2023 年（第十五届）四川省大学生程序设计大赛

四川成都都江堰

2023 年 6 月 4 日

Problem A. 旷野之息

给定 n 个人各自的财富值，找到一对 x，y 满足 $x\%$ 的人拥有 $y\%$ 的财富，并使 $y-x$ 尽可能大。

Problem A. 旷野之息

- 考虑将一个人 i 选上对答案的贡献，即 $\frac{a_i}{\sum a} - \frac{1}{n}$，若贡献大于 0，将其加上即可。
- 需要预处理 $\sum a$，时间复杂度 $O(n)$。

Problem B. 希卡之石

- 如果选手学过《编译原理》的话，他会发现这个题的解法与该课程所述的词法分析十分类似；如果选手学过《形式语言与自动机》的话，他会发现可以使用自动机的思想来解决此题。但是即便选手并没有学过上述课程，他也可以通过自己比较高超的分类讨论技巧在隐含自动机思想而不觉的情况下解决此题。毕竟这只是一道带有些许技巧的大模拟。
- 解题过程中，选手首先需要找到每行最后一个冒号，然后从此处阶段，后半部分为 $\%(message)s$；前半部分则需要进一步讨论。
- 此后，每当选手需要识别一个语法记号的时候，其诀窍在于考虑到题目保证输入数据的合法性，选手并不需要严格识别每一个语法记号并判别其是否合乎规则，只需要：
  - 把一个语法记号的类型与其他语法记号区别开来
  - 确定一个语法记号的结尾

Problem B. 希卡之石

- 按照上述思路可以大大降低编码复杂度，具体来说：
  - 在处理每个语法记号的首字符的时候，应该注意将其分成以下几种情况：
  - 空格：一个多余的空格应该跳过。
  - 数字 1 到 5：有可能是 $\%(levelno)s$ 或者是等同于其他数字
  - 其他数字：有可能是 $\%(asctime)s$ 或者是 $\%(filename)s$
  - 字母 ‘D’,‘I’,‘W’,‘E’,‘C’：有可能是 $\%(levelname)s$ 或者 $\%(pathname)s$ 或者 $\%(filename)s$
  - 其他大写字母：有可能是或者 $\%(pathname)s$ 或者 $\%(filename)s$
  - 字母 ‘l’,‘t’,‘p’: 分别有可能是 $\%(name)s$、$\%(thread)d$ 或者 $\%(thread)d$、$\%(process)s$，此外它们还有可能都是 $\%(filename)s$
  - 其他小写字母以及 ‘_’：只有可能是 $\%(filename)s$
  - 在处理好首字符后，使用类似的思想处理接下来的字符，以达到区分语法类型的目的。
  - 确定一个记号的类型后，根据题目所述的规则，找到其结尾。

Problem B. 希卡之石

- 一些细节
  - 如果选手未学习过《形式语言与自动机》或者不会诸如 JavaScript 等语言的话，可能没有接触过题目描述中所用到的正则表达式，不过选手应该可以通过此外的文字描述以及举的例子来理解题意。
  - 选手需要注意路径名和文件名中可能存在的空格以及日志时间中一定存在的空格，由于用 LaTeX 输入公式时显示的空格是很容易忽略的（类似 ）所以出题人特意用一些文字提醒了这些空格。
  - 需要注意在文件名或者路径名中可能会出现诸如"logger"、"process" 等字样，选手应该用其后是否有':' 或者'.' 来区分之。
  - 在确定一个记号的结尾时，一般情况下只需要数它的长度或者找到下一个空格即可。但是在路径名和文件名中可能会有内部空格，所以需要特殊处理。

Problem B. 希卡之石

- 一些细节（续）
  - 尽管题目描述并不要求选手注意时间数据的合法性，但是题目生成的数据确实保证了时间的合法性。
  - 理论上来说，文件名若与路径名同时出现，前者应该是后者的后缀。题目描述中并没有做这一要求，不过题目生成的数据却符合这一要求。
  - std 使用了 12 个左右的状态来区分记号类型，最终代码长度大约在 200 行左右。
  - std 是线性的，事实上只扫描了输入字符一次，故题目保证输入数据文件大小不超过 900KB 是很宽裕的。

Problem C. 神庙挑战

给定一 $n \times n$ 的棋盘，问最多可以放几个皇后使得通过调整每个皇后的 d 个 (左上、左下、右上、右下) 攻击方向满足每个皇后没有踏入别的皇后的攻击轨迹上。

Problem C. 神庙挑战

- 首先做 $d = 2$ 的情况。
  - 考虑对于每一条左上-右下的对角线，在这条对角线上的棋子最多有两个“左上”和“右下”攻击方向。
  - 对于左下-右上对角线，也是同理的。由于 $n * n$ 的棋盘一共有 $4n - 2$ 条对角线，每条对角线提供 2 个攻击方向，共有 $8n - 4$ 个攻击方向，而每个棋子需要 2 个攻击方向，所以这样算出来最多放 $4n - 2$ 个棋子。

Problem C. 神庙挑战

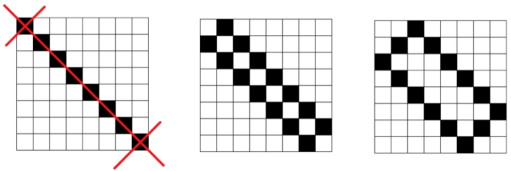
- 首先做 $d = 2$ 的情况。
  - 但是事实上，我们还需要考虑角落的情况，若右上角那个长为 1 的左上-右下对角线放了一个同时有左上和右下的棋子，那它对应的左下-右上对角线就不能有攻击方向右上的棋子，换句话说，这样的三个对角线（指最长的左下-右上对角线，右上角那个长度为 1 的左上-右下对角线与左下角那个长度为 1 的左上-右下对角线）只能提供 4 个攻击方向而不是 6 个，所以要在 $4n - 2$ 基础上减 1，同理，对于另一个方向，我们也要再减 1，所以 $d = 2$ 时，答案上界为 $4n - 4$。
  - 构造性证明是不难给出的，一个简单的构造是将棋子摆满第 1 行，第 n 行，第 1 列，第 n 列，第 1 行棋子攻击方向为左上与右上，第 n 行棋子攻击方向为左下与右下，第 1 列棋子攻击方向为左上与左下，第 n 列棋子范围为右上与右下，特别的，对于左上角的棋子，其攻击方向在左上与右上和左上与左下中任取一个即可，对于另外三个角同理。

Problem C. 神庙挑战

- 然后做 $d = 4$ 的情况。
  - 类似于 $d = 2$ 证明，每条对角线最多提供 2 个攻击方向，考虑角落的情况后，一共有 $8n - 8$ 个攻击方向，每个棋子需要 4 个攻击方向，这样算出来最多放 $2n - 2$ 个棋子。
  - 构造性证明同样不难给出，将第 1 行与第 n 行摆满后，拿掉第 n 行，第 1 列的棋子，拿掉第 n 行，第 n 列的棋子即可。

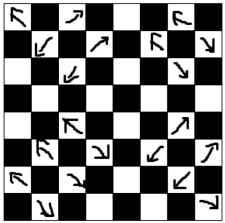
Problem C. 神庙挑战

- $d = 1$ 时，显然有一个很松的上边界 8n-4，因为我们考虑第一个图的三条对角线我们发现我们最多沿着对角线放两个棋子。于是有 4 个方向被浪费了。对于第二个的四条对角线也要浪费 4 个方向，第三个要浪费两个，对称一下又要浪费 10 个，于是我们就得到了上界，上界为 $8n - 24$。



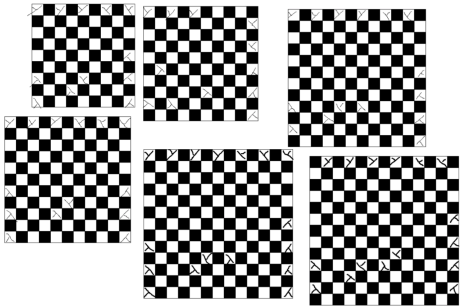
Problem C. 神庙挑战

- $d = 1$ 时，我们可以采用这一种从四周向中间的构造，构造一个类似 X 的形状后去掉最中间一行并微调中间那一行上面的那一行与下面的那一行即可。



Problem C. 神庙挑战

- $d = 3$ 时，我们发现我们能充分利用这些对角线。分奇偶讨论，奇数时上界为 $\lfloor \frac{4n}{3} \rfloor + \lfloor \frac{4n-4}{3} \rfloor$，偶数时上界为 $2 * \lfloor \frac{4n}{3} \rfloor$。



Problem D. 光鳞之枪

给定长度为 n 的两个字符串，问他们的最小编辑距离（删除、插入、替换）是否在 k（含 k）以内，若在 k 以内，则输出具体的最小编辑方案。

Problem D. 光鳞之枪

- 不难发现 $O(nK)$ 的 dp。
- 原本是想通过数据限制把该 naive 的方法卡掉，但是无奈现场评测机太快了，所以时限已改为让该做法通过。

Problem D. 光鳞之枪

- 由于两个串编辑距离很小，可以看出相同的部分很多，考虑用 LCP 来跳过相同的部分，让每次转移都用来处理 s,t 的当前匹配位置不同的情况。
- F[i][j] 表示进行了 i 次编辑后 $s[1\cdots a-1]$ 和 $t[1\cdots b-1]$ 匹配上，在 j=b-a 时最大的 a 是多少。
- 转移有三种：
  1. 在 s 的当前位置之后加入 t[b]，转移到 F[i+1][j+1]
  2. 在 t 的当前位置之后加入 s[a]，转移到 F[i+1][j-1]
  3. 将 s[a] 替换成 t[b]，转移到 F[i+1][j]
- 其中三种转移都需要求新的匹配位置的 LCP。
- 复杂度 $O(n\log n+K^2)$ 。

Problem E. 怪物商店

n 个商店，第 t 个商店提供价格为 1 到 a_t 的物品，每个物品 k 个，对每个商店，问从这个商店购买价格平均数为 x_t 的方案数。

Problem E. 怪物商店

- 考虑平均价格恰好为 x_t ，等价于购买的所有 $< x_t$ 的物品的价格的 $x_t - i$ 之和等于 $> x_t$ 的物品的 $i - x_t$ 之和
- 可以用一个简单的前缀和优化的 $O(ka^3)$ 背包计算两者的方案数，对应项相乘即得到答案。这样就得到了一个 $O(nka^3)$ 的做法，不可通过。
- 考虑对于多组询问，首先注意到 k 对于每组询问是相同的。
- 同时，对于每一组询问中 $< x_t$ 的物品，对和的贡献都是 $1, 2, .., x_t - 1$，对于 $> x_t$ 的物品，贡献是 $1, 2, ..., a_t - x_t$
- 也就是说我们只要求出所有物品集合是 $1, 2, ..., i$ 的背包结果就能回答所有询问

Problem F. 自建一始

给定一张图，找出最大的 k，使得该图采用最优策略下（每当选取两个节点 u, v 满足 $deg_u + deg_v \geq k$ 且 $(u, v) \notin E$ 时，将 (u, v) 加入 E）能够将原图补为完全图。

Problem F. 自建一始

- 首先二分答案，转化为判定能否加边成完全图。
- 我们可以用一个队列存储当前未处理的加边操作，每次取出队首，并将连锁反应要加的边加进队列。
- 由于最多进行 $\mathcal{O}(n^2)$ 次加边操作，总时间复杂度 $\mathcal{O}(n^2 \log n)$。

Problem G. 沃托里村

求 $(\sum_{i=1}^{n}\sum_{j=1}^{n}(a_i \bigoplus a_j)^2) \mod (10^9+7)$

Problem G. 沃托里村

- 将平方拆成两两位之间的贡献。
- 时间复杂度 $\mathcal{O}(n \log^2 V)$。

Problem H. 王国之泪

给定 n 个扭蛋机，第 i 个扭蛋机能随机扭编号 1 到 a_i 的扭蛋，问最优策略下，扭到所有可能的扭蛋的期望步数。

Problem H. 王国之泪

- 记 $m = max_{i=1}^{n}\{a_i\}$。
- 首先考虑寻找最优策略。
- 我们将扭蛋机按 a 的值从小到大排，那么最优策略显然是按 a 值从大往小扭。

Problem H. 王国之泪

- 记 cnt_i 表示排序后，第一个可以扭出扭蛋 i 的扭蛋机的 a 值。
- 对于一个扭蛋机，假设其 a 值为 x，那么对于它来说，扭出 1 至 x 中的任意一个整数的期望次数均为 x，证明考虑设扭出 $i(1 <= i <= x)$ 期望次数为 X_i，则有 $X_i = 1 + \frac{(x-1)}{x} X_i$。
- 假设现在还未扭出的扭蛋构成集合 S，设该集合中最大数为 $maxx$，那么我们接下来必然扭 cnt_{maxx}。
- 基于这样一些事实，再根据期望的线性性，我们可以从小到大枚举扭蛋编号 id，对每个扭蛋算它作为 $maxx$ 出现在集合 S 中的概率 P_{id}，以及扭出它所需期望次数 E_{id}，那么答案即为 $\sum P_{id} E_{id}$。
- E_{id} 是好算的，即为 cnt_{id}。

Problem H. 王国之泪

- 现在问题在于如何计算 P_{id}，即已经扭出 $[id+1,m]$，而 id 仍未扭出的概率。
- 不妨考虑这样一件事情，对于任意一个时刻，当前还未扭出的扭蛋的集合为 S，对于任意的 $i \in S$，下一个扭到的当前未出现的扭蛋为 i 的概率是相等的，也就是说我们只关注每个蛋第一次被扭出的时刻，同时我们并不关心 $[1,id-1]$ 是否被扭出来，则这些时刻形成一个 id 至 m 的排列，且每种排列出现概率相等。因为我们想获得 id 仍未被扭出的概率，我们可以假定 id 在这个时刻排列中最后出现，则 $P_{id}=\frac{(m-id)!}{(m-id+1)!}=\frac{1}{m-id+1}$。
- 需要对 a 排序，并且注意使用线性求逆元，不然快速幂求逆还有一个 log，时间复杂度为 $O(nlogn+m)$。

Problem I. 究极之手

给定一棵随机生成的边带权树，随机给点着色，每次着色完后求使得这些点互不连通的最小边权割集的权值。

Problem I. 究极之手

- 贪心。按照边权从大到小贪心的使每条边在尽可能晚的时间被删除。
- 一条边必须被删除当且仅当存在两个点同时在集合内且两个点路径上的其它边边权都大于这条边。
- 考虑边权从大到小并查集，当前边的两侧所在连通块都有点在集合内时，这条边就一定要被删除，因为此时两个连通块内其它边边权更大。并查集合并时维护连通块内的点加入集合时间的最小值。
- $O(n)$ or $O(n \log n)$ (排序的 $\log n$)

Problem I. 究极之手

- 由于随机，$\sum deep(v) = n \log n$。
- 对每个节点维护当子树加上当前节点的最小割集的权值和：若当前节点未染色，与当前节点相关的路径里留一条连接染色节点路径里割边最大的路径，并记录下该割边权值（因为其他着色节点均已被分割，可以留下一个边割最大的着色点等以后再割去），若当前节点着色，则需要把子树内所有其他着色点分隔开，然后当前节点的父边成为留下的边割。不难得到自叶子到根的 dp 转移，答案为根的最小边割权值和（留下的一条边割不用加进答案，因为不发生冲突）。
- 若有新的点被着色，则自该节点向根重新 dp 更新即可。
- $O(n \log n)$ (随机树深度的 $\log n$)
- PS：第一个做法不需要树随机生成的性质，但为了减少输入量、出题人比较好心等考虑，这个 naive 的做法也被放过去了，现场时限以该做法为准。

Problem J. 余料建造

给定一个字符串 S，需要构造一个字点序最小的字符串 T（T 初始为空，每次从 S 的开头或结尾取一个字符到 T 的尾部，直到 S 为空）。

Problem J. 余料建造

- 考虑贪心的构造 T，如果当前首尾字符不同，显然我们选择字典序更小的。那么相同的时候怎么办呢，我们考虑拿掉某个字符之后，下一步能否更优，如果两边下一个字符都大于当前字符，那么顺序无关紧要，否则我们要取下一个字符更优的。如果仍然相等，我们就需要继续比较再下一个字符。这样比较下去等价于我们比较正串和反串的字典序。取字典序更小的。
- 二分哈希即可解决。
- 当然这道题目其实 std 是线性的（每比较一次就有一个字符会被移动（可能不是立即移动，但当遇到不同字符时就会被移动，可以视作每比较一次，均摊移动一个字符））。

Problem K. 倒转乾坤

有两个圆环，其中小圆环的直径是 d，大圆环的直径是 $2d$。他将小圆环放在大圆环内，并让小圆环紧贴大圆环内壁进行无滑动的滚动。小圆环上等间隔地标记了 n 个点，求解在小圆环贴着大圆环运动一周后，所标记的 n 个点所经过的轨迹的长度之和。

Problem K. 倒转乾坤

- 容易观察发现轨迹在一条直线上。得答案是 $4dn$。

End

Thanks for attention!