2021 CSP 7 连测 day2

Xiejiadong

Sep. 2021

IP 地址

其实我们可以完全假设给出的 IP 是不合法的

唯一合法的时候，就是我们按照规则重新组织 IP 后，输入的字符串 = 输出的字符串

这样想以后，处理起来就容易得多，因为题目保证有 4 个被分割的数字，那就先找出这 4 个数字。技巧是找数字的时候不要用 int 来保存，直接用 $String$ 来保存

找到 4 个 $String$ 以后，我们只要纠正 (如果存在) 4 个 $String$ 的错误，再用 “.” 连接起来就行。

接下来就是如何纠正：

1 去掉前导 0，需要注意的地方是**只有一个 0 的时候是不能去掉**的

2 判断数字是否大于 255，显然如果 $String$ 的大小超过 3，就直接改成 255，不然就判断 String 代表的数字是否大于 255 就行

字符串

30 pts

用 if 判断然后手算即可，一共 16 种方案。

60 pts

用各种搜索算法或者随机性算法都可能拿到不错的分数。

100 pts

本题的关键在于，如果想要删除 A，那么必须是以 AP 的形式。不难发现尽可能删除 AP 一定会使答案更优（因为 A 会把一连串的 P 隔开）。

基于“A 优先删除”的思想，本题就变成了一道栈的模拟题：不断将 A 入栈，如果遇到 P 就将 A 出栈（即匹配到了一对 AP），最后一定会变成 P...PA....A 的形式，最后尽可能删除 P 即可。

继承类

首选对于类名的处理，需要考虑是不是出现过这个问题，我们可以简单的用 map<string,int> 或者 unordered_map<string,int> 来处理。这样就能够处理对于第一个条件的情况判断了。

对于其他两个子任务，我们需要考虑类层次结构的有向图中的路径。假设我们正在考虑一个新的声明“$K : P_1\ P_2 \cdots\ P_K$;” 并在添加时首先尝试确定菱形 A、B、X、Y 何时形成。

首先，显然它必须满足 $K = B$。此外，必须存在一条从 X 到 K 和从 Y 到 K 的路径，所以假设 X' 和 Y' 是这些路径上紧接在 K 之前的类。因为它们是路径到达 K 之前的最后一个类，所以 X' 和 Y' 需要是类 $P_1\ P_2 \cdots\ P_K$ 之一。注意到类 X' 和 Y' 不能从另一个派生，例如，当 X' 从 Y' 派生时，那么 A, X, Y , X' 是一个在添加新声明之前就存在的菱形。

因此，在添加新声明时，检查以下内容就足够了：在 K 继承的类中是否存在两个类 P_i 和 P_j 满足它们不是彼此派生的，并且都是从某个类 A 派生的。

我们可以计算 R_K，表示类 K 继承的所有类祖先的集合（相当于在图中的前缀类）。如果声明是不合法的，当且仅当在 K 中存在继承的类中存在两个类 P_i 和 P_j 使得他们满足以下条件：

- $P_i \notin R_{P_j}$（P_j 不是从 P_i 派生的）
- $P_j \notin R_{P_i}$（P_i 不是派生自 P_j ）
- $R_{P_i} \cap R_{P_j} \neq \varnothing$（$P_i$ 和 P_j 都源自某个类 A）

这样直接处理的时间复杂度 $O(n^3)$，但可以通过 bitset 之类的，可能也能优化到 100 分。

考虑到实际上就是检查是否存在满足上述条件的类 P_i 和 P_j。除此之外，需要注意的是，如果 P_a 是从 P_b 导出的，那么 P_b 甚至不需要考虑。对于每个声明，我们执行以下操作：

1 对类 $P_1\ P_2 \cdots\ P_K$ 进行排序，按照声明的顺序

2 维护集合 R，该集合对应于我们迄今为止考虑的类 P_i 的所有 R_{P_i} 的并集。集合 R 可以存储为 n 个布尔值的数组

3 对于每个类 P_i

  a 如果 P_i 已经在 R 中，忽略

  b 否则，将 R_{P_i} 中的所有元素添加到 R。如果我们在添加时遇到 R 中已经存在的元素，那么我们发现了一个菱形并且声明被驳回

对于每个声明，处理步骤的数量与添加到 R 的元素总数成正比，因此它受 n 的约束。

时间复杂度为 $O(n^2 \log n)$。

子图

首先可以发现，如果不考虑连通性问题，$(k+1)$-degree 一定是 k-degree 的子图

如果一个点属于 $(k+1)$-degree 而不属于 k-degree，我们可以把这个点标记为 $k+1$

所以我们可以考虑找到 k 最大的 k-degree，然后考虑不断的拓展这个子图，去得到 k 比较小的 k-degree

显然每一次加入一批相同标记的点，可以拓展得到 k 减小的 k-degree

假设我们已经知道了当前图的 $score$ ，我们考虑新增点/边会带来的 $score$ 变化

我们首先给所有的节点一个 $rank$

- $rank(v) > rank(u)$ 当且仅当
  - v 的标记 $>$ u 的标记
  - v 的标记 $=$ u 的标记且 $v > u$
- 这个部分可以用 bin-sort 完成

我们可以根据边两边点的 $rank$ 对边分类，用 $E(v,>)$ 表示 v 的边中另一个端点比 v $rank$ 大的边，其余的表示以此类推

于是，考虑新增一批点（标记相同的点）：对于每一个新家瑞的点 u：

- $\Delta n = 1$
- $\Delta m = |E(u,>)| + \frac{1}{2}|E(u,=)|$
- $\Delta b = |E(u,<)| - |E(u,>)|$

于是从标记大的点到小的点依次加入计算 $score$，得到最大 $score$ 的 k 即可

因为不断加入点，可能使得本来不连通的两个子图联通起来，这部分可以用并查集维护

时间复杂度 $O(m+n)$