windy 数

【BZOJ1026】windy 数

我们定义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 $[A, B]$ 之间的 windy 数的个数。

$1 \leq A \leq B \leq 2 \times 10^9$

【BZOJ1026】windy 数

【BZOJ1026】windy 数

数位 DP 入门题。
首先差分，变成了求 $[1, n]$ 之间 windy 数的个数。

【BZOJ1026】windy 数

数位 DP 入门题。

首先差分，变成了求 $[1, n]$ 之间 windy 数的个数。

接下来在数位上 DP。设 $f_{i,j}$ 表示已经确定了前 i 位，第 i 位的数字是 j 的 windy 数个数。

【BZOJ1026】windy 数

数位 DP 入门题。

首先差分，变成了求 $[1, n]$ 之间 windy 数的个数。

接下来在数位上 DP。设 $f_{i,j}$ 表示已经确定了前 i 位，第 i 位的数字是 j 的 windy 数个数。

然而光这样还不够。因为前 i 位可能与 n 一样也可能不一样，这直接影响到 $i+1$ 位的取值范围。

【BZOJ1026】windy 数

数位 DP 入门题。

首先差分，变成了求 $[1, n]$ 之间 windy 数的个数。

接下来在数位上 DP。设 $f_{i,j}$ 表示已经确定了前 i 位，第 i 位的数字是 j 的 windy 数个数。

然而光这样还不够。因为前 i 位可能与 n 一样也可能不一样，这直接影响到 $i+1$ 位的取值范围。

所以多记录一个 01 表示当前这些位是否和 n 一样。转移的时候枚举下一位填什么数。

【AHOI2009】同类分布

【AHOI2009】同类分布

求 $[a,b]$ 中各位数字之和能整除原数的数的个数。
$1 \le a \le b \le 10^{18}$

【AHOI2009】同类分布

【AHOI2009】同类分布

各位数字之和可能受到后面的影响，因此不可能直接 DP。

但是我们注意到，各位数字之和最大不过是 $9 \times 18 = 162$，这个数字是可以枚举的，为了方便记为 d。

【AHOI2009】同类分布

各位数字之和可能受到后面的影响，因此不可能直接 DP。

但是我们注意到，各位数字之和最大不过是 $9 \times 18 = 162$，这个数字是可以枚举的，为了方便记为 d。

枚举完之后我们进行 DP，设 $f_{i,j,k,0/1}$ 表示填好了前 i 位，总和为 j，模 d 的值是 k，是否卡上界的方案数。最后答案即为 $f_{18,d,0,0/1}$。

CodeForces 908G

CodeForces 908G

令 $S(i)$ 表示把 i 中的所有数位上的数拿出来，从小到大排序后形成的新的数。比如 $S(50394) = 3459$。

给你一个 X，问你 $\sum_{i=1}^{X} S(i)$。

$X \leq 10^{700}$

CodeForces 908G

CodeForces 908G

我们分开考虑每一位的贡献。

对于第 i 位，如果数字为 d 的方案数有 k_d 种，那么对答案的贡献就是 $10^i \sum dk_d$。

CodeForces 908G

我们分开考虑每一位的贡献。

对于第 i 位，如果数字为 d 的方案数有 k_d 种，那么对答案的贡献就是 $10^i \sum dk_d$。

先枚举一个 d。然后考虑前缀和，求这一位上的数字至多为 d 的方案数。相当于要求大于 d 的数不能超过 i 个。

CodeForces 908G

我们分开考虑每一位的贡献。

对于第 i 位，如果数字为 d 的方案数有 k_d 种，那么对答案的贡献就是 $10^i \sum dk_d$。

先枚举一个 d。然后考虑前缀和，求这一位上的数字至多为 d 的方案数。相当于要求大于 d 的数不能超过 i 个。

设 $f_{i,j,0/1}$ 表示进行到第 i 位，大于 d 的数已经有了 j 个，是否卡上界的方案数。转移直接看下一位是大于 d 还是小于等于 d。

【CCPC2020】Xor

【CCPC2020】Xor

给定 A, B, K, W，求有多少对 (x, y) 满足如下条件：

1. x, y 是整数
2. $x \in [0, A], y \in [0, B]$
3. $|x - y| \leq K$
4. $x \text{ xor } y \leq W$

$1 \leq T \leq 2000$

$0 \leq A, B, K, W \leq 10^9$

【CCPC2020】Xor

【CCPC2020】Xor

对于第二个限制，我们可以记两个 0/1，分别表示 x 和 y 是否卡上界。

对于第四个限制，我们可以记一个 0/1 表示是否卡上界。

【CCPC2020】Xor

对于第二个限制，我们可以记两个 0/1，分别表示 x 和 y 是否卡上界。

对于第四个限制，我们可以记一个 0/1 表示是否卡上界。

对于第三个限制，我们需要首先记录一个大小关系（0/1/2），然后记录一下当前两数之差是小于上界，等于上界还是刚好比上界多 1（0/1/2）。

【CCPC2020】Xor

对于第二个限制，我们可以记两个 0/1，分别表示 x 和 y 是否卡上界。

对于第四个限制，我们可以记一个 0/1 表示是否卡上界。

对于第三个限制，我们需要首先记录一个大小关系（0/1/2），然后记录一下当前两数之差是小于上界，等于上界还是刚好比上界多 1（0/1/2）。

总状态数 $9 \times 2 \times 2 \times 2 \times 3 \times 3 = 648$，可以通过本题。

基础概念

基础概念

概率：一件事情发生的可能性。
期望：也就是加权平均值，$E = \sum P_i w_i$。

基础概念

概率：一件事情发生的可能性。
期望：也就是加权平均值，$E = \sum P_i w_i$。
两者的联系：概率为 p 的事件期望 $\frac{1}{p}$ 次后发生。
独立事件：对于两个独立事件 A, B，$P(AB) = P(A)P(B)$。

期望的线性性

期望的线性性

$E(x+y)=E(x)+E(y)$，在任意情况下均成立。
$E(xy)=E(x)E(y)$，只在 x,y 互相独立的情况下成立。

期望的线性性

$E(x+y)=E(x)+E(y)$，在任意情况下均成立。
$E(xy)=E(x)E(y)$，只在 x,y 互相独立的情况下成立。
方差期望：

$$\begin{aligned}D(x)&=E\left((x-E(x))^2\right)\\&=E\left(x^2-2xE(x)+E^2(x)\right)\\&=E(x^2)-2E(x)E(x)+E^2(x)\\&=E(x^2)-E^2(x)\end{aligned}$$

【NOIP2016】换教室

【NOIP2016】换教室

一天一共有 n 节课，第 i 节课默认在第 c_i 间教室上，可以申请换到第 d_i 间教室上，申请有 k_i 的概率通过，从第 x 间教室到第 y 间教室有距离 $dis(x,y)$，你最多可以申请 m 门课，要求期望跑的距离尽量少。

$n, m \leq 2000$

【NOIP2016】换教室

【NOIP2016】换教室

第 i 节课和第 $i+1$ 节课之间期望路径长度只和这两节课是否申请换教室有关，并且可以根据期望的定义直接计算。

【NOIP2016】换教室

第 i 节课和第 $i+1$ 节课之间期望路径长度只和这两节课是否申请换教室有关，并且可以根据期望的定义直接计算。

接下来就很简单了，设 $f_{i,j,0/1}$ 表示到了第 i 节课，已经申请换了 j 节课，第 i 节课是否换教室的最短期望距离。

时间复杂度 $O(nm)$。

【BZOJ4318】OSU!

【BZOJ4318】OSU!

有一个长度为 n 的序列，第 i 个位置有 p_i 的概率为 1，$1-p_i$ 的概率为 0，一个序列的分数是所有极长连续的 1 的长度的三次方和。求期望分数。

$n \leq 100000$

【BZOJ4318】OSU!

【BZOJ4318】OSU!

考虑对于每一个位置，求出以它结尾的机场连续的 1 的三次方的期望 f_i。则最后的答案即为 $\sum_{i=1}^{n-1} f_i(1-p_{i+1})+f_n$。

【BZOJ4318】OSU!

考虑对于每一个位置，求出以它结尾的机场连续的 1 的三次方的期望 f_i。则最后的答案即为 $\sum_{i=1}^{n-1} f_i(1-p_{i+1})+f_n$。

推一下转移方程：$f_i = p_i E\left((x+1)^3\right) = p_i E(x^3+3x^2+3x+1) = p_i(f_{i-1}+3E(x^2)+3E(x)+1)$。

【BZOJ4318】OSU!

考虑对于每一个位置，求出以它结尾的机场连续的 1 的三次方的期望 f_i。则最后的答案即为 $\sum_{i=1}^{n-1} f_i(1-p_{i+1})+f_n$。

推一下转移方程：$f_i = p_i E\left((x+1)^3\right) = p_i E(x^3+3x^2+3x+1) = p_i(f_{i-1}+3E(x^2)+3E(x)+1)$。

看来还需要设一下二次方的期望 g_i 和一次方的期望 h_i。

【BZOJ4318】OSU!

考虑对于每一个位置，求出以它结尾的机场连续的 1 的三次方的期望 f_i。则最后的答案即为 $\sum_{i=1}^{n-1} f_i(1-p_{i+1})+f_n$。

推一下转移方程：$f_i = p_i E\left((x+1)^3\right) = p_i E(x^3+3x^2+3x+1) = p_i(f_{i-1}+3E(x^2)+3E(x)+1)$。

看来还需要设一下二次方的期望 g_i 和一次方的期望 h_i。

最后的转移方程如下：

- $f_i = p_i(f_{i-1}+3g_{i-1}+3h_{i-1}+1)$
- $g_i = p_i E\left((x+1)^2\right) = p_i E(x^2+2x+1) = p_i(g_{i-1}+2h_{i-1}+1)$
- $h_i = p_i(h_{i-1}+1)$

【BZOJ4318】OSU!

考虑对于每一个位置，求出以它结尾的机场连续的 1 的三次方的期望 f_i。则最后的答案即为 $\sum_{i=1}^{n-1} f_i(1-p_{i+1})+f_n$。

推一下转移方程：$f_i = p_i E\left((x+1)^3\right) = p_i E(x^3+3x^2+3x+1) = p_i(f_{i-1}+3E(x^2)+3E(x)+1)$。

看来还需要设一下二次方的期望 g_i 和一次方的期望 h_i。

最后的转移方程如下：

- $f_i = p_i(f_{i-1}+3g_{i-1}+3h_{i-1}+1)$
- $g_i = p_i E\left((x+1)^2\right) = p_i E(x^2+2x+1) = p_i(g_{i-1}+2h_{i-1}+1)$
- $h_i = p_i(h_{i-1}+1)$

时间复杂度 $O(n)$。

【CSP2019】Emiya 家今天的饭

【CSP2019】Emiya 家今天的饭

有 n 种烹饪方法和 m 种主要食材，使用第 i 中烹饪方法和第 j 种主要食材可以做出 $a_{i,j}$ 道不同的饭。

对于一道搭配 k 道菜的搭配方案，必须满足所有菜的烹饪方法不同，且任意一种食材至多在 $\lfloor \frac{k}{2} \rfloor$ 道菜中被使用。

求搭配方案的方案数对 998244353 取模的值。

$n \leq 100, m \leq 2000$

【CSP2019】Emiya 家今天的饭

【CSP2019】Emiya 家今天的饭

考虑容斥。先随便选取食材，接下来每次枚举一种食材使用超过 $\lfloor \frac{k}{2} \rfloor$ 次，然后 DP 出方案数再从答案里面减去。

【CSP2019】Emiya 家今天的饭

考虑容斥。先随便选取食材，接下来每次枚举一种食材使用超过 $\lfloor \frac{k}{2} \rfloor$ 次，然后 DP 出方案数再从答案里面减去。

DP 的过程可以设 $f_{i,j,k}$，表示考虑了前 i 种烹饪方法，枚举的食材使用了 j 次，其它食材使用了 k 次的方案。

转移的时候枚举这一道菜使用了哪种烹饪方法：

$$f_{i,j,k} = a_{i,x} f_{i-1,j-1,k} + (s_i - a_{i,x}) f_{i-1,j,k-1} + f_{i-1,j,k}$$

其中 $s_i = \sum_j a_{i,j}$。

【CSP2019】Emiya 家今天的饭

考虑容斥。先随便选取食材，接下来每次枚举一种食材使用超过 $\lfloor \frac{k}{2} \rfloor$ 次，然后 DP 出方案数再从答案里面减去。

DP 的过程可以设 $f_{i,j,k}$，表示考虑了前 i 种烹饪方法，枚举的食材使用了 j 次，其它食材使用了 k 次的方案。

转移的时候枚举这一道菜使用了哪种烹饪方法：

$$f_{i,j,k} = a_{i,x} f_{i-1,j-1,k} + (s_i - a_{i,x}) f_{i-1,j,k-1} + f_{i-1,j,k}$$

其中 $s_i = \sum_j a_{i,j}$。

最后所有 $f_{n,i,j}$（$i > j$）加起来即为不合法的方案。这样的时间复杂度是 $O(n^3 m)$ 的。

【CSP2019】Emiya 家今天的饭

【CSP2019】Emiya 家今天的饭

考虑优化这个算法。注意到，我们不关心 j,k 具体是多少，而只关心最后 $j-k$ 的正负。所以可以直接设 $f_{i,j}$，表示考虑了前 i 中烹饪方法，枚举的食材减去其它的食材差为 j 的方案数。

【CSP2019】Emiya 家今天的饭

考虑优化这个算法。注意到，我们不关心 j,k 具体是多少，而只关心最后 $j-k$ 的正负。所以可以直接设 $f_{i,j}$，表示考虑了前 i 中烹饪方法，枚举的食材减去其它的食材差为 j 的方案数。

转移可以直接写出来：

$$f_{i,j}=a_{i,x}f_{i-1,j-1}+(s_i-a_{i,x})f_{i-1,j+1}+f_{i-1,j}$$

【CSP2019】Emiya 家今天的饭

考虑优化这个算法。注意到，我们不关心 j,k 具体是多少，而只关心最后 $j-k$ 的正负。所以可以直接设 $f_{i,j}$，表示考虑了前 i 中烹饪方法，枚举的食材减去其它的食材差为 j 的方案数。

转移可以直接写出来：

$$f_{i,j}=a_{i,x}f_{i-1,j-1}+(s_i-a_{i,x})f_{i-1,j+1}+f_{i-1,j}$$

时间复杂度 $O(n^2m)$。

CodeForces 1216F

CodeForces 1216F

有一行 n 间屋子，你需要将它们连入互联网。

对于第 i 间屋子，你可以选择直接接入互联网，花费为 i，或者在这间屋子里建一个 Wi-Fi，花费仍然是 i，但是可以覆盖 $[i-k, i+k]$ 区间内所有的屋子。

给定一个 01 串 s，只有 $s_i = 1$ 的屋子可以建 Wi-Fi，你需要求出将它们都接入互联网的最小花费。

$1 \leq n, k \leq 2 \times 10^5$

CodeForces 1216F

CodeForces 1216F

首先可以 DP，设 f_i 表示将前 i 间屋子都接入互联网的最小代价。

CodeForces 1216F

首先可以 DP，设 f_i 表示将前 i 间屋子都接入互联网的最小代价。

转移的时候看这个位置是直接接还是建 Wi-Fi：

$$f_{i-1} + i \to f_i$$

$$f_{i-k-1} + i \to f_{i+k+1}$$

最后还需要让 $f_i = \min_{j \geq i} f_j$。

CodeForces 1216F

首先可以 DP，设 f_i 表示将前 i 间屋子都接入互联网的最小代价。

转移的时候看这个位置是直接接还是建 Wi-Fi：

$$f_{i-1} + i \to f_i$$

$$f_{i-k-1} + i \to f_{i+k+1}$$

最后还需要让 $f_i = \min_{j \geq i} f_j$。

用线段树即可实现区间取 min。时间复杂度 $O(n \log n)$。

随机游走

随机游走

给定一张 n 个点 m 条边的图，告诉你一个起点，每次从当前点出发随机选择一条边走过去，求 k 次之后位于每一个点的概率。

$n \leq 100, m \leq n(n-1), k \leq 10^9$

随机游走

随机游走

首先可以 DP。设 $f_{i,j}$ 表示走了 i 步位于 j 的概率。转移的时候枚举出边。

这样的时间复杂度是 $O(kn^2)$ 的。

随机游走

首先可以 DP。设 $f_{i,j}$ 表示走了 i 步位于 j 的概率。转移的时候枚举出边。

这样的时间复杂度是 $O(kn^2)$ 的。

注意到第二维很小，只有 100，所以可以用矩阵乘法优化。时间复杂度降到 $O(n^3 \log k)$。

【NOI2020】美食家

【NOI2020】美食家

给定一张 n 个点 m 条边的图，点有点权，边有边权，边权范围 $[1,5]$。

你需要从 1 号点出发走一条长度为 T 的回路最后回到 1，收益为路径上所有点的点权之和（经过多次的点计算多次）。

此外，还有 k 个特殊事件 (t_i, x_i, y_i)，表示如果你在 t_i 时刻刚好位于 x_i，那么收益额外加 y_i。

求最大收益。

$n \leq 50, m \leq 501, k \leq 200, 1 \leq t_i \leq T \leq 10^9$

【NOI2020】美食家

【NOI2020】美食家

类比上一道题，设 $f_{i,j}$ 表示长度为 i 到达 j 的最大收益。这样的时间复杂度是 $O(nT)$ 的。

【NOI2020】美食家

类比上一道题，设 $f_{i,j}$ 表示长度为 i 到达 j 的最大收益。这样的时间复杂度是 $O(nT)$ 的。

矩阵乘法优化的难点主要有两个，一是边权并不都是 1，二是中间有 k 个特殊事件。

【NOI2020】美食家

类比上一道题，设 $f_{i,j}$ 表示长度为 i 到达 j 的最大收益。这样的时间复杂度是 $O(nT)$ 的。

矩阵乘法优化的难点主要有两个，一是边权并不都是 1，二是中间有 k 个特殊事件。

对于第一点，我们可以把每一个点拆成五个，分别表示当前时刻和之前的四个时刻。由这些我们就可以推出下一个时刻，矩阵大小是 $5n \times 5n$。

【NOI2020】美食家

类比上一道题，设 $f_{i,j}$ 表示长度为 i 到达 j 的最大收益。这样的时间复杂度是 $O(nT)$ 的。

矩阵乘法优化的难点主要有两个，一是边权并不都是 1，二是中间有 k 个特殊事件。

对于第一点，我们可以把每一个点拆成五个，分别表示当前时刻和之前的四个时刻。由这些我们就可以推出下一个时刻，矩阵大小是 $5n \times 5n$。

对于第二点，我们可以用分段矩阵乘法，每次用矩阵乘法转移到特殊事件的时刻，处理好这个特殊时间再进行下一段矩乘。

【NOI2020】美食家

类比上一道题，设 $f_{i,j}$ 表示长度为 i 到达 j 的最大收益。这样的时间复杂度是 $O(nT)$ 的。

矩阵乘法优化的难点主要有两个，一是边权并不都是 1，二是中间有 k 个特殊事件。

对于第一点，我们可以把每一个点拆成五个，分别表示当前时刻和之前的四个时刻。由这些我们就可以推出下一个时刻，矩阵大小是 $5n \times 5n$。

对于第二点，我们可以用分段矩阵乘法，每次用矩阵乘法转移到特殊事件的时刻，处理好这个特殊时间再进行下一段矩乘。

时间复杂度为 $O((5n)^3 k \log T)$，难以通过本题。

【NOI2020】美食家

【NOI2020】美食家

考虑矩阵乘法的过程。由这个特殊事件到下一个特殊事件的过程中，本质上我们是在用一个向量（也就是 DP 数组）乘上一个矩阵的若干次方。如果预处理 2^k 次幂的话，相当于用一个向量乘上 log 个矩阵。

【NOI2020】美食家

考虑矩阵乘法的过程。由这个特殊事件到下一个特殊事件的过程中，本质上我们是在用一个向量（也就是 DP 数组）乘上一个矩阵的若干次方。如果预处理 2^k 次幂的话，相当于用一个向量乘上 log 个矩阵。

我们注意到，矩阵乘矩阵是 $O(n^3)$ 的，但是向量乘矩阵是 $O(n^2)$ 的，而矩阵乘法又满足结合律，所以我们可以直接拿这个向量一路乘过去，而不用先把所有矩阵乘起来。

【NOI2020】美食家

考虑矩阵乘法的过程。由这个特殊事件到下一个特殊事件的过程中，本质上我们是在用一个向量（也就是 DP 数组）乘上一个矩阵的若干次方。如果预处理 2^k 次幂的话，相当于用一个向量乘上 log 个矩阵。

我们注意到，矩阵乘矩阵是 $O(n^3)$ 的，但是向量乘矩阵是 $O(n^2)$ 的，而矩阵乘法又满足结合律，所以我们可以直接拿这个向量一路乘过去，而不用先把所有矩阵乘起来。

时间复杂度降到 $O((5n)^3 \log T + (5n)^2 k \log T)$，前面是预处理 2^k 次幂。

【NOI2020】命运

【NOI2020】命运

给定一棵 n 个点的树，再给定 m 条父子链，你需要将所有树边染为 01，使得每一条父子链上至少有一个 1。求方案数。

$n, m \leq 500000$

【NOI2020】命运

【NOI2020】命运

对于一条父子链 (u, v) 和这条链上的一个点 x，若 (x, v) 上没有 1 边，则相当于要求 x 向上到 v 的若干条边中必须有一个 1 边。

所以可以设 $f_{i,j}$，表示 i 的子树已经染完，并且 i 上方到达深度为 j 的祖先必须有至少一个 1 边的方案数。

【NOI2020】命运

对于一条父子链 (u, v) 和这条链上的一个点 x，若 (x, v) 上没有 1 边，则相当于要求 x 向上到 v 的若干条边中必须有一个 1 边。

所以可以设 $f_{i,j}$，表示 i 的子树已经染完，并且 i 上方到达深度为 j 的祖先必须有至少一个 1 边的方案数。

对于 (u, v) 这条边，状态转移方程如下：

$$f_{u,i} f_{v,j} \to f_{u,\min(i,j)} \text{（}(u,v)\text{这条边为 0）}$$

$$f_{u,i} f_{v,j} \to f_{u,i} \text{（}(u,v)\text{这条边为 1）}$$

【NOI2020】命运

对于一条父子链 (u,v) 和这条链上的一个点 x，若 (x,v) 上没有 1 边，则相当于要求 x 向上到 v 的若干条边中必须有一个 1 边。

所以可以设 $f_{i,j}$，表示 i 的子树已经染完，并且 i 上方到达深度为 j 的祖先必须有至少一个 1 边的方案数。

对于 (u,v) 这条边，状态转移方程如下：

$$f_{u,i}f_{v,j} \to f_{u,\min(i,j)} \text{（}(u,v)\text{这条边为 0）}$$

$$f_{u,i}f_{v,j} \to f_{u,i} \text{（}(u,v)\text{这条边为 1）}$$

直接做的时间复杂度是 $O(n^2)$ 的。

【NOI2020】命运

【NOI2020】命运

合并的过程等价于第二维取 min。

可以用线段树合并来优化。每一个点开一个线段树记录 DP 数组。

【NOI2020】命运

合并的过程等价于第二维取 min。

可以用线段树合并来优化。每一个点开一个线段树记录 DP 数组。

在合并两棵线段树 T_1, T_2 的过程中，每次把 T_1 的右子树的和乘到 T_2 的左子树中，把 T_2 的右子树的和乘到 T_1 的左子树中。

时间复杂度 $O(n \log n)$。

CodeForces 1402C

CodeForces 1402C

有 D 棵一模一样的 n 个点的树排成一排，分别即为 $T_1, T_2, \ldots, T_D$。你需要对于每一个 i 在 T_i 和 T_{i+1} 之间任取两个点连一条边，把这 D 棵树变成一棵树。

在你连完之后，Alice 和 Bob 会在这棵树上玩一个游戏。在原先第 1 棵树的 1 号点处有一枚棋子，先手后手轮流移动，每次可以把它移动到一个之前没有被移动到的点，不能移动的人算输。

现在你要求出，有多少个将这 D 棵树连成一棵树的方案，使得先手必胜。答案对 $10^9 + 7$ 取模。

$1 \leq n \leq 10^5, 1 \leq D \leq 10^{18}$

CodeForces 1402C

CodeForces 1402C

首先考虑 $D = 1$ 的情况，也就是如何判断是否先手必胜。

以 1 号点为根对整棵树进行 dfs，一个点为必胜点，当且仅当存在子节点为必败点。

CodeForces 1402C

首先考虑 $D = 1$ 的情况，也就是如何判断是否先手必胜。

以 1 号点为根对整棵树进行 dfs，一个点为必胜点，当且仅当存在子节点为必败点。

接下来考虑如何对每一个点，求出这个点出发是必胜点还是必败点。

用换根 DP 的思想，设一个 f_i 和一个 g_i 分别表示下方子树和上方子树是否是必胜点。

CodeForces 1402C

CodeForces 1402C

接下来考虑加上一棵子树会对答案造成怎样的影响。

CodeForces 1402C

接下来考虑加上一棵子树会对答案造成怎样的影响。

- 必胜子树加到必胜点上：无影响；
- 必胜子树加到必败点上：无影响；
- 必败子树加到必胜点上：无影响；
- 必败子树加到必败点上：必败点变为了必胜点。

CodeForces 1402C

接下来考虑加上一棵子树会对答案造成怎样的影响。

- 必胜子树加到必胜点上：无影响；
- 必胜子树加到必败点上：无影响；
- 必败子树加到必胜点上：无影响；
- 必败子树加到必败点上：必败点变为了必胜点。

显然只有第四种情况会对这棵树造成影响。

CodeForces 1402C

CodeForces 1402C

对于每一个点，我们预处理一个 f'_i 和一个 g'_i，分别表示有多少种选取一个必败点变为必胜点的方案，使得 i 下方/上方子树的胜负性发生改变。

CodeForces 1402C

对于每一个点，我们预处理一个 f'_i 和一个 g'_i，分别表示有多少种选取一个必败点变为必胜点的方案，使得 i 下方/上方子树的胜负性发生改变。

如果 i 本来是必败点，则将它变为必胜点等价于将它的其中一棵子树变为必败点或者直接将它变为必胜点。

如果 i 本来是必胜点，则若有两个必败子树那就不可能发生改变，否则等价于将唯一的必败子树变为必胜。

CodeForces 1402C

对于每一个点，我们预处理一个 f'_i 和一个 g'_i，分别表示有多少种选取一个必败点变为必胜点的方案，使得 i 下方/上方子树的胜负性发生改变。

如果 i 本来是必败点，则将它变为必胜点等价于将它的其中一棵子树变为必败点或者直接将它变为必胜点。

如果 i 本来是必胜点，则若有两个必败子树那就不可能发生改变，否则等价于将唯一的必败子树变为必胜。

接下来我们枚举一个 i，遍历它的所有出边，用类似的方式求出有多少种选取一个必败点变为必胜点的方案，使得从这个点出发的胜负性发生改变。（注意是从这个点出发，而不是这个点下方/上方的子树）

CodeForces 1402C

CodeForces 1402C

接下来 DP。设 $f_{i,0/1}$ 表示将 i 棵树连到一起，最后挑出来一个必败点/必胜点的方案数。

CodeForces 1402C

接下来 DP。设 $f_{i,0/1}$ 表示将 i 棵树连到一起，最后挑出来一个必败点/必胜点的方案数。

由 f_i 转移到 f_{i+1} 只需要考虑我们挑出来的是哪个点，以及 i 棵子树挑出来的那个点是否使它的胜负性发生改变。

CodeForces 1402C

接下来 DP。设 $f_{i,0/1}$ 表示将 i 棵树连到一起，最后挑出来一个必败点/必胜点的方案数。

由 f_i 转移到 f_{i+1} 只需要考虑我们挑出来的是哪个点，以及 i 棵子树挑出来的那个点是否使它的胜负性发生改变。

这样我们就能得到转移的系数。这个系数是不随 i 而改变的。时间复杂度为 $O(D)$。

CodeForces 1402C

接下来 DP。设 $f_{i,0/1}$ 表示将 i 棵树连到一起，最后挑出来一个必败点/必胜点的方案数。

由 f_i 转移到 f_{i+1} 只需要考虑我们挑出来的是哪个点，以及 i 棵子树挑出来的那个点是否使它的胜负性发生改变。

这样我们就能得到转移的系数。这个系数是不随 i 而改变的。时间复杂度为 $O(D)$。

最后再上一个 2×2 的矩阵乘法优化。

总时间复杂度为 $O(n + 2^3 \log D)$。