2019牛客暑期多校训练营 第八场

高铭鸿、李泽仁

第一关-All-one Matrices

- **前期题**
  - 单调栈+前缀和

对于每一个格子 (i,j)，记 Up[i][j] 为其向上的连续的 1 的个数。

然后枚举每一行作为矩阵的底边所在行，从前往后枚举每一列，并维护一个关于 Up[i][j]的单调上升的栈，对于栈中每一个 Up值，还要维护一个其向左最远能拓展的位置 pos。

那么每当有元素退栈的时候，设退栈元素是 (Up,pos)，那么可以得到一个全 1 矩阵 (i - Up + 1, pos) - (i, j)，可知其向上向左向右都是不可拓展的，所以只要判一下其往下能不能拓展即可。

可以给行记录前缀和，在 O(1) 时间求出一个横向子段是否为全 1。

时间复杂度：$O(nm)$

第二关- Beauty Values

- **前期题** （但好像大家都很懂这套理论就变签到题了）
  - 期望线性性

我们可以把每种数字对答案的贡献分开来计算，即枚举每个数字，求原序列有多少个子区间包含至少一个该数字，最后把答案累加起来即可。

问题在于求序列有多少个子区间包含至少一个某数字，可以考虑用全集减去不包含的子区间。不包含的子区间的个数可以枚举相邻两个数字，并累加中间空隙的子区间个数，再统计一下边界即可。

时间复杂度：O(n)

第三关- CDMA

- **前期题**
  - 构造

首先 m=2 时的答案是已经知道了的，考虑用 m 构造出 2m 的解：

不妨设方阵 A 为 m 的解，那么下面这个方阵则是 2m 的一个解：

[A A]

[A -A]

把每一行记为 (0/1, i)，表示是上半区/下半区的第 i 行。

首先如果 i 不相同，那么内积显然就是 0 了。

现在考虑 (0,i) 和 (1,i)，可知左半部分的内积贡献为 m，右半部分的内积贡献为 -m，加起来也就是 0 了。

第四关-Distance

- **前期题** （但看起来是个中后期题？）（kd-tree 想过？不存在的）
  - 定期重构

先假设所有询问都在加标记之后，那么我们可以用一次 bfs 求出网格中每个点离最近标记点的距离，询问就可以 O(1) 回答。

现在考虑用定期重构处理增量标记。我们可以记一个新增标记队列，每次拿出 bfs 预处理后的结果，再暴力枚举新增标记队列中的每一个新标记，把这些结果取个 min 即可。

当队列元素超过一个阈值 E 的时候，我们把队列中的标记与已有标记混合，并用 bfs 预处理每个位置的答案，最后清空新增标记队列。

可知复杂度是 O(qnmh/E + qE) 的，当 $E = \sqrt{nmh}$ 时可以取到复杂度的理论最小值 $O\left(nmh + q\sqrt{nmh}\right)$。

有把曼哈顿距离的8种情况暴力讨论然后上三维树状数组的 $O(q \log^3 n)$ 做法，常数挺小：
https://ac.nowcoder.com/acm/contest/view-submission?submissionId=41082796

第五关-Exciting Drifting

- **中后期**　（但好像大家都很懂这套理论就变前期题了）
  - 时间分治

首先可以把 size 离散化，然后建一棵以 size 为关键字的线段树，再把所有边都按照规定的 size 区段加进对应的线段树节点里。

然后就从线段树的根往下 dfs，每次用按秩合并优化并查集，因为回溯的时候要撤销并查集操作。

然后到了叶子节点的话就看一下 1 和 n 是否在同一个并查集里，如果是，则这个叶子节点对应的整数 size 段就都是可以带来贡献的，否则就不会带来贡献

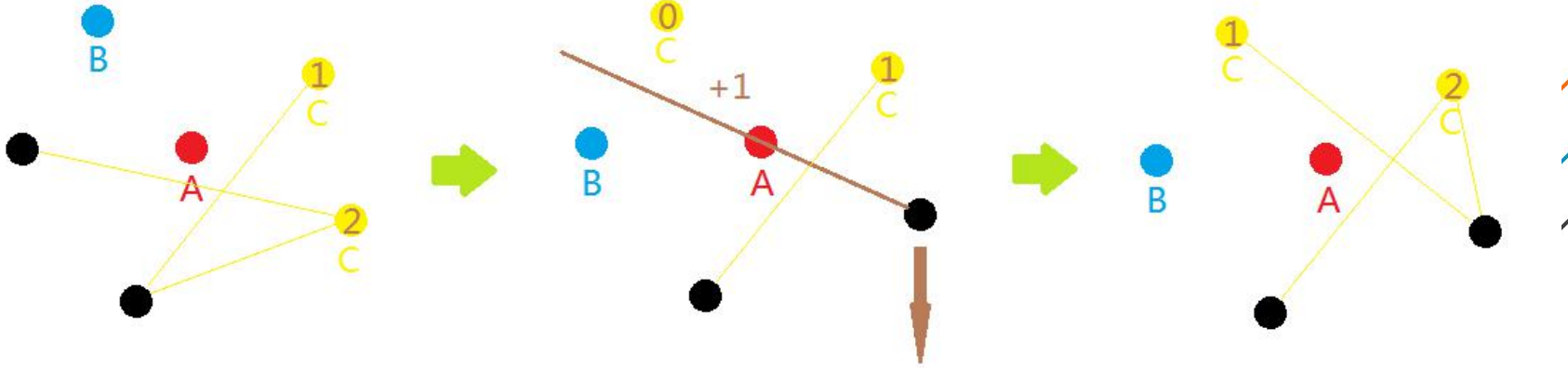
时间复杂度：$O(n \log^2 n)$ 。

第六关-Flower Dance

- **中后期**　　（那个 1h46min 就把这题 AC 了的哥哥队伍好厉害啊）
  - 计算几何
- 枚举点 A，对其他点进行极角排序，然后按照极角序枚举 B，不妨设 C 点必须满足 AB×AC < 0，即在 AB 的“右”侧，考虑维护其他每个满足以上条件的点作为 C 点的情况下，可取的 D 点的个数。

- 考虑 B 点每往逆时针方向转一下所带来的影响。

第六关-Flower Dance



由于B点的改变，这个点从可行C点变成D点，给对应极角范围内的C点带来1的贡献

第六关-Flower Dance

- 可以看到：每当有一个点从 C 点变成非 C 就会产生一次单点修改和区间加一，所以可以用线段树维护每个点作为 C 点的可行 D 点个数，每转移一次就统计一下答案即可。

- 复杂度：$O(n^2 \log n)$

第六关-Flower Dance

1h46min 把这个题 AC 了的哥哥队伍的做法大概是容斥，先令全集等于 C(n,4)，然后再把不是 flower 的情况都减掉，大概有四个点构成凸包和其中一个点在三角形一条边上这两种情况。

太强了 Orz

代码链接：https://ac.nowcoder.com/acm/contest/view-submission?submissionId=41083028

还有若干种枚举每个点，极角排序后 O(n) 的做法，跑起来快如闪电，全都 200ms 不到。。

我并说不出话来 QAQ 有兴趣的同学可以自行查看其他的 AC 代码。

第七关-Gemstones

- **签到**
  - 栈

从左到右依次把字符加入栈中，如果某个时刻栈顶的三个字符相同，则将其弹出栈顶并把答案加 1，最后输出一下答案即可。

第八关-How Many Schemes

- **中后期**
  - AC自动机dp+树上倍增

首先根据给定的模板串构架一个 AC 自动机，通过 fail 树求出每个点是否是一个终止状态，然后强制把终止节点的所有字母的出边都指向其本身，这样一旦走到了匹配点就会停留下来。

先考虑如何处理单个询问：可以记 f(w,x)表示走到从 u 走到了 w 号点，并停留在 AC 自动机的 x 状态下的方案数。转移就对于每个状态枚举树边中的每一个字母作为下一个字母，然后在 AC 自动机上对应转移一下并记录到 f[][] 中。最后答案就是所有终止节点 ed 的 f(v,ed) 值之和。

多个询问的话可以把每条边的转移矩阵求出来，然后用倍增维护转移矩阵的累乘结果。记 t 为模板串长度之和，Σ为字符集，那么预处理的复杂度是 $O(nt|\Sigma| + nt^3 \log n)$ 的。然后由于每次询问的时候是向量乘矩阵，所以单次询问的复杂度是 $O(t^2 \log n)$。

然而这样是会 TLE 的，本人费尽九牛二虎之力都只能卡到 4s，当然并不排除有更高深的卡常技巧来 AC。

第八关-How Many Schemes

- **中后期**
  - AC自动机dp+树上倍增

考虑一个优化：对于深度为 x 的点，记 x 的 lowbit 为 2 的 j 次幂，那么我们只用记录 x 往上跳 $2^0, 2^1, \cdots, 2^j$ 这 j+1 个矩阵，然后询问的时候先把 lca 求出来，再用类似树状数组的写法去求一个点到其某个祖先节点的转移矩阵的累乘。但是这样还是可以卡到 $O(nt^3 \log n)$，只要构造一棵很多点的深度都是 1024 之类的树即可。

所以可以考虑往根节点上面加若干个点来避免这种问题，问题在于加多少个点。我们先从最低位开始考虑，如果深度为偶数的点比奇数的多，那么我们可以考虑往根节点上面加一个点，从而保证深度为奇数的点不少于深度为偶数的，同理去考虑其他的二进制位。可知 $O(t^3)$ 的矩阵乘法的次数是 $\frac{n}{2} * 1 + \frac{n}{4} \times 2 + \cdots < 2n = O(n)$ 级别的。

所以总的时间复杂度降到了 $O(n \log n + nt|\Sigma| + nt^3 + qt^2 \log n)$。

加成关-Inner World

- **中后期**
  - dfs序+询问排序+扫描线
- 因为每次种树操作的v都不相同，可以考虑将所有的树都建到一棵树上去表示，每个节点 v 记录一个 $[l_v, r_v]$，即加入这个点的对应操作的 l,r，表示这个节点在哪个标号区间的树中出现。
- 考虑以节点的dfs序为一维，节点上的区间为另一维，那么这个问题就规约成了行加一和矩阵求和，其中可以把一次矩阵求和 $(x_1, y_1) - (x_2, y_2)$拆成 $(1, y_1) - (x_2, y_2)$ 减 $(1, y_1) - (x_1 - 1, y_2)$，离线给所有询问按在dfs序中的位置，即上述的 x 坐标排序后用扫描线+线段树维护答案即可。
- 时间复杂度：$O(n \log n)$。

第九关-Jumping Frogs

- **前期**
  - 组合数学

先不考虑攻击，则跳步方案数可以通过 dp 求出：设 f[i] 为跳 i 的距离的方案数，则有 $f[0]=1, f[i]=f[0]+f[1]+\cdots+f[i-d](i\geq d)$，这个可以用前缀和优化在 O(L) 的时间内求出。

考虑减去被攻击的方案，可知我们需要枚举每次攻击，并减去 (0,0) 走到 (t_i, p_i)的方案数乘以从 (t_i, p_i)走到 L号点的方案数，前面部分可以用组合数算出：$C(p_i - dt_i + t_i - 1, t_i - 1)$，后面部分就是 $f[L-p_i]$。然后考虑把经历两次攻击的方案数加回来，以此类推再减去经历三次攻击的方案数，加上经历四次攻击的方案数……所以我们可以先把所有攻击按照 t 或者 p 进行排序，那么如果经历了一次攻击 k，则这之后肯定不会经历 k 之前的攻击了。所以可以记 g[u][2] 表示当前处于排好序后的第 u 次攻击的状态，前面总共经历了奇数/偶数次攻击的方案数，转移可以暴力转移，最后减去奇数的方案数，加回偶数的方案数即可。

时间复杂度：$O(L+m^2)$

Thanks