扫码关注正睿教育

1 树

首先我们做一些简单的推理，这棵树的中序遍历是 1 到 n。对于一个子树，在中序遍历中，这些点一定是连续的，所以说编号一定是一个连续的区间。
接着假设我们拿到了一个拿到了一个区间，我们怎么知道这个区间的根呢？容易发现这个根一定是层先遍历中的第一个。
所以我们得到了一个做法，对于一个区间，我们先找到它的根也就是层先遍历中的最小值，然后把这个区间分成左右两部分，然后接着递归建出这棵树，时间复杂度 O(n^2)。建完之后直接先序遍历即可。这个做法获得 60 分。

对于满分算法，可以使用数据结构在 O(log n)的时间复杂度内算出最小值，那么时间复杂度为 O(n log n)。也可以定义一个数组 p，记 p[i]为 i 这个点在层先遍历中的位置，上面的过程相当于求 p 的笛卡尔树，可以使用单调栈解决。时间复杂度 O(n)。可以获得 100 分。

笛卡尔树可以百度一下。

2 石头剪刀布

考虑一个最简单的问题，如果给出一个序列，怎么求它的 w 值呢？
一个最直接的做法是记 dp[i]表示 i 这个位置结尾最长的长度，转移为 max(dp[j]+1)其中 j 能够战胜 i，然后这个 w 值就是 dp 值的最小值，总的时间复杂度为 O(n^2)，结合枚举能获得 20 分。

我们换个想法，记一下做到位置 i，结尾为剪刀石头布的的最长序列是多长，分别记作 p0, p1, p2。那么比如 i+1 这个位置是剪刀，我们就用 max(p1+1, p0)去更新一下 p0 这个值，也就是说如果选了这个剪刀，那么上一个位置必须是石头，所以最长序列的长度为 p1+1，否则是 p0。最后就是 max(p0, p1, p2)，时间复杂度 O(n)，结合枚举还是只能获得 20 分。

于是我们可以设计一个四维的状态，dp[i][p0][p1][p2]表示做到第 i 个位置，结尾为剪刀，石头，布的最长序列分别为 p0, p1, p2 的方案数。如果 i+1 这个位置我们选了剪刀，那么就转移到 dp[i+1][max(p0,p1+1)][p1][p2]这个地方。最后我们把 dp[n][p1][p2][p3]加到答案 answer[max(p1,p2,p3)]里面，表示有这么多个序列的 w 值为 max(p1,p2,p3)。时间复杂度为 O(n^4)，可以获得 40 分。

通过观察我们会发现这个 dp 的有用状态非常少，具体的我们有|p0-p1|<=2, |p0-p2|<=2，对于一个长度为 p0 的以剪刀结尾的序列，那么我们把最后一项去掉就能得到一个以石头结尾的序列长度，所以有 p1>=p0-1，同理有 p2>=p1-1, p0>=p2-1，结合这三式就能得到这三项的差不会超过 2。

所以我们可以把上面的状态改写成 dp[i][p0][d1][d2]其中 d1, d2 为-2 到 2 之间的数字，表示 p1-p0 和 p2-p0 的值，然后用上面说的方法转移和统计答案即可。

时间复杂度为 O(n^2)，其中可能有一个较大的常数，可以获得 100 分。如果你实现常数很大或者在 40 分算法的基础上直接使用 map 优化或者你只发现了可以压缩一维，那么可以获得 70 分。

3 剪刀石头布

首先我们考虑一下以知 l, r, wa, wb 之后怎么求期望胜局。这里记 wa, wb 就是出石头和剪刀的概率，记 wc=1-wa-wb，就是出布的概率。由于期望线性性，期望胜局等于和每个人的期望胜局的和，也就是和每个人玩胜率的和。
假设一个人出剪刀石头布的概率分别为a_i, b_i, c_i，那么期望也就是$wa * \sum_{i=l}^{r} c_i + wb * \sum_{i=l}^{r} a_i + wc * \sum_{i=l}^{r} b_i$。
又因为 wa + wb + wc = 1，且我们要求期望最大，所以说我们会把这些权值加到最大的项上。
比如 max 3*wa + 2*wb +1 * wc，那么肯定取 wa=1, wb=wc=0。
如果 max 3*wa + 3*wb +1 * wc，那么就取 wc=0, wa,wb 取任意非负实数并且加起来等于 1 即可。
如果 wa, wb, wc 都不等于 0，那么要求 a, b, c 的和相同，否则加到把非零的部分加到最大的里面一定更优。
如果 wa, wb 都不等于 0，那么要求 c 和 a 的和相同且大于等于 b 的和。
注意在这个题里面 wa, wb 都不等于 0，所以只有前面两种情况。

通过前面的分析，我们就能很快地得到 30 分算法，枚举一个区间，然后求出 a, b, c 的和，然后进行一下简单的判断即可，时间复杂度 O(n^2m)
然后可以发现对于每个询问，只有 wa+wb=1 和不等于 1 两种不同的情况和 wa, wb, wc 具体的值没有关系，理由见上面的分析。
所以说我们只要统计出 a, b, c 的和都相同的和 a, c 的和相同且大于 b 这两种情况即可的区间个数即可。
于是我们可以预处理这样的区间然后根据 wa+wb 是否等于 1, O(1)回答每组询问，时间复杂度为 O(n^2+m)，可以获得 60 分。

接下来考虑怎么加速预处理，这里只讨论第二种情况，第一种情况同理。
记 Sa, Sb, Sc 分别为 a, b, c 的前缀和，那么[l, r]这段区间的和为 Sa[r]-Sa[l-1], Sb[r]-Sb[l-1], Sc[r]-Sc[l-1]。条件为 Sa[r]-Sa[l-1]=Sc[r]-Sc[l-1], Sa[r]-Sa[l-1]>= Sb[r]-Sb[l-1]。
做一些简单的移项条件变为 Sa[r]-Sc[r]=Sa[l-1]-Sc[l-1], Sa[r]-Sb[r]>= Sa[l-1]-Sb[l-1]。
我们记 A[i]=Sa[i]-Sc[i], B[i]=Sa[i]-Sb[i]，那么变成了求二元组(i, j)的个数，满足 0<=i<j<=n, A[i]=A[j], B[i]<=B[j]。
于是我们把所有的 A 值相同的数字拿出来，形成一个组，求这些数里面 B 的顺序对(和逆序对一样)就可以了，这个可以用归并排序或者离散化+树状数组解决。
对于第一种情况，条件变成 A[i]=A[j], B[i]=B[j]，更加简单。
总的时间复杂度 O(n log n+m)，可以获得 100 分。