CCPC 2024 济南站题解

山东大学
上海交通大学

2024 年 10 月 27 日

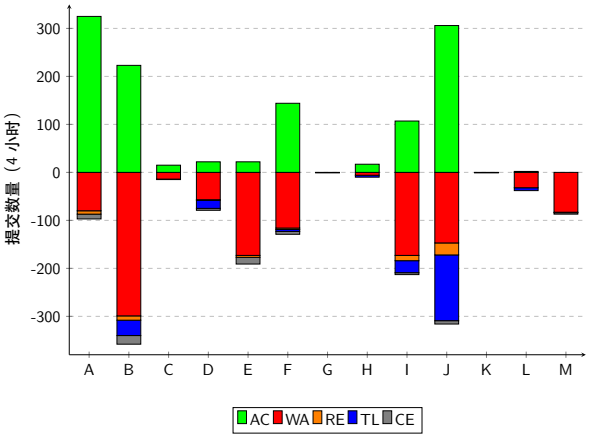
总览 - Fun facts

- 本场比赛共有 4080 份提交，3607 份 C++，423 份 Python，0 份 Java。
  - 其中，100% C 语言提交 CE 了；E 题贡献了 95.3% 的非 C++ 非 CE 提交。
- 所有题目均能使用 C++ 和 Python / Java 中的至少一种在现场的评测机上通过。
  - 一共有 173 份裁判代码（含部分 SJTU 队伍验题代码），其中 75 份为 AC 代码。
  - 所有裁判代码总长度为 $461\,325$ bytes.
- 皇帝、皇后 出题人 desprado2 老师本周五结婚了。皇帝 和 皇后 幸福地生活在了一起。

总览 - 题目分布

题目难度，与最短裁判代码长度 (bytes)

- Very easy: A (207), J (341)
- Easy: B (1023), I (606)
- Easy-mid: C (489), E (645), F (883)
- Medium: D (1065), H (1397), L (2393)
- Mid-hard: G (2095), K (3705), M (2379)
- Hard: —



A - 愚者

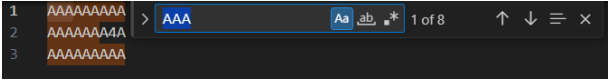
给定一个 $n \times mk$ 的 ASCII 可见字符网格，按顺序每 k 个代表一个字符串。
输出唯一一个和其他字符串不一样的字符串的位置。

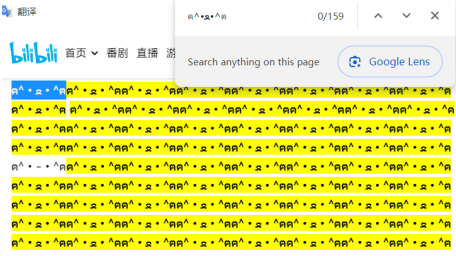
限制条件：

- $n, m \leq 200, k \leq 10$。

A - 愚者

- 如果在一个能够查找的文本编辑器或者浏览器里，这个问题用 Ctrl-F 很好解决：





- 使用你喜爱的编程语言也可以做类似的事情。时间复杂度 $O(nmk)$。

J - 节制

给定 n 个点 $A_i = (x_i, y_i, z_i)$。点 A_i 的密度定义为 $\max\{a, b, c\}$，其中 a, b, c 表示除了 A_i 之外的点中还有 a、b 和 c 个点与 A_i 分别有相同的 x、y 或 z 坐标。
你可以移除一些点，这么做会让所有点的密度重新计算。
对于 $k = 0, \cdots, n-1$，求至少要移除多少个点才能让剩余点的密度都大于等于 k。

限制条件：

- $n, |x_i|, |y_i|, |z_i| \leq 10^5$。

J - 节制

考虑固定 k 怎么做。

- 删除一个点不会使得其他点密度上升，所以初始密度 $< k$ 的点都必须被移除。
- 移除一个密度为 d 的点，不会影响密度 $> d$ 的点。所以也只需要移除初始密度 $< k$ 的点。

综上，我们只需要计算初始密度 $< k$ 的点有多少个。复杂度 $O(n)$。

B - 魔术师

给定一些扑克牌和六张塔罗牌的子集，塔罗牌可以看作能进行至多一次的操作，顺序任意：

- 星星、月亮、太阳、世界：将至多三张牌转化为方片、梅花、红桃、黑桃；
- 恋人：将一张牌转化为万能牌（视为任意花色）；
- 死神：将一张牌复制为另一张已有的牌。

五张花色可以视为一样的牌被称为同花，求给定牌最多能组成多少个不相交的同花。[1]

限制条件：

- 扑克牌为 52 张牌的子集，塔罗牌每张至多一张。
- $\sum n \leq 104$。

[1]游戏规则来源：Balatro, https://www.playbalatro.com/

B - 魔术师

一个直观的想法是进行搜索 / DP:

- 使用状态 $(d, c, h, s, w, t_{1\ldots6})$ 表示有 d 张梅花、c 张方片、h 张红桃、s 张黑桃、w 张万能牌，以及每张塔罗牌是否使用。
- 注意到我们能够新生成的一种花色的牌数量不会超过 4，初始只有 13 张，即使算上至多万能牌也只有至多 18 张，因此每种花色牌有效数至多为 15，状态数不超过 $15^4 \cdot 3 \cdot 2^6 = 9\,720\,000$。
- 由于操作其实限制很多，每次能进行的操作有限，且塔罗牌只能被单向消耗，有效的状态数会少很多，实现较好的搜索 / DP / 记忆化搜索也能通过。

B - 魔术师

太麻烦了，有没有简单做法？

B - 魔术师

太麻烦了，有没有简单做法？

我们首先注意到：

- 死神、恋人都可以看作将一张牌的花色直接修改为另一种任意花色；
- 一定可以先使用星星、月亮、太阳、世界，最后使用死神和恋人；
- 我们只需要关心最初状态和最后状态的差，把多的转换为少的。

因此，直接枚举最后每个花色组成了几个同花，贪心判断是否可行即可，需要判断的最终状态只有 4^4 个。同时，只依据部分观察也可以简化搜索实现，减小代码量。

Bonus: 修改限制条件为总共有 10^9 张牌，每张塔罗牌也有至多 10^9 张。

I - 倒吊人

给定 n 个点的树。构造加边方案使得每条边恰在一个简单环上，且图中不存在重边和自环。

限制条件：

- $n \leq 3 \times 10^5$。

I - 倒吊人 - DP

题意可以转化为：找到若干条路径不重不漏地覆盖整棵树的边，且每条路径长度 ≥ 2。

考虑树形 dp 判断合法性。记 $f_{x,0/1}$ 表示只考虑以 x 为根的子树，其中第二维表示：

0: 可以完成覆盖。

1: 可以留有一条链不完成覆盖，且链的一端是 x。

I - 倒吊人 - DP

考虑转移。

- $f_{x,0}$
  - 若有偶数个孩子，则为 True；
  - 否则，若有任意一个孩子 y 满足 $f_{y,1} = \text{True}$，则为 True。
  - 否则，一定有奇数个孩子且它们都 $f_{y,1} = \text{False}$，此时为 False。
- $f_{x,1}$
  - 若有奇数个孩子，则为 True。
  - 否则，若有任意一个孩子 y 满足 $f_{y,1} = \text{True}$，则为 True。
  - 否则，一定有偶数个孩子且他们都 $f_{y,1} = \text{False}$，此时为 False。
- 若最后根 r 满足 $f_{r,0} = \text{True}$ 则存在一组解。
- 按此讨论完成 DP 并在过程中记录方案/最后还原方案即可。复杂度 $O(n)$。

I - 倒吊人 - 贪心

太复杂了，有没有简单做法？

I - 倒吊人 - 贪心

太复杂了，有没有简单做法？

根据上述 DP，取任意偶度数点 r 为根时，都必然有 $f_{x,0} = \text{True}$，即存在偶度数点必然有解。为什么？

- 考虑如下构造：
  - 以某偶度数点为根 DFS。若当前子树内有偶数个点则两两连起来，若有奇数个点则任选一个不连。
  - 由于根是偶度数，最终构造一定是合法的。
- 若所有点都是奇度数，根据 DP 可证无解，亦可按树大小归纳证明。

E - 战车

一种出租车计价规则如下：

- 起步价 A 元，可以跑 X 米。
- 接下来 Y 米，每米 B 元。
- 接下来每米 C 元。

可以任意位置重新打车，问旅行 D 米的最优代价。

限制条件：

- $0 < A, B, C, D, X, Y < 10^{2077}$。
- 每个变量数码量总和不超过 $0\text{x}2077 = 8311$。

E - 战车

以下使用 A, B, C 表示对应计价等级，使用正则表达式表示策略：* 表示贪心选择直到里程将要溢出。

- $D \le X$：
  - A。
- $X < D \le X + Y$：
  - AB 或者 A*(A|B)：当起步价更划算时，重复使用起步价车辆；最后无法整除的一段可能由新车补齐，或者用已有（可能来自多辆）起步价车辆的单位计价 B 部分补齐。
- $D > X + Y$：
  - ABC 或 A*(A|B)：如果 C 是单位均价最划算的，那么只需一辆车跑到 $X + Y$ 米后，其余里程用 C；跑不到 $X + Y$ 米的原因只可能是跨过 B 太贵，会退化为全是起步价。
  - (AB)*(A|AB|C)：如果 AB 单位均价最划算，在里程为 $X + Y$ 倍数时一定可以贪心用 AB；最后一段可能用一辆新车补齐，或者用 C 补齐。有一个特殊情况：注意 (AB)*A 最后加入 A 时里程可能超出，计算时应把之前溢出的 B 退费。
  - A*(A|B|BC)：如果 A 最划算。

E - 战车

因此，可能的策略只有四种：

- 用一辆车：`A|AB|ABC`；
- 尽可能用 `A`：`A*(A|B|BC)`；
- 尽可能用 `AB`，且最后一段用 `A` 可能里程溢出，要退回部分 `B`：`(AB)*A,`；
- 尽可能用 `AB`：`(AB)*(A|AB|C)`。

对所有情况取 $\min$ 即可，需进行常数次四则运算。

E - 战车

本题写 Java 可能会因为表达式不直观显得痛苦，C++ 需要正确实现 $O(d^2)$ 的 $+, -, \times, \div, \mod$ 操作。用 Python 的话，代码会短很多。

p.s. 为什么数据限制是 10^{2077}?

- 答案最大约为 $\sim 10^{4154}$，如果再大则会触发神秘事件。
- Python 比赛版本（3.10.12）在输出超过 4300 位的时候会触发

```
ValueError: Exceeds the limit (4300) for integer string conversion;
    use sys.set_int_max_str_digits() to increase the limit.
```

F - 隐士

在 $\{1 \dots m\}$ 的所有大小为 n 的子集，询问每个子集至多能保留多少个数使得 $\gcd \neq \min$。所有答案求和对 $998\,244\,353$ 取模。

限制条件：

- $m \leq 10^5$。

F - 隐士

首先我们来解决对于一个集合应该怎么操作：

- 如果集合的 $\gcd \neq \min$，那么什么也不用做；
- 否则，此时集合里所有数整除 $\min$，不删除 $\min$ 的话集合不可能合法。

因此，一直删除 $\min$ 使得条件满足或者集合删空即可。

F - 隐士

对于 $\{1 \dots m\}$ 的所有大小为 n 的子集进行操作，直接维护状态或者容斥计数都较麻烦。换个角度，考虑每个元素被删了多少次，从而用原本集合大小 $\binom{m}{n} \cdot n$ 减去即可得到答案：

- 固定一个元素 x。如果 x 在某个集合内被删掉了，那么说明：
  - 比 x 小的数构成（长度不超过 $\log_2 x + 1$ 的）以 x 结尾的倍数链；
  - 比 x 大的数都是 x 的倍数。
- 小的部分可以直接枚举倍数求得 $f_{c,x}$ 表示倍数链长度为 c，且当前的数是 x 的方案数。
- 大的部分是在倍数里任选。

因此要删去的数总数为

$$\sum_{1 \le c \le n, 1 \le x \le m} f_{c,x} \cdot \binom{\lfloor m/x \rfloor - 1}{n - c}.$$

枚举倍数求得 $f_{c,x}$ 是调和级数的，粗略的时间复杂度上界估计为 $O(m \log^2 m)$。

F - 隐士 - 时间复杂度 $O(m \log m \log\log m)$

- $f_{c,x} > 0$ 仅当 x 能拆成 c 个大于 1 的数的乘积，最多拆成 $\Omega(x)$ 个乘积，即（重复出现计多次的）质因子数量。只枚举 $f_{c,x} > 0$ 的倍数，总复杂度为[2]

$$
\begin{aligned}
S &= \sum_{x=1}^{m} \lfloor m/x \rfloor \cdot \Omega(x) \\
&= \sum_{p^k,\mathrm{prime}(p),k\geq 1} \sum_{x,p^k \mid x} \lfloor m/x \rfloor \\
&= \sum_{p^k,\mathrm{prime}(p),k\geq 1} \sum_{j\geq 1} \lfloor m/(j \cdot p^k) \rfloor \\
&\leq \sum_{p^k,\mathrm{prime}(p),k\geq 1} m/p^k \cdot H(m/p^k) \\
&\leq m \cdot H(m) \cdot \sum_{p,\mathrm{prime}(p)} \sum_{k\geq 1} 1/p^k \\
&\leq m \cdot H(m) \cdot \sum_{p\leq m,\mathrm{prime}(p)} 1/p \cdot (1 + 1/2 + 1/4 + \dots) \\
&= O(m \log m) \cdot O(\log\log m) = O(m \log m \log\log m).
\end{aligned}
$$

数值计算：在 $m = 10^6$ 时，$S = 36\,696\,701$。

[2]感谢钱哥 (skip2004) 修对的证明。

C - 皇后

一种解释型语言由两种指令构成，初始有一个无限容量的空栈，并执行第一条指令。两种指令分别为：

- 若栈顶元素为 a，弹出 a，接下来执行第 x 条指令；否则将 b 推入栈中，接下来执行第 y 条指令。

```
POP a GOTO x; PUSH b GOTO y
```

- 若栈为空，程序停机；否则将 b 推入栈中，接下来执行第 y 条指令。

```
HALT; PUSH b GOTO y
```

构造一个不超过 64 条指令的程序使其恰好在执行 n 条指令后结束。

限制条件：

- $1 \leq n < 2^{31}$，n 为奇数。

C - 皇后

注意到除了最后一条停机指令，每次 POP 操作都对应一次 PUSH 操作。
考虑对 $\frac{n-1}{2}$ 进行二进制分解，如果我们能使用一条指令构造出 “$\times 2$” 或者 “$+2$” 两种操作，则可以用大约 $2\log_2 n$ 条指令构造出任意奇数 n。

归纳假设：已经构造出一个共 t 条指令的程序 S，满足：

- 从空栈进入指令 1 时，经 x 步后以空栈进入指令 $t-1$；
- 程序的最后两条指令 $t-1$ 和 t 必须为：

```
POP t-1 GOTO t; PUSH t-1 GOTO t
HALT; PUSH 100 GOTO 1
```

即，程序 S 共进行了 $x+2$ 次 PUSH 和 POP 操作后停机。

C - 皇后 - 归纳基础

归纳基础：$n \leq 5$ 的情况。

- 构造 $x = 2$ 时的程序 S，此时对应 $n = 5$ 的解。

```
1 POP 1 GOTO 2; PUSH 1 GOTO 1
2 POP 2 GOTO 3; PUSH 2 GOTO 2
3 HALT; PUSH 100 GOTO 1
```

- 删去第 2 条指令得到 $n = 3$ 的解，删除第 $1, 2$ 条指令得到 $n = 1$ 的解。

C - 皇后 - $\times 2$ 的构造

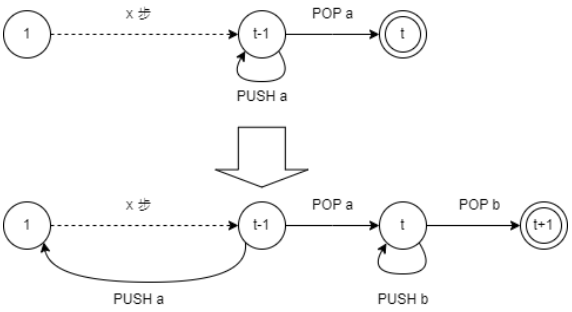
使用 $t+1$ 条指令构造 “$\times 2$”:

- 保持程序 S 的指令 1 至 $t-2$ 不变。
- 修改指令 $t-1$ 和 t 为:

```
POP t-1 GOTO t; PUSH t-1 GOTO 1
POP t GOTO t+1; PUSH t GOTO t+1
```

- 增加一条指令 $t+1$:

```
HALT; PUSH 100 GOTO 1
```



该程序共进行了 $2x+4$ 次 PUSH 和 POP 操作后停机。

C - 皇后 - $+2$ 的构造

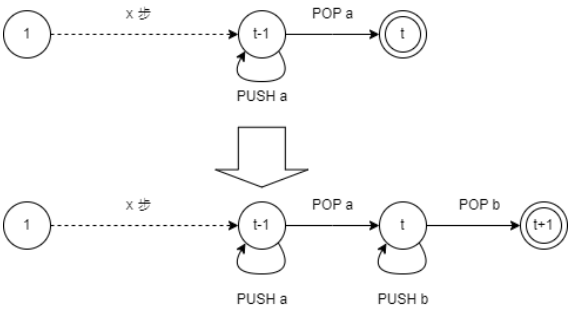
使用 $t+1$ 条指令构造 “$+2$”:

- 保持程序 S 的指令 1 至 $t-2$ 不变。
- 修改指令 $t-1$ 和 t 为:

```
POP t-1 GOTO t; PUSH t-1 GOTO t
POP t GOTO t+1; PUSH t GOTO t+1
```

- 增加一条指令 $t+1$:

```
HALT; PUSH 100 GOTO 1
```



该程序共进行了 $x+4$ 次 PUSH 和 POP 操作后停机。

C - 皇后 - 另解

如果不使用二进制分解的构造，使用 a 条指令完成形如 $(x+2)\times a$ 的构造也是可以接受的。

- 事实上仅考虑 $a\in[1,9]$，可以保证最多使用 64 条指令构造出 2^{31} 以内的任意奇数 n：使用动态规划暴力求出 2^{31} 以内仅考虑 $a\in[1,9]$ 的最优解，发现最优解小于等于 62。
- 由于空间和时间的限制，我们不可能预先求出所有 2^{31} 以内的奇数的解。因此可以指定一个阈值 $N=10^7$，对 N 以内的整数采用动态规划求最优解，对更大的数暴力搜索直到数字小于 N 且总指令数小于等于 64。
- 同理，一些类似的构造也可以通过本题。

D - 皇帝

TL; DR: 给皇后写一个（取模意义下的）裁判程序。

给定一个由 n 条指令组成的程序（见 C - 皇后 题面），求出其将在多少步以后停机，答案对 $998\,244\,353$ 取模。如果不能停机，输出 -1。

限制条件：

- $1 \leq n \leq 1024$。

D - 皇帝

考虑记忆化搜索，用二元组 $f(u,a)=(cnt,v)$ 表示若当栈顶为 a 时进入指令 u，将执行 cnt 条指令后将该元素 a 弹出，并进入指令 v。特别地，令 $a=0$ 表示此时栈为空，令 $v=0$ 表示程序停机。在搜索过程中需要记录在搜索栈内的所有 (u,a)。假设此时搜索到 (u,a)，依次讨论：

1. 如果 $a=0$ 且指令 u 为 `HALT`，返回 $(1,0)$。

D - 皇帝

2. 如果在指令 u 处为 `POP a GOTO x`，返回 $(1, x)$。否则将进行压栈操作，记指令 u 为 `PUSH b GOTO y`。
3. 如果 (u, a) 已经在搜索栈内，说明从 (u, a) 开始在**尚未将 a 弹出前**出现了循环，因此程序不停机。
4. 如果已经访问过 (u, a)，直接返回 $f(u, a)$。否则按下列步骤计算 $f(u, a)$。
5. 首先搜索 $f(y, b)$，记 $f(y, b) = (cnt_1, z_1)$。若 $z_1 = 0$ 则 $f(u, a) = (cnt_1 + 1, z_1)$；否则继续搜索 $f(z_1, a)$，记 $f(z_1, a) = (cnt_2, z_2)$，$f(u, a) = (cnt_1 + cnt_2 + 1, z_2)$。

时间复杂度 $O(n^2)$。

由于不停机时一定会死循环，还可以采用限制入队次数、限制总循环或递归次数的方式判断不停机。

H - 力量

给定一个数字 x，计算 $[0, z]$ 内可以激进舍入到数字 x 的数量。
激进舍入定义为若干次对着任意十进制数位进行四舍五入操作。

限制条件：

- $0 \leq x, z \leq 10^{18}$ ， 10^5 组数据。

H - 力量

- 首先考虑如何快速判断一个数 y 能否激进舍入到 x。
- 从低位向高位考虑。
- 若 x 和 y 个位相同，则个位不必舍入；
- 若 x 和 y 个位不同，那么我们必须让 y 的最低位舍入一次。如果 x 最低位是 0，那么此时也可行。如果 y 的最低位是 $5 \sim 9$，则会对高位进位 1。
- 然后再考虑 x 和 y 的十位。
- 若个位没有带来进位，则把十位当作最低位一样处理即可；
- 否则我们需要枚举个位舍入和十位舍入的先后顺序，再类似地讨论。
- 百位，千位……

H - 力量

- 上述过程实际可以写成 DP 的形式。
- 设 $f_{i,1/2/3}$ 表示从低考虑到高位，已经考虑到第 i 位时可行性，第二维表示：
  1. 上一位一定不对当前位进位；
  2. 上一位一定会对当前位进位；
  3. 可以通过调整操作顺序自由选择上一位是否对当前位进位。
- 转移则按照上一页所述进行讨论即可。

H - 力量

- 该 DP 可以直接套用来计数。
- 设 $f_{i,0/1,1/2/3}$ 表示从低考虑到高位，已经考虑到第 i 位，只考虑上界 z 前 i 位时当前 DP 的这个数是否小于等于上界 z，且
  1. 上一位一定不对当前位进位；
  2. 上一位一定会对当前位进位；
  3. 可以通过调整操作顺序自由选择上一位是否对当前位进位，

  的方案数。
- 转移只需枚举当前位数字填写 $0 \sim 9$ 中的哪一个数字，此时可以认为固定了 y 是什么，于是就可以按同样的方法 DP 了。
- 复杂度 $O(t \cdot 10 \log z)$

L - 高塔

维护一个顶部弹幕系统，支持三种操作：

- 发若干条弹幕，它们会占据最靠上的若干个空行；
- 删除之前某次发送的全部弹幕；
- 查询某一行是否有弹幕，如果有的话还要返回是哪一次发的。

限制条件：

- 操作数 $n \leq 5 \times 10^5$。
- 每次发送的弹幕数量和询问的行数 $\leq 10^9$。

L - 高塔

- 因为各个用户的弹幕之间没什么联系，因此没有必要把所有行都在一起维护，可以每个用户分别维护一个集合。
- 因为发弹幕的时候涉及到询问名次，可以使用线段树维护。用一棵线段树维护所有空行，那么发弹幕的时候就只需要把它的一个前缀分裂出来放到一棵新的线段树中。
- 删除的时候类似地，使用线段树合并将对应集合合并到代表空行的集合。

L - 高塔

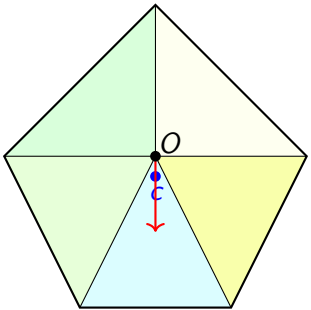
如何进行查询？

- 对于每一个线段树上的结点，维护它的直接祖先。同时用一个 map 储存每个根节点对应的发送者编号。查询的时候只需一步一步跳到根结点然后查询即可。
- 唯一的问题是，本题是动态开点的，所以询问时，对应的叶子结点可能根本没有建立！
- 由于本题允许离线回答，可以在一开始就把所有的询问结点建立出来。
- 复杂度 $O(n(\log v + \log n))$。原因是，以线段树总结点个数为势能，分裂操作使得势能增加 $O(\log v)$，合并操作每执行一次都会导致有一个结点被抛弃，即势能减少 1。
- 分析粗糙点，线段树结点数最多需要 $3n \log v$ 个，每个结点包含 4 个 int，大约需要 700MB 的空间。

Bonus：强制在线。

G - 命运之轮

一个匀质凸多边形转盘由旋转中心划分得奖区域。
当转盘的重心不在旋转中心时，转盘永远会停留在重心在最低点处，被向下的指针指向。
一个点配重 w 将会被均匀随机放在转盘上。问每个区域成为获奖区域的概率。



限制条件：

- $n \leq 10^5$。
- $|x|, |y| \leq 30000$，$1 \leq w \leq 10^9$。
- 转盘单位密度为 1，转盘面积 S 满足 $\frac{S}{w} \in [0.001, 1000]$。

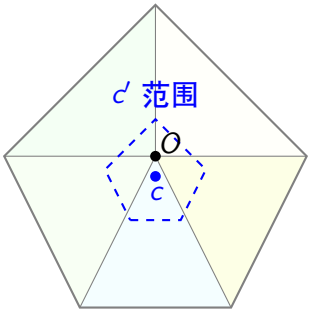
G - 命运之轮

设转盘的重心是 c，配重的坐标为 p，那么转盘新的重心是

$$c' = \frac{c \cdot S + p \cdot w}{S + w}.$$

其中只有 p 是一个在凸多边形内的变量，其余为常量。
因此，c' 是关于 p 的线性函数。

- 等价于在一个被缩放的凸多边形范围内随机一个点作为 c'。
- 在被缩放的凸多边形里，落在原来每个区域的面积占比即是答案。

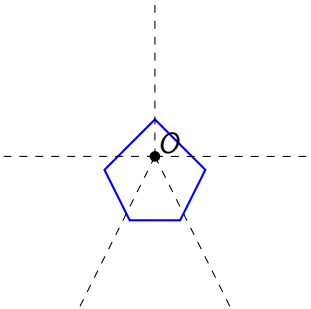


G - 命运之轮

由于 c' 范围严格小于原凸包，我们可以看做从 O 点作若干条射线，切分新 c' 范围。
实现凸包查询的话，可能需要分以下 3 种情况讨论 O 射线与 c' 范围的交。

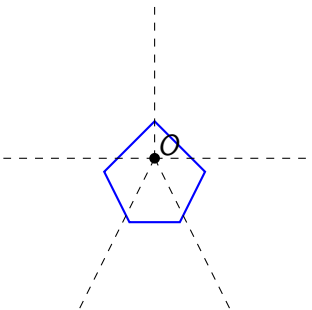
- 点在凸包内；
- 点在凸包上；
- 点在凸包外。

其中，后两种情况都需要正确实现 O 与 c' 范围上某些点、边重合 / 共线的情况，且要求复杂度都在 $O(\log n)$；最后需要把每个区域切出来的结构还原，求出面积。



G - 命运之轮

太麻烦了，有没有简单做法？



G - 命运之轮

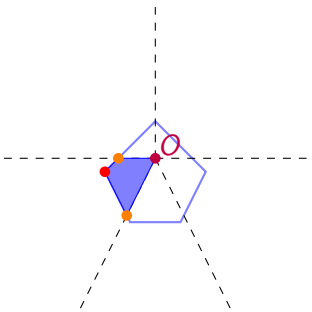
太麻烦了，有没有简单做法？

每个（答案非 0 的区域）里在缩放凸包中切出一个凸包，且所有区域点、边数量总和是 $O(n)$ 的。

对于每个区域，找出其凸包顶点，随后可以直接对这个凸包求面积。区域凸包顶点来源有：

- 缩放凸包顶点，以及缩放凸包的边与射线的交点。
- 如果原点 O 在缩放凸包里，那么每个区域凸包里都有。

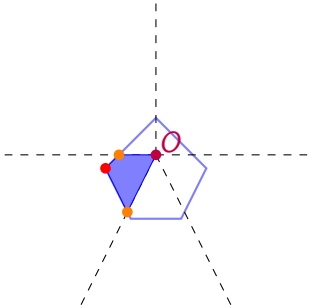
缩放凸包顶点和边与射线的交点可以通过沿着区域暴力跳的方式维护。



G - 命运之轮

缩放凸包顶点和边与射线的交点可以通过沿着区域暴力跳的方式维护。

- 假设 $u \to v$ 的过程中，从区域 i 移动到了某个新区域。
- 通过点 O 在 $u \to v$ 左侧还是右侧确定从 i 开始是逆时针转（++）还是顺时针转（--），转到 v 所在区域为止。
- 给途径区域加上交点，给 v 所在区域加上点 v。
- 暴力定位第一个点所在的区域。uv 直线穿过 O 时不能判别顺逆时针：此时，要么区域不变，要么 $u \to v$ 线段经过了 O。后者只有一次，暴力点定位即可。



主体部分复杂度 $O(n)$。求凸包仔细实现（如，拆为线积分）算面积也是 $O(n)$ 的，直接求凸包 $O(n \log n)$ 也是可以接受的。

- 一个小技巧：将计算面积的原点换为 O 可以有效避免所有过 O 直线的讨论。

G - 命运之轮 - 精度

精度要求看起来很宽松，正确使用 `double` 亦可以通过。但是，仍需要注意精度。

- 缩放凸包面积最小 $\sim 10^{-3}$，要求精度为 10^{-6}，因此在点定位时至多可以容许 10^{-9} 的坐标误差。
- 重心分母最大 $6 \cdot (S + w) \approx 2.76 \times 10^{10}$，亦可构造出坐标差值小于 10^{-10} 的数据。小误差不会影响答案正确性，但某些凸包写法里使用大 `eps` 会导致排序错位使凸包失效。
- 输入为整点，实际限制略松。以下两种组合经测试使用 `double` 可以通过：
  - 在点定位部分使用 $\texttt{eps} = 10^{-9}$，凸包不使用 `eps`；
  - 整体使用 $\texttt{eps} = 10^{-11}$。

参考代码在除了计算交点与面积外的所有部分使用了整数。

- 面积为半整数，重心为 $1/3$ 整数，因此只要整体乘 $6 \cdot (S + w)$ 即可使得重心与缩放凸包坐标为整数。
- 坐标值域为 $30000 \cdot 6 \cdot (60000^2 + 10^9) = 8.28 \times 10^{14}$，计算叉积需要 128 位整数。

K - 魔鬼

一个短语由不超过 N 个字符串构成，每个字符串长度不超过 N。
一个短语的缩写是由短语的每个单词的非空前缀按顺序拼接起来的字符串。
给定 $n \leq N$ 个短语，要求给每个短语分配一个不同的缩写，使得总长度最小。

限制条件：

- 字符串为大小写英文字母串。
- $N = 128$。

K - 魔鬼

存在一组最优解使得每个短语只会和其前 n 短缩写匹配：

- 不满足条件时可以调整答案使得与前 n 短匹配，且答案不会变差。

因此问题可以转换为在 $O(n^4)$ （题解记号中默认 $n = N$）内完成以下两件事情：

- 找到每个短语的前 n 短缩写；
- 找到短语到缩写的最小权匹配。

K - 魔鬼 - Part I

首先考虑如何在 $O(n^3)$ 内找到一个短语的前 n 短缩写。

- 如果每个前缀拼接起来构成的字符串都是两两不同的，那么直接按长度从小到大搜索就可以在 $O(n^2)$ 的时间内找到。但当字符串形如 AAAA… AAAA… 的最坏例子时，就会有大量重复结果，使得搜索上升到指数级。
- 考虑类似记忆化搜索的过程：对于每个 (i, s) 表示考虑到第 i 个字符串，当前组成的前缀为 s 的状态。同一个状态只应至多运行一次向后的搜索。
- 为了使复杂度正确，只应考虑每个 i 前 n 长字符串：因为它们在拼接后面每个单词首字母后就已经能形成 n 个互不相同的串，无需考虑其他的串。因此，每步向后递推时也只需考虑接至多 n 个字符。
- 在最坏的例子里，上述过程需要考虑至多 n^3 个串，每个串长度可以是 $n \sim 2n$，暴力去重复杂度达到了 $\Omega(n^4)$。

K - 魔鬼 - Part I

- 使用 Hash 去重可以使复杂度正确，但为了构造方案，需要额外 Hash 之间添加字符的关系来还原方案。
- 这个过程其实不太需要 Hash：一个字符串加一个字符等于另一个字符串的结构实际上可以用 Trie 维护。
- Trie 会额外引入一个空间的问题：如果实现使用的节点数达到了 $O(n^3)$，那么单次使用的空间可能达到 $O(n^3\alpha)$，导致清空复杂度不正确。
- 仔细实现的搜索不会引入这个问题：对于每个 i，Trie 树新增节点数量至多为 n 个，因为每新增一个 Trie 树结点就意味着一个不同的字符串出现。如果按长度顺序搜索，出现 n 个新结点一定就可以停下来了。总共用到的节点数是 $O(n^2)$ 的。

K - 魔鬼 - Part II

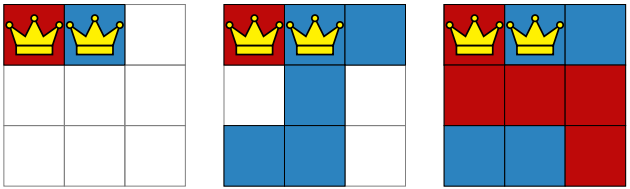
然后考虑如何在 $O(n^4)$ 内找到找到短语到缩写的最小权匹配。

- 匹配关系一个带权二分图，其中左侧有 n 个点，右侧有 $O(n^2)$ 个点，边数为 $O(n^2)$。
- 我们使用朴素的 Bellman-Ford 进行分析。考虑进行第 i 轮匹配时，增广路长度不超过 $2 \cdot i + 1$。因此, Bellman-Ford 需要的松弛次数至多为 $2n + 1$，匹配的复杂度是 $O(n \cdot nm) = O(n^4)$ 的。
- 同理, 用队列优化的 Bellman-Ford (a.k.a. SPFA) 的入队次数也不会超过 n 次，直接实现 SPFA 即可。
- 正确实现的 KM 算法也可以通过，但是直接抄一个补成满匹配的 KM 会导致复杂度退化到 $O(n^5)$ 或者更高。

M - 审判

给定一个 generals.io [3] 游戏局面，判断是否可能是合法状态。

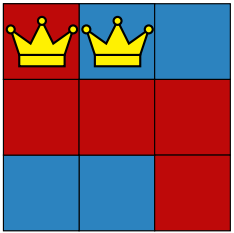
- 在 $n \times m$ 网格图上有红蓝两个玩家，每个玩家有一个不能被敌方占领的大本营。
- 从其余网格均未被占领的状态出发，会发生若干次以下事件：一名玩家占领了一个与己方已占领位置边相邻的格子。



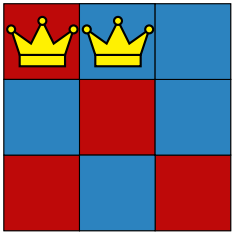
[3] https://generals.io/

M - 审判

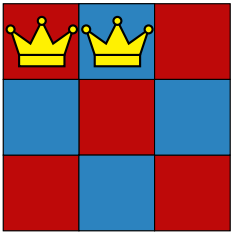
- 事实上，大部分情况都是合法的。



√



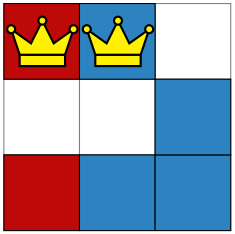
√



×

M - 审判

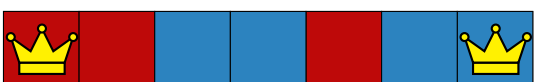
- 不合法情况 1：不连通。



M - 审判

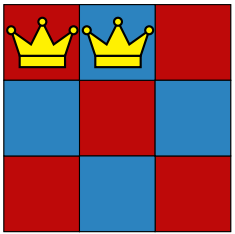
- 不合法情况 2：（样例 4）某颜色的格子，但不能走到该颜色大本营。

- 不合法情况 3：一条链，但是中间颜色切换多次。



M - 审判

- 不合法情况 4：红蓝相间。



- 这个观察是本题最重要的。

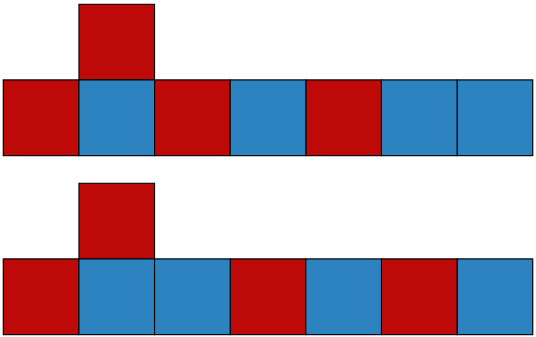
M - 审判

结论：排除掉某颜色无法到达对应大本营、链、不必移动的情况后，无解当且仅当图是红蓝相间。

- 首先证明红蓝相间是无解。如果有解，考察该解染色的最后一步，发现这一定会导致有两个相邻格子颜色相同。
- 其余状态有解可以直接构造。先构造一棵原图的生成树，要求大本营都在叶子上。由于排除掉了链的情况，必然存在一个点度数 ≥ 3。利用这个三/四叶草，可以将三叶草之外的点染色成任意方案，但这样三叶草本身无法染出红蓝相间。
- 其余状态中必然存在两个相邻点颜色相同，假设这两个点都在三叶草上，问题直接解决。

M - 审判

- 若不在三叶草上，则可以看它来自哪一瓣，把颜色做一次整体位移。



M - 审判

综上，算法流程如下：

1. 先去除掉两个大本营，对每个连通块 BFS，求出它们各自与两个大本营是否连通。特别的：当两个大本营相邻时，不应当认为它们之间有边。
2. 然后对于每个连通块分别检查。

复杂度 $O(nm)$。

Thank you!