2020 Multi-University Training Contest 2 Editorial

By Claris

2020 年 7 月 23 日

1 Total Eclipse

Shortest judge solution: 911 Bytes.

每次一定是选择一个极大连通块，将里面所有数同时减小，直到最小值变成 0，然后将变成 0 的点删除，分裂成多个连通块再接着做。

为了实现这个策略，可以将整个过程倒过来看，变成按照 b 的值从大到小依次加入每个点。加入每个点 x 时遍历与 x 相连的所有边 (x,y)，如果 y 在 x 之前加入且 x 和 y 不连通则将 x 和 y 合并，并将 y 所在连通块的树根的父亲设为 x，得到一棵有根树。那么每个点 x 在成为最小值之前已经被做了 b_{father_x} 次操作，所以每个点 x 对答案的贡献为 $b_x - b_{father_x}$。

使用并查集支持路径压缩，时间复杂度 $O((n+m)\log n)$。

2 Blood Pressure Game

Shortest judge solution: 1720 Bytes.

将所有数字从小到大排序得到 $b_1, b_2, \ldots, b_n$，那么 $|a_i - a_j|$ 等价于排序后某些相邻段的差值 $b_k - b_{k-1}$ 的区间和，可以将 $\sum |a_j - a_{j-1}|$ 表示成 $\sum cnt_i(b_i - b_{i-1})$，即考虑每对排序后相邻的数 (b_{i-1}, b_i)，统计 $b_i - b_{i-1}$ 对最终结果的贡献。

按照 $b_1, b_2, \ldots, b_n$ 的顺序依次将每个数字插入到最终的序列中，设 $f[i][j][t]$ 表示考虑了前 i 个数，这 i 个数目前已经形成了 j 个连通块，其中有 $t(0 \le t \le 2)$ 个数作为 a_1 或 a_n 的前 k 大方案以及对应的方案数，那么初始状态是 $f[1][1][0]$（b_1 不作为 a_1 或 a_n，有一种方案）和 $f[1][1][1]$（b_1 作为 a_1 或 a_n，有两种方案），目标是求出 $f[n][1][2]$。

对于 $f[i][j][t]$ 到 $f[i+1][][]$ 的转移，则 $b_{i+1} - b_i$ 对最终结果的贡献是 $(2j-t)(b_{i+1} - b_i)$，有以下几种转移：

- 新建一个不含 a_1 和 a_n 连通块，转移到 $f[i+1][j+1][t]$，有 $j+1-t$ 种方案。
- 新建一个作为 a_1 或 a_n 连通块，转移到 $f[i+1][j+1][t+1]$，有 $2-t$ 种方案。
- 合并两个连通块，转移到 $f[i+1][j-1][t]$，有 $j-1$ 种方案。
- 在某个连通块左侧或右侧扩展，且不作为 a_1 和 a_n，转移到 $f[i+1][j][t]$，有 $2j-t$ 种方案。

- 在某个连通块左侧或右侧扩展，且作为 a_1 或 a_n，转移到 $f[i+1][j][t+1]$，有 $2-t$ 种方案。

对于每个状态，收集到所有转移后，将贡献相同的方案合并，然后保留前 k 大即可，使用快速排序的时间复杂度为 $O(n^2k\log k)$，使用归并的时间复杂度为 $O(n^2k)$。

3 Count on a Tree II Striking Back

Shortest judge solution: 2812 Bytes.

k 个 $[0,1]$ 的随机实数的最小值的期望为 $\frac{1}{k+1}$。对于一个大小为 k 的集合，如果给每个元素随机一个正整数，那么多次采样得到的平均最小值越小就说明 k 的值越大。

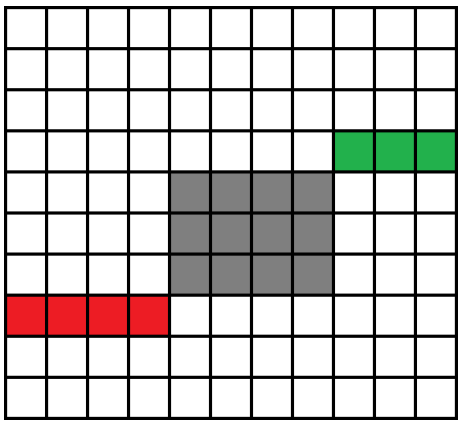
回到本题，进行 k 次采样，每次采样时对每种颜色随机一个正整数，令每个点的点权为其颜色对应的随机数，然后统计询问的树链上点权的最小值，将 k 次采样的结果相加以粗略比较两条树链的颜色数的大小，因为不要求精确值所以 k 取几十即可得到正确结果。

使用树链剖分 + 线段树的时间复杂度为 $O(nk+mk\log^2 n)$，使用全局平衡二叉树可以做到 $O(nk+mk\log n)$。

4 Diamond Rush

Shortest judge solution: 2411 Bytes.

如下图所示，令左上角为起点，右下角为终点，灰色部分为不可走的区域，那么任意一条合法路线必定经过了至少一个红点或绿点。



DP 出起点到每个点的最长路径、终点到每个点的最长路径，相加以得到经过每个点的最长路径 $f[i][j]$，再预处理每一行的前后缀 f 的最大值，则每个询问可以 $O(1)$ 回答。

最后的问题是如何处理权值非常大的情况。对于任意一个方案，令 cnt_i 表示该方案中经过了多少个权值为 $\left(n^2\right)^i$ 的点，则比较两个方案的大小等价于比较字符串 $cnt_{n^2}, cnt_{n^2-1}, \ldots, cnt_1$ 的字典序大小。注意到 DP 的时候每个 DP 值对应的方案只会在之前某个 DP 值对应的方案中将某个 cnt 增加 1，于是可以用可持久化线段树来记录 cnt 数组，并维护区间 Hash 值用于 $O(\log n)$ 比较大小。

时间复杂度 $O((n^2+q)\log n)$。

5 New Equipments

Shortest judge solution: 2287 Bytes.

对于每个二次函数 $a_i \times j^2 + b_i \times j + c_i$，根据函数性质以及公式找到它在 $[1, m]$ 中的最小值，并在最小值附近找到前 n 小的值，由工人 i 向这 n 个位置连边，则可以建出一张点数边数均为 $O(n^2)$ 的费用流图，然后直接增广 n 次找到流量 $= 1, 2, \ldots, n$ 的最小费用流即可。因为左边任何一个大小为 $k(1 \le k \le n)$ 的子集都与右边至少 $n \ge k$ 个点相连，所以根据 Hall 定理一定存在完美匹配，又因为每个点都保留了最小的 n 个取值，所以一定可以找到最优解。

朴素 SPFA 实现的时间复杂度为 $O(n^5)$，常数很小，使用 Dijkstra 费用流的时间复杂度为 $O(n^3 \log n)$。

6 The Oculus

Shortest judge solution: 471 Bytes.

令修改的那一位为 k，则要找到这个 k 满足 $A \times B = C + F_k$，其中 $k \le 2000001$。假设存在一个 P 满足 $F_1, F_2, \ldots, F_{2000001}$ 模 P 两两不同余，则可以根据 $F_k \bmod P = (A \times B - C) \bmod P$ 得到 k。事实上取 $P = 2^{64}$，即自然溢出就是满足条件的。

7 In Search of Gold

Shortest judge solution: 1239 Bytes.

二分答案，判断是否存在树直径长度不超过 mid 的方案。

考虑树 DP，设 $f[i][j]$ 表示考虑了 i 的子树，其中有 j 条边的取值来自 a 数组的所有树直径 $\le mid$ 的方案中，与 i 点距离最远的点到 i 的距离的最小可能值。转移时合并之前已经 DP 过的部分和新加入的子树，如果两部分到 i 的最长距离之和超过 mid 则不转移，因为这表示树直径超过了 mid。

在转移时加上 $j \le \min(k, size[i])$ 的剪枝，时间复杂度为 $O(nk \log ans)$。

8 Dynamic Convex Hull

Shortest judge solution: 2855 Bytes.

考虑离线对操作序列按时间建立线段树，那么每个函数在时间轴上存在的部分一定是一个区间，将其作为标记插入线段树的 $O(\log m)$ 个节点中；对于每个询问，其在时间轴上对应一个点，那么它的答案对应的函数一定在线段树对应叶子节点到根这 $O(\log m)$ 个节点的标记之中，将其作为询问插入对应的 $O(\log m)$ 个节点中。

那么对于线段树的每个节点，它有若干标记和若干询问，这是一个静态的问题，对于每个询问 x，分别找到 $a_i \le x$ 和 $a_i \ge x$ 的最优函数即可。

以 $a_i \le x$ 为例，考虑两个函数 $(x - a_i)^4 + b_i$ 以及 $(x - a_j)^4 + b_j$，其中 $a_i \le a_j$。如果 i 不比 j 优，则说明 $(x - a_i)^4 + b_i \ge (x - a_j)^4 + b_j$，随着 x 的增大这个不等式将会一直成立，所以将函数按照 a 从小到大排序、将询问按照 x 从小到大排序，则最优决策具有单调性，可以分治求解。

时间复杂度 $O((n+m)\log^2 m)$。

9 It's All Squares

Shortest judge solution: 1309 Bytes.

对于每个询问，求出经过的点的横纵坐标的最小值和最大值，显然可以只在框出的矩形里做。在框出的矩形里递推出每个点往左边射线会经过多少次多边形的边界，则根据这个值的奇偶性可以判断出每个点是否在多边形里从而得到答案。

不妨设 n, m 同阶，当询问的形状是正方形时该算法的时间复杂度最大，令正方形的边长为 $k(k \le n)$，则消耗 $4k$ 的输入量需要支付 $O(k^2)$ 的时间。所以最坏情况下需要支付 $O(\frac{\sum |S|n}{4})$ 的时间，这个值只有 10^8 左右，所以可以接受。

10 Lead of Wisdom

Shortest judge solution: 781 Bytes.

令第 i 种装备的数量为 cnt_i，显然如果 cnt_i 不为 0 那么这一种装备不空的方案一定比空的方案优，在这时需要考虑的总方案数为 $\prod \max(cnt_i, 1)$，其中 $\sum cnt_i \le 50$。最坏情况下所有 cnt 的值都相同，令它们都等于 k，则方案数为 $k^{\frac{n}{k}}$，当 k 取 3 时取到最大值 $3^{\frac{n}{3}}$，在 $n = 50$ 时并不算太大，因此可以直接爆搜所有方案得到最优解。

需要注意的是，$cnt_i = 0$ 的部分应该直接跳过，以保证搜索树上每一层的节点数至少是上一层的两倍，使得时间复杂度为 $O(3^{\frac{n}{3}})$，否则会退化成 $O(n3^{\frac{n}{3}})$ 而 TLE。

11 King of Hot Pot

Shortest judge solution: 3389 Bytes.

首先可以发现最优解可以增量构造，即往吃 k 份肉的最优解加入一份肉可以得到吃 $k+1$ 份肉的最优解，因此存在一个顺序 $ord_1, ord_2, \ldots, ord_n$ 满足 $ord_1, ord_2, \ldots, ord_k$ 是吃 k 份肉的一个最优解吃的肉对应的集合。

假设确定了要吃哪些肉，那么肯定是按照捞出锅的时间从小到大吃。按照出锅时间从小到大依次考虑每份肉，在 $ord_1, ord_2, \ldots, ord_{i-1}$ 中找到一个位置插入第 i 份肉。令第 i 份肉的出锅时间为 a，吃掉它的时间为 b，设 $ord_1, ord_2, \ldots, ord_{i-1}$ 里按照出锅时间顺序吃掉前 k 份肉 $ord_1, ord_2, \ldots, ord_k$ 所需的时间为 t_k，则将 ord_k 替换成第 i 份肉后，对应的方案所需的时间为 $\max(t_{k-1}, a) + b$，如果 $\max(t_{k-1}, a) + b < t_k$，则第 i 份肉应该插在 ord_k 之前。注意到满足条件的 k 是 ord 序列的一个后缀，所以可以二分找到对应的位置，将第 i 份肉插入 ord 序列，并将后面部分的 t 都修正为 $\max(t, a) + b$。

为了加速这个过程，可以用平衡树维护 ord 序列，每个位置记录 t_k 以及 t_{k-1} 方便二分，在修正 t 为 $\max(t, a) + b$ 时，根据 t 的单调性将 t 的一个区间赋值为 a，再将一个后缀加上 b。最后得到的 t 序列就是答案。

时间复杂度 $O(n \log n)$。

12 String Distance

Shortest judge solution: 879 Bytes.

考虑修改完毕后的 A 串和 B 串，它们对应位置的字符都相等，对于每一位：

- 如果两个串在这一位都做了插入操作，那么可以同时不做插入操作使得操作次数减少 2。
- 如果一个串 A 在这一位做了插入操作，另一个串 B 这一位不动，那么可以通过 A 这一位不动，B 删除这一位达到同样的效果。

因此可以发现插入操作是没用的，所以两个串 A 和 B 的距离等于 $|A|+|B|-2LCS(A,B)$，其中 LCS 表示最长公共子序列。预处理出 $g[i][j]$ 表示 $A[i..n]$ 里字符 j 最早出现的下标。对于每个询问通过 DP 求出 LCS，设 $f[i][j]$ 表示与 $B[1..i]$ 的公共序列长度达到 j 的 $A[l..r]$ 的最短前缀的长度，利用 g 数组进行转移。

时间复杂度 $O(26n+qm^2)$。