BITACM 俱乐部 2023 年暑假集训 Rating#2 Solution

2023 年 7 月 29 日

A ○国大选

对于每个事件，需要查询区间内等于 x 的数的数量，并把区间内的数全部替换为 x。此题把询问和修改绑在一起，就是为了通过修改降低序列中数的多样性，来让询问易于维护。

下面介绍分块做法：把序列分块后，每个块维护一个标记。若块中所有数都相等，标记就等于这个数；否则，标记为 -1。对于每个询问区间，我们自然是暴力统计涉及到非完整块的部分，暴力统计标记为 -1 的块，然后简单地统计标记值和询问数相同的块数。由于询问后涉及到的数全变成同一个值，刚被暴力枚举过的块都可以打上标记，下次就不会再暴力了。

每次操作至多破坏两个块的标记，结合暴力统计非完整块的复杂度，不难发现是最经典的分块模型。取块大小为 $\sqrt{n}$ 即可做到 $O(n\sqrt{n})$。

B 七星瓢虫

查询区间内出现次数与 v 互质的瓢虫有多少种。

先简单考虑，查询区间内出现次数为 x 的瓢虫有多少种。首先需要知道每种瓢虫的出现次数，但如果枚举各种瓢虫来查询那复杂度不能接受，纯随机数据就会让一个区间内有非常多种瓢虫。所以还需要在维护各种瓢虫出现次数的基础上，维护各种瓢虫出现次数的出现次数，比如有 100 种瓢虫出现 1 次，就可以合在一起来高效计算答案。这部分用莫队算法即可维护。根据莫队算法移动当前区间的原理，需要支持“某种瓢虫出现次数 +1 或-1”，那对于次数的次数就要支持插入或删除某个值。如果就这样，直接用桶记录就行了。但对于原问题，如果没有高妙的数学变形，只能在判定与 v 互质时把各个次数都拿出来验证一下，这样复杂度似乎很高。

此时可以发现一个性质：“瓢虫出现次数”的种数不多，最多有大约 $\sqrt{r-l}$ 种，即使每种瓢虫的出现次数都不同，且都尽可能小，也就是从 1 开始的递增数列。当然，实际上出现次数可能有小有大，在桶上位置不连续，那么把莫队算法中的桶换成双向链表，即可维护插入删除且在遍历时不涉及值域，高效统计答案。遍历一遍判是否互质就行了。

C 集合点

可以证明，对三个点两两求一遍 LCA 后，一定存在至少两个 LCA 重合。

如果只有两个 LCA 重合，则最优集合点为剩下的那个 LCA 点；

如果三个 LCA 均重合，则最优集合点就是三个 LCA 重合的点。

D 最好的序列

先不考虑区间的限制，考虑每个下标 i 作为子序列的起始位置，好的序列的长度最大是多少，这个可以用滑动窗口线性处理出所有下标对应的最大长度。

这样结合区间求最值是不是就得到答案了呢？并不是，因为可能会出现这样的情况。假设区间 $[L,R]$ 内长度最大的起始位置是 i，长度是 m，可能有 $i+m-1>R$，即这个最长的子序列超出了区间的范围。但如果假设 j 是 $[L,R]$ 中最长子序列超出区间范围的最小坐标，容易发现 $[j,R]$ 内所有下标对应的最长子序列都会超出区间。因此我们可以二分得到 j，然后 $[L,j-1]$ 和 $[j,R]$ 两个区间分别处理，前者中最长子序列一定不会超出 R，因此最长长度就是求区间最值，后者的最长长度是 $R-j+1$，两者中较大那个就是答案。时间复杂度 $O(m\log n)$。

E 地铁

预处理出每个点坐一次车，可以到达的最靠左的点和最靠右的点。继而倍增预处理出每个点坐 2^k 次车，可以到达的最靠左的点和最靠右的点。

现在我们把这些区间展开成一棵树：如果区间 a 完全包含了区间 b，且区间 a 代表了从 i 出发跳 k 次覆盖的区间，区间 b 代表了从 i 出发跳 $k-1$ 次覆盖的区间，那么我们将 b 挂在 a 下面（事实上并不需要显式建树，这么解释只是方便直观理解）。注意到，树上的任意两个区间之间，只有包含和不相交（只在端点重合也归为这一类）两类关系，不会出现部分相交的情况。

现在来转化问题：我们现在让 $[A,A]$ 和 $[B,B]$ 两个点向上跳，直到它们的父节点相同为止（此时这两个区间紧挨着）。这意味着，一个从 A 点出发，坐若干次车之后的人，和另一个从 B 点出发，坐若干次车的人在一个点相遇了，两个人的行程拼接起来，就得到了 $A\to B$ 的完整路程。因此，只要把两个区间向上跳的次数相加，即可得到答案。

时间复杂度 $O(q\log n)$。

F 最幸运的一次

考虑 Kruskal 算法的过程，按照贪心法的原则逐次加入树边。

此时假如我们想得到一颗次小生成树，只能把至多一条“最优”的树边换成非树边。

因为换掉两条树边的方案一定不优于只换一条的方案，与“次小”矛盾。

所以我们可以得到一个非严格（注意，非严格！）次小生成树的暴力算法：逐次枚举最小生成树上的每一条边并删去，在新图中寻找最小生成树。

时间复杂度 $O(nm\log m)$。

考虑这样的优化：

在暴力算法中，我们每次都求出一颗新的最小生成树，但这是没有必要的，因为由上述讨论，最小生成树与次小生成树最多只有一条边的差距，所以我们可以考虑如何利用当前最小生成树的已知信息。

现在我们考虑把一条边加入最小生成树中，要维持树的特性显然需要再删除一条边。

所以我们令加入的这一条边为 (x,y)，然后删去最小生成树上 x 到 y 的路径中最大的一条边就可以了。

最大边可以使用倍增的带权 lca，树剖，LCT 等算法维护。

最后我们考虑怎么处理“严格”这个条件。

发现非严格之所以非严格，是因为有可能 $w(x,y)$ 刚好等于最小生成树上的 x 到 y 路径上的最大边，这个时候怎么办呢？我们显然需要把操作改为删去 x 到 y 路径上的严格次大边。

这个过程仍然可以采用上述维护方法。

G 搬砖

本题题意用数学语言表达，即为寻找一个与序列 $\{a_i\}$ 等长的非严格单调序列 $\{b_i\}$，使得 $\sum_{i=1}^{n}|a_i-b_i|$ 最小。

通过样例推导不难发现，当我们想要修改 a_i 时，我们只需要令其等于 a_{i-1} 或 a_{i+1}，即可满足非严格的单调性，再多改就浪费了体力。

明确这一点后，我们可以确定，最终的单调序列中的每一个数，一定和原序列中的某一个数相等。因此，我们取 $\{a_i\}$ 的副本 $\{a_i'\}$ 并对其排序（不妨先假设最终序列是非严格单增的，这里就对副本排升序），用 $dp[i][j]$ 表示最终序列从左往右的第 i 个数 b_i 和排序副本的第 j 项 a_j' 相等时耗费的最小体力，于是有

$$dp[i][j]=\min_{1\le k\le j}dp[i-1][k]+|a_i-a_j'|$$

得到将原序列改为不下降序列的最小耗费体力为

$$\min_{1\le i\le n}dp[n][i]$$

再将 a_i 倒序排一次，重复上述操作，答案即为两次结果取 min。

上述算法时间复杂度 $O(n^3)$，空间复杂度 $O(n^2)$，可通过前缀 min 将时间复杂度优化至 $O(n^2)$, 通过滚动数组将空间复杂度优化至 $O(n)$。

H 线

两点确定一条直线：

当 $K=1$ 时，答案自然是无穷多条。

当 $K\ge 2$ 时，我们可以枚举两点，求出其确定的直线，再枚举所有点，判断该直线经过的点数是否不少于 K。

求直线方程：用直线的一般式方程 $Ax+By+C=0$（普适性）来表示直线。

已知经过点 $(x1\ y1),(x2\ y2)$，那么 $A=y2-y1, B=x1-x2, C=x2*y1-x1*y2$。

为了方便直线的判重，对参数进行处理（唯一性），使得：ABC 三数公因数为 1，$A>0$ 或 $A=0$ 且 $B>0$。

I 恐怖的末位淘汰

这是可并堆模板。维护一些小根堆，支持合并操作和弹出堆顶最小值即可。由于测试数据是完全随机造的，需要在调用函数前判掉查询值已被删除的情况，别卡死了。

J 交

以下按逆时针方向讨论，即多边形的内部位于边的左侧。平移后的边和原来的边是平行的，取平移后的边上一点，向原边的两端连线形成三角形，显然向左平移得越多，该三角形的有向面积就越大。

要判断某点是否在所有多边形内部，即判断该点是否在所有边左侧，那么我们可以对每条边所有的平移形成的三角形面积取最大值，若要判断的点计算出的面积小于最大值，则该点就不在所有多边形内部。

注意最大值要初始化为负无穷，因为向右平移的话，有向面积为负值。

K 凸包

首先凸多边形的价值转化为凸多边形内部点数的选择方案（每个点选或不选）。

先考虑没有多点共线的情况。

本题中，对于每个方案，凸多边形外面没有点。

对于一个若干点的图，只有唯一的凸多边形包括整个图。

由上可知， 和 等价，也就是对于整个图枚举点选或不选的方案，唯一对应了一个答案，总价值为 $2^n - n^2 - 1$（n 为总点数，减去选 $0, 1, 2$ 个点的方案）。

再来处理多点共线的情况，显然不能使选择的所有点都在同一直线上，所以枚举同一直线上的点数减去，细节见代码。

本题套路：枚举同一直线上的点，利用在同一直线上的点必定在其中两个点组成的直线上的原理，只需枚举任意两点，再枚举第三点是否在该直线上即可，复杂度 $O(n^3)$。

题外套路：枚举多少直线交于一点，利用交于同一点的直线必然经过其中两条直线相交点的原理，直接枚举两条直线后再枚举第三条即可。

L 序列

最优策略显然是取区间和最大的那 k 个区间，然而暴力枚举所有合法区间的时间复杂度难以接受。

注意到，设序列 $\{A_i\}$ 的前缀和为 $\{s_i\}$，由于 $l \sim r$ 的区间和等于 $s_r - s_{l-1}$，因此在固定区间左端点 l 的前提下，求解最大区间和等价于找到最大的 s_r。

我们设 $f(l, r_1, r_2)$ 表示以 l 为区间左端点，区间右端点的范围在 $[r_1, r_2]$ 之间时的最大区间和（同时记录一下区间最大值对应的 r），根据前面的观察，这个可以转化为求 $[r_1, r_2]$ 区间内 s 的最大值，从而方便用 ST 表维护。

维护一个大根堆，一开始枚举左端点 l，求出对应的 r 的完整区间，算出相应的 $f(l, r_1, r_2)$ 后加入大根堆。

取出堆顶元素一次，代表选择一个区间，设该区间的右端点为 r，这时候需要将原来的 $f(l, r_1, r_2)$ 分拆成 $f(l, r_1, r-1)$ 和 $f(l, r+1, r_2)$ 两个子问题，分别计算完成后再插入堆中。

容易发现，堆中覆盖了所有还没取过的区间的信息，且已经被取过区间的信息均不会再参与计算，因此该算法是正确的。

在 n, k 同阶的情况下，预处理（ST 表建立，堆的初始化）时间复杂度 $O(n \log n)$，每次取区间更新信息时间复杂度 $O(\log n)$，总时间复杂度 $O(n \log n)$。

M 铸币

设 $dp[i][j]$ 表示价值为 i，n 进制数字的最后一位为 j 的方案数，不难写出转移方程

$$dp[i][j] = \sum dp[i-k^2][j-k] + dp[i-k^2][j+k]$$

但由于 W 的范围是 10^9，不能用这样的转移方程直接递推；但原方程是一个纯粹的递推，我们可以考虑矩阵快速幂加速，将线性递推优化至 log 级别。k 的范围是 $1 \le k \le n-1$，j 的范围是 $0 \le k \le n-1$，故转移矩阵应当为一个 $(n-1)^2 n$ 阶的 01 方阵，内部参数自行推导即可。

N 代肝

本题的剩余时间 T 实际已经根据数据 n, s, t_i 确定了，如果所有任务都在这一时刻完成，那么可以获得 $T * \sum v_i$ 的报酬。但现在这些任务都在这之后才能完成，所以可以看做完成这些任务的时间就是在消耗本来能得到的报酬。

考虑 $dp[i][j]$ 表示把前 i 个任务分成 j 组扣掉的最小报酬，容易得到

$$dp[i][j] = \min_{0 \le k < i} dp[k][j-1] + \left(j * s + \sum_{l=1}^{i} t_l\right)\left(\sum_{l=k+1}^{i} v_l\right)$$

其中，所有的 $\sum$ 都可以用前缀和处理，但时间复杂度 $O(n^3)$，空间复杂度 $O(n^2)$，需要优化。

先做费用提前优化，考虑每分一组就会增加 s 的时间，这 s 的时间将作用到这组以及这组之后的所有组，即所有这些组的完成时刻都要往后延 s，我们将这一影响带来的损失全部计算在 $dp[i][j]$ 中，于是原方程可被优化为

$$dp[i][j] = \min_{0 \le k < i} dp[k][j-1] + \left(\sum_{l=1}^{i} t_l\right)\left(\sum_{l=k+1}^{i} v_l\right) + s \sum_{l=k+1}^{n} v_l$$

可以发现 $dp[i][j]$ 的第二维可以被直接优化掉，于是进一步化简方程：

$$dp[i] = \min_{0 \le j < i} dp[j] + \left(\sum_{k=1}^{i} t_k\right)\left(\sum_{k=j+1}^{i} v_k\right) + s \sum_{k=j+1}^{n} v_k$$

利用前缀和形式整理式子可得

$$dp[i] = \min_{0 \le j < i} dp[j] - (T_i + s)V_j + (T_i V_i + sV_n)$$

其中 T_i, V_i 分别表示 t_i, v_i 的前缀和。令

$$y = dp[j], x = V_j, k = T_i + s, C = dp[i] - (T_i V_i + sV_n)$$

原方程可写为 $y = kx + C$ 的形式，即找到一条斜率为 k，过某个 $(V_j, dp[j])$ 的截距最小的直线，进行常规的斜率优化操作即可，时间复杂度 $O(n)$，空间复杂度 $O(n)$。