NOIP2020 信心赛

Day1 solution

染色大战

本题是一道简单的签到题，题目背景为真实发生的事。

算法 1（$n,m,q \leq 100$）

对于每一次染色，都暴力更改连通块中的颜色。每一次重新计算连通块大小。只要不死循环，效率都是对的。

算法 2（$n \leq 2$）

假设连通块最远到 r，那么对于 $x \leq r$ 的所有还不在连通块的点一定与连通块相邻。只需要记一下这些点每种颜色有多少个即可。

~~感觉和正解没差多少，这个子任务是不是只是防止写挂啊。~~

算法 3（全部子任务）

我们只需要维护与当前连通块相邻的所有点即可。

我们维护若干个桶，把这些点丢到对应的桶中。那么每一次只需要访问对应的桶即可。

由于一个元素只可能被加入桶一次。那么总的时间复杂度就是 $O(nm+q)$ 的。

树的重心

本题的定位是一道不太难的 dp 计数题，不过缺点是子任务对正解没有什么提示作用。

算法 1（$n \leq 20$）

因为是 NOIP 模拟，所以就白送了 12 分。怎么搜索都过得去吧。

算法 2（一些乱搞）

对于一条链的情况，连通块一定是一个区间，重心一定是区间中点，直接枚举即可。
对于菊花图的情况。如果 $k=1$，答案就是每一个点。
如果 $k=2$，那么直接考虑每一条边的贡献。
如果 $k \geq 3$，那么重心一定是 1 号点了，只需要计算出有多少个连通块即可。
对于所有数据中，如果 $k=2$，那么直接枚举每一条边的贡献。
如果 $k=3$，可以枚举重心，方案为与这个点相连的点选 2 个。
如果 $k=4$，分讨一下，对于两种情况分别计数。
如果 $k=5$，也可以分讨解决。

算法 3（$k \leq 100$）

先考虑一个 $O(n^2k)$ 的做法。我们可以枚举每一个点当重心，然后以他为根做 dp。
状态可以设计 $F[i][j]$ 表示第 i 个点，包含它和它子树中共 j 个节点的连通块数。

对于根来说，只需要它的每一个子树都不超过 $\left\lfloor \frac{k}{2} \right\rfloor$ 即可，所以可以再单独 dp 一下。

事实上，如果我们采用换根 dp 的思想，可以把上述做法优化到 $O(nk^2)$，具体方式就不在这里说了。

如果使用 FFT 优化还可以做到 $O(nk \cdot log\, k)$，但这样并不能获得更多分。

算法 4（全部子任务）

事实上多记几个 dp 即可。
先保留 $F[i][j]$ 表示第 i 个点，包含它和它子树中共 j 个节点的连通块数。
我们考虑一个点是否可以成为重心，那么只需要在这个点的所有子树的大小都不超过 $\left\lfloor \frac{k}{2} \right\rfloor$，并且自己的大小大于等于 $\left\lfloor \frac{k}{2} \right\rfloor$。

要实现这个我们可以再原来的 dp 上加点限制。$F[i][j][k], k \in \{0,1\}$ 表示是否存在一个子树的大小超过 $\left\lfloor \frac{k}{2} \right\rfloor$。这样效率是不变的。

然后在记 $G[i][j]$ 为第 i 个点，包含它和它子树中共 j 个节点中所有可能的重心的编号之和。

两个 dp 相互转移即可。
最后时间复杂度为 $O(nk)$ 的。

随机的排列

本道题属于不太难的第三题。不过子任务中有很多是我都不会写的。

算法 1（$n \leq 20$）

这部分还是比较容易的。可以直接暴力建出整张图，然后在搜索每一个点是否选择即可。

算法 2（$n \leq 50$）

因为是随机的数据，所以搜索的时候加点剪枝可以过？
出题人没有实现这个算法。（验题人跟我说可以网络流？）
同样的，对于 $n \leq 100$ 的做法出题人同样不会做。

算法三 3（大概可以拿 $38 \sim 51$ 分）

首先因为排列是随机的，所以从每一个位置开始，把前面的数建立一个单调递减的队列。这个队列的长度大概只有十几。

在单调队列上，不难发现前一个数一定向后一个数连边。

此时如果新加入一个数，它可以弹出单调队列中的若干数。而这个数一定向这些弹出的数连边。

以上两种连边方式可以表示出所有的边。

那么我们可以记 $F[i][S]$ 表示前 i 个点，当前单调队列里的选择情况为 S。此时我们可以分类第 $i+1$ 个点选不选。

假设单调队列中基本保持 k 个元素，那么时间复杂度为 $O(nq \cdot 2^k)$。

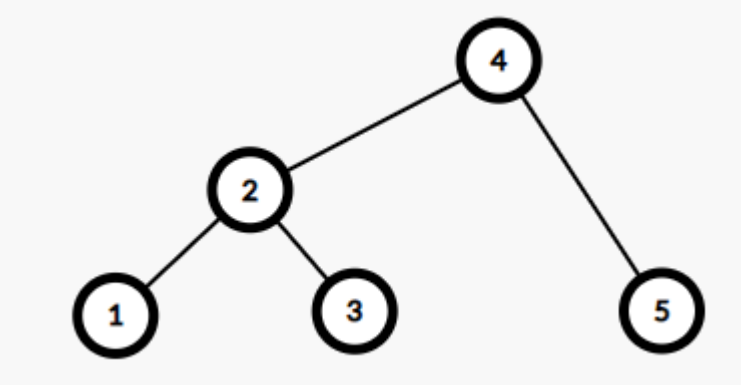
此方法可以通过 $1 \sim 5$ 和 7 号子任务，基本上不能通过子任务 6。对于子任务 8，如果常数优秀也可以通过。

算法 4（子任务 $1 \sim 8$）

我们可以先对序列建立一棵笛卡尔树。

建法就是每一次找到区间中的最大值，然后把这个点作为根。这样这个点的左子树就是以这个点切开的左区间，而右子树就是以这个点切开的右区间。

以样例序列 $\{2,4,1,5,3\}$ 为例，它的笛卡尔树如下：



这棵树有一些性质。

比如以 4 号点为例，那么 4 号点会与 2,3,5 三个点连边。

可以发现，一个点连边的点的集合为这个点左子树的最右侧的一条链和这个点右子树最靠左的一条链。

这时候就可以 dp 了。

我们可以记 $F[i][a][b][c], a,b,c \in \{0,1\}$。其中 i 表示以 i 为节点的子树， a 表示点 i 是否被选择， b 表示点 i 左子树的左链是否全部被覆盖， c 表示点 i 右子树的右链是否全部被覆盖，且除了以上三种点外的所有点都已经被覆盖的最小选点数。

建立笛卡尔树是 $O(n)$ 的，dp 是 $O(n \cdot c)$ 的。而常数大概只有 30 左右，所有总时间复杂度为 $O(nqc)$， c 为常数。

算法 5（全部测试点）

算法 4 的瓶颈在于每一次都要重新建立笛卡尔树和重新 dp，这样很耗时间。

注意到排列和操作都是随机的，并且操作是交换相邻的两个数，所以就有些性质。

性质 1：因为数据是随机的，所以笛卡尔树的树高大概是 $log\, n$ 级别的。

证明（不知道是不是对的）：

考虑每一个区间取出根后新分成的两个区间，我们都往大的区间走。一直到区间长度为 1 时停止。

很显然每一次大的区间长度的期望为 $\frac{3}{4}l$。那么这样大概会走 $log_{\frac{4}{3}} n$ 步。

性质 2：笛卡尔树的形态大致不变。

证明：

我们设交换的位置为 x，不妨设 $p_x < p_{x+1}$。

那么 x 一定在 $x+1$ 的左子树的右链上，并且 x 没有右儿子。

我们先考虑在 $x+1$ 的左子树上删去 x 这个节点。那么只需要把 x 的左子树接到 x 的父亲节点的右儿子位置即可。

我们再考虑在 $x+1$ 的右子树中加入 x。不难发现， x 一定在 $x+1$ 右子树的左链上。

那么只需要在 $x+1$ 的右子树上一直往左跳，直到找到合适位置即可。

这样构造下来，只有原 x 所在位置到根，新 x 所在位置到根， $x+1$ 所在位置到根的所有点的 dp 值会改变。

由性质 1 可知，改变的数量大概为 $log\, n$，因此可以暴力修改这些 dp 值。

最后效率大概可以归为 $O(n \cdot log^2 n)$。