P3349 [ZJOI2016] 小星星

考虑把 p 是排列的条件看做：每个值必须在 p 中出现至少一次。

考虑容斥，强制一些值不会出现，剩下的任意。接下来方案数怎么计算呢？

可以树形 dp ，令 $dp_{u,i}$ 是考虑了 u 子树内合法，且 $p_u = i$ 的方案数。转移是容易的，若 (i,j) 在图上有边，则可以从 $dp_{u,i}dp_{v,j}$ 转移到 $dp'_{u,i}$ 。

总复杂度 $O(2^n n^3)$ 。

AGC032D Rotation Sort

首先观察到对一个数操作两次一定不优：不如让它直接到终点。

先思考 $A = B$ 时怎么做：相当于一个数可以移动到序列的任意位置，转化成：让不移动的数尽量多，这个就是求 LIS。

再来思考 $A \neq B$ 时怎么做。先在排列开头添加 0 ，末尾添加 $n+1$ 并强制要求这两个数不被操作。接下来考虑如果已经确定了不移动的位置，那剩下的数该如何操作。对于数 x 我们设其往左往右碰到的第一个不动的数分别是 l, r 。首先如果 $l < x < r$ 你发现让 x 也不操作肯定更优；否则 $x < l$ 时 x 一定往左移动，$x > r$ 时 x 一定往右移动。容易发现这样也是一定能构造出方案的。转移是容易的，复杂度 $O(n^2)$ 。

CF1523F Favorite Game

首先有一个容易想到的状态是：$f_{S,i,j}$ 表示当前完成了 i 个任务并且人在点 j ，激活的传送门集合为 S ，所需的最小时间。这里 j 可能是某个任务点，也可能是某个传送门。

考虑优化：

如果 j 是传送门，那我们根本不需要记下 j 。只需记 $g_{S,i}$ 为：激活传送门集合为 S ，且当前在 S 内任意一个传送门，完成 i 个任务的最小时间。

如果 j 是任务点，那时间一定是固定的。只需记 $h_{S,j}$ 为：传送门集合为 S ，当前在任务点 j 能完成的最多任务个数。

转移直接枚举下一步要干什么就好了，这里需要预处理 $T_{S,i}$ 为：S 内的传送门到点 i 的最短时间。复杂度 $O(2^n(n+m)^2)$ 。

qoj7504 HearthStone

把 a 从小到大排序。设 b 是最终的数组，容易证明 b 一定是单调不降的；根据题目的条件又有 $b_{i+1} \leq b_i + 1$ ，其中不妨设 $b_0 = 0$ ；代价是 $|a_i - b_i|$ 的和。

这样容易得到 $O(n^2)$ 的做法，就是 $dp_{i,j}$ 表示 $b_i = j$ 的最小代价。考虑怎么优化。

slope trick。

观察 $dp_i(j)$ ，这是一个下凸的分段函数。利用数据结构维护函数的折点。

qoj8528 Chords

初步的想法是设 $f_{i,j}$ 为只考虑 $i \le l \le r \le j$ 的线，最多能选出多少条。转移是容易的：

如果不取端点 j 就是 $f_{i,j-1}$ 更新；

如果取了的话，设 j 的连线的另一个端点为 x_j ，则用 $f_{i,x_j-1} + f_{x_j+1,j-1} + 1$ 来更新 $f_{i,j}$ 。

这是 $O(n^2)$ 的，考虑怎么利用随机的性质，猜测此时答案一定是不大的，事实上可以证明是 $\sqrt{n}$ 级别。

考虑交换 dp 的值域和下标。注意到 j 固定时 i 越小 f 越大，于是考虑设 $g_{j,k}$ 为最大的 l 满足 $f_{l,j} \ge k$ ，发现 $g_{j,k}$ 要么从 $g_{j-1,k}$ 转移而来，要么从 $g_{x_j-1,k-1-f_{x_j+1,j-1}}$ 转移而来。

CF1616G Just Add an Edge

先想想怎么 check $y \to x$ 是否合法，不妨在两侧加两个点 0 和 $n+1$ ，那首先看 0 能否只通过 $i \to i+1$ 走到 $x-1$ ，$y+1$ 能否走到 $n+1$ 。

再看能否取出两条不交路径，分别是 $x-1$ 到 y 及 x 到 $y+1$ ，且覆盖了中间所有点。那这个 dp 一下就好了，记 $dp_{i,j}$ 表示 i 在第 j 条路径，$i+1$ 在另一条是否可行，转移是容易的。

考虑优化，令 L_u, R_u 是只看 $i \to i+1$ 时能到 u 的最左点/u 能到的最右点，于是我们有 $x \le R_0$ ，$L_n - 1 \le y$ 。可以发现 R_0 和 $R_0 + 1$ 所在的路径一定是不相同的，于是考虑以 $(R_0, R_0 + 1)$ 为中心对两头分别做上面的 dp ，然后合并一下就好了。

CF1781F Bracket Insertion

把 (看成 1 ，) 看成 -1 ，然后观察括号序列的前缀和数组。

发现一开始只有一个数 0 ，然后插入 () 等同于把 x 变成 x x+1 x；)(等同于 x x-1 x。合法的条件就是不能出现 < 0 的数。

设 $dp_{x,i}$ 是对只有 x 一个数的序列操作 i 次的结果，答案即为 $dp_{0,n}$ 。

一次操作，以变成 $x, x+1, x$ 的情况为例，可以发现这三个数此时已经互相不影响了，我们可以把它们分成三个序列来看。于是枚举对第一个、第二个序列操作了多少次，有式子：

$dp_{x,n} \leftarrow p(n-1)!\sum\limits_i\sum\limits_j \frac{dp_{x,i}}{i!}\frac{dp_{x+1,j}}{j!}\frac{dp_{x,n-1-i-j}}{(n-1-i-j)!}$ 。

直接算是 $O(n^4)$ 的，思考怎么优化：发现可以分两步计算。为了方便令 $f_{x,n}=\frac{dp_{x,n}}{n!}$ 。

先求出 $g_{x,n}=\sum\limits_i f_{x,i}f_{x+1,n-i}$ 。于是上面的式子可以写成 $dp_{x,n}\leftarrow p(n-1)!\sum\limits_i g_{x,i}f_{x,n-1-i}$ 。

$x,x-1,x$ 的情况同理。

复杂度 $O(n^3)$ 。

P9824 Fountains

考虑把所有区间按 C 从大到小排序，按照这个顺序依次考虑它们会不会被选取，同时更新总权值。这里我们说一个区间 L,R 的权值就是满足 $L\le l\le r\le R$ 且 (l,r) 被选取者中 $C(l,r)$ 最大值。

你发现此时我们只关心：第一，选取了多少个区间；第二，哪些 L,R 的权值已经固定：当我们确定 l_i,r_i 被选取后，所有满足 $L\le l_i\le r_i\le R$ 且还未确定权值的 (L,R) ，其权值就会确定。于是设 $dp_{i,k,S}$ 是考虑了前 i 个区间，当前选了 k 个区间，已知权值者状态为 S 的方案数。然后发现，如果我们令 a_L 是最小的 R 满足 (L,R) 权值已知，那可以发现两件容易说明的事情：1. a 单调不降。2. 若 (L,i) 权值已知，$(L,i+1)$ 权值一定也是已知的 。于是 S 可以看成是 $(0,0)$ 到 (n,n) 的一条路径上方的格子形成的集合。所以可能的 S 只有 $\binom{2n}{n}$ 种。（俗称：轮廓线）

预处理出 状态 S 加入 (l,r) 后变成了哪个状态。于是复杂度 $O(n^4\binom{2n}{n})$ 。

可以发现 C 是区间和这档事，对做法没有影响。

qoj8049 Equal Sums

考虑设 $f_{i,j,k}$ 是有多少组方案使得 $\sum\limits_{p=1}^{i}x_p-\sum\limits_{p=1}^{j}y_p=k$ ，最终答案就是 $f_{n,m,0}$ 。

不妨认为 n,m 及值域是同阶的。那首先至少可以发现 $|k|$ 是在 n^2 以内的。但这样状态数是 $O(n^4)$ 的，不能通过。

思考一下 f 的计算方式，发现它既可以从 $f_{i-1,j}$ 推过来，也可以从 $f_{i,j-1}$ 推过来。而我们只需要算 $f_{n,m,0}$ ，那能不能存在一种转移方式使得我们全程都不需要 $|k|$ 较大的状态呢?

考虑这样一个思路：对于 $k>0$ ，用 $f_{i-1,j,k-x}$ 来更新 $f_{i,j,k}$ $(lx_i\le x\le rx_i)$ ；否则用 $f_{i,j-1,k+x}$ 更新 $f_{i,j,k}$ $(ly_i\le x\le ry_i)$ 。

这样处理之后，可以发现 k 控制在了 $[-n,n]$ 间。枚举 x 可以用前缀和优化掉，复杂度 $O(n^3)$ 。这个题的技巧被人称为：拔河背包。

关路灯，但是 O(n)

先说 $O(n^2)$ 做法：$dp_{l,r,0/1}$ 表示把 $[l,r]$ 都关掉且当前在 l/r ，耗电量的最小值。

考虑经典的 dp 状态所以只能两维是因为：如果我们走到一端，不记另一端关了多少灯，会导致重复计算。

考虑把原来的提前计算的方式重新表达：令 $E(x)$ 表示我们认为的 x 最小可能被关的时间，则每走一步后计算 $E(x)$ 的增量。最后 $E(x)$ 全都固定成真实值了，也就算出答案了。传统的做法就是认为，对于还未关的灯， $E(x)$ 等于当前的时刻 T。

但我们可以这样设计：设 $E(x)=T+|x-X|$ ，其中 X 是你现在所在的位置。

现在再看我们走的过程，比如说从起点左边的 i ，走到起点右边的 j。考察 E 的变化，你发现右端的这些点 E 无论状态如何，都是不变的！改变的只有左端未被推的点，改变量即 $2s_{i-1}(x_j-x_i)$ ，其中 s 是功率的前缀和。

这样就容易设计 dp 了，我们只需设 dp_i 表示走到 i 时的最小势能和，输出 $dp_1+\sum|x_c-x_i|s_i$ 即可，原因是走到 1 时即使右部灯没有关完，我们走过去关了也不影响势能。

发现可以斜率优化，最后复杂度 $O(n)$ 。注意转移顺序，类似于 dij ，先用更小的 dp_i 来转移，具体维护两个指针即可。而且也得维护两个凸包。