T1

考虑从前往后依次贪心，假设我们已经确定了 $a_{1\dots i}$。可以发现 $a_{i+1\dots n}$ 的所有填法贡献的 $a_i \neq a_{i+1}$ 的个数一定是一段区间，且这段区间只与 a_i 的值有关。因此我们记录 $L_{i,j}, R_{i,j}$ 表示 $a_i = j$ 时区间为 $[L_{i,j}, R_{i,j}]$。容易做到 $O(n)$。

T2

首先进行容斥。我们钦定树上 k 个点满足条件，这个方案就贡献 $(m-1)^k$ 的权值。假设这个方案中实际上共有 k' 个点满足条件，那么这些点的每个子集都会恰好被计算一次贡献，总贡献恰好为 $\sum_{k=0}^{k'} \binom{k'}{k}(m-1)^k = m^{k'}$。

于是我们可以进行 dp：设 $dp_{u,i}$ 表示只考虑 u 的子树，其中被钦定满足条件的点中编号最小的一个为 i 的方案数。

转移只需要考虑两个排列的合并方式即可，时间复杂度与树形背包相同，为 $O(n^2)$。

T3

不妨设 $c_1 > c_2$。显然有 $O(n2^{c_1})$ 的状压 dp 算法。

另一方面，我们可以将所有点 i 按照 $i \bmod c_1$ 分类。显然只有同一类点之间存在差为 c_1 的。

固定一个 $r \in [0, c_1)$，我们拿出所有 $\{(r + kc_2) \bmod c_1\}$ 对应的类，并将它们按照 k 从小到大排序。

显然排序后只有相邻两类以及首尾两类点之间可能存在边。

如果不考虑首尾两类点之间的边，那么我们可以直接进行状压 dp，状态中记录当前这一类中的每个点是否被匹配。依次转移每一类点即可，转移时需要用一些类似于轮廓线 dp 的技巧。

否则我们需要断环为链。直接枚举首尾两类点之间的每条边是否在匹配中，然后进行上述 dp 即可。

这种算法的时间复杂度为 $O(n2^{\frac{2n}{c_1}})$。

因此我们按照 $c_1 \le \sqrt{2n}$ 分治，可得时间复杂度为 $O(n2^{\sqrt{2n}})$ 的算法。

T4

对于一组合法的 u, v, w，一定存在恰好一个点 x 满足 $dis(x,u) = dis(x,v) = dis(x,w)$。

因此我们可以考虑枚举 u, d，假设已经求出 u 每个方向中分别有多少个与它距离为 d 的点，我们可以快速计算出从其中三个不同的方向各选出一个点的方案数。

设 len_u 表示 u 往下的最长链。再设 $len'_u =$ 可重集 $\{len_v + 1 : v \in son_u\}$ 中的非严格次大值。

显然 $d > len'_u$ 时方案数为 0。而根据类似于长链剖分的分析，我们知道 $\sum len'_u$ 是 $O(n)$ 的。因此我们只需要暴力地求出每个距离对应的点数。

u 到其子树内的部分可以用长链剖分解决。

对于剩下的部分，一种暴力的方法是用点分治直接对于每个 u, d 求出距离 u 为 d 的点数，然后减去子树内的部分即可。时间复杂度 $O(n \log n)$。

但实际上这一部分我们也可以用类似于长链剖分的方法。对于每个 $d \le len_u$ 维护 u 子树外有多少个与它距离为 d 的点。这个东西是可以往下转移的，时间复杂度为 $O(n)$，复杂度分析与长链剖分类似。

因此总时间复杂度为 $O(n)$。