上海大学
Shanghai University

第 18 届上海大学程序设计联赛夏季赛

试题分析

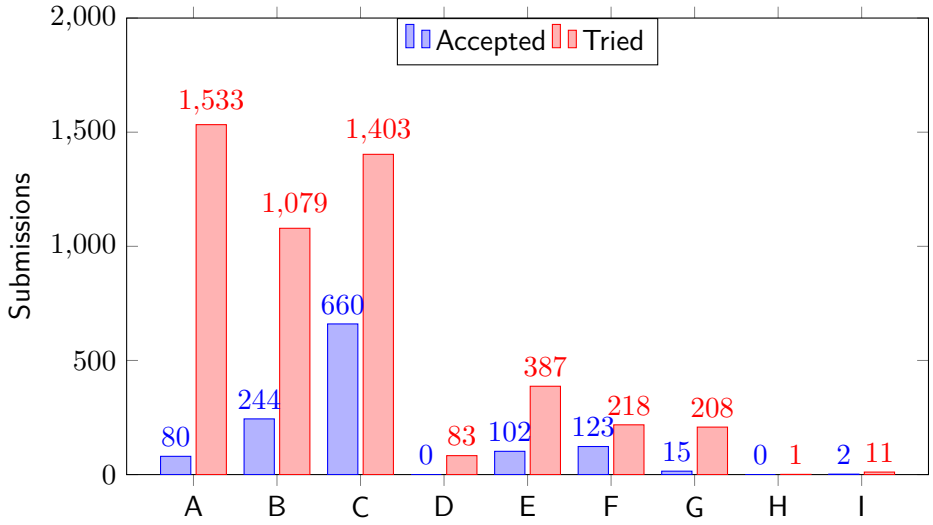
compute cubercsl ybmj
LIN88 Lemon_412

2020 年 8 月 5 日

难度预估

- Very easy: C
- Easy: B, E
- Medium: A, F
- Medium hard: G, I
- Hard: D, H

通过情况


Accepted
Tried
Submissions
2,000
1,500
1,000
500
0
1,533
1,403
1,079
660
387
244
218
208
80
83
102
123
15
0
1
2
11
A
B
C
D
E
F
G
H
I

C. 爵士

Tag

implement, sabit

题解

- 简单的实现题，本场唯一指定签到题。
- 只要会基础的语法和浮点数的输出方法即可通过。
- 出题人费了好大劲想出来的爵士 Sabit。

首次通过：华东师范大学 — 汪杰 0:03(+)

B. 分子

Tag

implement, stack

花絮

- 考虑到没有学过编译原理的小朋友，题目难度进行过适当删减；
- 但所有裁判代码都是可以解决括号嵌套问题的；
- 如果您的代码无法解决括号嵌套，可以尝试改进自己的代码。
- 题干中的 CHTHOLLY 拼错了，虽然不影响选手答题。

B. 分子 （续）

题解

- 观察到该文法是 LL(1) 的，可以通过递归下降分析法分析。
- 对于已经学过编译原理的同学来说，这题相对简单。递归下降正是编译原理课程的实验之一。如果不会做，建议重修。
- 如果在数据结构课上学过表达式求值，也可以试着使用一个栈来实现。
- 当然，由于题目不允许括号嵌套，也可以使用其他更简单的方法替代。
- 调用函数时应当正确设计参数。不采用引用传参会造成大量不必要的数据拷贝（如输入的字符串），从而可能超时。
- 时间复杂度：$O(n)$

首次通过：上海大学 — 周天澜 0:07(+2)

E. 内存

Tag

implement, bitmask

题解

- 把每个虚页的所在的实页号用 `std::map` 或哈希表等可快速按内容查找的数据结构存起来。
- 每次在所问的地址转换为二进制表示，截取相应的位转换为十进制数，去查找其对应的实页号，转换为二进制覆盖虚地址中的页号，最后转换为十六进制输出即可。
- 由于计算机本身就是以二进制存储数据的，因此更好的实现方式是直接使用位运算来操作地址。
- 时间复杂度：$O(q)$

首次通过：上海理工大学 — 阙寅清 0:50(+)

A. 同源

Tag

brute force, math

题解

- 显然答案存在的必要条件是 n 能被 k 整除。
- 问题转化为：求三个两两互质的大于 1 的整数，使它们的和为 $N = \frac{n}{k}$。
- 直觉告诉我们，当 N 足够大的时候，这样方案会有很多。
- 根据 N 的奇偶性分类讨论暴力即可。
- 问 a, b, c 能不能相等的，建议复习一下 $\gcd$ 的定义。
- 时间复杂度：$O(T \cdot C \log n)$

首次通过：上海大学 — 徐正阳 0:29(+2)

F. 游戏

Tag

game

花絮

- 出题人半夜三点在床上打滚的时候证出来的。
- 不过对于更普遍的情况还是不知道什么时候平局。
- 为了使得想不出如何证明的同学也能通过，我们把数据范围和时限设置成了这样。
- 可以通过对树的点集进行状压来进行记忆化搜索而通过，因为并非所有点集都合法，所以并跑不满。

F. 游戏 （续）

可以有多种方式证明先手一定不会输，这里给出一种。

证明.

- 考虑每次取重心的最大的一颗子树。
- 设这颗子树的大小为 x，其他子树大小为 y_i，必有 $\frac{n}{2} \geq x \geq y_i$，所以可以取走 x。
- 假设可以在这棵树上找到一个大小大于 x 的子树 t，选点的中心势必要往一侧移动。
- 此时找到的 $\forall t > x$，都有另一侧 $n - x - t < y_i$，而 $x \geq y_i$，故 $n - x - t < t$。
- 因为 t 是两部分中较大的一部分，根据题设条件我们无法取走 t。
- 所以后手没有办法超过先手。

□

首次通过：上海大学 — 田汇捷 0:58(+)

G. 选择

Tag

dynamic programming

花絮

- 原来没有必选 x 这个限制，为了卡掉一些做法和一定程度上避免直接交原题代码加了这个限制。
- 同时加大了一点点的难度，希望大家用 dp 的方法予以解决。

G. 选择 （续）

题解

- 考虑最直接的 dp，用 $f_{i,j}$ 表示前 i 个数，选了 j 个的最大和。
- 但是时空都不太能接受，注意到无用状态非常多，因为前 i 个数里必定选择了约 $\lfloor \frac{i}{2} \rfloor$ 个数。
- 差距不会超过 2，所以将第二维向前偏移 $\lfloor \frac{i}{2} \rfloor$ 就好了，转移的数量非常少。
- 复杂度：$O(n)$

首次通过：同济大学 — 赵屹雄 0:44(+)

I. 露营

Tag

search, constructive algorithms

花絮

- 出题人没想到为什么大家都不开这题。
- 摇曳露营 Season 2 2021 年 1 月开始放送啦，强烈推荐大家一起来看哦。

I. 露营 （续）

题解

- 首先有两个必要条件，任意两点的距离必须小于等于高度差，并且距离与高度差的奇偶性必须相同。
- 并且这两个条件是充分的，只要令所有点都取 $\max(h_i - d_i)$ 即可。
- 可直接这样做的复杂度是 $O(nmk)$ 的，无法接受。
- 我们可以考虑直接从大往小填，并在这个过程中顺便 check 是否合法。
- 使用一个队列从关键点向外 BFS 即可。
- 不要忘记 $k = 0$ 的情况。
- 时间复杂度：$O((nm + k)\log(nm))$
- 使用桶排序和两个普通的队列可以做到 $O(nm + k)$ 。

首次通过：华东师范大学— 赵云翔 3:40(+1)

D. 旅行

Tag

graph, brute force, topo sort

花絮

- 由于数据随机，所以可以乱搞。
- 至于怎么乱搞，那就看大家各显神通了。

D. 旅行 （续）

题解

- 这里提供一个比较容易理解的方法：
  - 先把强联通分量缩点，问题就变成在 DAG 上的询问了。
  - 将询问离线后，对每个点开一个 `std::bitset`，按拓扑序维护可达性。同时对每个点的询问进行引用计数，在没有用的情况下就释放其内存。
  - 由于边数不多且数据随机，缩完点之后的强联通分量数量会比 n 小，并且在拓扑排序中实际存放在内存中的 `std::bitset` 的个数期望不多，可以满足规定的内存限制。
  - 时间复杂度：$O(\frac{n^2}{64})$
- 也有相关加剪枝的搜索可以通过本题，可以参考论文 — GRAIL: Scalable reachability index for large graphs：
  `http://link.springer.com/article/10.1007/s00778-011-0256-4`

H. 病毒

Tag

string

回文自动机的增量构造过程

当前串 S 长为 L，在尾部新增一个字符 c，即 $S_L = c$（标号从 0 开始）

1. 从加入 c 前后缀的最长回文对应的自动机节点开始；
2. 记当前节点的回文长度为 len，对比是否 $S_{L-1-\text{len}} = S_L$。
3. 若成立说明可在当前节点对应的回文两边各加上一个 c 构成的回文即为当前后缀的最长回文；
4. 若不成立跳到当前节点的 fail 节点，即当前节点的回文串的最长回文后缀的节点，回到步骤 2。

以上过程只与加入 c 前**后缀最长回文的节点**、加入的**字符** c 和 **前面的字符**有关。

H. 病毒 （续）

Trie 上广义回文自动机构造

- 每个 Trie 的节点对应的串即为节点父亲对应的串尾部新增所连边的字符 c 。
- 因此加入此字符，就是用父亲对应的节点，执行加入 c 的构造过程。
- 普通回文自动机中的 $S_{L-1-\text{len}}$ 就是 Trie 节点的第 len 祖先，可以使用树上倍增快速得到。

用此方法即可正确构建 Trie 的回文自动机。

H. 病毒 （续）

普通回文自动机的均摊的复杂度

在回文自动机的构造过程有暴力跳 fail 的过程。在单串的构造中这复杂度是通过均摊来保证为 $O(n)$ 级别的。因为只有形如 “……aaaaaa…aaaa” 中加入一个非 a 的字符才会导致多次跳 fail ，而加入一个非 a 字符后，“……aaaaaa…aaaa” 的形状就不再存在了。
只有用 $O(n)$ 级别的 a 才会形成 $O(n)$ 次暴力跳 fail 。

广义回文自动机的退化

但在 Trie 中会有链状的 $O(n)$ 长度的 “aaaa……aaaaa” 再在最后一个 a 上接 $O(n)$ 个非 a 字符，那么每个非 a 字符都会导致 $O(n)$ 次跳 fail ，总复杂度退化为 $O(n^2)$。

H. 病毒 （续）

优化

- 由于暴力跳 fail 会导致复杂度退化。但可以发现，对于在自动机的一个节点 P 上，面对字符 c 时比较失败，就要跳 fail ，到其最长回文后缀去寻找。
- 记从 P 跳 fail 能到的节点集为 $G(P)$ ，P 对应的回文串为 $S(P)$ 。显然 $G(P)$ 中所有节点的回文长度都比 P 的小，且都是 $S(P)$ 的真后缀。在这些点上对 c 寻找节点时，都在 $S(P)$ 里面进行，往左看一定不会超出 $S(P)$ 。且 $S(P)$ 是确定的，那么在从 P 跳 fail 去找 c 的可行节点是唯一的且不会变，则可记从 P 点跳 fail 对字符 c 去找到的节点为 $f(P,c)$ 。
- 当字符集较小时，则可以在对任一节点 P 、任一字符 c 第一次暴力跳 fail 找到节点后，把 $f(P,c)$ 存下来。下一次可以在表中 $O(1)$ 取值。
- 如此优化之后，每一个节点对每个字符只会跳一次 fail 。记字符集大小为 $|\Sigma|$,Trie 的节点数为 n，则优化后复杂度为 $O(n|\Sigma|)$，与普通回文自动机为同一级别。

H. 病毒 （续）

题解

- 解决完上面所有的问题以后，本题就是一个回文自动机上的经典问题了，由于篇幅原因，此处不再赘述。
- 更详细地说明可以阅读回文自动机论文 — *EERTREE: An Efficient Data Structure for Processing Palindromes in Strings*：`https://arxiv.org/pdf/1506.04862.pdf`

谢谢