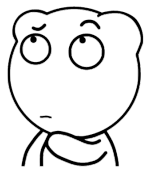
2019 暑期集训新队员选拔赛第 1 场

试题分析

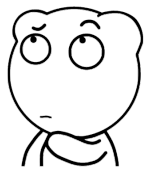
你以为你 CF 过了四题

2019 年 8 月 16 日



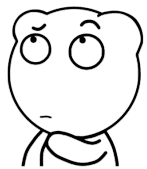
难度预估

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



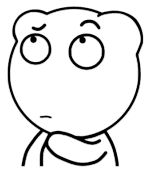
通过情况

	通过人数	提交人数	首次通过
A	27	28	0:06
B	0	18	N/A
C	18	25	0:28
D	0	0	N/A
E	1	3	4:05
F	0	6	N/A
G	6	7	1:49
H	0	3	N/A
I	0	11	N/A
J	11	18	0:54



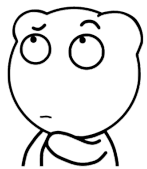
Problem A. 最大公约数

- **签到题。**
- $\gcd(x, x+1) = 1$，因此 $a \neq b$ 时输出 1，否则输出 a。



Problem A. 最大公约数

- **签到题**。
- $\gcd(x, x+1) = 1$，**因此** $a \neq b$ **时输出** 1，**否则输出** a。

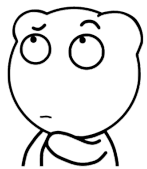


Problem B. 最长公共子序列

- 定义状态 $f_{i,j}$ 为 S 的前 i 位与 T 的前 j 位最长公共子序列，则有

$$f_{i,j} = \begin{cases} \max(f_{i-1,j}, f_{i,j-1}) & , S_i \neq T_j \\ f_{i-1,j-1} + 1 & , S_i = T_j \end{cases}$$

- 上述做法的时间复杂度 $O(nm)$，无法通过本题。
- 最终答案不会超过 m 。
- 更改状态定义 $f_{i,j}$ 为与 T 前 i 位的最长公共子序列长度为 j 的 S 的最短前缀长度（即将朴素做法的答案与第一维状态对调）
- 可以通过预处理 S 的每一位的下一个 $a, b, \cdots, z$ 的出现位置进行 $O(1)$ 的顺推转移。
- 复杂度 $O(m^2 + 26n)$，可以通过本题。

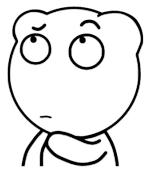


Problem B. 最长公共子序列

- 定义状态 $f_{i,j}$ 为 S 的前 i 位与 T 的前 j 位最长公共子序列，则有

$$f_{i,j} = \begin{cases} \max(f_{i-1,j}, f_{i,j-1}) & , S_i \neq T_j \\ f_{i-1,j-1} + 1 & , S_i = T_j \end{cases}$$

- 上述做法的时间复杂度 $O(nm)$ ，无法通过本题。
- 最终答案不会超过 m 。
- 更改状态定义 $f_{i,j}$ 为与 T 前 i 位的最长公共子序列长度为 j 的 S 的最短前缀长度（即将朴素做法的答案与第一维状态对调）
- 可以通过预处理 S 的每一位的下一个 $a, b, \cdots, z$ 的出现位置进行 $O(1)$ 的顺推转移。
- 复杂度 $O(m^2 + 26n)$ ，可以通过本题。

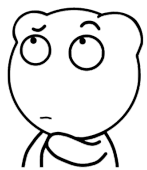


Problem B. 最长公共子序列

- 定义状态 $f_{i,j}$ 为 S 的前 i 位与 T 的前 j 位最长公共子序列，则有

$$f_{i,j} = \begin{cases} \max(f_{i-1,j}, f_{i,j-1}) & , S_i \neq T_j \\ f_{i-1,j-1} + 1 & , S_i = T_j \end{cases}$$

- 上述做法的时间复杂度 $O(nm)$ ，无法通过本题。
- 最终答案不会超过 m 。
- 更改状态定义 $f_{i,j}$ 为与 T 前 i 位的最长公共子序列长度为 j 的 S 的最短前缀长度（即将朴素做法的答案与第一维状态对调）
- 可以通过预处理 S 的每一位的下一个 $a, b, \cdots, z$ 的出现位置进行 $O(1)$ 的顺推转移。
- 复杂度 $O(m^2 + 26n)$ ，可以通过本题。

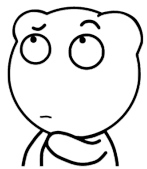


Problem B. 最长公共子序列

- 定义状态 $f_{i,j}$ 为 S 的前 i 位与 T 的前 j 位最长公共子序列，则有

$$f_{i,j} = \begin{cases} \max(f_{i-1,j}, f_{i,j-1}) & , S_i \neq T_j \\ f_{i-1,j-1} + 1 & , S_i = T_j \end{cases}$$

- 上述做法的时间复杂度 $O(nm)$ ，无法通过本题。
- 最终答案不会超过 m 。
- 更改状态定义 $f_{i,j}$ 为与 T 前 i 位的最长公共子序列长度为 j 的 S 的最短前缀长度（即将朴素做法的答案与第一维状态对调）
- 可以通过预处理 S 的每一位的下一个 $a, b, \cdots, z$ 的出现位置进行 $O(1)$ 的顺推转移。
- 复杂度 $O(m^2 + 26n)$ ，可以通过本题。

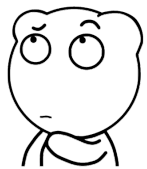


Problem B. 最长公共子序列

- 定义状态 $f_{i,j}$ 为 S 的前 i 位与 T 的前 j 位最长公共子序列，则有

$$f_{i,j} = \begin{cases} \max(f_{i-1,j}, f_{i,j-1}) & , S_i \neq T_j \\ f_{i-1,j-1} + 1 & , S_i = T_j \end{cases}$$

- 上述做法的时间复杂度 $O(nm)$ ，无法通过本题。
- 最终答案不会超过 m 。
- 更改状态定义 $f_{i,j}$ 为与 T 前 i 位的最长公共子序列长度为 j 的 S 的最短前缀长度（即将朴素做法的答案与第一维状态对调）
- 可以通过预处理 S 的每一位的下一个 $a, b, \cdots, z$ 的出现位置进行 $O(1)$ 的顺推转移。
- 复杂度 $O(m^2 + 26n)$ ，可以通过本题。

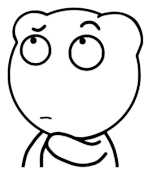


Problem B. 最长公共子序列

- 定义状态 $f_{i,j}$ 为 S 的前 i 位与 T 的前 j 位最长公共子序列，则有

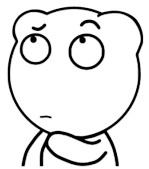
$$f_{i,j} = \begin{cases} \max(f_{i-1,j}, f_{i,j-1}) & , S_i \neq T_j \\ f_{i-1,j-1} + 1 & , S_i = T_j \end{cases}$$

- 上述做法的时间复杂度 $O(nm)$ ，无法通过本题。
- 最终答案不会超过 m 。
- 更改状态定义 $f_{i,j}$ 为与 T 前 i 位的最长公共子序列长度为 j 的 S 的最短前缀长度（即将朴素做法的答案与第一维状态对调）
- 可以通过预处理 S 的每一位的下一个 $a, b, \cdots, z$ 的出现位置进行 $O(1)$ 的顺推转移。
- 复杂度 $O(m^2 + 26n)$ ，可以通过本题。



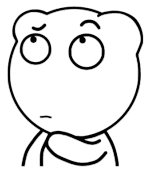
Problem C. 看不见黑板了

- 求右边第一个严格大于的数是一个经典的问题。
- 通过维护一个单调递减的栈来求解。
- 复杂度 $O(n)$
- 本题时限较大，使用其它数据结构（权值线段树/树状数组）也可以通过。



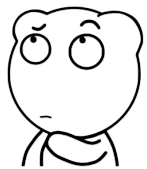
Problem C. 看不见黑板了

- 求右边第一个严格大于的数是一个经典的问题。
- 通过维护一个单调递减的栈来求解。
- 复杂度 $O(n)$
- 本题时限较大，使用其它数据结构（权值线段树/树状数组）也可以通过。



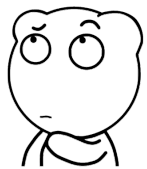
Problem C. 看不见黑板了

- 求右边第一个严格大于的数是一个经典的问题。
- 通过维护一个单调递减的栈来求解。
- 复杂度 $O(n)$
- 本题时限较大，使用其它数据结构（权值线段树/树状数组）也可以通过。



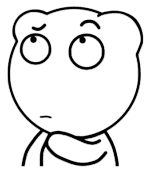
Problem C. 看不见黑板了

- 求右边第一个严格大于的数是一个经典的问题。
- 通过维护一个单调递减的栈来求解。
- 复杂度 $O(n)$
- 本题时限较大，使用其它数据结构（权值线段树/树状数组）也可以通过。



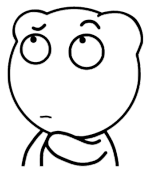
Problem D. 小喵的微波炉

- 按题意 BFS，并求出大于等于 t 的第一个可行解。
- 复杂度：$O(3600n)$



Problem D. 小喵的微波炉

- 按题意 BFS ，并求出大于等于 t 的第一个可行解。
- 复杂度：$O(3600n)$

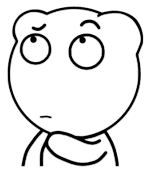


Problem E. 小喵的数列

- 原问题等价于求解第 k 个与 x 互质的数。
- 可以发现答案具有单调性，考虑二分答案。
- 求解 n 以内与 x 互质的数的个数。
- 根据容斥原理，答案为

$$\sum_{d|x} \mu(d) \lfloor \frac{n}{d} \rfloor$$

- 复杂度：$O(p \log p + t \log n \max(\tau(p)))$

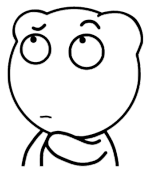


Problem E. 小喵的数列

- 原问题等价于求解第 k 个与 x 互质的数。
- 可以发现答案具有单调性，考虑二分答案。
- 求解 n 以内与 x 互质的数的个数。
- 根据容斥原理，答案为

$$\sum_{d|x} \mu(d) \lfloor \frac{n}{d} \rfloor$$

- 复杂度：$O(p \log p + t \log n \max(\tau(p)))$

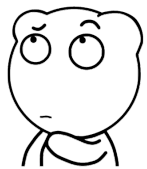


Problem E. 小喵的数列

- 原问题等价于求解第 k 个与 x 互质的数。
- 可以发现答案具有单调性，考虑二分答案。
- 求解 n 以内与 x 互质的数的个数。
- 根据容斥原理，答案为

$$\sum_{d|x} \mu(d) \lfloor \frac{n}{d} \rfloor$$

- 复杂度：$O(p \log p + t \log n \max(\tau(p)))$

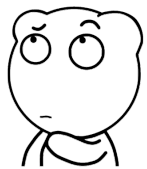


Problem E. 小喵的数列

- 原问题等价于求解第 k 个与 x 互质的数。
- 可以发现答案具有单调性，考虑二分答案。
- 求解 n 以内与 x 互质的数的个数。
- 根据容斥原理，答案为

$$\sum_{d|x} \mu(d) \lfloor \frac{n}{d} \rfloor$$

- 复杂度：$O(p \log p + t \log n \max(\tau(p)))$

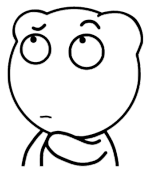


Problem E. 小喵的数列

- 原问题等价于求解第 k 个与 x 互质的数。
- 可以发现答案具有单调性，考虑二分答案。
- 求解 n 以内与 x 互质的数的个数。
- 根据容斥原理，答案为

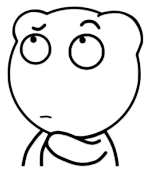
$$\sum_{d|x} \mu(d) \lfloor \frac{n}{d} \rfloor$$

- 复杂度：$O(p \log p + t \log n \max(\tau(p)))$



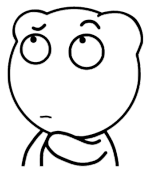
Problem F. 象棋比赛

- 依次考虑每一个人，判断他是否能成为冠军。
- 不妨设该选手剩余比赛全部获胜。那么问题变为：如何分配剩下与这位选手无关的比赛的胜负情况，使得其它选手的分数都不高于该选手。
- 这是一个经典的网络流模型，我们可以这样建模：
  - 源点 S 向比赛连边，容量为 2。（为了防止平局造成非整数的流量，统一将分数扩大一倍）
  - 每场比赛向对应的两个选手连边，容量为 ∞。
  - 每个选手向汇点 T 连边，容量为每个选手与该选手的分差。
- 如果最大流等于剩余比赛数量（的两倍），说明存在可行的分配方案，即该选手有机会成为冠军。
- 复杂度：$O(n^4)$



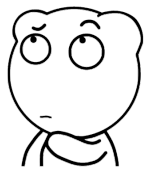
Problem F. 象棋比赛

- 依次考虑每一个人，判断他是否能成为冠军。
- 不妨设该选手剩余比赛全部获胜。那么问题变为：如何分配剩下与这位选手无关的比赛的胜负情况，使得其它选手的分数都不高于该选手。
- 这是一个经典的网络流模型，我们可以这样建模：
  - 源点 S 向比赛连边，容量为 2。（为了防止平局造成非整数的流量，统一将分数扩大一倍）
  - 每场比赛向对应的两个选手连边，容量为 ∞。
  - 每个选手向汇点 T 连边，容量为每个选手与该选手的分差。
- 如果最大流等于剩余比赛数量（的两倍），说明存在可行的分配方案，即该选手有机会成为冠军。
- 复杂度：$O(n^4)$



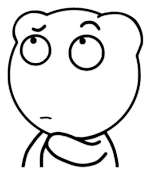
Problem F. 象棋比赛

- 依次考虑每一个人，判断他是否能成为冠军。
- 不妨设该选手剩余比赛全部获胜。那么问题变为：如何分配剩下与这位选手无关的比赛的胜负情况，使得其它选手的分数都不高于该选手。
- 这是一个经典的网络流模型，我们可以这样建模：
  - 源点 S 向比赛连边，容量为 2。（为了防止平局造成非整数的流量，统一将分数扩大一倍）
  - 每场比赛向对应的两个选手连边，容量为 ∞。
  - 每个选手向汇点 T 连边，容量为每个选手与该选手的分差。
- 如果最大流等于剩余比赛数量（的两倍），说明存在可行的分配方案，即该选手有机会成为冠军。
- 复杂度：$O(n^4)$



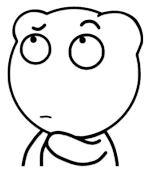
Problem F. 象棋比赛

- 依次考虑每一个人，判断他是否能成为冠军。
- 不妨设该选手剩余比赛全部获胜。那么问题变为：如何分配剩下与这位选手无关的比赛的胜负情况，使得其它选手的分数都不高于该选手。
- 这是一个经典的网络流模型，我们可以这样建模：
  - 源点 S 向比赛连边，容量为 2。（为了防止平局造成非整数的流量，统一将分数扩大一倍）
  - 每场比赛向对应的两个选手连边，容量为 ∞。
  - 每个选手向汇点 T 连边，容量为每个选手与该选手的分差。
- 如果最大流等于剩余比赛数量（的两倍），说明存在可行的分配方案，即该选手有机会成为冠军。
- 复杂度：$O(n^4)$



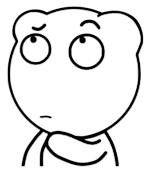
Problem F. 象棋比赛

- 依次考虑每一个人，判断他是否能成为冠军。
- 不妨设该选手剩余比赛全部获胜。那么问题变为：如何分配剩下与这位选手无关的比赛的胜负情况，使得其它选手的分数都不高于该选手。
- 这是一个经典的网络流模型，我们可以这样建模：
  - 源点 S 向比赛连边，容量为 2。（为了防止平局造成非整数的流量，统一将分数扩大一倍）
  - 每场比赛向对应的两个选手连边，容量为 ∞。
  - 每个选手向汇点 T 连边，容量为每个选手与该选手的分差。
- 如果最大流等于剩余比赛数量（的两倍），说明存在可行的分配方案，即该选手有机会成为冠军。
- 复杂度：$O(n^4)$



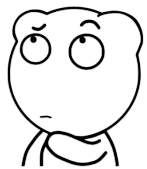
Problem F. 象棋比赛

- 依次考虑每一个人，判断他是否能成为冠军。
- 不妨设该选手剩余比赛全部获胜。那么问题变为：如何分配剩下与这位选手无关的比赛的胜负情况，使得其它选手的分数都不高于该选手。
- 这是一个经典的网络流模型，我们可以这样建模：
  - 源点 S 向比赛连边，容量为 2。（为了防止平局造成非整数的流量，统一将分数扩大一倍）
  - 每场比赛向对应的两个选手连边，容量为 ∞。
  - 每个选手向汇点 T 连边，容量为每个选手与该选手的分差。
- 如果最大流等于剩余比赛数量（的两倍），说明存在可行的分配方案，即该选手有机会成为冠军。
- 复杂度：$O(n^4)$



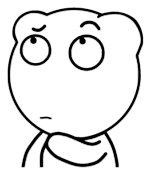
Problem F. 象棋比赛

- 依次考虑每一个人，判断他是否能成为冠军。
- 不妨设该选手剩余比赛全部获胜。那么问题变为：如何分配剩下与这位选手无关的比赛的胜负情况，使得其它选手的分数都不高于该选手。
- 这是一个经典的网络流模型，我们可以这样建模：
  - 源点 S 向比赛连边，容量为 2。（为了防止平局造成非整数的流量，统一将分数扩大一倍）
  - 每场比赛向对应的两个选手连边，容量为 ∞。
  - 每个选手向汇点 T 连边，容量为每个选手与该选手的分差。
- 如果最大流等于剩余比赛数量（的两倍），说明存在可行的分配方案，即该选手有机会成为冠军。
- 复杂度：$O(n^4)$



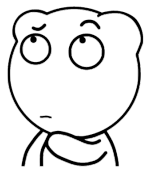
Problem F. 象棋比赛

- 依次考虑每一个人，判断他是否能成为冠军。
- 不妨设该选手剩余比赛全部获胜。那么问题变为：如何分配剩下与这位选手无关的比赛的胜负情况，使得其它选手的分数都不高于该选手。
- 这是一个经典的网络流模型，我们可以这样建模：
  - 源点 S 向比赛连边，容量为 2。（为了防止平局造成非整数的流量，统一将分数扩大一倍）
  - 每场比赛向对应的两个选手连边，容量为 ∞。
  - 每个选手向汇点 T 连边，容量为每个选手与该选手的分差。
- 如果最大流等于剩余比赛数量（的两倍），说明存在可行的分配方案，即该选手有机会成为冠军。
- 复杂度：$O(n^4)$



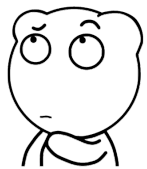
Problem G. Nim 游戏

- $\oplus_{i=0}^{n-1} a_i = 0$ 为必败态。
- 令 $\oplus_{i=0}^{n-1} a_i = k$，每次只要找到一个 i 满足 $a_i \oplus k < a_i$ 即可。
- 复杂度：$O(n \sum a_i)$



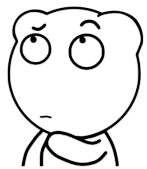
Problem G. Nim 游戏

- $\oplus_{i=0}^{n-1} a_i = 0$ **为必败态。**
- **令** $\oplus_{i=0}^{n-1} a_i = k$**，每次只要找到一个** i **满足** $a_i \oplus k < a_i$ **即可。**
- 复杂度：$O(n \sum a_i)$



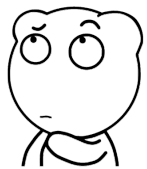
Problem G. Nim 游戏

- $\oplus_{i=0}^{n-1} a_i = 0$ **为必败态。**
- **令** $\oplus_{i=0}^{n-1} a_i = k$**，每次只要找到一个** i **满足** $a_i \oplus k < a_i$ **即可。**
- **复杂度：** $O(n \sum a_i)$



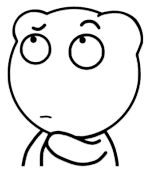
Problem H. 小喵的守护战

- 当坐标范围很小的时候，可以使用二维前缀和的方式解决。
- 根据这个启发，我们可以将一个询问拆分成四个前缀和的询问。
- 由于没有修改操作，我们可以**离线**询问。
- 将询问按纵坐标从小到大依次处理，处理的时候，将纵坐标小于等于询问的点加入树状数组，用树状数组维护前缀和。
- 也可以使用嵌套数据结构（树状数组套线段树）来求解，这样可以支持修改，但代码较繁琐。



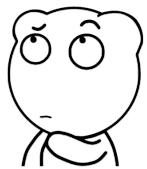
Problem H. 小喵的守护战

- 当坐标范围很小的时候，可以使用二维前缀和的方式解决。
- 根据这个启发，我们可以将一个询问拆分成四个前缀和的询问。
- 由于没有修改操作，我们可以离线询问。
- 将询问按纵坐标从小到大依次处理，处理的时候，将纵坐标小于等于询问的点加入树状数组，用树状数组维护前缀和。
- 也可以使用嵌套数据结构（树状数组套线段树）来求解，这样可以支持修改，但代码较繁琐。



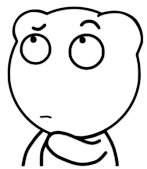
Problem H. 小喵的守护战

- 当坐标范围很小的时候，可以使用二维前缀和的方式解决。
- 根据这个启发，我们可以将一个询问拆分成四个前缀和的询问。
- 由于没有修改操作，我们可以**离线**询问。
- 将询问按纵坐标从小到大依次处理，处理的时候，将纵坐标小于等于询问的点加入树状数组，用树状数组维护前缀和。
- 也可以使用嵌套数据结构（树状数组套线段树）来求解，这样可以支持修改，但代码较繁琐。



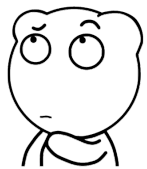
Problem H. 小喵的守护战

- 当坐标范围很小的时候，可以使用二维前缀和的方式解决。
- 根据这个启发，我们可以将一个询问拆分成四个前缀和的询问。
- 由于没有修改操作，我们可以**离线**询问。
- 将询问按纵坐标从小到大依次处理，处理的时候，将纵坐标小于等于询问的点加入树状数组，用树状数组维护前缀和。
- 也可以使用嵌套数据结构（树状数组套线段树）来求解，这样可以支持修改，但代码较繁琐。



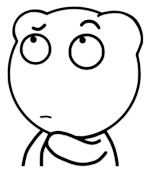
Problem H. 小喵的守护战

- 当坐标范围很小的时候，可以使用二维前缀和的方式解决。
- 根据这个启发，我们可以将一个询问拆分成四个前缀和的询问。
- 由于没有修改操作，我们可以**离线**询问。
- 将询问按纵坐标从小到大依次处理，处理的时候，将纵坐标小于等于询问的点加入树状数组，用树状数组维护前缀和。
- 也可以使用嵌套数据结构（树状数组套线段树）来求解，这样可以支持修改，但代码较繁琐。



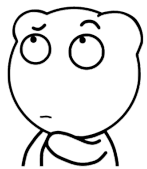
Problem I. 代码重构

- 按题意解析输入的内容。
- 维护一个大小为 65536 bool 数组即可。
- 有一些需要注意的边界条件在样例中已经给出。



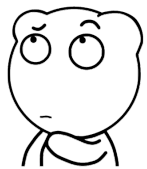
Problem I. 代码重构

- 按题意解析输入的内容。
- 维护一个大小为 65536 bool 数组即可。
- 有一些需要注意的边界条件在样例中已经给出。



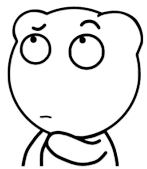
Problem I. 代码重构

- 按题意解析输入的内容。
- 维护一个大小为 65536 bool 数组即可。
- 有一些需要注意的边界条件在样例中已经给出。



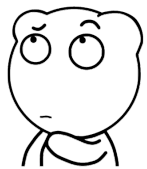
Problem J. 数星星

- **显然，最优策略一定是排序之后将连续的 $m-1$ 颗星星升级。**
- 使用前缀和或滑动窗口等方式实现即可。
- 使用桶排序可以做到线性复杂度。
- 复杂度：$O(n)$



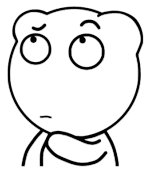
Problem J. 数星星

- 显然，最优策略一定是排序之后将连续的 $m-1$ 颗星星升级。
- 使用前缀和或滑动窗口等方式实现即可。
- 使用桶排序可以做到线性复杂度。
- 复杂度：$O(n)$



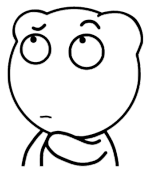
Problem J. 数星星

- 显然，最优策略一定是排序之后将连续的 $m-1$ 颗星星升级。
- 使用前缀和或滑动窗口等方式实现即可。
- 使用桶排序可以做到线性复杂度。
- 复杂度：$O(n)$



Problem J. 数星星

- 显然，最优策略一定是排序之后将连续的 $m-1$ 颗星星升级。
- 使用前缀和或滑动窗口等方式实现即可。
- 使用桶排序可以做到线性复杂度。
- 复杂度：$O(n)$



谢谢 ~