2024牛客暑期多校训练营#4 题解

A. LCT

题意

给定一颗有根树，每次询问前i条边组成的森林中，已第c_i个点为根的树的深度。

题解

1. 带权并查集维护每个点到当前根节点的距离 $deep_u$
2. 每个根节点维护一个当前深度$f(u)$
3. 当连接u, v，其中u为父亲时，设$root_u$为此时u节点的根，那么可以更新此时的$f(root_u)$
   $f(root_u) = \max(f(root_u), deep_u + f(v) + 1)$

如上可以在线维护答案，时间复杂度为并查集的复杂度，空间复杂度O(1)。

B. Pull The Box

题意

有一个无向图，箱子在1号点，人在x号点，人要把箱子拉到n号点，问最少需要几步。人不允许跨过箱子。

题解

直接讨论是比较困难的，有很多case，例如人和终点在箱子两侧，人的那侧是否有环，有环的时候如何最优...因此需要一种一般化的方法

- 结论1：首先有一个很显然的结论：在最优方案中，人不会走回头路。否则人可以直接不走这一段路，显然更优。
- 结论2：人一旦拉上箱子就不松手，不会让答案变得更劣。因为箱子只限制了人不能走回头路，而结论1说明了人不会走回头路。

由结论2，整个问题可以分成两部分：

1. 人单独走，直到与箱子接触
2. 人拉着箱子走到终点

不会出现 '单走-拉-单走-拉' 的情况。下面是两个部分的具体处理方法

- 单走：这种情况比较简单，直接bfs即可，注意箱子不可以跨过，得到dis_p。
- 拉箱子到终点：由于人-箱不会分开，即人和箱子是相邻的两个点，因此{人，箱}可以看作一条有向边，于是可以直接用有向边做bfs，得到dis_e。

  这里还有一些细节需要考虑：

  - 考虑一个车轮图，中心点为o，每次从其他点指向o点的边都要遍历一次o的所有临点，复杂度退化为n^2。
  - 注意到，第一次走到指向o点的边$u-o$后，只有其反向边$o-u$不会被更新，其他的$o-v$都会被更新，而下一次访问o时一定会更新$o-u$。
  - 因此每个点最多被访问两次，用一个vis_p数组记一下即可。

考虑到人第一次接触箱子的点不确定，可以以终点为起点做反向bfs，最终对1号点的所有邻点的两个dis距离求和取min。

C. Sort 4

题意

给定一个排列，每次选择四个位置任意交换其中的元素，求排序的最小次数。

题解

考虑$i \to p_i$构成的图中的环，对于长度为len的环

- $len = 3,4$可以一次排序好
- 两个$len = 2$的环可以一次排好
- 对于$len > 4$的环，每次操作可以排好其中三个元素，环长变为$len - 3$

因此记录有多少个环最终长度为2，将其两两合并处理即可。

D. Plants vs. Zombies (Sunflower Edition)

题意

有两种向日葵，你每回合可以花阳光种植各至多一棵。种在场上的向日葵每回合生产阳光。你需要最大化 t 回合后的阳光量。

向日葵的价格和每回合产量不超过 1023。

回合数和初始阳光不超过 2E7。

分析

看起来像是 Easy 贪心题，实则不然。阳光存量与每回合增量无法合并到一起优化，因此计算最优解时必须同时保留两个信息。

注意到阳光花费和产量的值域很小，考虑设计状态进行 DP。DP 的复杂度和回合数有关，还要分析如何处理 2E7 的回合数。

题解

观察一：如果每回合阳光增量达到两种向日葵的价格之和，实现阳光自由，后续的决策就很平凡了。可以直接计算得出每种向日葵应持续购入至哪个回合截止。

观察二：需要维护的回合数不多。在最坏情况下，每种向日葵价格 p = 1023 且产量 1，只要持续种植向日葵，依然能在约为调和级数的回合内实现阳光自由。

因此 DP 的回合到 2p ln(p) 的数量级就很充裕了。

设计状态：第 i 回合，单回合阳光产量为 j 时，阳光存量的最大值为 f(i,j)。回合数 i 的取值如前文所述。阳光产量 j 的上界只需考虑到两种向日葵价格之和，超出上界可直接计算 t 回合时最终答案。

考虑一种劣的情况：在 x 回合时阳光足够种植向日葵 1 但不种植，却在 x + 1 回合种植。这样显然亏了一回合的产量。换言之，对于最优策略而言，在确定下一次种植的向日葵后，一定尽早购入。

因此容易想到刷表转移。已知当前 i 和 j，枚举下一次种植时选择的向日葵集合，计算出最早哪一回合有足够阳光，刷表转移过去。注意一回合可同时种植两种向日葵，转移时要考虑到这种情况。

为了在 DP 过程中阳光产量过界时获得最终值，可以先 O(t) 预处理，也可以通过简单分类讨论 O(1) 计算。

本题的预期是一道有些反直觉的题目。经准确判断，可以用并不复杂的算法解决本题。

E. String God's String

约定字符串是从下标 1 开始的。

首先对 t 串建出 kmp 自动机。具体而言，是求出这样一个数组：$\text{kmp}[i][0/1]$ 表示在 t 的第 i 个位置后面放 $'a'$ 或者 $'b'$ 最长会匹配到多长的前缀。求法也很简单，要么 $\text{kmp}[i][c] = i + 1$，要么 $\text{kmp}[i][c] = \text{kmp}[\text{next}[i]][c]$，其中 next 是 kmp 过程中求出的 next 数组。

想象这个自动机对应的图，可以发现除了最后一个位置之外，每个点连了两条边：一条连向下一个点，一条连向前面的某个点。假设对于 i 来说，有 $\text{kmp}[i][c] \le i$，这样就有 $(i, \text{kmp}[i][c], \text{kmp}[i][c] + 1, \text{kmp}[i][c] + 2, \ldots i)$ 这样的环。根据题意，当 i 走向了 $\text{kmp}[i][c]$ 后，必须走这个环回到 i。

那么问题就变成，选择若干个这样的环，让它们的长度之和等于 $n - |t|$。同时根据题意，每个环只能走一次，而且显然走完了 i 的环才能走 $i + 1$ 的环。

于是我们就可以将这个问题抽象成一个 01 背包，第 i 个物品的重量为 v_i，$v_i = i - \text{kmp}[i][c] + 1$，求选出若干物品使得总重量为 $n - |t|$ 的方案数。每个物品的生成函数为 $1 + x^{v_i}$，所求即是 $\prod_{i=0}^{n-1}(1 + x^{v_i})$。

取一个 $\ln$，最后再 $\exp$ 回来。取 $\ln$ 就是 $\sum_{i=0}^{n-1} \ln(1 + x^{v_i})$，将每一项展开，得到 $\sum_{i=0}^{n-1} \sum_{j=1}^{+\infty} (-1)^{j-1} \dfrac{x^{v_i j}}{j}$。这个式子可以直接 $\mathcal{O}(n \ln n)$ 求，最后再 $\exp$ 回去。

时间复杂度 $\mathcal{O}(n \log n)$。

F. Good Tree

题目大意

对树上节点 u 定义 $f(u) = \Sigma_v dis(u, v)$，给定 x 求满足“存在两点 u, v 使得 $f(u) - f(v) = x$”成立的树最少有多少个节点。

做法

- 先给出答案：记 $sq = [\sqrt{x}]$
- if $n = sq \times sq$，答案为 $2sq + 1$ （结论一）
- else if $n > sq \times (sq + 1)$，答案为 $2sq + 3$ （结论二）
- else （此时 $sq \times sq < n \le sq \times (sq + 1)$）：若 $n - sq \times sq \equiv sq \mod 2$ 则答案为 $2sq + 2$ 否则答案为 $2sq + 3$ （结论三）

证明

- 首先考虑固定了 u, v 两点和它们之间的路径长，如何放点能最大化 $f(u) - f(v)$：不难发现一定是要把这些点放在 v 的子树里（远离 u 的方向），且只要是在子树里无论具体怎么放，贡献都是 $dis(u, v)$
- 那么，我们考虑奇数个点，$2k + 1$ 个点，最大能多大？由均值不等式和上述分析，可以得出，最大是 k^2，且方案是：
  1. u 到 v 的距离为 k;
  2. v 的子树里有另外 k 个点
- 对于偶数个点，$2k + 2$ 个点，最大能多大？同理，最大是 $k(k + 1)$
- 再下一个奇数，$2k + 3$ 个点，最大是 $(k + 1)^2$ 个点。所以我们发现，只要能分析清楚落在区间 $[k^2, (k + 1)^2]$ 里的 x 的答案都是什么，就做到不失一般性了，下面开始分析
- 我们用 $ans(x) = y$ 表示输入 x 的时候答案是 y
- 根据上页内容首先可以得出：$ans(k^2) = 2k + 1,\ ans(k(k + 1)) = 2k + 2,\ ans((k + 1)^2) = 2k + 3$ 所以结论一是对的
- 对于区间 $[k^2 + 1, k^2 + k]$，我们知道其答案一定大于 $2k + 1$（至少 $2k + 2$）；对于区间 $[k^2 + k + 1, (k + 1)^2]$，其答案一定大于 $2k + 2$（至少 $2k + 3$）
- 对于区间 $[k^2 + k + 1, (k + 1)^2]$，可以构造出一定可以做到 $2k + 3$ 的方法，结论三成立
- 对于区间 $[k^2 + 1, k^2 + k]$，不一定总能做到 $2k + 2$，不能的话就需要 $2k + 3$，手玩几个样例可以发现这与 k 的奇偶性有关，结论二成立

- 上面这俩具体为何结论二和奇偶性有关、结论三为何对，写起来太麻烦了、也不好懂，反而是手玩一下就明白了，所以大家手玩一下输入 4 到 16 时的答案就理解了。这里附上 4 到 16 对应的答案： (4, 5) (5, 7) (6, 6) (7, 7) (8, 7) (9, 7) (10, 8) (11, 9) (12, 8) (13, 9) (14, 9) (15, 9) (16, 9)

G. Horse Drink Water

题意

经典将军饮马问题，但是河变为x正半轴和y正半轴。

题解

签到，分别按x, y轴求答案取个min即可。

H. Yet Another Origami Problem

题目大意

给两种操作：排序数组后在值域上翻折一个前缀或翻折一个后缀，问任意次操作能做到的最小极差

做法

- 先直接给出答案。将输入数组 a 排序，则答案为 $g = gcd(a_2 - a_1, a_3 - a_2, \dots, a_n - a_{n-1})$
- $n = 1$ 时答案为 0，所有数字都一样时答案也为 0（这个应该不需要写特判，但如果写丑了的话可能就需要特判）

证明

- 下面给出证明
- 首先容易说明答案不会比 g 更小：考虑差分数组 $d_i = a_{i+1} - a_i \quad (i \le n-1)$，则初始 d_i 都是 g 的倍数，因此每个数都可以表示成 $a_i = a_1 + k_i g$ 的形式。那么在一次操作之后，容易说明每个数依然可以表示为 $a'_i = a_1 + k'_i g$ 的形式，不难看出他们排序后求相邻项之差的 gcd 仍然形如 kg，不会更小
- 接下来说明总能达到 g，即若当前极差还不是 g，我们总有一种操作能将极差减小：计当前数组先去重再排序得到的 k 个数的数组是 $b_1, \dots, b_k$，$k \ge 3$ 时选择下标 $p = k - 1$ 执行操作二，一定可以减小极差。
- $k = 2$ 怎么办：此时已经做到极差是 g 了，不用再操作了。为什么此时极差一定是 g 而不是 $kg\ (k > 1)$？这是因为，也容易证明上述操作完全不改变操作前后的 g，上一页已经说明了不会更小，现在只要说明不会更大：
-   - 操作前：$g = gcd(b_2 - b_1, b_3 - b_2, \dots, b_k - b_{k-1})$
-   - 操作后：$g = gcd(b_2 - b_1, b_3 - b_2, \dots, b_{k-1} - b_{k-2})$，少了一项 $b_k - b_{k-1}$，但其实操作后 $b_{k-1} - b'_k = b_k - b_{k-1}$，所以这项其实也还在，因此 g 不会更大

I. Friends

注意到如果 $[l, r], l < r$ 为好区间，则 $[l, r-1], [l+1, r]$ 也会是好区间。考虑每次对固定左端点求出最大的右端点。设 $[l, r]$ 是好区间，且 $[l, r+1]$ 不是好区间，则 $[l+1, r]$ 一定是好区间。于是对于 $l+1$ 为左端点的区间，我们只需要考虑 $r+1$ 是否与 $[l+1, r]$ 是好朋友，以此类推。$r+1$ 是否与 $[l+1, r]$ 是好朋友这个信息暴力从 r 开始往前检验即可，时间复杂度 $\mathcal{O}(m \log m)$。
两个端点移动都是单调的，所以移动端点是 $\mathcal{O}(n)$。
总时间复杂度 $\mathcal{O}(m \log m + n)$。

J. Zero

设 X_i 为一随机变量，其中 $P(X_i = p)$ 表示 i 为右端点时，连续 1 的数量为 p 时的概率。设 $f_{i,j} = \sum_{p \ge 0} P(X_i = p)(\sum_{l=1}^{p} l^j)$，也即如果 $s_i = 0$ 时，我们将 $f_{i-1,k}$ 加到答案里；或者 $s_i = ?$ 时，我们将 $f_{i-1,k}/2$ 加到答案里。
接下来就是递推 f。
如果 $s_i = 0$，则 $P(X_i = 0) = 1$，此时 f 容易得到。

如果 $s_i=1$，则 $P(X_i=p)=P(X_{i-1}=p-1)$。注意到 $f_{i-1,j}=\sum_{p\ge 0}P(X_{i-1}=p)\sum_{l=1}^{p}l^j$。
又

$$f_{i,j}=\sum_{p\ge 1}P(X_{i-1}=p-1)\sum_{l=1}^{p}l^j=\sum_{p\ge 0}P(X_{i-1}=p)\sum_{l=1}^{p+1}l^j=\sum_{p\ge 0}P(X_{i-1}=p)\sum_{l=1}^{p}l^j+\sum_{p\ge 0}P(X_{i-1}=p)(p+1)^j$$

前半部分是 $f_{i-1,j}$，后半部分使用二项式定理。即

$$\sum_{p\ge 0}P(X_{i-1}=p)\sum_{l=0}^{j}\binom{j}{l}p^l=\sum_{l=0}^{j}\binom{j}{l}\sum_{p\ge 0}P(X_{i-1}=p)p^l=\sum_{l=0}^{j}\binom{j}{l}\mathbb{E}[X_{i-1}^l]$$

于是如果可以求出 $\mathbb{E}[X_i^j]$，则上述式子可以快速求出答案，而很明显这也是可以递推的。即考虑 $E[(X_{i-1}+1)^j]$，一样的二项式定理拆开，则可以每次 $\mathcal{O}(k^2)$ 递推出所有 $\mathbb{E}[X_i^j]$ 了。
$s_i=?$ 同理。
时间复杂度 $\mathcal{O}(nk^2)$。
Bonus: 看起来是可以做到 $\mathcal{O}((n+k)\text{polylog})$ 的。

K. Calculate the Expression

题目大意

给一个只含加和乘运算的表达式，每次修改表达式的一个字符，或对一个子串求值，输出答案模998244353。

题解

本题是一道码量题，主要考察做题人的代码能力，花费一定时间即可完成。

容易想到，使用线段树维护单点修改和区间查询。线段树需要两段信息能够合并，因此，我们考虑一个表达式的基本类型，发现有一个数、仅含有乘法以及含有加法三种类型，分别考虑这三种类型的子表达式在拼接合并的时候发生的信息变化。

一个数的情况最简单，无论是和哪种表达式进行拼接，均只需要考虑表达式的第一个数或最后一个数拼接这个数的情况即可。

仅含有乘法的表达式在拼接过程中实际上发生的变化是，被拼接的表达式第一个数或最后一个数进行了数的拼接，随后对应的一项再整体进行了乘法。

最后是含有加法的表达式，这种表达式在拼接过程中，实际上相当于第一项或最后一项以乘法表达式的形式进行拼接，随后再加上该表达式的剩余部分。

总结以上拼接方式，对于一个数的情况，我们维护该数的长度和数值；对于仅含有乘法的情况，我们维护这个表达式首尾两个数以及中间其他数的乘积；对于含有加法的情况，我们维护表达式第一个数，第一项其余数的乘积，最后一个数，最后一项其余数的乘积，以及中间其他项的和。使用这些信息，我们就可以维护在表达式的首位拼接其他表达式后的信息变化情况。

具体的实现细节只需要将表达式以上各部分的信息拆开后花时间处理即可，有点繁琐但思维难度不高。

L. Deceleration

题意

车的初始速度为1，处理以下两种询问

- 添加一个时间点x，使车在x时间点速度减半
- 询问一个距离x，假设车第0s从0重新出发，走到x要多久

题解

发现无论以时间为轴维护距离，还是以距离为轴维护时间，都会遇到一个问题：取模之后无法比较大小，也就没法在平衡树里操作；线段树的深度会退化到O(n)。

因此需要维护其他东西。

- 定义“等效距离”，初始的剩余等效距离为x，每秒等效距离减1，那么每次速度减半就等价于剩余等效距离$\times 2$。
- 构造一个时间为下标线段树，维护三个值：
  - k: 区间减速次数
  - x: 至少需要多少等价距离才可以通过这个时间区间
  - r: 以x（即最少需要的等价距离）的等价距离通过这个区间之后，还剩多少等价距离

注意，此时已经没有速度的概念了。减速已经转化为了等价距离$\times 2$

这样的线断树就足以维护添加、合并、查询了。但是还有一些细节需要讨论：

- 我们还需要维护r是否已经被取模过了
- 由于减速时间点最大为10^9，因此一旦等价距离被取模过$(> 10^9)$，则一定会通过所有时间节点。也就是s一定小于10^9。
- 是否取模会有一些简单但繁琐的讨论，可以参考std代码