第 19 届浙大城市学院程序设计竞赛题解

ZUCC ACM Group

Zhejiang University City College

April 10, 2022

A. Who is The 19th ZUCCPC Champion

签到题，输出一行长度 100 以内仅由数字、大小写字母、空格组成的任意字符串即可。

B. Jiubei and Overwatch

签到题 2，仅需要考虑血量最高的敌人 a_{max}，分 $a_{max} \leq kx$ 和 $a_{max} > kx$ 两种情况考虑即可。

C. Ah, It's Yesterday Once More

- 可以发现算法 1 每次执行 i 时，[1, i-1] 都必定有序且 $a_{i-1} = n$，所以相当于将 a_i 向左冒泡到其应该在的位置
- 对于冒泡排序，其交换次数 = 逆序对数量
- 可以发现若构造的排列 $a_1 = n$，两个算法的交换次数都是其逆序对数量
- 若 $a_1 \neq n$，则算法 1 执行完 $i = 1$ 的轮次后，整个序列的逆序对增加，再加上 $i = 1$ 时的交换次数，算法 1 的交换次数一定更大
- 因此随便输出一个首位为 n 的排列即可

D. Reflectilon

- 由于光线只能从四个方向射到镜子，因此可以通过 vector+ 二分预处理出光线从方向 j 射到第 i 号镜子后会反射到的位置 $to_{i,j}$
- 对于每个询问，可以直接进行记忆化搜索
- 也可以先通过倍增处理出从每个镜子开始最后到达的镜子编号，即 $to_{ij} = to_{to_{ij},j}{'}$，其中 $j^{'}$ 是方向 j 发生反射后的方向

E. Disjoint Path On Tree

- 不相交路径对数 = 总路径对数 − 相交路径对数
- 当树上两条路径相交时，某条路径的 LCA 必为交点
- 枚举点 i 作为一条路径的 LCA，则此时的相交方案数为经过 i 且 LCA 不在 i 的路径数 $\times$ LCA 为 i 的路径数 +LCA 都在 i 的路径对数

F. Sum of Numerators

- 将所有数表示为 $2^a \times b$，最终要求的答案即为 $\sum_{i=1}^{n} 2^{\min(a_i,k)} \times b_i$
- 由于 $[1, n]$ 内能被 t 整除的数有 $\lfloor \frac{n}{t} \rfloor$，令 $S(n) = \frac{n(n+1)}{2}$，答案初始为 $S(n)$，我们可以对于所有的 $i \in [1, k]$，令答案减去 $S(\lfloor \frac{n}{2^i} \rfloor)$ 即可消去多余的贡献

G. Guaba and Computational Geometry

- 若两个矩形不相交，则将两个矩形投影到 x 轴或 y 轴上形成的两条线段不相交
- 将所有矩形和 x, y 轴平行的边分别拆出来排序，记录前缀最大权值更新答案即可

H. Distance

- 考虑某个时刻，使答案最小的 x 一定满足 x 左侧 r 的数量等于右侧 l 的数量（如果数量不相等，往多的那一块靠近会使答案减小），x 的意义即为中位数
- 因此用两个堆维护中位数和左右对答案的贡献即可

I. Array Division

- f_i 表示前 i 个数可以划分的最多段数，则有 $f_i = fj + 1$ 当且仅当 $\sum_{k=j+1}^{i} a_k - b_k \geq 0$，答案即为 f_n，复杂度 $O(n^2)$
- 令 $c_i = \sum_{j=1}^{i} a_j - b_j$，不难发现我们只需要求出从非负的位置开始且以 c_n 结束的的最长上升子序列的长度，经典算法，复杂度 $O(n \log n)$

J. Substring Inversion (Easy Version)

大体上有两种做法：

- 取出所有的子串（n^2 个），带上子串的起点按照字典序 sort，对于子串 s_i，$[s_1, s_{i-1}]$ 中起点在 s_i 右边的会产生贡献，维护各起点上有多少个子串即可，时间复杂度 $O(n^3 logn)$，常数比较小，可以轻易跑过去
- 枚举一对后缀 s_i 和 s_j，考虑所有 $a = i, c = j$ 的四元组的数量，方便起见另 $x = n - i + 1, y = n - j + 1, lcp = lcp(s_i, s_j)$
  - 若字典序 $s_i < s_j$，则贡献为 $\sum_{k=1}^{lcp} x - k = xlcp - \frac{(lcp+1)lcp}{2}$
  - 若字典序 $s_i > s_j$，则贡献为 $\frac{(lcp-1)lcp}{2} + (x - lcp)y$

lcp 可以直接暴力求，时间复杂度 $O(n^3)$

K. Substring Inversion (Hard Version)

本题思路和 easy version 的做法 2 类似：

- 首先将所有后缀排好序，这一步可以用 SA 来实现，也可以用二分哈希进行字典序比较然后直接 sort 来实现，后者是 $O(nlog^2 n)$ 的，但是没有特意去卡，经验题人验证常数比较小的话也可以跑过去
- 处理出 height 数组，$height_i = lcp(s_{i-1}, s_i)$（此处 s_i 指字典序从小到大第 i 个后缀），有 $lcp(s_i, s_j) = \min_{k=i+1}^{j} height_k$
- 方便起见还是用 x 表示后缀 si 的长度，y 表示后缀 sj 的长度，首先考虑字典序 $s_i < s_j$ 的情况，s_i 的贡献为 $x \sum lcp - \sum \frac{(lcp+1)lcp}{2}$ [s_j 在 s_i 的右边]，考虑 segment tree beats，相当于每次对 $[1, n]$ 区间与 $height_i$ 取 min，维护区间平方和以及区间和
- 字典序 $s_i < s_j$ 的情况类似，$x \sum y - \sum ylcp + \sum \frac{(lcp-1)lcp}{2}$，多记录一个最值的 $\sum y$ 即可

K. Substring Inversion (Hard Version)

另外贴一份验题人的后缀树做法

先建出后缀树。

对于同一个节点代表的x个字符串，有以下几种答案贡献方式:

1)

节点u和节点v，u的字典序>v的字典序，但是edp[u]<edp[v]。

这里$(len[u]-len[lca])\times(len[v]-len[lca])$作为第一部分答案。

对于这部分答案可以拆成$len[u]\times len[v]-(len[u]+len[v])\times len[lca]+len[lca]\times len[lca]$

这三部分答案都可以通过线段树合并快速求出。

2)

对于lca以上的子串集合，只要子树内出现一对edp不同的位置u和v，并且lca(u,v)等于当前子树的根x，那么答案就加上$len[x]\times(len[x]-1)/2$。
这里子树内edp不同的位置u和v的数量，也可以通过线段树合并快速求出。

3)

对于子树内任何一对edp不同的位置u和v，且满足edp[u]<edp[v]，

$(len[u]-len[lca])\times len[lca]$也是答案的一部分。

可以拆成$len[u]\times len[lca]-len[lca]\times len[lca]$。

这部分答案也是可以通过线段树合并求出来的。

L. Monster Tower

- 答案具有单调性，可以二分答案然后用堆维护最下面 k 层的最小值判断是否能把怪杀完，复杂度 $O(n\log n\log W)$
- 注意到每次只选取的最小值的序列是唯一的，因此可以直接用堆维护最下面 k 层的最小值，如果最小值大于当前能力值，则将答案加上当前能力值和堆顶之差，复杂度 $O(n\log n)$

END