2025 联赛测试 1 题解

目录

A 数数（count）

1.1 40pts

小于 10 的每个数的和仅和自己是一对默契数，$N=10$ 时答案为 9 ，其他情况答案为 n 。时间复杂度 $O(1)$ 。

1.2 70pts

考虑暴力枚举 A 与 B，然后每次算出 A 的个位和最高位，以及 B 的个位和最高位，比较是否对应相等。

时间复杂度：$O(n^2)$

1.3 100pts

我们设 $num[i][j]$ 表示最高位是 i，个位是 j 的数有多少个，$co[i][0]$ 表示 i 的最高位，$co[i][1]$ 表示 i 的最低位。这两个数组都可以通过枚举每一个 i 来求到。

统计的时候对于每一个 i，和它能配成默契数的数的个数即为 $num[co[i][1]][co[i][0]]$。

时间复杂度：$O(n)$

B 画画（paint）

2.1 25pts

模拟这个过程并枚举这个副本，暴力计算差。

时间复杂度：$O(k^2nm)$

2.2 40pts

考虑优化计算差的过程，由原来的枚举每个格子变为枚举每行，那么两个副本的差就转变为区间的 3 种关系 (相离、相交、包含)，讨论一下即可。

时间复杂度：$O(k^2n)$

2.3 50pts

考虑继续优化计算差的过程，那么两个副本的差就转变为矩形的 3 种关系 (相离、相交、包含)，(大力) 讨论一下即可。

时间复杂度：$O(k^2)$

2.4 60pts

对于所有的矩形位置都相同的情况，就不用 (大力) 讨论了。

转变一下思路，考虑每个坐标与其他坐标的差，最后对于每个副本求和。

只要记录那个矩形中每个颜色出现了几次。

时间复杂度：$O(k+s)$

2.5 100pts

考虑记录每个坐标上每个颜色出现了几次，并由此算出每个颜色在这个坐标上的贡献。观察每个副本的答案，一定是原图的答案扣去矩形的答案，再加上那个矩形里同一种颜色的贡献。这个部分可以用二维前缀和维护。对于每个坐标上每个颜色出现了几次，需要用到二维差分。

设 $\delta_{i,j} = a_{i,j} - a_{i-1,j} - a_{i,j-1} + a_{i-1,j-1}$，那么把 a_{x_1,y_1} a_{x_2,y_2} 都加上 1，只需要在 的 4 个位置进行修改，最后把 数组二维前缀和还原成 a。

时间复杂度：$O(k+nms)$

C 打字机（typer）

考虑一个性质，令 $h(i)$ 表示 S 长度为 i 的后缀和 T 的编辑距离，则 $i-h(i)$ 单调递增，并且有 $|i-h(i)|\le |T|$。

首先证明单调递增，每次 i 增加 1 的时候，$h(i)$ 至多增加 1，这个很显然，因此单调递增。

其次 $-|T|\le i-h(i)\le |T|$，这个也很显然。

于是我们就可以进行分段函数 dp 了。即 $f(i,j,k)$ 表示考虑了 S 长度为 i 的前缀，T 长度为 j 的前缀，最大的 x 使得 $x-h(x)\le k$。

转移方程仔细思考一下即可，也就是魔改原来编辑距离的 dp。记 g 为暴力编辑距离的 dp

$g(i,j)=\min g(i-1,j)+1, g(i,j-1)+1, g(i-1,j-1)+[S_i\ne T_j]$

于是 $f(i,j,k)=\min f(i-1,j,k)+1, f(i,j-1,k+1), f(i-1,j-1,k-[S_i=T_j])+1$。

初始值需要注意一下，$f(i,j,-|T|-1\cdots -j-1)=-1$，$f(0,0,-|T|-1\cdots -1)=-1$，$f(0,0,0\cdots |T|)=0$，剩下全是 $+\infty$。

这个 dp 的复杂度为 $O(|S||T|^2)$，查询 $[l,r]$ 的话直接找最小的 k 使得 $f(r,|T|,k)\ge r-l+1$ 即可，答案为 $r-l+1-k$，因此单次询问的复杂度可以做到 $O(|T|)$ 或 $O(\log |T|)$。

D 堆（heap）

显然是按位确定的套路，我们考虑确定了前 $i-1$ 个点的值，来确定第 i 个点的值。

首先若前 $i-1$ 个点构成的前缀已经严格比给定的前缀小了，那么当前显然就没有任何限制了，否则当前数字会有一个上界。

假设我们暴力枚举当前数字是 k，于是问题转化为了钦定前 i 个点的数字，求此时合法的方案数。

不难发现现在就是若干个限制，每条限制都是限制子树内的所有数字要大于某个数。

设 s_i 表示被它限制的数的数量 (注意不是子树大小)，t_i 表示限制，我们把所有数字按照 t_i 从大到小排序，那么方案数显然是：

$$\Pi\binom{n-t_i-s_1-\cdots-s_{i-1}-i+1}{s_i}$$

于是我们就获得了一个 $O(n^3)$ 的做法。

优化成 $O(n^2)$ 也十分简单，考虑枚举当前数字的那部分，直接 two pointers 扫描更新乘积即可。