2023“钉耙编程”中国大学生算法设计超级联赛（3）题解

By Claris

2023 年 7 月 25 日

1 Magma Cave

Shortest judge solution: 2785 Bytes.

对于一个询问，首先特判图不连通以及对应边权的边不存在的情况，然后找到这条边 $E=(u,v,w)$，将边权小于 w 的边染成黑边，将边权大于 w 的边染成白边。接下来需要判断是否存在一棵生成树满足 E 必选，且黑边选了恰好 $k-1$ 条。注意到满足条件的 k 是一个区间，只需求出合法情况下黑边数量的最小值和最大值，即可得到 k 的取值范围。

- 求 k 的最大值：首先选择 E，然后贪心地考虑每条黑边，如果加入后无环则加入。这等价于求最小生成树上边权不超过 w 的边数。
- 求 k 的最小值：首先选择 E，然后贪心地考虑每条白边，如果加入后无环则加入。同理可从最大生成树上得到结果。

根据上述讨论，只需使用两棵 Link-Cut Tree 分别维护最小生成树、最大生成树，再分别用两棵树状数组维护两棵生成树上每种边权的出现次数即可。

时间复杂度 $O(q\log q)$。

2 King’s Ruins

Shortest judge solution: 2561 Bytes.

设 f_i 表示最后拍照的是第 i 个骑士时的最大总价值，则 $f_i=\max\{f_j\}+v_i$，其中 $j<i$ 且 j 的每一维能力值都不超过 i 对应的能力值。

将所有骑士按标号分块，每块 k 个骑士，从前往后依次计算每块骑士的 DP 值。对于当前正在计算的某块，块内两两之间可以暴力 $O(k^2)$ 转移。计算完毕整块的 DP 值后，遍历位于该块之后的每个骑士，用该块的某个子集的 DP 值的最大值更新其 DP 值。子集的 DP 值最大值可以 $O(2^k)$ 预处理，$O(1)$ 查询。每个骑士对于该块需要进行转移的子集可以通过 bitset 将每一维能力值不超过它的集合取交集来得到。

时间复杂度 $O(nk+\frac{n^2}{64}+\frac{n2^k+n^2}{k})$，当 k 取 $O(\log n)$ 时为 $O(\frac{n^2}{\log n})$。

3 Leshphon

Shortest judge solution: 1509 Bytes.

令 $f_{E,k}$ 表示从边集 E 中删去恰好 k 条边，使得剩下的图不是强连通图的方案数：

- 如果 E 本来就不强连通，则 $f_{E,k} = C(|E|, k)$。这一步只需检查 1 是否可以到达每个点以及每个点是否可以到达 1，通过正反两次 BFS 实现，BFS 的过程可以通过位运算优化至 $O(\frac{n^2}{64})$。
- 否则如果 $k = 0$，则 $f_{E,k} = 0$。
- 否则考虑第一条中的两棵 BFS 生成树包含的 $O(n)$ 条树边，显然它们之中必须至少要删除一条才可能使得图变得不连通。任意赋予树边一种顺序，枚举其中一条树边 e，将其从 E 中删去，并将它前面的所有树边都标记为无法删除，得到新的边集 E'，则对 $f_{E,k}$ 的贡献为 $f_{E',k-1}$。

一共需要递归 k 层，每层 $O(n)$ 个子问题，底层需要 $O(\frac{n^2}{64})$ 的代价，总时间复杂度为 $O(\frac{n^{k+2}}{64}) = O(\frac{n^5}{64})$。

4 Chaos Begin

Shortest judge solution: 2285 Bytes.

考虑原始 n 个点的凸包，将它复制一份，沿着向量 $(\Delta x, \Delta y)$ 平移的过程将会拉扯出两条平行边，边的两端点恰好对应平移前后的一对点，且差值为 $(\Delta x, \Delta y)$。这启发我们找到输入的 $2n$ 个点的凸包，枚举凸包上的一条边，再枚举边上的两个点，作差得到一个可能的 $(\Delta x, \Delta y)$，检查它是否合法。由于点的坐标在给定的正方形范围内随机，因此一个点集的凸包的期望顶点数为 $O(\log n)$，需要检查的次数也为 $O(\log n)$。

对于一个待检查的 $(\Delta x, \Delta y)$，如果 $(-\Delta x, -\Delta y)$ 检查过，则无需重复检查。否则不妨设 $\Delta x \geq 0$ 且 $\Delta y \geq 0$，找到 $2n$ 个点中横坐标最小的点，如果有多个则取纵坐标最小的点，那么它一定属于原始的 n 个点，将它 (x, y) 与 $(x + \Delta x, y + \Delta y)$ 配对并删去，重复 n 次即可。

时间复杂度 $O(n \log^2 n)$。

5 Out of Control

Shortest judge solution: 725 Bytes.

将数组 x 中的数从小到大排序，那么一个可能的前缀最大值序列 $s_1, s_2, \ldots, s_k$ 合法当且仅当 s 仅包含 x 中的数，且对于每个可能的 i $(1 \leq i \leq k)$ 都有 $s_i \geq x_i$。将数值离散化后，设 $f_{i,j}$ 表示有多少合法的长度为 i 的前缀最大值序列 s 满足 $s_i = j$，则 $f_{i,j} = \sum_{1 \leq k \leq j} f_{i-1,k}$，其中 $j \geq x_i$。通过前缀和可以将转移优化至 $O(1)$。

时间复杂度 $O(n^2)$，亦可以通过分治 FFT 做到 $O(n \log^2 n)$。

6 Dragon Seal

Shortest judge solution: 3254 Bytes.

任取一点作为树根，将树转化为有根树，然后树形 DP。设 $f_{x,j,k}$ 表示考虑了 x 节点的子树，x 点选择第 j 颗宝石，x 点的父亲选择第 k 颗宝石的所有合法方案中，x 子树（不含父亲）的宝石最大总力量值。

考虑如何进行状态转移。对于 x 点的每个儿子 y，需要从状态 $f_{y,0,j}$ 和状态 $f_{y,1,j}$ 中恰好选择一个，满足对应的魔法值线性无关，且 DP 值之和最大。为每个儿子新建一个大小为 2 的集合，集合中每个元素有它的魔法值以及权值，其中权值为对应状态的 DP 值。为了将 x 点和 x 点的父亲也一并考虑进去，新建两个大小为 1 的集合，元素的权值为对应宝石的力量值。那么问题等价于从这些元素中挑选恰好 deg_x+1 个元素，满足魔法值线性无关、每个集合选择了不超过 1 个元素，且权值和最大。容易发现这是两个拟阵的交，利用带权拟阵交算法可以在 $O(deg_x^4)$ 的时间内完成状态转移。

由于所有点的度数之和为 $O(n)$，因此总时间复杂度为 $O(n^4)$。

7 Casino Royale

Shortest judge solution: 2088 Bytes.

对于一个游戏序列 $s_1, s_2, \ldots, s_n$，令 $sum(s) = s_1 + s_2 + \cdots + s_n$ 为 s 的所有数字之和，$diff(s)$ 为最优策略下先手与后手的最终分差，那么根据 $diff(s)$ 以及两者之和为 $sum(s)$ 可以解方程得到两个人的最终分值。

如果一个游戏序列形如 V 字，即存在一个分界点 k 满足 $s_1 \geq s_2 \geq s_3 \geq \cdots \geq s_k$ 且 $s_k \leq s_{k+1} \leq \cdots \leq s_{n-1} \leq s_n$，那么每次一定可以拿到序列中的最大数，于是贪心选择左右端点中的较大数为最优策略。否则序列不为 V 字，那么一定存在相邻三个数 A, B, C 满足 $A \leq B \geq C$，第一个拿走 A 或 C 的人一定是被逼无奈或者为了同时拿走 A 和 C，无论哪种情况对面一定会接着拿走中间较大的 B，于是 A 和 C 必定属于一方，而 B 属于另一方，因此可以用一个数 $A-B+C$ 来等效替代 A, B, C，替换前后的两个序列的 $diff$ 值不变。

通过不断地等效替代，任何游戏序列都能被转化为 V 字，然后贪心计算最终结果。容易发现 V 字中所有数字的绝对值之和为 $O(n)$ 级别，且分界线前后是有序的，这提示我们合法的 V 字的数量不多，因为它很像 n 的整数拆分方案数的平方。但是仅仅知道 $diff$ 值是不够的，还需要知道序列的 sum 值才能解出双方的最终得分，于是有状态 $f_{i,sum,S}$ 表示考虑了序列的前 i 位，前 i 位的数字总和为 sum，且前 i 位的序列化简为 V 字后为 S 时，有多少种可能的填法。写程序可得 $n = 50$ 时，合法的状态数也只有不到 150 万个。预处理出状态机后可以 $O(\text{状态数})$ 计算每组数据的答案。

8 Teyberrs

Shortest judge solution: 2189 Bytes.

离线询问，按照横坐标 $x = 1, 2, 3, \ldots, n$ 的顺序依次处理地图的每一列。对于当前的一列，从起点能飞到的纵坐标集合显然要么是空集，要么是一个公差为 2 的等差数列 $S, S+2, S+4, \ldots, E-4, E-2, E$，其中 $S \leq E$ 且 $(E-S) \bmod 2 = 0$。如果可行的纵坐标集合是空集，显然后续所有询问的答案都是 -1，接下来不考虑这种特殊情况。

如果可行的纵坐标集合非空，那么 S 和 E 可以很方便地维护出来。设起点到 $S, S+2, S+4, \dots, E$ 的最小代价依次为 $f_0, f_1, f_2, \dots, f_m$，令 $d_i = f_i - f_{i-1}$，$s_i = d_1 + d_2 + \cdots + d_i$，则

$$f_i = \begin{cases} f_0 & (i = 0) \\ f_0 + s_{i-1} + d_i & (i > 0) \end{cases}$$

假设当前列的所有信息已经被正确地维护，当横坐标增加 1 时，首先无视新一列的障碍，那么 S, E 将分别变为 $S-1, E+1$。设往右下飞的代价为 A，往右上飞的代价为 B。如果强制往右下飞，那么 $f_0, f_1, \dots, f_m$ 的每一项都要增加 A，并需要在 f_m 后面添加一项 $f_{m+1} = +\infty$，表示 $E+1$ 无法通过上一列往右下飞得到，此时 d 和 s 末尾都要新增一项 $+\infty$，但是 d 和 s 中其它项的数值均没有变化。接下来考虑往右上飞的情况，对于 $1 \le i \le m+1$，需要用 $f_{i-1} + B - A = f_0 + s_{i-1} + B - A$ 来更新 f_i。因此整理可得：

$$f_i = \begin{cases} f_0 + A & (i = 0) \\ f_0 + s_{i-1} + \min(d_i, B - A) & (i > 0) \end{cases}$$

如果差分序列 d 满足 $d_1 \le d_2 \le d_3 \le \cdots \le d_m$，观察 $\min(d_i, B-A)$ 从 d_i 变为 $B-A$ 的分界点，上面的转移式子等价于将 $B-A$ 插入序列 d 中，满足插入后仍然是升序。由于当 $x \le 1$ 时差分序列显然是升序，而每次插入 $B-A$ 不会破坏升序，因此差分序列一定是升序。于是只要维护出 S, E, f_0 以及序列 d，就能求出起点到每个点的最小代价。最后考虑新一列的障碍部分，显然等价于移除 d 的一段前缀和一段后缀，并调整 S, E, f_0 的值。

综上所述，我们需要一个数据结构来维护 d 中的所有元素，支持删除最小值、删除最大值、插入元素以及查询前 k 小元素的和。注意到 d 中的元素一定是某个 $b_i - a_i$，于是可以离散化后用权值线段树来维护。

时间复杂度 $O((n+q)\log n)$。

9 Operation Hope

Shortest judge solution: 1738 Bytes.

二分答案，判定是否存在一种方案满足每种属性的极差都不超过 mid。2-SAT 建模，将每个职业的加强与否作为 n 个变量，如果两个方案在某种属性的差值超过 mid，则选了一个就要选另一个的逆命题。

判定 2-SAT 问题是否有解时，需要求出对应有向图的强连通分量，在这里可以使用 Kosaraju 算法，分别在正图和反图上 DFS，每个遍历过的点后续不需要重复遍历。遍历到一个点 x 时，需要快速找到与它相连的且尚未遍历过的点。枚举三种属性中的一种，那么与 x 相连的点一定是该属性值最小或最大的若干个点。对于每种属性，将所有点按属性值从小到大排序，并维护头指针和尾指针，那么在 DFS 的过程中只需不断消费头指针和尾指针对应的点。由于每个点只会被消费 $O(1)$ 次，因此 2-SAT 判定的时间复杂度为 $O(n)$，总时间复杂度 $O(n \log v)$。

10 The Mine of Abyss

Shortest judge solution: 1933 Bytes.

按下标建立线段树，线段树上每个节点维护区间内所有可能的子集和。在随机数据下，可以用 $O(1)$ 段互不重叠的值域区间来表示合法的子集和，于是在合并两个节点的信息时将每个值域区间对应的左右端点相加，然后将重叠的区间合并即可。

时间复杂度 $O(n + q \log n)$。

11 8-bit Zoom

Shortest judge solution: 726 Bytes.

直接模拟像素放大的过程，判断每个格子是否包含两种以上颜色即可。

时间复杂度 $O(n^2)$。

12 Noblesse Code

Shortest judge solution: 2061 Bytes.

对于一个二元组 (A, B)，它的前驱二元组唯一确定：

- 如果 $A = B$，则它没有前驱二元组。
- 如果 $A > B$，则它的前驱二元组是 $(A - B, B)$。
- 如果 $A < B$，则它的前驱二元组是 $(A, B - A)$。

如果看作二叉树，那么前驱二元组就是父亲，两种变换分别看作左右儿子，一个二元组 (A, B) 所在的树的根节点为 $(\gcd(A, B), \gcd(A, B))$。对于每个二元组，找到树根到它的路径，用仅包含大写字母 “L” 和 “R” 的字符串表示一路上每一步是往左走还是往右走，并在字符串开头加上数字 $\gcd(A, B)$ 以区分不同树根。那么对于一个询问二元组，令它的字符串表示为 S，则答案为字典序小于 $S\$$ 的字符串数量减去字典序小于 S 的字符串数量。

注意到我们可以把连续一段相同的字符压缩起来，那么拐点数量是 $O(\log v)$ 的，可以通过辗转相除法得到，于是暴力比较两个压缩串的时间复杂度为 $O(\log v)$，将所有串排序即可求出每个询问的答案。

时间复杂度 $O((n + q) \log(n + q) \log v)$。