第 18 届上海大学程序设计联赛春季赛

试题分析

上海大学程序设计集训队

2020 年 4 月 18 日

通过情况

	通过人数	提交次数	首次通过
A	376	798	0:01
B	289	879	0:09
C	312	609	0:04
D	172	423	0:24
E	291	1059	0:03
F	217	856	0:48
G	1	25	4:59
H	3	117	1:21
I	1	7	3:14
J	0	8	N/A
K	0	6	N/A
L	4	14	1:50

Problem A. 组队比赛

难度： ★
最短裁判代码： 76B (Python3)
最短选手代码： 82B (Python3)

Problem A. 组队比赛

难度： ★
最短裁判代码： 76B (Python3)
最短选手代码： 82B (Python3)

- 可以证明，最强的和最弱的组队时，实力差距最小。

Problem A. 组队比赛

难度：★
最短裁判代码：76B (Python3)
最短选手代码：82B (Python3)

- 可以证明，最强的和最弱的组队时，实力差距最小。
- 可以使用排序，也可以直接枚举所有可能的组队方式。

Problem A. 组队比赛

难度：★
最短裁判代码：76B (Python3)
最短选手代码：82B (Python3)

- 可以证明，最强的和最弱的组队时，实力差距最小。
- 可以使用排序，也可以直接枚举所有可能的组队方式。
- 复杂度：$O(1)$

Problem B. 每日一报

难度： ★★
最短裁判代码： 793B (C++)
最短选手代码： 225B (Python3)

Problem B. 每日一报

难度： ★★
最短裁判代码： 793B (C++)
最短选手代码： 225B (Python3)

- 多关键字排序，正确设计和实现结构体和排序的比较函数即可。

Problem B. 每日一报

难度： ★★
最短裁判代码： 793B (C++)
最短选手代码： 225B (Python3)

- 多关键字排序，正确设计和实现结构体和排序的比较函数即可。
- 鼓励每一位同学会手写至少一种复杂度为 $O(n \log n)$ 的排序算法。

Problem B. 每日一报

难度：★★
最短裁判代码：793B (C++)
最短选手代码：225B (Python3)

- 多关键字排序，正确设计和实现结构体和排序的比较函数即可。
- 鼓励每一位同学会手写至少一种复杂度为 $O(n \log n)$ 的排序算法。
- 鼓励每一位同学学会调用至少一种语言中内置的排序算法。

Problem B. 每日一报

难度： ★★
最短裁判代码： 793B (C++)
最短选手代码： 225B (Python3)

- 多关键字排序，正确设计和实现结构体和排序的比较函数即可。
- 鼓励每一位同学会手写至少一种复杂度为 $O(n \log n)$ 的排序算法。
- 鼓励每一位同学学会调用至少一种语言中内置的排序算法。
- 复杂度： $O(n \log n)$ 或 $O(n^2)$

Problem C. 最长非公共子序列

难度： ★
最短裁判代码： 69B (Python3)
最短选手代码： 78B (Python3)

Problem C. 最长非公共子序列

难度： ★
最短裁判代码： 69B (Python3)
最短选手代码： 78B (Python3)

- 这是一道脑筋急转弯。

Problem C. 最长非公共子序列

难度： ★
最短裁判代码： 69B (Python3)
最短选手代码： 78B (Python3)

- 这是一道脑筋急转弯。
- 当 s_1 和 s_2 相同时，是找不到非公共子序列的，答案为 -1 。

Problem C. 最长非公共子序列

难度： ★
最短裁判代码： 69B (Python3)
最短选手代码： 78B (Python3)

- 这是一道脑筋急转弯。
- 当 s_1 和 s_2 相同时，是找不到非公共子序列的，答案为 -1 。
- 当 s_1 和 s_2 不同的时，我们发现：

Problem C. 最长非公共子序列

难度： ★
最短裁判代码： 69B (Python3)
最短选手代码： 78B (Python3)

- 这是一道脑筋急转弯。
- 当 s_1 和 s_2 相同时，是找不到非公共子序列的，答案为 -1 。
- 当 s_1 和 s_2 不同的时，我们发现：
  - 如果 s_1 和 s_2 长度不同，那么较长的一定是两者的非公共子序列；

Problem C. 最长非公共子序列

难度：★
最短裁判代码：69B (Python3)
最短选手代码：78B (Python3)

- 这是一道脑筋急转弯。
- 当 s_1 和 s_2 相同时，是找不到非公共子序列的，答案为 -1 。
- 当 s_1 和 s_2 不同的时，我们发现：
  - 如果 s_1 和 s_2 长度不同，那么较长的一定是两者的非公共子序列；
  - 如果 s_1 和 s_2 长度相同，那么两者都是彼此的非公共子序列。

Problem C. 最长非公共子序列

难度： ★
最短裁判代码： 69B (Python3)
最短选手代码： 78B (Python3)

- 这是一道脑筋急转弯。
- 当 s_1 和 s_2 相同时，是找不到非公共子序列的，答案为 -1 。
- 当 s_1 和 s_2 不同的时，我们发现：
  - 如果 s_1 和 s_2 长度不同，那么较长的一定是两者的非公共子序列；
  - 如果 s_1 和 s_2 长度相同，那么两者都是彼此的非公共子序列。
- 复杂度： $O(n)$

Problem D. 最大字符集

难度：★★
最短裁判代码：197B (Python3)
最短选手代码：188B (Python3)

Problem D. 最大字符集

难度： ★★
最短裁判代码： 197B (Python3)
最短选手代码： 188B (Python3)

- 当 $n \geq 3$ 时，我们可以构造一种答案为 $n-1$ 的方案：

  11
  101
  1001
  10001
  100001
  ...

Problem D. 最大字符集

难度：★★
最短裁判代码：197B (Python3)
最短选手代码：188B (Python3)

- 当 $n \geq 3$ 时，我们可以构造一种答案为 $n-1$ 的方案：

  11
  101
  1001
  10001
  100001
  ...
- 并且我们可以证明在 $n \geq 3$ 时不存在答案为 n 的方案。

Problem D. 最大字符集

难度：★★
最短裁判代码：197B (Python3)
最短选手代码：188B (Python3)

- 当 $n \geq 3$ 时，我们可以构造一种答案为 $n-1$ 的方案：

  11
  101
  1001
  10001
  100001
  $\cdots$

- 并且我们可以证明在 $n \geq 3$ 时不存在答案为 n 的方案。
- 同时需要特判 $n \leq 2$ 的情况。

Problem E. 美味的序列

难度： ★
最短裁判代码： 87B (Python3)
最短选手代码： 90B (Python3)

Problem E. 美味的序列

难度： ★
最短裁判代码： 87B (Python3)
最短选手代码： 90B (Python3)

- 吃的顺序根本不会影响最后的答案。

Problem E. 美味的序列

难度：★
最短裁判代码：87B (Python3)
最短选手代码：90B (Python3)

- 吃的顺序根本不会影响最后的答案。
- 因为不论怎么吃，美味度的减少总是一个以 0 为首项，1 为公差，长度为 n 的等差数列。

Problem E. 美味的序列

难度： ★
最短裁判代码： 87B (Python3)
最短选手代码： 90B (Python3)

- 吃的顺序根本不会影响最后的答案。
- 因为不论怎么吃，美味度的减少总是一个以 0 为首项，1 为公差，长度为 n 的等差数列。
- 答案为：$\sum_{i=1}^{n} a_i - i$ 。

Problem E. 美味的序列

难度：★
最短裁判代码：87B (Python3)
最短选手代码：90B (Python3)

- 吃的顺序根本不会影响最后的答案。
- 因为不论怎么吃，美味度的减少总是一个以 0 为首项，1 为公差，长度为 n 的等差数列。
- 答案为：$\sum_{i=1}^{n} a_i - i$ 。
- 注意数据范围，不要溢出。

Problem E. 美味的序列

难度：★
最短裁判代码：87B (Python3)
最短选手代码：90B (Python3)

- 吃的顺序根本不会影响最后的答案。
- 因为不论怎么吃，美味度的减少总是一个以 0 为首项，1 为公差，长度为 n 的等差数列。
- 答案为：$\sum_{i=1}^{n} a_i - i$ 。
- 注意数据范围，不要溢出。
- 复杂度：$O(n)$

Problem F. 日期小助手

难度：★★★
最短裁判代码：1816B (C++)
最短选手代码：798B (C)

Problem F. 日期小助手

难度： ★★★
最短裁判代码： 1816B (C++)
最短选手代码： 798B (C)

- 注意到 $365 \equiv 1 \bmod 7$。

Problem F. 日期小助手

难度：★★★
最短裁判代码：1816B (C++)
最短选手代码：798B (C)

- 注意到 $365 \equiv 1 \bmod 7$。
- 可以发现每年五月的第二个周日是在一个日期范围内，并且每年变化 1 天（闰年除外）。

Problem F. 日期小助手

难度：★★★
最短裁判代码：1816B (C++)
最短选手代码：798B (C)

- 注意到 $365 \equiv 1 \bmod 7$。
- 可以发现每年五月的第二个周日是在一个日期范围内，并且每年变化 1 天（闰年除外）。
- 预处理所有时间范围内的母亲节和父亲节的日期即可。

Problem F. 日期小助手

难度：★★★
最短裁判代码：1816B (C++)
最短选手代码：798B (C)

- 注意到 $365 \equiv 1 \bmod 7$。
- 可以发现每年五月的第二个周日是在一个日期范围内，并且每年变化 1 天（闰年除外）。
- 预处理所有时间范围内的母亲节和父亲节的日期即可。
- 也可以使用个别语言标准库内自带的日期类。

Problem F. 日期小助手

难度：★★★
最短裁判代码：1816B (C++)
最短选手代码：798B (C)

- 注意到 $365 \equiv 1 \bmod 7$。
- 可以发现每年五月的第二个周日是在一个日期范围内，并且每年变化 1 天（闰年除外）。
- 预处理所有时间范围内的母亲节和父亲节的日期即可。
- 也可以使用个别语言标准库内自带的日期类。
- 注意不要将 `21st` 错写成 `21th`，以及 2100 年不是闰年。

Problem F. 日期小助手

难度：★★★
最短裁判代码：1816B (C++)
最短选手代码：798B (C)

- 注意到 $365 \equiv 1 \bmod 7$。
- 可以发现每年五月的第二个周日是在一个日期范围内，并且每年变化 1 天（闰年除外）。
- 预处理所有时间范围内的母亲节和父亲节的日期即可。
- 也可以使用个别语言标准库内自带的日期类。
- 注意不要将 `21st` 错写成 `21th`，以及 2100 年不是闰年。
- 本题并不会出现 `21st` 以外以 `st` 结尾的日期，以及除 5 月和 6 月之外的月份，而这些情况均在已在样例中给出。

Problem G. 血压游戏

难度：★★★★
最短裁判代码：2540B (C++)
最短选手代码：1628B (C++)

Problem G. 血压游戏

难度：★★★★
最短裁判代码：2540B (C++)
最短选手代码：1628B (C++)

注意到每层之间是互不影响的，可以分别处理。

- 对于每种深度，我们只需要关心它们两两的 LCA，总点数是同阶的。

Problem G. 血压游戏

做法一

- 对于每种深度，我们只需要关心它们两两的 LCA，总点数是同阶的。
- 将同一深度的点拿出来建虚树，DFS 一遍统计答案即可。

Problem G. 血压游戏

做法一

- 对于每种深度，我们只需要关心它们两两的 LCA，总点数是同阶的。
- 将同一深度的点拿出来建虚树，DFS 一遍统计答案即可。
- 复杂度：$O(n \log n)$

Problem G. 血压游戏

做法二

- 考虑用 $f_{i,j}$ 记录以 i 号节点为根，深度为 j 的节点的答案。

Problem G. 血压游戏

做法二

- 考虑用 $f_{i,j}$ 记录以 i 号节点为根，深度为 j 的节点的答案。
- 没有 -1 操作的时候是动态开 f 数组启发式合并的经典问题。

Problem G. 血压游戏

做法二

- 考虑用 $f_{i,j}$ 记录以 i 号节点为根，深度为 j 的节点的答案。
- 没有 -1 操作的时候是动态开 f 数组启发式合并的经典问题。
- 注意到 -1 操作可以叠加执行，我们只要在 f 数组上加懒惰标记，那么这样就只需要在需要做更新操作的时候再做即可。

Problem G. 血压游戏

做法二

- 考虑用 $f_{i,j}$ 记录以 i 号节点为根，深度为 j 的节点的答案。
- 没有 -1 操作的时候是动态开 f 数组启发式合并的经典问题。
- 注意到 -1 操作可以叠加执行，我们只要在 f 数组上加懒惰标记，那么这样就只需要在需要做更新操作的时候再做即可。
- 由于同一层节点只会在他们的 LCA 处被合并一次，所以复杂度为：$O(n)$

Problem H. 纸牌游戏

难度：★★★
最短裁判代码：1999B (C++)
最短选手代码：868B (C++)

Problem H. 纸牌游戏

难度： ★★★
最短裁判代码： 1999B (C++)
最短选手代码： 868B (C++)

显然有一个贪心逐位确定的做法。

- 逐位确定的 check 方法如下：

Problem H. 纸牌游戏

做法一

- 逐位确定的 check 方法如下：
  - 设当前剩下的位中模 3 为 0，1，2 的个数为 s_0, s_1, s_2；

Problem H. 纸牌游戏

做法一

- 逐位确定的 check 方法如下：
  - 设当前剩下的位中模 3 为 0 ，1 ，2 的个数为 s_0, s_1, s_2 ；
  - 当前还需要的位数是 k ，需要在剩下的位中得到的余数为 m 。

Problem H. 纸牌游戏

做法一

- 逐位确定的 check 方法如下：
  - 设当前剩下的位中模 3 为 0，1，2 的个数为 s_0, s_1, s_2；
  - 当前还需要的位数是 k，需要在剩下的位中得到的余数为 m。
  - 先令 $s_i = \min(s_i, k)$；

Problem H. 纸牌游戏

做法一

- 逐位确定的 check 方法如下：
  - 设当前剩下的位中模 3 为 0，1，2 的个数为 s_0, s_1, s_2 ；
  - 当前还需要的位数是 k ，需要在剩下的位中得到的余数为 m 。
  - 先令 $s_i = \min(s_i, k)$ ；
    - 如果 $s_0 + s_1 + s_2 < k$ ，返回 `False`；

- 逐位确定的 check 方法如下:
  - 设当前剩下的位中模 3 为 0，1，2 的个数为 s_0, s_1, s_2；
  - 当前还需要的位数是 k，需要在剩下的位中得到的余数为 m。
  - 先令 $s_i = \min(s_i, k)$；
    - 如果 $s_0 + s_1 + s_2 < k$，返回 `False`;
    - 如果 $s_0 + s_1 + s_2 = k$，则直接判断 $s_1 + 2s_2 \equiv m \bmod 3$；

Problem H. 纸牌游戏

做法一

- 逐位确定的 check 方法如下：
  - 设当前剩下的位中模 3 为 0，1，2 的个数为 s_0, s_1, s_2；
  - 当前还需要的位数是 k，需要在剩下的位中得到的余数为 m。
  - 先令 $s_i = \min(s_i, k)$；
    - 如果 $s_0 + s_1 + s_2 < k$，返回 `False`；
    - 如果 $s_0 + s_1 + s_2 = k$，则直接判断 $s_1 + 2s_2 \equiv m \bmod 3$；
    - 如果 $s_0 \cdot s_1 \cdot s_2 \neq 0$，则返回 `True`。

Problem H. 纸牌游戏

做法一

- 逐位确定的 check 方法如下：
  - 设当前剩下的位中模 3 为 0，1，2 的个数为 s_0, s_1, s_2；
  - 当前还需要的位数是 k，需要在剩下的位中得到的余数为 m。
  - 先令 $s_i = \min(s_i, k)$；
    - 如果 $s_0 + s_1 + s_2 < k$，返回 `False`；
    - 如果 $s_0 + s_1 + s_2 = k$，则直接判断 $s_1 + 2s_2 \equiv m \bmod 3$；
    - 如果 $s_0 \cdot s_1 \cdot s_2 \neq 0$，则返回 `True`。
  - 否则最多存在两种余数，然后分类讨论：

Problem H. 纸牌游戏

做法一

- 逐位确定的 check 方法如下：
  - 设当前剩下的位中模 3 为 0，1，2 的个数为 s_0, s_1, s_2；
  - 当前还需要的位数是 k，需要在剩下的位中得到的余数为 m。
  - 先令 $s_i = \min(s_i, k)$；
    - 如果 $s_0 + s_1 + s_2 < k$，返回 `False`；
    - 如果 $s_0 + s_1 + s_2 = k$，则直接判断 $s_1 + 2s_2 \equiv m \bmod 3$；
    - 如果 $s_0 \cdot s_1 \cdot s_2 \neq 0$，则返回 `True`。
  - 否则最多存在两种余数，然后分类讨论：
    - 当余下的是 1 和 2 两种余数时，我们先算出 $s_1 + s_2 - k$ 的值 x，当 x 大于 1 时返回 `True`；

Problem H. 纸牌游戏

做法一

- 逐位确定的 check 方法如下：
  - 设当前剩下的位中模 3 为 0，1，2 的个数为 s_0, s_1, s_2；
  - 当前还需要的位数是 k，需要在剩下的位中得到的余数为 m。
  - 先令 $s_i = \min(s_i, k)$；
    - 如果 $s_0 + s_1 + s_2 < k$，返回 `False`；
    - 如果 $s_0 + s_1 + s_2 = k$，则直接判断 $s_1 + 2s_2 \equiv m \bmod 3$；
    - 如果 $s_0 \cdot s_1 \cdot s_2 \neq 0$，则返回 `True`。
  - 否则最多存在两种余数，然后分类讨论：
    - 当余下的是 1 和 2 两种余数时，我们先算出 $s_1 + s_2 - k$ 的值 x，当 x 大于 1 时返回 `True`；
    - 否则计算 $s_1 + 2s_2 + 1$ 与 $s_1 + 2s_2 + 2$ 是否与 m 模 3 同余即可。

Problem H. 纸牌游戏

做法一

- 逐位确定的 check 方法如下:
  - 设当前剩下的位中模 3 为 0 , 1 , 2 的个数为 s_0, s_1, s_2 ;
  - 当前还需要的位数是 k , 需要在剩下的位中得到的余数为 m 。
  - 先令 $s_i = \min(s_i, k)$;
    - 如果 $s_0 + s_1 + s_2 < k$, 返回 `False`;
    - 如果 $s_0 + s_1 + s_2 = k$, 则直接判断 $s_1 + 2s_2 \equiv m \bmod 3$;
    - 如果 $s_0 \cdot s_1 \cdot s_2 \neq 0$, 则返回 `True`。
  - 否则最多存在两种余数，然后分类讨论:
    - 当余下的是 1 和 2 两种余数时，我们先算出 $s_1 + s_2 - k$ 的值 x , 当 x 大于 1 时返回 `True`;
    - 否则计算 $s_1 + 2s_2 + 1$ 与 $s_1 + 2s_2 + 2$ 是否与 m 模 3 同余即可。
  - 其他情况类比上述做法。

- 逐位确定的 check 方法如下：
  - 设当前剩下的位中模 3 为 0，1，2 的个数为 s_0, s_1, s_2；
  - 当前还需要的位数是 k，需要在剩下的位中得到的余数为 m。
  - 先令 $s_i = \min(s_i, k)$；
    - 如果 $s_0 + s_1 + s_2 < k$，返回 `False`；
    - 如果 $s_0 + s_1 + s_2 = k$，则直接判断 $s_1 + 2s_2 \equiv m \bmod 3$；
    - 如果 $s_0 \cdot s_1 \cdot s_2 \neq 0$，则返回 `True`。
  - 否则最多存在两种余数，然后分类讨论：
    - 当余下的是 1 和 2 两种余数时，我们先算出 $s_1 + s_2 - k$ 的值 x，当 x 大于 1 时返回 `True`；
    - 否则计算 $s_1 + 2s_2 + 1$ 与 $s_1 + 2s_2 + 2$ 是否与 m 模 3 同余即可。
  - 其他情况类比上述做法。
- 也有其它类似的讨论方法。

Problem H. 纸牌游戏

做法一

- 逐位确定的 check 方法如下：
  - 设当前剩下的位中模 3 为 0，1，2 的个数为 s_0, s_1, s_2；
  - 当前还需要的位数是 k，需要在剩下的位中得到的余数为 m。
  - 先令 $s_i = \min(s_i, k)$；
    - 如果 $s_0 + s_1 + s_2 < k$，返回 `False`；
    - 如果 $s_0 + s_1 + s_2 = k$，则直接判断 $s_1 + 2s_2 \equiv m \bmod 3$；
    - 如果 $s_0 \cdot s_1 \cdot s_2 \neq 0$，则返回 `True`。
  - 否则最多存在两种余数，然后分类讨论：
    - 当余下的是 1 和 2 两种余数时，我们先算出 $s_1 + s_2 - k$ 的值 x，当 x 大于 1 时返回 `True`；
    - 否则计算 $s_1 + 2s_2 + 1$ 与 $s_1 + 2s_2 + 2$ 是否与 m 模 3 同余即可。
  - 其他情况类比上述做法。
- 也有其它类似的讨论方法。
- 复杂度：$O(n)$

Problem H. 纸牌游戏

做法二

- 从贪心的角度来考虑，肯定是大的数取得越多越好。

Problem H. 纸牌游戏

做法二

- 从贪心的角度来考虑，肯定是大的数取得越多越好。
- 这对之后的影响只有余数的变化，而余数只有 $0, 1, 2$ 三种取值。

Problem H. 纸牌游戏

做法二

- 从贪心的角度来考虑，肯定是大的数取得越多越好。
- 这对之后的影响只有余数的变化，而余数只有 $0, 1, 2$ 三种取值。
- 假设当前数取的上限为 x ，我们只需要考虑 $x-2, x-1, x$ 最多三种情况。

Problem H. 纸牌游戏

做法二

- 从贪心的角度来考虑，肯定是大的数取得越多越好。
- 这对之后的影响只有余数的变化，而余数只有 $0, 1, 2$ 三种取值。
- 假设当前数取的上限为 x ，我们只需要考虑 $x-2, x-1, x$ 最多三种情况。
- DFS 递归搜索即可。

Problem H. 纸牌游戏

做法二

- 从贪心的角度来考虑，肯定是大的数取得越多越好。
- 这对之后的影响只有余数的变化，而余数只有 $0, 1, 2$ 三种取值。
- 假设当前数取的上限为 x ，我们只需要考虑 $x-2, x-1, x$ 最多三种情况。
- DFS 递归搜索即可。
- 复杂度：$O(3^9+n)$

Problem I. 古老的打字机

难度：★★★
最短裁判代码：524B (Python3)
最短选手代码：1166B (C++)

Problem I. 古老的打字机

难度：★★★
最短裁判代码：524B (Python3)
最短选手代码：1166B (C++)

- 根据期望的线性性，考虑分开计算每个串的价值的期望，再求和。

Problem I. 古老的打字机

难度：★★★
最短裁判代码：524B (Python3)
最短选手代码：1166B (C++)

- 根据期望的线性性，考虑分开计算每个串的价值的期望，再求和。
- 注意到长度相同的串打出的概率是相同的，因此打出串的期望和给定的串的内容无关。

Problem I. 古老的打字机

难度：★★★
最短裁判代码：524B (Python3)
最短选手代码：1166B (C++)

- 根据期望的线性性，考虑分开计算每个串的价值的期望，再求和。
- 注意到长度相同的串打出的概率是相同的，因此打出串的期望和给定的串的内容无关。
- 对于一个长度为 x 的串，它在 s_i 的期望价值也可以通过期望的线性性计算，即考虑每个匹配位置的贡献：

$$\begin{cases} v_i \cdot (x - |s_i| + 1) \cdot 26^{x-|s_i|} & x \le |s_i| \\ 0 & x > |s_i| \end{cases}$$

Problem I. 古老的打字机 (续)

- 枚举打出的长度，根据期望的定义计算 s_i 对 t 的期望。

Problem I. 古老的打字机 (续)

- 枚举打出的长度，根据期望的定义计算 s_i 对 t 的期望。
- 这里我们需要得到在敲了 m 次得到长度为 x 的串的概率，这可以通过一个二维的 DP 来计算。

Problem I. 古老的打字机 (续)

- 枚举打出的长度，根据期望的定义计算 s_i 对 t 的期望。
- 这里我们需要得到在敲了 m 次得到长度为 x 的串的概率，这可以通过一个二维的 DP 来计算。
- $f_{i,j}$ 表示敲了 i 次得到长度为 j 的方案数，转移显然。

Problem I. 古老的打字机 (续)

- 枚举打出的长度，根据期望的定义计算 s_i 对 t 的期望。
- 这里我们需要得到在敲了 m 次得到长度为 x 的串的概率，这可以通过一个二维的 DP 来计算。
- $f_{i,j}$ 表示敲了 i 次得到长度为 j 的方案数，转移显然。
- 复杂度：$O(m^2 + \sum |s_i|)$

Problem J. 能到达吗

难度：★★★★★
最短裁判代码：3704B (C++)
最短选手代码：N/A

Problem J. 能到达吗

难度：★★★★★
最短裁判代码：3704B (C++)
最短选手代码：N/A

- 可以发现图中的联通块的数量为 $O(k)$ 。

Problem J. 能到达吗

难度：★★★★★
最短裁判代码：3704B (C++)
最短选手代码：N/A

- 可以发现图中的联通块的数量为 $O(k)$ 。
- 我们把每一行中相邻的两个障碍物之间的空地看成一个连通块。

Problem J. 能到达吗

难度：★★★★★
最短裁判代码：3704B (C++)
最短选手代码：N/A

- 可以发现图中的联通块的数量为 $O(k)$ 。
- 我们把每一行中相邻的两个障碍物之间的空地看成一个连通块。
- 通过双指针找出上一行中和当前这个联通块相交的连通块，然后并查集即可。

Problem J. 能到达吗

难度：★★★★★
最短裁判代码：3704B (C++)
最短选手代码：N/A

- 可以发现图中的联通块的数量为 $O(k)$ 。
- 我们把每一行中相邻的两个障碍物之间的空地看成一个连通块。
- 通过双指针找出上一行中和当前这个联通块相交的连通块，然后并查集即可。
- 对于一个大小为 x 的联通块，对答案的贡献为 $\binom{x}{1} + \binom{x}{2}$ 。

Problem J. 能到达吗

难度：★★★★★
最短裁判代码：3704B (C++)
最短选手代码：N/A

- 可以发现图中的联通块的数量为 $O(k)$ 。
- 我们把每一行中相邻的两个障碍物之间的空地看成一个连通块。
- 通过双指针找出上一行中和当前这个联通块相交的连通块，然后并查集即可。
- 对于一个大小为 x 的联通块，对答案的贡献为 $\binom{x}{1} + \binom{x}{2}$ 。
- 复杂度：$O(k \log k)$

Problem K. 迷宫

难度：★★★★★
最短裁判代码：3066B (C++)
最短选手代码：N/A

Problem K. 迷宫

难度： ★★★★★
最短裁判代码： 3066B (C++)
最短选手代码： N/A

- 分别从起点和终点进行 BFS 得到从起点出发到此处的最短路 $s_{i,j}$ 和此处到终点的最短路 $t_{i,j}$ 。

难度：★★★★★
最短裁判代码：3066B (C++)
最短选手代码：N/A

- 分别从起点和终点进行 BFS 得到从起点出发到此处的最短路 $s_{i,j}$ 和此处到终点的最短路 $t_{i,j}$ 。
- 由于“切比雪夫距离”的约束为一个正方形，因此，对于一个固定的点，我们可以穿越到达的点一定在一个正方形内。

Problem K. 迷宫

难度：★★★★★
最短裁判代码：3066B (C++)
最短选手代码：N/A

- 分别从起点和终点进行 BFS 得到从起点出发到此处的最短路 $s_{i,j}$ 和此处到终点的最短路 $t_{i,j}$ 。
- 由于“切比雪夫距离”的约束为一个正方形，因此，对于一个固定的点，我们可以穿越到达的点一定在一个正方形内。
- 问题就转化为对于每个起点能到达的点 (i_1, j_1) ，求它所覆盖的正方形范围的最小的 t_{i_2,j_2} 。

Problem K. 迷宫

难度：★★★★★
最短裁判代码：3066B (C++)
最短选手代码：N/A

- 分别从起点和终点进行 BFS 得到从起点出发到此处的最短路 $s_{i,j}$ 和此处到终点的最短路 $t_{i,j}$ 。
- 由于“切比雪夫距离”的约束为一个正方形，因此，对于一个固定的点，我们可以穿越到达的点一定在一个正方形内。
- 问题就转化为对于每个起点能到达的点 (i_1, j_1) ，求它所覆盖的正方形范围的最小的 t_{i_2,j_2} 。
- 所需步骤为 $s_{i_1,j_1} + 1 + t_{i_2,j_2}$ 。

Problem K. 迷宫 (续)

- 如何求正方形内的最小值？

Problem K. 迷宫 (续)

- 如何求正方形内的最小值？
  - 二维 ST 表是开不下的。

Problem K. 迷宫 (续)

- 如何求正方形内的最小值？
  - 二维 ST 表是开不下的。
  - 这是经典的滑动窗口最小值的二维版本，用单调队列做两次就好了。

Problem K. 迷宫 (续)

- 如何求正方形内的最小值？
  - 二维 ST 表是开不下的。
  - 这是经典的滑动窗口最小值的二维版本，用单调队列做两次就好了。
  - 当然你用一维的 ST 表做两次可能也是可以的。

Problem K. 迷宫 (续)

- 如何求正方形内的最小值？
  - 二维 ST 表是开不下的。
  - 这是经典的滑动窗口最小值的二维版本，用单调队列做两次就好了。
  - 当然你用一维的 ST 表做两次可能也是可以的。
- 至于输出方案，写就完事了⋯⋯

Problem K. 迷宫 (续)

- 如何求正方形内的最小值？
  - 二维 ST 表是开不下的。
  - 这是经典的滑动窗口最小值的二维版本，用单调队列做两次就好了。
  - 当然你用一维的 ST 表做两次可能也是可以的。
- 至于输出方案，写就完事了……
- 注意可能需要特殊处理的**不使用穿越**的情况。

Problem K. 迷宫 (续)

- 如何求正方形内的最小值？
  - 二维 ST 表是开不下的。
  - 这是经典的滑动窗口最小值的二维版本，用单调队列做两次就好了。
  - 当然你用一维的 ST 表做两次可能也是可以的。
- 至于输出方案，写就完事了⋯⋯
- 注意可能需要特殊处理的**不使用穿越**的情况。
- 对不使用穿越进行特判，或者观察到结论：当且仅当 $d = 0$ 时不使用穿越。

Problem K. 迷宫 (续)

- 如何求正方形内的最小值？
  - 二维 ST 表是开不下的。
  - 这是经典的滑动窗口最小值的二维版本，用单调队列做两次就好了。
  - 当然你用一维的 ST 表做两次可能也是可以的。
- 至于输出方案，写就完事了……
- 注意可能需要特殊处理的**不使用穿越**的情况。
- 对不使用穿越进行特判，或者观察到结论：当且仅当 $d = 0$ 时不使用穿越。
- 复杂度：$O(nm)$

Problem L. 动物森友会

难度：★★★
最短裁判代码：2771B (C++)
最短选手代码：504B (C++)

Problem L. 动物森友会

难度： ★★★
最短裁判代码： 2771B (C++)
最短选手代码： 504B (C++)

- 二分答案。

Problem L. 动物森友会

难度：★★★
最短裁判代码：2771B (C++)
最短选手代码：504B (C++)

- 二分答案。
- 一周的每一天对应一个点，每个事件对应一个点。事件往对应的天连边，每一天都有一个次数的限制。

Problem L. 动物森友会

难度：★★★
最短裁判代码：2771B (C++)
最短选手代码：504B (C++)

- 二分答案。
- 一周的每一天对应一个点，每个事件对应一个点。事件往对应的天连边，每一天都有一个次数的限制。
- 二分图匹配或最大流检查是否满足条件。

Problem L. 动物森友会

难度：★★★
最短裁判代码：2771B (C++)
最短选手代码：504B (C++)

- 二分答案。
- 一周的每一天对应一个点，每个事件对应一个点。事件往对应的天连边，每一天都有一个次数的限制。
- 二分图匹配或最大流检查是否满足条件。
- 注意估算二分上界。

Problem L. 动物森友会

难度：★★★
最短裁判代码：2771B (C++)
最短选手代码：504B (C++)

- 二分答案。
- 一周的每一天对应一个点，每个事件对应一个点。事件往对应的天连边，每一天都有一个次数的限制。
- 二分图匹配或最大流检查是否满足条件。
- 注意估算二分上界。
- 也可以根据 Hall 定理直接判断是否存在完备匹配。

Problem L. 动物森友会

难度：★★★
最短裁判代码：2771B (C++)
最短选手代码：504B (C++)

- 二分答案。
- 一周的每一天对应一个点，每个事件对应一个点。事件往对应的天连边，每一天都有一个次数的限制。
- 二分图匹配或最大流检查是否满足条件。
- 注意估算二分上界。
- 也可以根据 Hall 定理直接判断是否存在完备匹配。
- 复杂度：$O(\log(\sum c_i)\text{maxflow}(n+7,7n))$ 或 $O(\log(\sum c_i)\max\{2^7,n\})$

谢谢观看