Contest Info

date: 2020-07-18 12:00~17:00

2020牛客暑期多校训练营（第三场）

Solutions

A. Clam and Fish

题目大意

题解

B. Classical String Problem

题目大意

题解

C. Operation Love

题目大意：给你一个“手掌”的图形，可能会被平移和旋转，给的点会按正序或者逆序给出，问你所给的点组成了右手还是左手。

题解

手掌最底部长度是 9，唯一的 9，找一下。大拇指和小拇指就是相邻的，长度不一样，判一下位置，作为条件 1；大拇指的向量与手掌底部的向量叉积，符号作为条件 2；两个条件异或一下，答案。

D. Points Construction Problem

题目大意：在平面上，开始时所有整点都是白点，你需要将其中 n 个点染黑，使得相邻的（四联通）黑白点对数量为 m

题解m 必然为偶数，可以归纳证明m 至多为 $4n$而要使 m 小，注意到图形内部不应存在空行和空列，否则将两边合并起来显然不坏。这样一来，注意到答案数至少为黑点外接矩形的周长。因而最小值就是按螺旋线来摆放，或者说拼成长、宽尽可能接近的图形。

理论分析完后，并不需要真的去讨论、构造。由于数据范围很小，可以事先打表。首先枚举一个 L 形，然后在它上面依次摆放黑点——这不会改变周长，但是可以调节黑点数量。此外还需要一些散点以向最大值靠近。实践证明，`ac is ok`

E. Two Matchings

题目大意：给偶数个 a_i 需要进行配对，记第 i 个元素与第 p_i 个配对，称 $\{p_i\}$ 为配对方案。配对后记配对的代价为配对的两个元素的绝对差的总和。现在要你求出两个不同的配对方案，对每个 i 要满足 $p_i \ne p'_i$ 使得两个配对方案的代价和最小。

题解 DP

首先，我们肯定会先考虑给 a_i 排一下序，不会影响到配对。假设我们已经有最优的两个配对方案了，我们把配对关系看成边，两个配对方案放在一张图上，那么就会出现若干个连通块。连通块内边数和点数一样，且度数均为 2，所以是个环。环上的边是方案 A 方案 B ……交替着出现的。

考虑一下答案的物理意义，对于排序后相邻的两个元素 a_i 和 a_{i+1} 他俩的绝对差的贡献。如果我们划一刀将 $a_1, \ldots, a_i$ 和 $a_{i+1}, \ldots, a_n$ 切分开，那么被切到的边的数量，就是 $a_{i+1} - a_{i}$ 对答案的贡献。而被切到的边数也必然是偶数个。

那么对于确定的一个连通块，答案的下界显然就是最大值减最小值的差的两倍，且很容易能构造出来这个方案（按顺序连起来，然后最后最大和最小连一下），能够取到答案的下界。

接下来考虑将连通块内位置最大的 4 个或 6 个取出来，记这些元素为 A 组，剩下的元素记为 B 组。我们将 B 组中与 A 组相连的边随便接起来，答案显然不会变大。而 B 组又总有办法弄到只有这几个元素组成的答案的下界的方案。又由于只用 4 和 6 就能组合出所有可能的元素数量，因此取到最优的答案的前提下，连通块的大小总有办法限制为 4 或 6 个元素。

类似上面的证明，也可以证明一个连通块必然对应到了排序后序列的一段区间。

所以很简单 DP 一下，取当前末尾的 4 个或 6 个，转移一下就好。记得 dp[0] = 0, dp[2] = inf

F. Fraction Construction Problem

题目大意

题解

G. Operating on a Graph

题目大意

题解

H. Sort the Strings Revision

题目大意 $s_0 = 01234567890123\ldots$ s_{i+1} 通过将 s_i 的第 p_i 个字符换成 d_i 来生成 p_i 是个排列。求给这些字符串排序后的排名。

题解：笛卡尔树

想一下第一次替换第 0 个字符时，如果变大了，那么就相当于确定好了某个时刻，接下来生成的字符串必定比前面的大。然后可以递归着考虑排序的过程。相当于第 i 个字符如果有大小变化，那么我们就把下标的区间切开来，然后标一下大小顺序。

由于 p_i 是个排列，所以每个位置也只会被切一次。如果某个位置没变化，那实际上大小关系不切就是了，那最后肯定会有几个区间没有被切过，也显然下标在区间内的字符串都是一样的，按标号时根据题意按下标大小排一下。

最后我们整个递归、切区间的过程，可以看成是一棵二叉树，所以如果我们这颗二叉树出来了，题就活了。

我们队现场没过，但是复杂度和题解是一模一样的。由于给的 p_i 是排列，记一下 $pp_{p_i} = i$ 就好，我们按 p_i 从大到小，去合并区间，由于每个分界点只会被合并一次，所以可以 $\mathcal{O}(1)$ 的时间内合并，合并过程中创建节点、记录二叉树的信息。然后这么一个 $\mathcal{O}(n)$ 的东西它就是过不了。

正解怎么做呢，考虑一下 $\mathcal{O}(n)$ 为 p_i 构造笛卡尔树，不给没带来变化的 p_i 做。笛卡尔树的根也就是 p_i 中的最小值，也就是相当于第一次切开区间的地方，递归着下去。所以实际上给笛卡尔树的最后的叶子再加俩节点，就和之前说的切区间切出来的树一模一样了。

我们比正解差在什么地方呢。数据是随机的，我们合并区间的两个端点也是随机的，疯狂 cache miss所以合并区间的常数很大。刚好卡不过去。

I. Sorting the Array

题目大意

题解

J. Operating on the Tree

题目大意

题解

K. Eleven Game

题目大意

题解

L. Problem L is the Only Lovely Problem

题目大意

题解