国庆集训第一场题解

黑白棋 (chess)

Algorithm1

对于 k 为 0 和 1 的情况：首先对于两种棋子数量相差超过1的可以直接输出 -1，剩下的我们可以考虑两种方案：以B开头或以W开头，分别计算两种情况的最小cost，最后再取一次两者的最小值输出即可。

计算 $cost$ 时，维护一个保存位置不正确的棋子的队列，对于任意cost函数类型都用同一种贪心原则：不改变位置正确的棋子，对于每个位置不正确的棋子，如果队列左边异色棋子，则左边棋子出队列并与之交换，计算cost。如果队列左边为同色，则入队列。这种方法只对cost为1次及以下成立，期望得分40。

Algorithm2

对于 100% 的数据，$k \geq 2$ 时，可以考虑将一次交换拆解成 $|i-j|$ 次距离为1的交换（i,j 为需要交换的棋的位置，把中间所有颜色与 j 相同的棋子依次移动到左边的同色棋子的位置），就等效为了 $k=1$ 的情况，复杂度为 $O(n)$，期望得分 100

魔法大陆 (magic)

前言

考察动态规划及转移状态的条件问题。

Subtask1
对于 p_i 全为 1：

设 $dp_{i,j}$ 表示前 i 个覆盖了 j 个时的最小花费，$a_{i,j}$ 表示覆盖 i 到 j 的最小花费。

枚举右端点 i，枚举覆盖个数 j，再枚举覆盖区间左端点 l。

状态转移：当前区间的最小覆盖代价+当前区间之前的最小选取代价

状态转移方程：$dp_{i,j}=\min(dp_{i,j},dp_{l-1,j-(i-l+1)}+a_{l,i})$。

Subtask2
对于任意p_i，注意到如果一个点被覆盖三次则一定不优（只需要取经过这个点，且最靠左和最靠右的两个区间即可）。

则状态转移过程中只需考虑p_i为1和p_i不为1的情况。

在Subtask1的基础上，$[p_l,p_i]$全不为1,表示区间左端点为$l(l\leq i)$的区间,在l点之后依然可以作为$[l+k,i]$覆盖区间。$(1\leq k\leq i-l)$

若p_l为1，则区间$[l,i]$只能贡献$dp_{i,j}=\min(dp_{i,j},dp_{l-1,j-(i-l+1)}+a_{l,i})$一次转移（只有这一次转移是有价值的）。

因此，只需记录当前区间覆盖的最小代价$tcost$=min($tocst$,$a_{l,i}$)，进行状态转移即可。若遇到$p_l=1$,则左端点在l之前（包含l）的区间在之后无法再产生贡献，则修改$tcost$的值为无穷大。

时间复杂度$O(n^3)$

树上修路 (treeroad)

前言

考察对树上路径、dfs 序的理解，并查集、数据结构的运用。

Algorithm1

结论：i 的父亲在 $[1,i-1]$ 中等概率随机，这样的树的深度是 $O(\log n)$ 级别的，树上路径的长度也是 $\log$ 级别的

那么直接用并查集维护每个连通块，集合代表选为连通块中深度最小的节点。

暴力模拟该过程即可，可以获得 30 分。

Algorithm2

可以发现，算法一的瓶颈在于查询时可能跨越的连通块数很多。

我们考虑快速维护这个过程：每个点的权值可以为 $0/1$，分别表示它是否是一个连通块中深度最小的点。那么一条路径的答案即为路径上点权之和，特殊处理 LCA 后得到的结果。于是可以直接使用树转为链的技巧，得到如下算法：

1. 每个节点权值初始化为 1，树链剖分。
2. 合并：对路径 (u,v) 上的点改为 0，LCA 处的权值保持不变。
3. 求和：即树链求和。

复杂度 $O(n\log^2 n)$ ，期望获得 $60\sim 100$ 分

Algorithm3

考虑用其他数据结构技巧加速上述算法。容易发现求和具有可减性，因此问题可以转化为查询点到根的和，容斥掉 LCA 到根的和。

一个子树内的 DFS 序是连续的，点到根的和可以进行树上前缀和，对每个 1 点做子树 $+1$，然后单点查询。

区间加和单点查询可以再进行一次差分转换为单点修改和前缀和查询，使用树状数组即可方便地维护。

复杂度 $O(n\log n)$，期望获得 100 分

破损的棋盘 (chessboard)

Algorithm1

可以发现每行每列至少摆放一个车，那么直接全排列枚举即可。

期望得分 20 。

Algorithm2

对于 $m=0$ 的情况，考虑容斥，答案等于不考虑对角线的方案数，减去一条对角线为空的方案数，再加上两条对角线均为空的方案数。

不考虑对角线的方案数，答案为 $n!$，后两种情况可以容斥/打表/找规律得到答案，不是本题重点。结合前面的算法期望得分 40。

Algorithm3

依旧是考虑容斥，有 $m \leq 10$ 个位置不能放置，考虑容斥强制打破其中的 k 个限制，答案带上容斥系数 $(-1)^k$。

答案的构成仍然是不考虑对角线的方案数，减去一条对角线为空的方案数，再加上两条对角线均为空的方案数。

- 不考虑对角线的方案数：假设行列各少了 k 个，方案数为 $(n-k)!$
- 一条对角线为空：考虑再次容斥，先求出对角线上还有多少个格子可以用，枚举在这条对角线上强制选了 t 个格子，剩下的行列同不考虑对角线的情况，带容斥系数 $(-1)^t$。
- 两条对角线均为空：依旧是容斥，但钦定了 t 个格子的答案计算方法有所不同。考虑 DP，从外向内按环来做。对于第 i 环，考虑 (i,i)，$(n-i+1,i)$，$(i,n-i+1)$，$(n-i+1,n-i+1)$ 这四个格子，它们显然是独立的，每一环可以选择 $0,1,2$ 个格子（N 为奇数时，中心点所在的环只算 $0,1$ 个格子）。那么就转化为了一个背包问题，$F[i][j]$ 表示前 i 个环选了 j 个格子时的方案数是多少，如此可以得到钦定任意个格子时的答案。

时间复杂度 $O(2^m \times n^2)$，背包可以利用生成函数 + FFT 优化，但范围较小意义不大。