CodeChef 题目选讲

THU CST 胡泽聪
Email: huzecong@163.com

关于题目

为照顾各个层次的选手，讲义中的题目分为了四类。

- Warmup。大家都能想出来的热身题。
- Average。比热身题稍难的一般题。
- Intermediate。有一定难度的进阶题。
- Hard。具有挑战性的难题。

每个部分内的题目按照个人主观感受的难度排序。

LEMOUSE

有一个 $n \times m$的格子，有一头大象，初始时在 $(1, 1)$，要移动到 (n, m)，每次只能向右或者向下走。有些格子中有老鼠，如果大象所在的格子和某个有老鼠的格子的曼哈顿距离 ≤ 1，大象就会被那只老鼠吓到。求一条移动路径，使得吓到过大象的老鼠数量最少。

$n, m \leq 100$。

LEMOUSE - Solution

如果多次经过一只老鼠旁边也会被吓多次，直接 DP 就好了。而这里只会被吓一次，我们需要往状态中加点东西。

LEMOUSE - Solution

如果多次经过一只老鼠旁边也会被吓多次，直接 DP 就好了。而这里只会被吓一次，我们需要往状态中加点东西。

注意到，一只老鼠最多能影响我们三步，因此只要记录上两步的走法即可。转移时分情况讨论。

MLCHEF

有一棵 n 个点的有根树，1号节点为根，每个点有一个正点权。有 m 次操作，操作有下面两种：

- 将一棵子树内的所有点的点权减去某个正整数。
- 询问一棵子树内有多少个点的点权为正数。

$n, m \leq 100000$。

MLCHEF - Solution

既然是子树操作，先求出 DFS 序。问题就变成了区间操作的序列问题。

MLCHEF - Solution

既然是子树操作，先求出 DFS 序。问题就变成了区间操作的序列问题。

分块套平衡树显然可做，但是复杂度略高（还不能调set）。有更简单的做法吗？

MLCHEF - Solution

既然是子树操作，先求出 DFS 序。问题就变成了区间操作的序列问题。

分块套平衡树显然可做，但是复杂度略高（还不能调set）。有更简单的做法吗？

由于每次减去的都是正整数，一个数变成非正数后就不会再变回正数。

线段树维护区间内有多少个正数，同时维护区间最小值。当最小值 ≤ 0时，就把这个数改成正无穷，同时更新区间正数个数。

复杂度 $O(n \log n)$。

METEORAK

一个 $n \times m$的网格中，有 k个坏点。有 q次询问，每次给定 L_i和 R_i，询问在网格的第 L_i到第 R_i行中最大的内部没有坏点的矩形的面积。

$n, m \leq 1000$，$k \leq n \times m$，$q \leq 10^6$。

METEORAK - Solution

令 $f(L,R)$代表询问 (L,R)的答案，令 $g(L,R)$代表矩形上下边界为 L和 R的最大矩形宽度，显然有

$$f(L,R) = \max(g(L,R) \times (R-L+1), f(L+1,R), f(L,R-1))$$

问题就在于怎么求出 $g(L,R)$。

METEORAK - Solution

一个暴力做法是，把 $L \sim R$行压缩成一行，然后变成一维的问题。利用单列的前缀和，可以做到单次操作 $O(n)$。

METEORAK - Solution

一个暴力做法是，把 $L \sim R$行压缩成一行，然后变成一维的问题。利用单列的前缀和，可以做到单次操作 $O(n)$。

如何优化？直接压位！单次就变成了 $O(n/32)$。已经可以通过了。

METEORAK - Solution

一个暴力做法是，把 $L \sim R$行压缩成一行，然后变成一维的问题。利用单列的前缀和，可以做到单次操作 $O(n)$。

如何优化？直接压位！单次就变成了 $O(n/32)$。已经可以通过了。但是还能更优吗？

METEORAK - Solution

一个暴力做法是，把 $L \sim R$行压缩成一行，然后变成一维的问题。利用单列的前缀和，可以做到单次操作 $O(n)$。

如何优化？直接压位！单次就变成了 $O(n/32)$。已经可以通过了。但是还能更优吗？

考虑求最大全 0子矩阵的传统 $O(n^2)$做法。枚举下边界之后，求出每列向上拓展的最大高度。

用单调队列可以求出，以一列的最大高度向左向右拓展，可以拓出的最大长度。设此时的高度为 h，可以用这个长度来更新 $g(R-h+1 \sim R, R)$。

用树状数组之类的数据结构可以做到 $O(n^2 \log n)$，直接打标记之后扫一遍可以做到 $O(n^2)$。

SHGAME

一张 $n \times m$的网格，其中某个格子 (x, y)被涂黑了。两个玩家轮流操作，每次沿一条水平或竖直的格线将网格切成两部分，并舍弃不含黑格子的一部分。无法操作者败。

双方均以最优策略行动。问有多少种 (x, y)的选择使得先手必胜。

$n, m \leq 10^6$，100 组测试数据。

SHGAME - Solution

我们固定住 (x, y)，然后考虑切一刀造成的影响。

SHGAME - Solution

我们固定住 (x, y)，然后考虑切一刀造成的影响。

可以发现，如果我们切 (x, y)上方，则显然其上方的行数会减少，但其左右列数及下方行数不变。如果我们分别记上下左右的行（列）数为 u, d, l, r，则一次操作相当于将一个变量减去一个不大于其自身的数，其他变量不变。

这其实就是一个 4 堆石子的 Nim 游戏。

SHGAME - Solution

我们固定住 (x, y)，然后考虑切一刀造成的影响。

可以发现，如果我们切 (x, y)上方，则显然其上方的行数会减少，但其左右列数及下方行数不变。如果我们分别记上下左右的行（列）数为 u, d, l, r，则一次操作相当于将一个变量减去一个不大于其自身的数，其他变量不变。

这其实就是一个 4 堆石子的 Nim 游戏。

那么一个朴素的做法就是枚举 (x, y)，然后判断 u, d, l, r的异或和是否为 0，若不为零则先手必胜。

SHGAME - Solution

但这样显然太慢了。
一个（显然）的观察是，$u + d = n - 1$，$l + r = m - 1$。

SHGAME - Solution

但这样显然太慢了。
一个（显然）的观察是，$u + d = n - 1$，$l + r = m - 1$。
这意味着 u xor d与 l xor r的值分别是 $O(n)$和 $O(m)$级别的。

SHGAME - Solution

但这样显然太慢了。

一个（显然）的观察是，$u+d=n-1$，$l+r=m-1$。

这意味着 u xor d与 l xor r的值分别是 $O(n)$和 $O(m)$级别的。

我们只需预处理 u xor $d = u$ xor $(n-1-u)$的所有取值，并计算每种取值的个数。接下来枚举 l，只要 u xor $(n-1-u) \neq l$ xor $r = l$ xor $(m-1-l)$，这一选择就合法。故可以用总方案数 $n \times m$减去不合法方案数，即异或值相同的方案数。

复杂度为 $O(n+m)$。

SEABAL

一排有 n 个数，初始均为 0。给定 m 个区间，有 k 次操作，每次会将一个数改成 1，并询问此时有多少个给定区间中所有数均为 1。

$n, m \le 100000$，强制在线。

SEABAL - Solution

将一个数改为 1后，可能会将两个分开的全 1区间合并到了一起。第一步要做的就是找出这个区间。可以用并查集维护向左和向右的位置。

SEABAL - Solution

将一个数改为 1后，可能会将两个分开的全 1区间合并到了一起。第一步要做的就是找出这个区间。可以用并查集维护向左和向右的位置。

记上面找到的左右端点为 L和 R。接下来的问题就是，找出所有在加入前还不全为 1，加入后就全为 1的区间。这些区间一定满足左右端点在 [L, R]中，且并未在之前被删除。我们只需要快速找到这些区间，并删除即可。

SEABAL - Solution

将一个数改为 1后，可能会将两个分开的全 1区间合并到了一起。第一步要做的就是找出这个区间。可以用并查集维护向左和向右的位置。

记上面找到的左右端点为 L和 R。接下来的问题就是，找出所有在加入前还不全为 1，加入后就全为 1的区间。这些区间一定满足左右端点在 $[L, R]$中，且并未在之前被删除。我们只需要快速找到这些区间，并删除即可。

这是一个二维查询，很容易想到用线段树套平衡树实现。内层的平衡树可以用set。

复杂度 $O(n \log^2 n)$。

SEABAL - Extras

如果没有强制在线，有更好的做法吗？

欢迎大家讨论。

BNGAME

有一排 n 个格子，从左到右编号为 $1 \sim n$。第 i 个格子中有两个数 a_i 和 b_i。玩家初始时站在所有格子左边（0 号格子），每次移动玩家可以前进 $1 \sim k$ 个格子，并在停下的那个格子落子。当玩家移出所有格子，即到达 n 号格子右边时，游戏结束。

设落子的格子为 $S = \{p_1, \ldots, p_{|S|}\}$，则一盘游戏的得分为

$$\max_{x \in S}(a_x) + \max_{y \in S}(b_y)$$

求最小得分。

$n \leq 500000$，$|a_i|, |b_i| \leq 32000$。

BNGAME - Solution

朴素做法是枚举 $\max(a_x)$，禁止所有 a_i 大于枚举值的格子，并动态规划求出只使用剩下的格子能取到的最小的 $\max(b_y)$。这样的复杂度为 $O(nW)$，其中 $W = \max\{a_i, b_i\}$。

BNGAME - Solution

朴素做法是枚举 $\max(a_x)$，禁止所有 a_i大于枚举值的格子，并动态规划求出只使用剩下的格子能取到的最小的 $\max(b_y)$。这样的复杂度为 $O(nW)$，其中 $W = \max\{a_i, b_i\}$。

记枚举的 $\max(a_x) = K$，并记此时的 $\max(b_y) = f(K)$。显然随着K增加，$f(K)$单调不增。

如果要使 $f(K)$减少，那么我们不能在任意 $b_i = f(K)$的格子中落子，也即，禁止所有这样的格子后仍然可以到达终点。

从小到大枚举 K，每次 K增加，就会有原来被禁止的格子取消禁止。此时我们尝试逐个禁止 $b_i = f(K)$的格子，如果能全部禁止，则说明$f(K)$可以减小，并对减小后的 $f(K)$重新尝试禁止。

BNGAME - Solution

朴素做法是枚举 $\max(a_x)$，禁止所有 a_i大于枚举值的格子，并动态规划求出只使用剩下的格子能取到的最小的 $\max(b_y)$。这样的复杂度为 $O(nW)$，其中 $W = \max\{a_i, b_i\}$。

记枚举的 $\max(a_x) = K$，并记此时的 $\max(b_y) = f(K)$。显然随着 K增加，$f(K)$单调不增。

如果要使 $f(K)$减少，那么我们不能在任意 $b_i = f(K)$的格子中落子，也即，禁止所有这样的格子后仍然可以到达终点。

从小到大枚举 K，每次 K增加，就会有原来被禁止的格子取消禁止。此时我们尝试逐个禁止 $b_i = f(K)$的格子，如果能全部禁止，则说明 $f(K)$可以减小，并对减小后的 $f(K)$重新尝试禁止。

如何快速判断可以到达终点？

BNGAME - Solution

我们只需要知道，需要禁止的点的前一个和后一个未被禁止的点之间，是否能通过一步移动到达。

BNGAME - Solution

我们只需要知道，需要禁止的点的前一个和后一个未被禁止的点之间，是否能通过一步移动到达。

使用平衡树维护位置即可。可以直接用 STL 中的 set。

DECORATE

给定字符串 S，选择 S的 n个回文子串（可以相同）构成一个序列。如果两个序列经过旋转和/或翻转后相同，则认为两个序列同构。如，序列{"ab","a","b"}与序列{"a","ab","b"}同构。

求有多少种本质不同的序列。

$n \leq 600$，$|S| \leq 100000$。

DECORATE - Solution

我们可以把问题分为两个子问题：

- 求出 S有多少本质不同的回文子串。记这个值为 x。
- 求出用 x种不同的颜色给长度为 n的序列染色，有多少种本质不同的方案。

DECORATE - Solution

先来解决第二个子问题。

DECORATE - Solution

先来解决第二个子问题。

根据 Burnside 引理，本质不同的着色数等于各置换下不动点的平均数。

这里一共有 $2n$种置换。由于 n很小，可以暴力计算出每个置换的循环个数。

对于一个有 k个循环的置换，其不动点个数就是 x^k。

DECORATE - Solution

第二个问题被轻松愉快地解决了，再来看看第一个问题。

DECORATE - Solution

第二个问题被轻松愉快地解决了，再来看看第一个问题。

由于回文子串的数量可以达到 $O(n^2)$，所以我们需要高效的算法。

DECORATE - Solution

第二个问题被轻松愉快地解决了，再来看看第一个问题。

由于回文子串的数量可以达到 $O(n^2)$，所以我们需要高效的算法。先考虑一个与之相关的问题。求出一个字符串有多少回文子串的最快的算法是什么？

DECORATE - Solution

第二个问题被轻松愉快地解决了，再来看看第一个问题。

由于回文子串的数量可以达到 $O(n^2)$，所以我们需要高效的算法。

先考虑一个与之相关的问题。求出一个字符串有多少回文子串的最快的算法是什么？

Manacher 算法！

DECORATE - Solution

第二个问题被轻松愉快地解决了，再来看看第一个问题。

由于回文子串的数量可以达到 $O(n^2)$，所以我们需要高效的算法。
先考虑一个与之相关的问题。求出一个字符串有多少回文子串的最快的算法是什么？

Manacher 算法！

而一个字符串的本质不同回文子串只会是 Manacher 算法在向右拓展时的子串，因此本质不同的回文子串的个数是 $O(n)$的。

DECORATE - Solution

第二个问题被轻松愉快地解决了，再来看看第一个问题。

由于回文子串的数量可以达到 $O(n^2)$，所以我们需要高效的算法。

先考虑一个与之相关的问题。求出一个字符串有多少回文子串的最快的算法是什么？

Manacher 算法！

而一个字符串的本质不同回文子串只会是 Manacher 算法在向右拓展时的子串，因此本质不同的回文子串的个数是 $O(n)$的。

不过这里面还是可能有相同的子串。怎么办呢？

DECORATE - Solution

第二个问题被轻松愉快地解决了，再来看看第一个问题。

由于回文子串的数量可以达到 $O(n^2)$，所以我们需要高效的算法。

先考虑一个与之相关的问题。求出一个字符串有多少回文子串的最快的算法是什么？

Manacher 算法！

而一个字符串的本质不同回文子串只会是 Manacher 算法在向右拓展时的子串，因此本质不同的回文子串的个数是 $O(n)$的。

不过这里面还是可能有相同的子串。怎么办呢？

Hash 一下不就完了吗？

DECORATE - Solution

结合这两个算法，我们就得到了这道题的最终算法。
子问题一算法的复杂度为 $O(|S|\log|S|)$（因为要排序去重）。
子问题二算法的复杂度为 $O(n^2)$。
总复杂度为 $O(|S|\log|S|+n^2)$。
顺带一提这题要高精度。

LEMOVIE

对于一个序列，定义其“激动值”为序列中严格大于前面所有数的元素的个数。比如，$\{1, 1, 5, 6, 5\}$的激动值为 3。

给定 n个数 $p_1, p_2, \ldots, p_n$，求这 n个数的所有排列中，激动值不超过 k的个数。

$1 \leq k \leq n \leq 200$，$1 \leq p_i \leq 200$。

LEMOVIE - Solution

先记录相同的数的个数，然后去重。

LEMOVIE - Solution

先记录相同的数的个数，然后去重。

从大到小考虑每组相同的数。令 $f[i][j]$代表已经插入前 i大的数，且激动值为 j的序列方案数。

LEMOVIE - Solution

先记录相同的数的个数，然后去重。

从大到小考虑每组相同的数。令 $f[i][j]$代表已经插入前 i大的数，且激动值为 j的序列方案数。

考虑当前这组数插入在哪些位置。每一组数的贡献至多为 1，有贡献当且仅当至少一个数被插在了最前面。无论插入在什么位置，已有的激动值不会减少。

LEMOVIE - Solution

先记录相同的数的个数，然后去重。

从大到小考虑每组相同的数。令 $f[i][j]$代表已经插入前 i大的数，且激动值为 j的序列方案数。

考虑当前这组数插入在哪些位置。每一组数的贡献至多为 1，有贡献当且仅当至少一个数被插在了最前面。无论插入在什么位置，已有的激动值不会减少。

假设已经插入了 x个数，当前这组有 y个数，没有一个数被插在最前面的方案数为

$$y!\binom{x+y-1}{x-1}$$

至少一个被插在最前面的可以类似算。

复杂度 $O(nk)$。

LEMOVIE - Extras

如果激动值的定义为，序列中相邻的且构成逆序对的数对呢？
（为啥会有这个问题，是因为我最开始就把题目看成这个了）

LEMOVIE - Extras

如果激动值的定义为，序列中相邻的且构成逆序对的数对呢？
（为啥会有这个问题，是因为我最开始就把题目看成这个了）
这个问题实际上稍难于原问题。

LEMOVIE - Extras

如果激动值的定义为，序列中相邻的且构成逆序对的数对呢？
（为啥会有这个问题，是因为我最开始就把题目看成这个了）
这个问题实际上稍难于原问题。

做法类似。从小到大考虑，此时插入一个数可能会抵消之前的激动值。枚举有几个数插在原来的非逆序对之间，可以类似地算出方案数。
复杂度 $O(nk^2)$。

RRTREE2

给定 n 个点的有根树，1 号点为根，每个节点有权值 w_i。我们称一个数 S 合法，当且仅当存在一个 $1 \sim n$ 的排列 p，使得 p 为这棵树的某个 DFS 序列，且存在 $1 \leq k \leq n$ 使得

$$\sum_{i=1}^{k} w_{p_i} = S$$

对于所有 $1 \leq S \leq \sum w_i$，判断 S 是否合法。
$n \leq 500$，$\sum w_i \leq 100000$。

RRTREE2 - Solution

从特殊情况入手：假设树的深度为 1，即除了根节点之外都是叶子节点。

RRTREE2 - Solution

从特殊情况入手：假设树的深度为 1，即除了根节点之外都是叶子节点。

此时问题可以简化为 01 背包问题。

RRTREE2 - Solution

从特殊情况入手：假设树的深度为 1，即除了根节点之外都是叶子节点。

此时问题可以简化为 01 背包问题。

考虑原问题，假设我们凑出 S 时在 x 节点。此时必须和可以被加进背包的有哪些东西？

RRTREE2 - Solution

从特殊情况入手：假设树的深度为 1，即除了根节点之外都是叶子节点。

此时问题可以简化为 01 背包问题。

考虑原问题，假设我们凑出 S时在 x节点。此时必须和可以被加进背包的有哪些东西？

必须被加进去的是从 x到根路径上的所有点；可以被加进去的是 x及祖先的所有兄弟的**整棵子树**。

RRTREE2 - Solution

我们在 DFS 的过程中处理完一个点时，就把它所有儿子的子树中所有节点权值之和视为一个物品，加进背包；在往下 DFS 的时候，把选择的儿子的子树权值和对应的物品从背包中删除。接下来就可以递归处理了。

如何从背包中删除物品？

RRTREE2 - Solution

我们在 DFS 的过程中处理完一个点时，就把它所有儿子的子树中所有节点权值之和视为一个物品，加进背包；在往下 DFS 的时候，把选择的儿子的子树权值和对应的物品从背包中删除。接下来就可以递归处理了。

如何从背包中删除物品？

存储方案数，对多个数取模。

RRTREE2 - Solution

我们在 DFS 的过程中处理完一个点时，就把它所有儿子的子树中所有节点权值之和视为一个物品，加进背包；在往下 DFS 的时候，把选择的儿子的子树权值和对应的物品从背包中删除。接下来就可以递归处理了。

如何从背包中删除物品？

存储方案数，对多个数取模。

由于每棵子树只会被加入一次删除一次，故总复杂度为 $O(nW)$，其中 $W = \sum w_i$。

RRTREE2 - Solution

我们在 DFS 的过程中处理完一个点时，就把它所有儿子的子树中所有节点权值之和视为一个物品，加进背包；在往下 DFS 的时候，把选择的儿子的子树权值和对应的物品从背包中删除。接下来就可以递归处理了。

如何从背包中删除物品？

存储方案数，对多个数取模。

由于每棵子树只会被加入一次删除一次，故总复杂度为 $O(nW)$，其中 $W = \sum w_i$。

有没有更加优美的做法呢？

RRTREE2 - Solution

当然是有的。我们要求的只是：n个物品删去每一个分别得到的背包。

RRTREE2 - Solution

当然是有的。我们要求的只是：n个物品删去每一个分别得到的背包。

可以考虑分治：每次处理删去前一半中物品的背包，将后一半物品依次加入前一半的背包。反之亦然。

这样每个物品会被加入 $O(\log s)$个背包中，其中 s为其兄弟个数。

故总复杂度为 $O(nW\log n)$。

PERMUTE

求有多少个 $1 \sim n$的排列满足，任意相邻的两个数之和不超过 m。
$2 \le n \le 1000000$，$n < m < 2n$，有 100000组测试数据。

PERMUTE - Solution

考虑从大到小插入所有数字。考虑插入一个数字时，其左边和右边相邻的数字是什么。

对于最开始的状态，可行的数字是 $1, 2, \ldots, m-n$，以及左右端点。我们可以选择两个数字，或者一个端点和一个数字。

PERMUTE - Solution

考虑从大到小插入所有数字。考虑插入一个数字时，其左边和右边相邻的数字是什么。

对于最开始的状态，可行的数字是 $1, 2, \ldots, m - n$，以及左右端点。我们可以选择两个数字，或者一个端点和一个数字。

插入之后，我们把 n 以及两边的数字（或端点）看做一个整体，记作 $s(n)$。显然，这个整体也可以与之后的任意数字相邻。那么我们从此时的可行集合中删去这次选择的两个数字（或端点），再插入 $s(n)$ 和 $m - n + 1$。这就是下一步的可行集合。可以发现可行集合的大小没有变。

PERMUTE - Solution

考虑从大到小插入所有数字。考虑插入一个数字时，其左边和右边相邻的数字是什么。

对于最开始的状态，可行的数字是 $1, 2, \ldots, m-n$，以及左右端点。我们可以选择两个数字，或者一个端点和一个数字。

插入之后，我们把 n 以及两边的数字（或端点）看做一个整体，记作 $s(n)$。显然，这个整体也可以与之后的任意数字相邻。那么我们从此时的可行集合中删去这次选择的两个数字（或端点），再插入 $s(n)$ 和 $m-n+1$。这就是下一步的可行集合。可以发现可行集合的大小没有变。

而当插入到 $\lceil \frac{m}{2} \rceil$ 时，就可以任意排列了。此时尚未确定位置的就是可行集合中的元素个数。当 m 为偶数时，方案数是 $(m-n)!$；当 m 为奇数时，方案数是 $(m-n+1)!$。

PERMUTE - Solution

令 $f(n,k)$为可行集合大小为 k（这里忽略端点）时，填入 n到 1 的方案数。我们有

$$
\begin{aligned}
f(n,k) &= k(k-1)f(n-1,k) + 2k \cdot f(n-1,k) \\
&= k(k+1)f(n-1,k) \\
&= (k(k+1))^i f(n-i,k)
\end{aligned}
$$

前面的部分是选择两个数字，后面则是选择一个端点和一个数字。

PERMUTE - Solution

令 $f(n,k)$为可行集合大小为 k（这里忽略端点）时，填入 n到 1 的方案数。我们有

$$
\begin{aligned}
f(n,k) &= k(k-1)f(n-1,k) + 2k\cdot f(n-1,k) \\
&= k(k+1)f(n-1,k) \\
&= (k(k+1))^i f(n-i,k)
\end{aligned}
$$

前面的部分是选择两个数字，后面则是选择一个端点和一个数字。

因此答案就是

$$
f(n,m-n) = \begin{cases} (k(k+1))^{n-\lceil \frac{m}{2} \rceil}(m-n)! & m \text{ is even} \\ (k(k+1))^{n-\lceil \frac{m}{2} \rceil}(m-n+1)! & m \text{ is odd} \end{cases}
$$

QTREE6

给定一棵 n个点的树，每个点可以是黑色或者白色。初始时所有点都是黑色。有 m次操作，操作有下面两种：

- 询问有多少个节点与 u相连。两个节点 u和 v相连当且仅当 u到 v的路径上所有点（包括 u、v两点）的颜色相同。
- 改变节点 u的颜色。

$n, m \leq 100000$。

QTREE6 - Solution

一个比较棘手的地方在于，修改影响的点可能有儿子、祖先甚至没有祖先与子孙关系的节点。

QTREE6 - Solution

一个比较棘手的地方在于，修改影响的点可能有儿子、祖先甚至没有祖先与子孙关系的节点。

如果只询问子树中相连的节点有多少个，要怎么做呢？

QTREE6 - Solution

一个比较棘手的地方在于，修改影响的点可能有儿子、祖先甚至没有祖先与子孙关系的节点。

如果只询问子树中相连的节点有多少个，要怎么做呢？

维护黑白两种颜色的答案。修改时查询子树的答案，然后将所有相连的祖先的答案都减去这棵子树的答案。

找到最浅的相连祖先，就变成了链修改问题。

而查询就直接是点查询。

QTREE6 - Solution

一个比较棘手的地方在于，修改影响的点可能有儿子、祖先甚至没有祖先与子孙关系的节点。

如果只询问子树中相连的节点有多少个，要怎么做呢？

维护黑白两种颜色的答案。修改时查询子树的答案，然后将所有相连的祖先的答案都减去这棵子树的答案。

找到最浅的相连祖先，就变成了链修改问题。

而查询就直接是点查询。

如何找最浅的相连祖先？

QTREE6 - Solution

一个比较棘手的地方在于，修改影响的点可能有儿子、祖先甚至没有祖先与子孙关系的节点。

如果只询问子树中相连的节点有多少个，要怎么做呢？

维护黑白两种颜色的答案。修改时查询子树的答案，然后将所有相连的祖先的答案都减去这棵子树的答案。

找到最浅的相连祖先，就变成了链修改问题。

而查询就直接是点查询。

如何找最浅的相连祖先？

维护到根路径上的黑点个数，倍增时查询。

还是只要链修改和点查询。

QTREE6 - Solution

我们需要一个支持链修改和点查询的数据结构。

QTREE6 - Solution

我们需要一个支持链修改和点查询的数据结构。
动态树显然可以，但还有更简单的方法。

QTREE6 - Solution

我们需要一个支持链修改和点查询的数据结构。

动态树显然可以，但还有更简单的方法。

用轻重链剖分的思想，在 DFS 的时候先走重链。这样得到的 DFS 序列中重链会是一个区间。

一条链也就会被划分成 $O(\log n)$个区间。链修改就成了区间修改。

树状数组即可胜任。

QTREE6 - Solution

回到最开始的问题。

QTREE6 - Solution

回到最开始的问题。
其实问题已经解决了对吧？
因为一个点的答案就是其最浅相连祖先的子树的答案。
直接套用上面的方法就可以解决了。

总复杂度为 $O(n\log^2 n)$，瓶颈在于倍增部分。
如果有同学对于如何优化到 $O(n\log n)$有想法，欢迎在课后来讨论。

COT5

Treap 是一棵平衡树，每个点有两个值，key 和 weight。Treap 的形态使得 key 值满足 BST 的性质，且 weight 值满足堆的性质。

维护一个大根堆 Treap，有 n次操作，操作有下面三种：

- 插入一个 key 值为 k，weight 值为 w的节点。
- 删除 key 值为 k的元素。
- 查询 Treap 中 key 值为 k_u和 k_v的元素在 Treap 中的距离。这里距离定义为树上两个点之间唯一路径的长度。

$n \leq 200000$，任意两个点的 key 值和 weight 值均不相同。

COT5 - Solution

Treap 期望 $O(\log n)$的复杂度来源于 weight 值的随机性。如果 weight 值随机，那么期望树高也是 $O(\log n)$级别的，我们完全可以暴力。

但题目中 weight 值是给定的，Treap 可以被构造成一条链，我们不能直接建立一棵 Treap 来模拟操作。

我们需要挖掘 Treap 特别的性质。

COT5 - Solution

我们可以把问题分成两个部分：

- 求出两个点的 LCA。
- 求出一个点的深度。

解决了这两个问题，就可以回答询问了。

COT5 - Solution

先考虑如何求出两个点的 LCA。

COT5 - Solution

先考虑如何求出两个点的 LCA。

设两个点为 u 和 v，不妨设 $k_u < k_v$。设其 LCA 为 x。

COT5 - Solution

先考虑如何求出两个点的 LCA。

设两个点为 u和 v，不妨设 $k_u < k_v$。设其 LCA 为 x。
有这样一个结论：$k_u \le k_x \le k_v$。可以用反证法证明。

COT5 - Solution

先考虑如何求出两个点的 LCA。

设两个点为 u和 v，不妨设 $k_u < k_v$。设其 LCA 为 x。
有这样一个结论：$k_u \le k_x \le k_v$。可以用反证法证明。
而由 Treap 的堆的性质可得，x应该是 key 值在 $[k_u, k_v]$内且 weight 值最大的元素。
因此，我们只需要一个支持动态 RMQ 的数据结构，即可找到 LCA。

COT5 - Solution

再考虑如何求出一个点的深度。设需要求深度的点为点 x。

COT5 - Solution

再考虑如何求出一个点的深度。设需要求深度的点为点 x。

一个点的深度即其不同的祖先的个数。结合上面求 LCA 的算法，我们有一个暴力的想法：

枚举所有其他的点，和 x求 LCA，看有多少个不同的 LCA。

直接这么做肯定不行，我们想想可不可以优化这个方法。

COT5 - Solution

再考虑如何求出一个点的深度。设需要求深度的点为点 x。

一个点的深度即其不同的祖先的个数。结合上面求 LCA 的算法，我们有一个暴力的想法：

枚举所有其他的点，和 x求 LCA，看有多少个不同的 LCA。

直接这么做肯定不行，我们想想可不可以优化这个方法。

这个方法等价于，固定区间的一个端点，另一个端点扫过所有的数，问最大值变化了几次。

COT5 - Solution

再考虑如何求出一个点的深度。设需要求深度的点为点 x。

一个点的深度即其不同的祖先的个数。结合上面求 LCA 的算法，我们有一个暴力的想法：

枚举所有其他的点，和 x求 LCA，看有多少个不同的 LCA。

直接这么做肯定不行，我们想想可不可以优化这个方法。

这个方法等价于，固定区间的一个端点，另一个端点扫过所有的数，问最大值变化了几次。

我们如果把另一个端点从固定的端点开始移动，会发现这个变化次数实际上就是，从固定的端点向左向右看，比之前的所有数字都大的数的个数。

而这个问题是可做的，我们分开考虑左右两个方向的答案，实际上这个问题就是 2012 年集训队清华集训的《楼房重建》。

COT5 - Solution

《楼房重建》要怎么做？

COT5 - Solution

《楼房重建》要怎么做？

线段树套平衡树？应该是能做的，但是细节特别多（我写到一半就放弃了）。

COT5 - Solution

《楼房重建》要怎么做？

线段树套平衡树？应该是能做的，但是细节特别多（我写到一半就放弃了）。

事实上，我们可以只用线段树完成这个任务。

COT5 - Solution

《楼房重建》要怎么做？

线段树套平衡树？应该是能做的，但是细节特别多（我写到一半就放弃了）。

事实上，我们可以只用线段树完成这个任务。

很明显，满足条件的数是单调递增的。我们修改一个数只会对后面的数字造成影响。

考虑线段树划分出来的若干个区间，有两种情况：

- 区间中的最大值小于等于修改的数。显然这个线段的贡献为 0。
- 否则，我们将这个区间分成两部分。如果左侧的最大值大于修改的数，那么是不影响右侧的贡献的，只需递归处理左侧；否则又变成了第一种情况，只需递归处理右侧。

COT5 - Solution

实现上的伪代码大致如下：

```
calc(SegNode x, l, r, value):
    if (l == r)
        if (x.val > value) return 1
        else return 0
    mid = (l + r) / 2
    if (x.left.val <= value) return calc(x.right, mid + 1, r, value)
    else return x.count - x.left.count + calc(x.left, l, mid, value)

modify(SegNode x, l, r, pos, value):
    if (l == r)
        x.val = value, x.count = 1
        return
    mid = (l + r) / 2
    if (pos <= mid) modify(x.left, l, mid, pos, value)
    else modify(x.right, mid + 1, r, pos, value)
    x.val = max(x.left.val, x.right.val)
    x.count = x.left.count + calc(x.right, mid + 1, r, x.left.val)
```

February Challenge 2014 - Count on a Treap

这里x.count为，仅考虑x代表的区间时，比前面所有数都大的数的个数。注意calc函数中最后一个return那里的实现。

查询类似，此处从略。

COT5 - Solution

回到原问题。

由于问题没有强制在线，我们可以先读入所有操作，无视删除，处理出最后的排好序的 key 值序列。这样插入和删除操作都可以视为 weight 值的修改，插入即改为应有的 weight 值，删除则改为 0。

COT5 - Solution

回到原问题。

由于问题没有强制在线，我们可以先读入所有操作，无视删除，处理出最后的排好序的 key 值序列。这样插入和删除操作都可以视为 weight 值的修改，插入即改为应有的 weight 值，删除则改为 0。

那么两部分都可以用线段树完成了。总复杂度 $O(n \log^2 n)$。

COT5 - Solution

回到原问题。

由于问题没有强制在线，我们可以先读入所有操作，无视删除，处理出最后的排好序的 key 值序列。这样插入和删除操作都可以视为 weight 值的修改，插入即改为应有的 weight 值，删除则改为 0。

那么两部分都可以用线段树完成了。总复杂度 $O(n\log^2 n)$。

如果强制在线呢？

COT5 - Solution

回到原问题。

由于问题没有强制在线，我们可以先读入所有操作，无视删除，处理出最后的排好序的 key 值序列。这样插入和删除操作都可以视为 weight 值的修改，插入即改为应有的 weight 值，删除则改为 0。
那么两部分都可以用线段树完成了。总复杂度 $O(n \log^2 n)$。

如果强制在线呢？
改成平衡树就好了。

The End

谢谢！
欢迎课后交流！