路径方案

一个比较熟知的结论: 最短路图是一张拓扑图，因此可以直接按照每个点离S的距离dis_i从小到大依次考虑

考虑对于两条路径依次往路径结尾添加点

令$dp_{i,j}$表示当前路径结尾为i,j的方案数，转移时可以通过保证$dis_i < dis_j \lor (dis_i = dis_j \land i < j)$的顺序来避免重复

每次转移时，考虑把当前节点接上去，枚举当前节点所有可能的前驱，枚举另一个节点然后转移

可能的前驱总数量上限为m，因此复杂度上限为$O(nm)$，实际应该远远不到

排列价值

Part1 暴力dp

一种最暴力的做法是令$dp_{l,r,i}$为区间l,r跟节点为i的权值总大小，

枚举两个子树，乘上子树大小的组合数合并，复杂度为$O(n^4 - n^5)$

Part2 $O(n^2)$做法

设F_i为长度为i的排列权值总和

令 $G_i = \sum_{j=1}^{i} j \cdot (i-1) = \frac{i(i+1)}{2}(i-1)!$，即为长度为$i$的排列所有根可能出现的位置之和

考虑dp求出F_i，假设选择了j作为根，两边剩下$a = j-1, b = i-j$

则贡献分为两部分

1.子树自己的贡献$b! \cdot dp_a + a! \cdot dp_b$

2.当$a > 0, b > 0$时，两个儿子的贡献为$G_b \cdot a! - G_a \cdot b! + j \cdot a! \cdot b!$，其中$j$为下标偏移量

注意最后所有的方案都要乘上组合数$C(a+b,a)$，即合并两边序列的方案数

由此得到一个$O(n^2)$的dp

Part3 $O(n)$

上述dp转移可以形式化的描述为

$$F_i=\sum_{j=1}^{i}(C(i-1,j-1)(j-1)!F_{i-j}+C(i-1,j-1)(i-j)!F_{j-1})+\sum_{j=2}^{i-1}(G_{i-j}(i-1)!+j\cdot(j-1)!(i-j)!-G_{i-1}(i-j)!)C(i-1,i-j)$$

化简为

$$F_i=2\sum_{j=1}^{i}C(i-1,j-1)(j-1)!F_{i-j}+\sum_{j=2}^{i-1}j\cdot(j-1)!(i-j)!C(i-1,i-j)$$

上式两边同除以$(i-1)!$

$$\frac{F_i}{(i-1)!}=2\sum_{j=2}^{i}\frac{F_{j-1}}{(j-1)!}+\sum_{j=2}^{i-1}j$$

$$\frac{F_i}{(i-1)!}=2\sum_{j=2}^{i}\frac{F_{j-1}}{(j-1)!}+\frac{(i-1+2)(i-2)}{2}$$

令$H_i=\frac{F_i}{i!}$，得到H_i的递推公式为

$$H_i=\begin{cases}0 & i\le 2\\ \dfrac{2\sum_{j=1}^{i-1}H_j+\dfrac{(i+1)(i-2)}{2}}{i} & i\ge 3\end{cases}$$

显然已经可以用前缀和优化在$O(n)$时间内求解

尝试过求H_i的通项公式，但是发现答案包含调和级数，可能无法求解

序列操作

$type = 1$

直接暴力状压即可，令$dp_{i,S}$表示当前操作了i次，当前序列情况为S的方案数

暴力枚举翻转的区间l, r，复杂度为n^2

实际上，每次操作序列一定减少一个1，所以dp状态中i这一维完全可以用S中1的个数代替

因此状态数为2^n，总复杂度上限为$O(2^n n^2)$，需要一点常数优化

$type = 2$

发现一个明显的性质：翻转了一段序列$[l, r]$之后，两边的序列$[1, l-1], [r+1, n]$都和$[l+1, r-1]$这一部分相差至少2个0，已经不可能组成交替序列

因此翻转一段序列$[l, r]$后，可以将问题分解为3个互不相干的子问题

令$dp_{i,j}$表示序列包含i个1，还需操作j次的方案数，枚举每个子问题操作的次数，组合数合并

最劣的实现复杂度为$O(n^3k^3)$
将枚举三个子问题分成两步转移，额外记录$f_{i,j}$为合并了两个子问题之后的答案，然后再枚举一个合并即可

处理两个数组的复杂度均为$O(n^2k^2)$，这里不卡常

实际上可以用NTT优化到$O(n^2 \log n + n^3)$，但是实际考试情况不允许

对于$k = n + 1$的情况，实际上就是删除了所有的1，最后序列为全0

因此只需要令dp_i为删除整个$i - 1$阶交替序列的方案数，类似的枚举，即可做到$O(n^2)$处理

实际上，打表可以明显发现答案就是$\prod_{i=1}^{n}(2i-1)$，带入该式可以在$O(n)$时间内求解

对于$2n - 1 \geq 998244353$的情况，显然答案为0

剩下的情况可以分段打表来解决，~~更一般的解法需要类似快速阶乘算法~~

路径差值

朴素的暴力就是枚举起点dfs全树，复杂度为$O(n^2)$

比较优的暴力可以根据权值大小来计算，暴力点分治即可做到$O(n\log nw^?)$

接下来的做法就是容斥了，考虑边权在$[L,R]$范围内的路径数量为$F(L,R)$

容易想到通过计算$\sum F(L,L+k)$来得到差值$\leq k$的答案，但是显然这样的计算会重复

实际上，容斥部分应该是
$F(L,L+k)-F(L+1,L+k)-F(L,L+k-1)+F(L+1,L+k-1)$

形式化地理解这个容斥：

设$\min\max$为L,R的路径条数为$G(L,R)$，显然
$F(L,R)=\sum_{i=L}^{R}\sum_{j=i}^{R}G(L,R)$

那么发现实际上$\sum F(L,L+k)=\sum_{R-L\leq k}G(L,R)\cdot(k-(R-L))$

则
$\sum F(L,L+k)-F(L+1,L+k)-F(L,L+k-1)+F(L+1,L+k-1)$

$=\sum F(L,L+k)-2\sum F(L,L+k-1)+\sum F(L,L+k-2)$

$=\sum_{R-L\leq k}G(L,R)\cdot(k-(R-L))-2\sum_{R-L\leq k-1}G(L,R)\cdot(k-1-(R-L))+$

$\sum_{R-L\leq k-2}G(L,R)\cdot(k-2-(R-L))$

$=\sum G(L,L+k)$

如果暴力枚举区间，然后并查集插入和维护所有在范围内的边权，复杂度为$O(nk\alpha(n))$

发现每次枚举的区间形式非常单一，每条边能贡献的L是一段范围，可以考虑用LCT维护，复杂度为$O(n\log n)$

但是实际上可以用线段树分治+按秩合并查集的回撤操作来完成，复杂度为$O(n\log^2 n)$

没有测试过LCT的做法，不知道效率如何

或许std比较丑，常数比较大吧