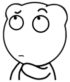
2019 Summer Newbie Selection Contest 3

Presentation of solutions

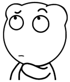
Three Fat House

August 20, 2019



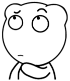
Difficulty

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



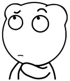
Statistics

	Solved	Attempted	First Solved
A	1	2	03:15 by 朱剑翔
B	0	1	N/A
C	9	59	01:52 by 王如嫣
D	27	43	00:08 by 王如嫣
E	4	41	01:42 by 孙逸融
F	0	4	N/A
G	27	60	00:04 by 孙逸融
H	2	40	00:59 by 孙逸融
I	1	16	04:29 by 孙逸融



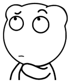
一些心得

- 时间复杂度的概念：判题机 1 秒可以进行约 10^8 次运算
- 至少要学会通过循环次数估计运行时间，提前判断做法会不会超时
- 签到题 “Ei” 写成 “E1” -> 可以复制题面上的输出
- “YE5”,“N0”
- 事实上，一片红的题可能比没人做的题更恐怖
- 准备一套自己熟悉的，正确实现的板子



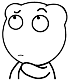
一些心得

- 时间复杂度的概念：判题机 1 秒可以进行约 10^8 次运算
- 至少要学会通过循环次数估计运行时间，提前判断做法会不会超时
- 签到题 “Ei” 写成 “E1” -> 可以复制题面上的输出
- “YE5”,“N0”
- 事实上，一片红的题可能比没人做的题更恐怖
- 准备一套自己熟悉的，正确实现的板子



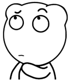
一些心得

- 时间复杂度的概念：判题机 1 秒可以进行约 10^8 次运算
- 至少要学会通过循环次数估计运行时间，提前判断做法会不会超时
- 签到题 “Ei” 写成 “E1” -> 可以复制题面上的输出
- “YE5”,“N0”
- 事实上，一片红的题可能比没人做的题更恐怖
- 准备一套自己熟悉的，正确实现的板子



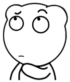
一些心得

- 时间复杂度的概念：判题机 1 秒可以进行约 10^8 次运算
- 至少要学会通过循环次数估计运行时间，提前判断做法会不会超时
- 签到题 “Ei” 写成 “E1” -> 可以复制题面上的输出
- “YE5”,“N0”
- 事实上，一片红的题可能比没人做的题更恐怖
- 准备一套自己熟悉的，正确实现的板子



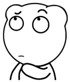
一些心得

- 时间复杂度的概念：判题机 1 秒可以进行约 10^8 次运算
- 至少要学会通过循环次数估计运行时间，提前判断做法会不会超时
- 签到题 “Ei” 写成 “E1” –> 可以复制题面上的输出
- “YE5”,“N0”
- 事实上，一片红的题可能比没人做的题更恐怖
- 准备一套自己熟悉的，正确实现的板子



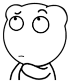
一些心得

- 时间复杂度的概念：判题机 1 秒可以进行约 10^8 次运算
- 至少要学会通过循环次数估计运行时间，提前判断做法会不会超时
- 签到题 “Ei” 写成 “E1” −> 可以复制题面上的输出
- “YE5”,“N0”
- 事实上，一片红的题可能比没人做的题更恐怖
- 准备一套自己熟悉的，正确实现的板子



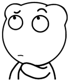
Problem D. 诶诶诶

- 寻找是否包含子序列“`father`”，不区分大小写
- 把所有字符转化成小写，计算最长公共子序列 LCS
- b 串是固定的，直接开 7 种状态，记录当前能匹配最大位数
- 送温暖的签到题，时间复杂度 $O(7 \cdot n)$
- 为了送温暖 $O(n \log n)$ 、$O(n^2)$ 同样可以通过



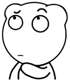
Problem D. 诶诶诶

- 寻找是否包含子序列 “`father`”，不区分大小写
- 把所有字符转化成小写，计算最长公共子序列 LCS
- b 串是固定的，直接开 7 种状态，记录当前能匹配最大位数
- 送温暖的签到题，时间复杂度 $O(7 \cdot n)$
- 为了送温暖 $O(n \log n)$ 、$O(n^2)$ 同样可以通过



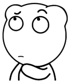
Problem D. 诶诶诶

- 寻找是否包含子序列“`father`”，不区分大小写
- 把所有字符转化成小写，计算最长公共子序列 LCS
- b 串是固定的，直接开 7 种状态，记录当前能匹配最大位数
- 送温暖的签到题，时间复杂度 $O(7 \cdot n)$
- 为了送温暖 $O(n \log n)$ 、$O(n^2)$ 同样可以通过



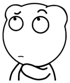
Problem D. 诶诶诶

- 寻找是否包含子序列 “`father`”，不区分大小写
- 把所有字符转化成小写，计算最长公共子序列 LCS
- b 串是固定的，直接开 7 种状态，记录当前能匹配最大位数
- 送温暖的签到题，时间复杂度 $O(7 \cdot n)$
- 为了送温暖 $O(n \log n)$ 、$O(n^2)$ 同样可以通过



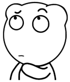
Problem D. 诶诶诶

- 寻找是否包含子序列 “`father`”，不区分大小写
- 把所有字符转化成小写，计算最长公共子序列 LCS
- b 串是固定的，直接开 7 种状态，记录当前能匹配最大位数
- 送温暖的签到题，时间复杂度 $O(7 \cdot n)$
- 为了送温暖 $O(n \log n)$ 、$O(n^2)$ 同样可以通过



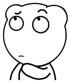
Problem G. CSL 的区间和

- 对于固定的数组求 $\sum_{i=l}^{r} a_i$
- 记 $s_0 = 0, s_n = \sum_{i=1}^{n} a_i$
- 则有 $s_i = s_{i-1} + a_i$ 可 $O(n)$ 预处理
- 且 $\sum_{i=l}^{r} a_i = s_r - s_{l-1}$
- 时间复杂度 $O(n+q)$ ，放宽到 $O(n \log n)$ 可过
- 因此，这也是一道线段树或树状数组裸题
- 答案可能爆 int
- $^{(*)}$ 若先有 p 个区间增减，再询问 q 个区间和，如何实现 $O(p+n+q)$ ？
- $^{(*)}$ $O(p)$ 打差分标记，$O(n)$ 计算结果，$O(q)$ 查询



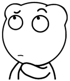
Problem G. CSL 的区间和

- 对于固定的数组求 $\sum_{i=l}^{r} a_i$
- 记 $s_0 = 0, s_n = \sum_{i=1}^{n} a_i$
- 则有 $s_i = s_{i-1} + a_i$ 可 $O(n)$ 预处理
- 且 $\sum_{i=l}^{r} a_i = s_r - s_{l-1}$
- 时间复杂度 $O(n+q)$ ，放宽到 $O(n \log n)$ 可过
- 因此，这也是一道线段树或树状数组裸题
- 答案可能爆 int
- $^{(*)}$ 若先有 p 个区间增减，再询问 q 个区间和，如何实现 $O(p+n+q)$ ？
- $^{(*)}$ $O(p)$ 打差分标记，$O(n)$ 计算结果，$O(q)$ 查询



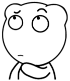
Problem G. CSL 的区间和

- 对于固定的数组求 $\sum_{i=l}^{r} a_i$
- 记 $s_0 = 0, s_n = \sum_{i=1}^{n} a_i$
- 则有 $s_i = s_{i-1} + a_i$ 可 $O(n)$ 预处理
- 且 $\sum_{i=l}^{r} a_i = s_r - s_{l-1}$
- 时间复杂度 $O(n+q)$ ，放宽到 $O(n \log n)$ 可过
- 因此，这也是一道线段树或树状数组裸题
- 答案可能爆 int
- $^{(*)}$ 若先有 p 个区间增减，再询问 q 个区间和，如何实现 $O(p+n+q)$ ？
- $^{(*)}$ $O(p)$ 打差分标记，$O(n)$ 计算结果，$O(q)$ 查询



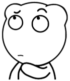
Problem G. CSL 的区间和

- 对于固定的数组求 $\sum_{i=l}^{r} a_i$
- 记 $s_0 = 0, s_n = \sum_{i=1}^{n} a_i$
- 则有 $s_i = s_{i-1} + a_i$ 可 $O(n)$ 预处理
- 且 $\sum_{i=l}^{r} a_i = s_r - s_{l-1}$
- 时间复杂度 $O(n+q)$ ，放宽到 $O(n \log n)$ 可过
- 因此，这也是一道线段树或树状数组裸题
- 答案可能爆 int
- $^{(*)}$ 若先有 p 个区间增减，再询问 q 个区间和，如何实现 $O(p+n+q)$ ？
- $^{(*)}$ $O(p)$ 打差分标记，$O(n)$ 计算结果，$O(q)$ 查询



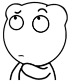
Problem G. CSL 的区间和

- 对于固定的数组求 $\sum_{i=l}^{r} a_i$
- 记 $s_0 = 0, s_n = \sum_{i=1}^{n} a_i$
- 则有 $s_i = s_{i-1} + a_i$ 可 $O(n)$ 预处理
- 且 $\sum_{i=l}^{r} a_i = s_r - s_{l-1}$
- 时间复杂度 $O(n+q)$，放宽到 $O(n \log n)$ 可过
- 因此，这也是一道线段树或树状数组裸题
- 答案可能爆 int
- $^{(*)}$ 若先有 p 个区间增减，再询问 q 个区间和，如何实现 $O(p+n+q)$ ？
- $^{(*)}$ $O(p)$ 打差分标记，$O(n)$ 计算结果，$O(q)$ 查询



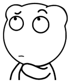
Problem G. CSL 的区间和

- 对于固定的数组求 $\sum_{i=l}^{r} a_i$
- 记 $s_0 = 0, s_n = \sum_{i=1}^{n} a_i$
- 则有 $s_i = s_{i-1} + a_i$ 可 $O(n)$ 预处理
- 且 $\sum_{i=l}^{r} a_i = s_r - s_{l-1}$
- 时间复杂度 $O(n+q)$，放宽到 $O(n \log n)$ 可过
- 因此，这也是一道线段树或树状数组裸题
- 答案可能爆 int
- $^{(*)}$ 若先有 p 个区间增减，再询问 q 个区间和，如何实现 $O(p+n+q)$？
- $^{(*)}$ $O(p)$ 打差分标记，$O(n)$ 计算结果，$O(q)$ 查询



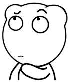
Problem G. CSL 的区间和

- 对于固定的数组求 $\sum_{i=l}^{r} a_i$
- 记 $s_0 = 0, s_n = \sum_{i=1}^{n} a_i$
- 则有 $s_i = s_{i-1} + a_i$ 可 $O(n)$ 预处理
- 且 $\sum_{i=l}^{r} a_i = s_r - s_{l-1}$
- 时间复杂度 $O(n+q)$ ，放宽到 $O(n \log n)$ 可过
- 因此，这也是一道线段树或树状数组裸题
- 答案可能爆 int
- $^{(*)}$ 若先有 p 个区间增减，再询问 q 个区间和，如何实现 $O(p+n+q)$ ？
- $^{(*)}$ $O(p)$ 打差分标记，$O(n)$ 计算结果，$O(q)$ 查询



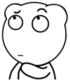
Problem G. CSL 的区间和

- 对于固定的数组求 $\sum_{i=l}^{r} a_i$
- 记 $s_0 = 0, s_n = \sum_{i=1}^{n} a_i$
- 则有 $s_i = s_{i-1} + a_i$ 可 $O(n)$ 预处理
- 且 $\sum_{i=l}^{r} a_i = s_r - s_{l-1}$
- 时间复杂度 $O(n+q)$ ，放宽到 $O(n \log n)$ 可过
- 因此，这也是一道线段树或树状数组裸题
- 答案可能爆 int
- $^{(*)}$ 若先有 p 个区间增减，再询问 q 个区间和，如何实现 $O(p+n+q)$ ？
- $^{(*)}$ $O(p)$ 打差分标记，$O(n)$ 计算结果，$O(q)$ 查询



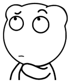
Problem G. CSL 的区间和

- 对于固定的数组求 $\sum_{i=l}^{r} a_i$
- 记 $s_0 = 0, s_n = \sum_{i=1}^{n} a_i$
- 则有 $s_i = s_{i-1} + a_i$ 可 $O(n)$ 预处理
- 且 $\sum_{i=l}^{r} a_i = s_r - s_{l-1}$
- 时间复杂度 $O(n+q)$ ，放宽到 $O(n \log n)$ 可过
- 因此，这也是一道线段树或树状数组裸题
- 答案可能爆 int
- $^{(*)}$ 若先有 p 个区间增减，再询问 q 个区间和，如何实现 $O(p+n+q)$ ？
- $^{(*)}$ $O(p)$ 打差分标记，$O(n)$ 计算结果，$O(q)$ 查询



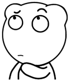
Problem C. 异或

- **对于给定的 x 求满足 $x \oplus a \oplus b$ 取得最大值的同时，$|a-b|$ 取得最小值的方案数**
- $x \oplus a \oplus b = 2^{31} - 1$
- 对 x, a, b 进行二进制拆分
  - 若 $x_i = 0$，$a_i \oplus b_i = 1$，此时 a_i 与 b_i 相反
  - 若 $x_i = 1$，$a_i \oplus b_i = 0$，此时 a_i 与 b_i 相同
  - $x_i = 0$ 会对 $|a-b|$ 产生 $\pm 2^i$ 的贡献
  - $x_i = 1$ 会对 $|a-b|$ 产生 0 贡献
- 贪心让 $|a-b|$ 最小，$x_i = 0$ 的分配方案是唯二的
- “`cout << (1ll << (__builtin_popcount(x) + 1)) << endl;`”
- 注意 $2^{31} - 1$ 二进制下没有 0
- 时间复杂度 $O(1)$ 或 $O(31)$



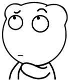
Problem C. 异或

- **对于给定的 x 求满足 $x \oplus a \oplus b$ 取得最大值的同时，$|a-b|$ 取得最小值的方案数**
- $x \oplus a \oplus b = 2^{31} - 1$
- 对 x, a, b 进行二进制拆分
  - 若 $x_i = 0$，$a_i \oplus b_i = 1$，此时 a_i 与 b_i 相反
  - 若 $x_i = 1$，$a_i \oplus b_i = 0$，此时 a_i 与 b_i 相同
  - $x_i = 0$ 会对 $|a-b|$ 产生 $\pm 2^i$ 的贡献
  - $x_i = 1$ 会对 $|a-b|$ 产生 0 贡献
- 贪心让 $|a-b|$ 最小，$x_i = 0$ 的分配方案是唯二的
- “`cout << (1ll << (__builtin_popcount(x) + 1)) << endl;`”
- 注意 $2^{31} - 1$ 二进制下没有 0
- 时间复杂度 $O(1)$ 或 $O(31)$



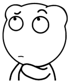
Problem C. 异或

- 对于给定的 x 求满足 $x \oplus a \oplus b$ 取得最大值的同时，$|a-b|$ 取得最小值的方案数
- $x \oplus a \oplus b = 2^{31} - 1$
- 对 x, a, b 进行二进制拆分
  - 若 $x_i = 0$，$a_i \oplus b_i = 1$，此时 a_i 与 b_i 相反
  - 若 $x_i = 1$，$a_i \oplus b_i = 0$，此时 a_i 与 b_i 相同
  - $x_i = 0$ 会对 $|a-b|$ 产生 $\pm 2^i$ 的贡献
  - $x_i = 1$ 会对 $|a-b|$ 产生 0 贡献
- 贪心让 $|a-b|$ 最小，$x_i = 0$ 的分配方案是唯二的
- “`cout << (1ll << (__builtin_popcount(x) + 1)) << endl;`”
- 注意 $2^{31} - 1$ 二进制下没有 0
- 时间复杂度 $O(1)$ 或 $O(31)$



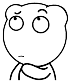
Problem C. 异或

- 对于给定的 x 求满足 $x \oplus a \oplus b$ 取得最大值的同时，$|a-b|$ 取得最小值的方案数
- $x \oplus a \oplus b = 2^{31} - 1$
- 对 x, a, b 进行二进制拆分
  - 若 $x_i = 0$ ，$a_i \oplus b_i = 1$ ，此时 a_i 与 b_i 相反
  - 若 $x_i = 1$ ，$a_i \oplus b_i = 0$ ，此时 a_i 与 b_i 相同
  - $x_i = 0$ 会对 $|a-b|$ 产生 $\pm 2^i$ 的贡献
  - $x_i = 1$ 会对 $|a-b|$ 产生 0 贡献
- 贪心让 $|a-b|$ 最小，$x_i = 0$ 的分配方案是唯二的
- “cout << (1ll << (__builtin_popcount(x) + 1)) << endl;”
- 注意 $2^{31} - 1$ 二进制下没有 0
- 时间复杂度 $O(1)$ 或 $O(31)$



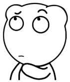
Problem C. 异或

- 对于给定的 x 求满足 $x \oplus a \oplus b$ 取得最大值的同时，$|a-b|$ 取得最小值的方案数
- $x \oplus a \oplus b = 2^{31}-1$
- 对 x, a, b 进行二进制拆分
  - 若 $x_i = 0$ ，$a_i \oplus b_i = 1$ ，此时 a_i 与 b_i 相反
  - 若 $x_i = 1$ ，$a_i \oplus b_i = 0$ ，此时 a_i 与 b_i 相同
  - $x_i = 0$ 会对 $|a-b|$ 产生 $\pm 2^i$ 的贡献
  - $x_i = 1$ 会对 $|a-b|$ 产生 0 贡献
- 贪心让 $|a-b|$ 最小，$x_i = 0$ 的分配方案是唯二的
- “`cout << (1ll << (__builtin_popcount(x) + 1)) << endl;`”
- 注意 $2^{31}-1$ 二进制下没有 0
- 时间复杂度 $O(1)$ 或 $O(31)$



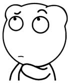
Problem C. 异或

- 对于给定的 x 求满足 $x \oplus a \oplus b$ 取得最大值的同时，$|a-b|$ 取得最小值的方案数
- $x \oplus a \oplus b = 2^{31} - 1$
- 对 x, a, b 进行二进制拆分
  - 若 $x_i = 0$ ，$a_i \oplus b_i = 1$ ，此时 a_i 与 b_i 相反
  - 若 $x_i = 1$ ，$a_i \oplus b_i = 0$ ，此时 a_i 与 b_i 相同
  - $x_i = 0$ 会对 $|a-b|$ 产生 $\pm 2^i$ 的贡献
  - $x_i = 1$ 会对 $|a-b|$ 产生 0 贡献
- 贪心让 $|a-b|$ 最小，$x_i = 0$ 的分配方案是唯二的
- “`cout << (1ll << (__builtin_popcount(x) + 1)) << endl;`”
- 注意 $2^{31} - 1$ 二进制下没有 0
- 时间复杂度 $O(1)$ 或 $O(31)$



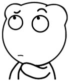
Problem C. 异或

- 对于给定的 x 求满足 $x \oplus a \oplus b$ 取得最大值的同时，$|a-b|$ 取得最小值的方案数
- $x \oplus a \oplus b = 2^{31} - 1$
- 对 x, a, b 进行二进制拆分
  - 若 $x_i = 0$ ，$a_i \oplus b_i = 1$ ，此时 a_i 与 b_i 相反
  - 若 $x_i = 1$ ，$a_i \oplus b_i = 0$ ，此时 a_i 与 b_i 相同
  - $x_i = 0$ 会对 $|a-b|$ 产生 $\pm 2^i$ 的贡献
  - $x_i = 1$ 会对 $|a-b|$ 产生 0 贡献
- 贪心让 $|a-b|$ 最小，$x_i = 0$ 的分配方案是唯二的
- “`cout << (1ll << (__builtin_popcount(x) + 1)) << endl;`”
- 注意 $2^{31} - 1$ 二进制下没有 0
- 时间复杂度 $O(1)$ 或 $O(31)$



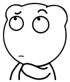
Problem C. 异或

- 对于给定的 x 求满足 $x \oplus a \oplus b$ 取得最大值的同时，$|a-b|$ 取得最小值的方案数
- $x \oplus a \oplus b = 2^{31} - 1$
- 对 x, a, b 进行二进制拆分
  - 若 $x_i = 0$ ，$a_i \oplus b_i = 1$ ，此时 a_i 与 b_i 相反
  - 若 $x_i = 1$ ，$a_i \oplus b_i = 0$ ，此时 a_i 与 b_i 相同
  - $x_i = 0$ 会对 $|a-b|$ 产生 $\pm 2^i$ 的贡献
  - $x_i = 1$ 会对 $|a-b|$ 产生 0 贡献
- 贪心让 $|a-b|$ 最小，$x_i = 0$ 的分配方案是唯二的
- “`cout << (1ll << (__builtin_popcount(x) + 1)) << endl;`”
- 注意 $2^{31} - 1$ 二进制下没有 0
- 时间复杂度 $O(1)$ 或 $O(31)$



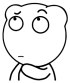
Problem C. 异或

- 对于给定的 x 求满足 $x \oplus a \oplus b$ 取得最大值的同时，$|a-b|$ 取得最小值的方案数
- $x \oplus a \oplus b = 2^{31} - 1$
- 对 x, a, b 进行二进制拆分
  - 若 $x_i = 0$ ，$a_i \oplus b_i = 1$ ，此时 a_i 与 b_i 相反
  - 若 $x_i = 1$ ，$a_i \oplus b_i = 0$ ，此时 a_i 与 b_i 相同
  - $x_i = 0$ 会对 $|a-b|$ 产生 $\pm 2^i$ 的贡献
  - $x_i = 1$ 会对 $|a-b|$ 产生 0 贡献
- 贪心让 $|a-b|$ 最小，$x_i = 0$ 的分配方案是唯二的
- “`cout << (1ll << (__builtin_popcount(x) + 1)) << endl;`”
- 注意 $2^{31} - 1$ 二进制下没有 0
- 时间复杂度 $O(1)$ 或 $O(31)$



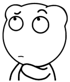
Problem C. 异或

- 对于给定的 x 求满足 $x \oplus a \oplus b$ 取得最大值的同时，$|a-b|$ 取得最小值的方案数
- $x \oplus a \oplus b = 2^{31} - 1$
- 对 x, a, b 进行二进制拆分
  - 若 $x_i = 0$ ，$a_i \oplus b_i = 1$ ，此时 a_i 与 b_i 相反
  - 若 $x_i = 1$ ，$a_i \oplus b_i = 0$ ，此时 a_i 与 b_i 相同
  - $x_i = 0$ 会对 $|a-b|$ 产生 $\pm 2^i$ 的贡献
  - $x_i = 1$ 会对 $|a-b|$ 产生 0 贡献
- 贪心让 $|a-b|$ 最小，$x_i = 0$ 的分配方案是唯二的
- “`cout << (1ll << (__builtin_popcount(x) + 1)) << endl;`”
- 注意 $2^{31} - 1$ 二进制下没有 0
- 时间复杂度 $O(1)$ 或 $O(31)$



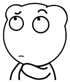
Problem C. 异或

- 对于给定的 x 求满足 $x \oplus a \oplus b$ 取得最大值的同时，$|a-b|$ 取得最小值的方案数
- $x \oplus a \oplus b = 2^{31}-1$
- 对 x, a, b 进行二进制拆分
  - 若 $x_i = 0$ ，$a_i \oplus b_i = 1$ ，此时 a_i 与 b_i 相反
  - 若 $x_i = 1$ ，$a_i \oplus b_i = 0$ ，此时 a_i 与 b_i 相同
  - $x_i = 0$ 会对 $|a-b|$ 产生 $\pm 2^i$ 的贡献
  - $x_i = 1$ 会对 $|a-b|$ 产生 0 贡献
- 贪心让 $|a-b|$ 最小，$x_i = 0$ 的分配方案是唯二的
- “`cout << (1ll << (__builtin_popcount(x) + 1)) << endl;`”
- 注意 $2^{31}-1$ 二进制下没有 0
- 时间复杂度 $O(1)$ 或 $O(31)$



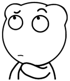
Problem E. 秦神的签到题

- $f(l,r)$ 表示序列 A 区间 $[l,r]$ 中不同数的数值总和，求 $\sum_{l=1}^{n}\sum_{r=l}^{n} f(l,r)$
- $pos[i]$ 表示数 i 上一次出现时的下标，当 i 第一次出现时记为 0
- A_i 对答案的贡献为 $A_i \cdot (i - pos[i]) \cdot (n - i + 1)$
- 如 $[2,1,2,3]$ 第一个 2 贡献区间为 $[1,1],[1,2],[1,3],[1,4]$
- 第二个 2 贡献区间为 $[2,3],[2,4],[3,3],[3,4]$
- 每个区间贡献为 2
- 注意取模的过程
- 时间复杂度 $O(n)$



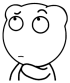
Problem E. 秦神的签到题

- $f(l,r)$ 表示序列 A 区间 $[l,r]$ 中不同数的数值总和，求 $\sum_{l=1}^{n}\sum_{r=l}^{n} f(l,r)$
- $pos[i]$ 表示数 i 上一次出现时的下标，当 i 第一次出现时记为 0
- A_i 对答案的贡献为 $A_i \cdot (i - pos[i]) \cdot (n - i + 1)$
- 如 $[2,1,2,3]$ 第一个 2 贡献区间为 $[1,1],[1,2],[1,3],[1,4]$
- 第二个 2 贡献区间为 $[2,3],[2,4],[3,3],[3,4]$
- 每个区间贡献为 2
- 注意取模的过程
- 时间复杂度 $O(n)$



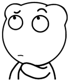
Problem E. 秦神的签到题

- $f(l,r)$ 表示序列 A 区间 $[l,r]$ 中不同数的数值总和，求 $\sum_{l=1}^{n}\sum_{r=l}^{n} f(l,r)$
- $pos[i]$ 表示数 i 上一次出现时的下标，当 i 第一次出现时记为 0
- A_i 对答案的贡献为 $A_i \cdot (i - pos[i]) \cdot (n - i + 1)$
- 如 $[2,1,2,3]$ 第一个 2 贡献区间为 $[1,1],[1,2],[1,3],[1,4]$
- 第二个 2 贡献区间为 $[2,3],[2,4],[3,3],[3,4]$
- 每个区间贡献为 2
- 注意取模的过程
- 时间复杂度 $O(n)$



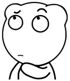
Problem E. 秦神的签到题

- $f(l, r)$ 表示序列 A 区间 $[l, r]$ 中不同数的数值总和，求 $\sum_{l=1}^{n} \sum_{r=l}^{n} f(l, r)$
- $pos[i]$ 表示数 i 上一次出现时的下标，当 i 第一次出现时记为 0
- A_i 对答案的贡献为 $A_i \cdot (i - pos[i]) \cdot (n - i + 1)$
- 如 $[2, 1, 2, 3]$ 第一个 2 贡献区间为 $[1, 1], [1, 2], [1, 3], [1, 4]$
- 第二个 2 贡献区间为 $[2, 3], [2, 4], [3, 3], [3, 4]$
- 每个区间贡献为 2
- 注意取模的过程
- 时间复杂度 $O(n)$



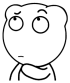
Problem E. 秦神的签到题

- $f(l,r)$ 表示序列 A 区间 $[l,r]$ 中不同数的数值总和，求 $\sum_{l=1}^{n}\sum_{r=l}^{n} f(l,r)$
- $pos[i]$ 表示数 i 上一次出现时的下标，当 i 第一次出现时记为 0
- A_i 对答案的贡献为 $A_i \cdot (i - pos[i]) \cdot (n - i + 1)$
- 如 $[2,1,2,3]$ 第一个 2 贡献区间为 $[1,1],[1,2],[1,3],[1,4]$
- 第二个 2 贡献区间为 $[2,3],[2,4],[3,3],[3,4]$
- 每个区间贡献为 2
- 注意取模的过程
- 时间复杂度 $O(n)$



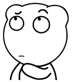
Problem E. 秦神的签到题

- $f(l,r)$ 表示序列 A 区间 $[l,r]$ 中不同数的数值总和，求 $\sum_{l=1}^{n}\sum_{r=l}^{n} f(l,r)$
- $pos[i]$ 表示数 i 上一次出现时的下标，当 i 第一次出现时记为 0
- A_i 对答案的贡献为 $A_i \cdot (i - pos[i]) \cdot (n - i + 1)$
- 如 $[2,1,2,3]$ 第一个 2 贡献区间为 $[1,1],[1,2],[1,3],[1,4]$
- 第二个 2 贡献区间为 $[2,3],[2,4],[3,3],[3,4]$
- 每个区间贡献为 2
- 注意取模的过程
- 时间复杂度 $O(n)$



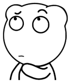
Problem E. 秦神的签到题

- $f(l,r)$ 表示序列 A 区间 $[l,r]$ 中不同数的数值总和，求 $\sum_{l=1}^{n}\sum_{r=l}^{n} f(l,r)$
- $pos[i]$ 表示数 i 上一次出现时的下标，当 i 第一次出现时记为 0
- A_i 对答案的贡献为 $A_i \cdot (i - pos[i]) \cdot (n - i + 1)$
- 如 $[2,1,2,3]$ 第一个 2 贡献区间为 $[1,1],[1,2],[1,3],[1,4]$
- 第二个 2 贡献区间为 $[2,3],[2,4],[3,3],[3,4]$
- 每个区间贡献为 2
- 注意取模的过程
- 时间复杂度 $O(n)$



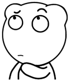
Problem E. 秦神的签到题

- $f(l,r)$ 表示序列 A 区间 $[l,r]$ 中不同数的数值总和，求 $\sum_{l=1}^{n}\sum_{r=l}^{n} f(l,r)$
- $pos[i]$ 表示数 i 上一次出现时的下标，当 i 第一次出现时记为 0
- A_i 对答案的贡献为 $A_i \cdot (i - pos[i]) \cdot (n - i + 1)$
- 如 $[2,1,2,3]$ 第一个 2 贡献区间为 $[1,1],[1,2],[1,3],[1,4]$
- 第二个 2 贡献区间为 $[2,3],[2,4],[3,3],[3,4]$
- 每个区间贡献为 2
- 注意取模的过程
- 时间复杂度 $O(n)$



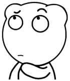
Problem A. 推箱子

- 给你一个 12×12 的地图和至多两个箱子，求将所有箱子推到目的地的最短时间及方案
- 简单 BFS，按题意模拟即可
- 技巧 1：只有一个箱子等价于将第二个箱子卡在地图之外的地方（并且这个地方恰好是一个目的地）
- 技巧 2：可以给地图中每一个点编号 $1 \sim 144$ ，设计函数转移。这样做不仅能减少状态维数，还能在记录路径时提供方便
- 需特别注意的情况有：人或箱子撞（穿）墙、出界，两个箱子卡在一起



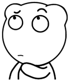
Problem A. 推箱子

- 给你一个 12×12 的地图和至多两个箱子，求将所有箱子推到目的地的最短时间及方案
- 简单 BFS，按题意模拟即可
- 技巧 1：只有一个箱子等价于将第二个箱子卡在地图之外的地方(并且这个地方恰好是一个目的地)
- 技巧 2：可以给地图中每一个点编号 $1 \sim 144$ ，设计函数转移。这样做不仅能减少状态维数，还能在记录路径时提供方便
- 需特别注意的情况有：人或箱子撞（穿）墙、出界，两个箱子卡在一起



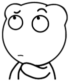
Problem A. 推箱子

- 给你一个 12×12 的地图和至多两个箱子，求将所有箱子推到目的地的最短时间及方案
- 简单 BFS，按题意模拟即可
- 技巧 1：只有一个箱子等价于将第二个箱子卡在地图之外的地方(并且这个地方恰好是一个目的地)
- 技巧 2：可以给地图中每一个点编号 $1 \sim 144$ ，设计函数转移。这样做不仅能减少状态维数，还能在记录路径时提供方便
- 需特别注意的情况有：人或箱子撞（穿）墙、出界，两个箱子卡在一起



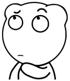
Problem A. 推箱子

- 给你一个 12×12 的地图和至多两个箱子，求将所有箱子推到目的地的最短时间及方案
- 简单 BFS，按题意模拟即可
- 技巧 1：只有一个箱子等价于将第二个箱子卡在地图之外的地方 (并且这个地方恰好是一个目的地)
- 技巧 2：可以给地图中每一个点编号 $1 \sim 144$ ，设计函数转移。这样做不仅能减少状态维数，还能在记录路径时提供方便
- 需特别注意的情况有：人或箱子撞（穿）墙、出界，两个箱子卡在一起



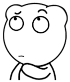
Problem A. 推箱子

- 给你一个 12×12 的地图和至多两个箱子，求将所有箱子推到目的地的最短时间及方案
- 简单 BFS，按题意模拟即可
- 技巧 1：只有一个箱子等价于将第二个箱子卡在地图之外的地方 (并且这个地方恰好是一个目的地)
- 技巧 2：可以给地图中每一个点编号 $1 \sim 144$ ，设计函数转移。这样做不仅能减少状态维数，还能在记录路径时提供方便
- 需特别注意的情况有：人或箱子撞（穿）墙、出界，两个箱子卡在一起



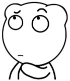
Problem A. 推箱子

- 最强测试数据步数高达 458 步
- 因为有些情况下为了让箱子移动一步，玩家需要绕整个地图一圈
- 时间复杂度 $O(4 \cdots 12^6)$ （宽松上界）
- 此题有不少可行性剪枝以及优化空间复杂度的方法，感兴趣的同学可以尝试
- 建议去看一下 CSL 的代码实现



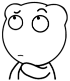
Problem A. 推箱子

- 最强测试数据步数高达 458 步
- 因为有些情况下为了让箱子移动一步，玩家需要绕整个地图一圈
- 时间复杂度 $O(4 \cdot \cdot \cdot 12^6)$ （宽松上界）
- 此题有不少可行性剪枝以及优化空间复杂度的方法，感兴趣的同学可以尝试
- 建议去看一下 CSL 的代码实现



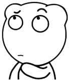
Problem A. 推箱子

- 最强测试数据步数高达 458 步
- 因为有些情况下为了让箱子移动一步，玩家需要绕整个地图一圈
- 时间复杂度 $O(4 \cdots 12^6)$ （宽松上界）
- 此题有不少可行性剪枝以及优化空间复杂度的方法，感兴趣的同学可以尝试
- 建议去看一下 CSL 的代码实现



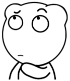
Problem A. 推箱子

- 最强测试数据步数高达 458 步
- 因为有些情况下为了让箱子移动一步，玩家需要绕整个地图一圈
- 时间复杂度 $O(4 \cdots 12^6)$ （宽松上界）
- 此题有不少可行性剪枝以及优化空间复杂度的方法，感兴趣的同学可以尝试
- 建议去看一下 CSL 的代码实现



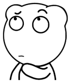
Problem A. 推箱子

- 最强测试数据步数高达 458 步
- 因为有些情况下为了让箱子移动一步，玩家需要绕整个地图一圈
- 时间复杂度 $O(4 \cdots 12^6)$ （宽松上界）
- 此题有不少可行性剪枝以及优化空间复杂度的方法，感兴趣的同学可以尝试
- 建议去看一下 CSL 的代码实现



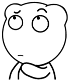
Problem H. 小王的旅游计划

- n 个点的连通图，对每一个询问 t ，求边权不大于 t 且包含 1 号点的联通子图中节点个数
- 预处理 1 号节点到其他所有节点的路径中的最长边最小值
- 然后对每组询问可以二分得到答案
- 满足上述条件的路径一定在最小生成树上
- 求最小生成树，然后从 1 号节点 dfs 即可
- 时间复杂度 $O((n+q)\log n)$
- 非 void 函数要写返回值



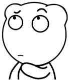
Problem H. 小王的旅游计划

- n 个点的连通图，对每一个询问 t ，求边权不大于 t 且包含 1 号点的联通子图中节点个数
- 预处理 1 号节点到其他所有节点的路径中的最长边最小值
- 然后对每组询问可以二分得到答案
- 满足上述条件的路径一定在最小生成树上
- 求最小生成树，然后从 1 号节点 dfs 即可
- 时间复杂度 $O((n+q)\log n)$
- 非 void 函数要写返回值



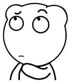
Problem H. 小王的旅游计划

- n 个点的连通图，对每一个询问 t ，求边权不大于 t 且包含 1 号点的联通子图中节点个数
- 预处理 1 号节点到其他所有节点的路径中的最长边最小值
- 然后对每组询问可以二分得到答案
- 满足上述条件的路径一定在最小生成树上
- 求最小生成树，然后从 1 号节点 dfs 即可
- 时间复杂度 $O((n+q)\log n)$
- 非 void 函数要写返回值



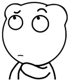
Problem H. 小王的旅游计划

- n 个点的连通图，对每一个询问 t ，求边权不大于 t 且包含 1 号点的联通子图中节点个数
- 预处理 1 号节点到其他所有节点的路径中的最长边最小值
- 然后对每组询问可以二分得到答案
- 满足上述条件的路径一定在最小生成树上
- 求最小生成树，然后从 1 号节点 dfs 即可
- 时间复杂度 $O((n+q)\log n)$
- 非 void 函数要写返回值



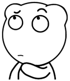
Problem H. 小王的旅游计划

- n 个点的连通图，对每一个询问 t ，求边权不大于 t 且包含 1 号点的联通子图中节点个数
- 预处理 1 号节点到其他所有节点的路径中的最长边最小值
- 然后对每组询问可以二分得到答案
- 满足上述条件的路径一定在最小生成树上
- 求最小生成树，然后从 1 号节点 dfs 即可
- 时间复杂度 $O((n+q)\log n)$
- 非 void 函数要写返回值



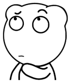
Problem H. 小王的旅游计划

- n 个点的连通图，对每一个询问 t ，求边权不大于 t 且包含 1 号点的联通子图中节点个数
- 预处理 1 号节点到其他所有节点的路径中的最长边最小值
- 然后对每组询问可以二分得到答案
- 满足上述条件的路径一定在最小生成树上
- 求最小生成树，然后从 1 号节点 dfs 即可
- 时间复杂度 $O((n+q)\log n)$
- 非 void 函数要写返回值



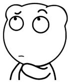
Problem H. 小王的旅游计划

- n 个点的连通图，对每一个询问 t ，求边权不大于 t 且包含 1 号点的联通子图中节点个数
- 预处理 1 号节点到其他所有节点的路径中的最长边最小值
- 然后对每组询问可以二分得到答案
- 满足上述条件的路径一定在最小生成树上
- 求最小生成树，然后从 1 号节点 dfs 即可
- 时间复杂度 $O((n+q)\log n)$
- 非 void 函数要写返回值



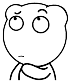
Problem H. 小王的旅游计划

- n 个点的连通图，对每一个询问 t ，求边权不大于 t 且包含 1 号点的联通子图中节点个数
- 或者还有离线做法
- 先读入所有询问，并对询问排序
- 离线处理出所有的询问的答案，然后再按输入顺序一一输出
- 由于询问从小到大排序，等价于向图中逐渐添加一些边
- 每次询问的是和 1 号点联通的点的个数
- 并查集维护即可
- 时间复杂度同为 $O((n+q)\log n)$



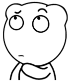
Problem H. 小王的旅游计划

- n 个点的连通图，对每一个询问 t ，求边权不大于 t 且包含 1 号点的联通子图中节点个数
- 或者还有离线做法
- 先读入所有询问，并对询问排序
- 离线处理出所有的询问的答案，然后再按输入顺序一一输出
- 由于询问从小到大排序，等价于向图中逐渐添加一些边
- 每次询问的是和 1 号点联通的点的个数
- 并查集维护即可
- 时间复杂度同为 $O((n+q)\log n)$



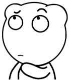
Problem H. 小王的旅游计划

- n 个点的连通图，对每一个询问 t ，求边权不大于 t 且包含 1 号点的联通子图中节点个数
- 或者还有离线做法
- 先读入所有询问，并对询问排序
- 离线处理出所有的询问的答案，然后再按输入顺序一一输出
- 由于询问从小到大排序，等价于向图中逐渐添加一些边
- 每次询问的是和 1 号点联通的点的个数
- 并查集维护即可
- 时间复杂度同为 $O((n+q)\log n)$



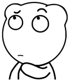
Problem H. 小王的旅游计划

- n 个点的连通图，对每一个询问 t ，求边权不大于 t 且包含 1 号点的联通子图中节点个数
- 或者还有离线做法
- 先读入所有询问，并对询问排序
- 离线处理出所有的询问的答案，然后再按输入顺序一一输出
- 由于询问从小到大排序，等价于向图中逐渐添加一些边
- 每次询问的是和 1 号点联通的点的个数
- 并查集维护即可
- 时间复杂度同为 $O((n+q)\log n)$



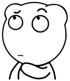
Problem H. 小王的旅游计划

- n 个点的连通图，对每一个询问 t ，求边权不大于 t 且包含 1 号点的联通子图中节点个数
- 或者还有离线做法
- 先读入所有询问，并对询问排序
- 离线处理出所有的询问的答案，然后再按输入顺序一一输出
- 由于询问从小到大排序，等价于向图中逐渐添加一些边
- 每次询问的是和 1 号点联通的点的个数
- 并查集维护即可
- 时间复杂度同为 $O((n+q)\log n)$



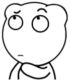
Problem H. 小王的旅游计划

- n 个点的连通图，对每一个询问 t ，求边权不大于 t 且包含 1 号点的联通子图中节点个数
- 或者还有离线做法
- 先读入所有询问，并对询问排序
- 离线处理出所有的询问的答案，然后再按输入顺序一一输出
- 由于询问从小到大排序，等价于向图中逐渐添加一些边
- 每次询问的是和 1 号点联通的点的个数
- 并查集维护即可
- 时间复杂度同为 $O((n+q)\log n)$



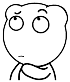
Problem H. 小王的旅游计划

- n 个点的连通图，对每一个询问 t ，求边权不大于 t 且包含 1 号点的联通子图中节点个数
- 或者还有离线做法
- 先读入所有询问，并对询问排序
- 离线处理出所有的询问的答案，然后再按输入顺序一一输出
- 由于询问从小到大排序，等价于向图中逐渐添加一些边
- 每次询问的是和 1 号点联通的点的个数
- 并查集维护即可
- 时间复杂度同为 $O((n+q)\log n)$



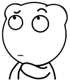
Problem H. 小王的旅游计划

- n 个点的连通图，对每一个询问 t ，求边权不大于 t 且包含 1 号点的联通子图中节点个数
- 或者还有离线做法
- 先读入所有询问，并对询问排序
- 离线处理出所有的询问的答案，然后再按输入顺序一一输出
- 由于询问从小到大排序，等价于向图中逐渐添加一些边
- 每次询问的是和 1 号点联通的点的个数
- 并查集维护即可
- 时间复杂度同为 $O((n+q)\log n)$



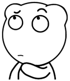
Problem I. 无聊的数学题

- 对于给定的正整数 l, r, k ，问有多少个数 y 满足 $y+k$ 与 $y-k$ 互质，且 $y+k, y-k \in [l, r]$
- 显然有 $\gcd(y+k, y-k) = \gcd(y+k, 2k)$
- 即求有多少个数 y 满足 $\gcd(y+k, 2k) = 1$
- $2k$ 是常数，因此简单容斥即可
- 时间复杂度 $O(2^{\tau(n)})$



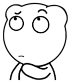
Problem I. 无聊的数学题

- 对于给定的正整数 l, r, k ，问有多少个数 y 满足 $y+k$ 与 $y-k$ 互质，且 $y+k, y-k \in [l, r]$
- 显然有 $\gcd(y+k, y-k) = \gcd(y+k, 2k)$
- 即求有多少个数 y 满足 $\gcd(y+k, 2k) = 1$
- $2k$ 是常数，因此简单容斥即可
- 时间复杂度 $O(2^{\tau(n)})$



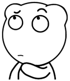
Problem I. 无聊的数学题

- 对于给定的正整数 l, r, k ，问有多少个数 y 满足 $y+k$ 与 $y-k$ 互质，且 $y+k, y-k \in [l, r]$
- 显然有 $\gcd(y+k, y-k) = \gcd(y+k, 2k)$
- 即求有多少个数 y 满足 $\gcd(y+k, 2k) = 1$
- $2k$ 是常数，因此简单容斥即可
- 时间复杂度 $O(2^{\tau(n)})$



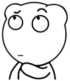
Problem I. 无聊的数学题

- 对于给定的正整数 l, r, k ，问有多少个数 y 满足 $y+k$ 与 $y-k$ 互质，且 $y+k, y-k \in [l, r]$
- 显然有 $\gcd(y+k, y-k) = \gcd(y+k, 2k)$
- 即求有多少个数 y 满足 $\gcd(y+k, 2k) = 1$
- $2k$ 是常数，因此简单容斥即可
- 时间复杂度 $O(2^{\tau(n)})$



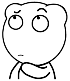
Problem I. 无聊的数学题

- 对于给定的正整数 l, r, k ，问有多少个数 y 满足 $y+k$ 与 $y-k$ 互质，且 $y+k, y-k \in [l, r]$
- 显然有 $\gcd(y+k, y-k) = \gcd(y+k, 2k)$
- 即求有多少个数 y 满足 $\gcd(y+k, 2k) = 1$
- $2k$ 是常数，因此简单容斥即可
- 时间复杂度 $O(2^{\tau(n)})$



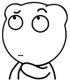
Problem B. 鸡老师配对

- n 男 n 女，$|i-j| \le e$ 时第 i 个男生和第 j 个女生可以进行配对，有 k 对学生是不愿意配对。求配对方案总数。
- 观察 1：$e \le 4$ 范围非常小
- 观察 2：一个男生的配对方案只和他附近 $2e+1$ 个女生是否配对有关
- 考虑进行状压 dp
- $dp[i][j]$ 表示男生匹配到第 i 位时，且这位男生附近女生被匹配状态为 j 的匹配方案数
- 时间复杂度 $O(2^{2e+1} \cdot n)$
- 注意初始值设定，需要用到一些位运算技巧



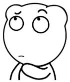
Problem B. 鸡老师配对

- n 男 n 女，$|i-j| \le e$ 时第 i 个男生和第 j 个女生可以进行配对，有 k 对学生是不愿意配对。求配对方案总数。
- 观察 1： $e \le 4$ 范围非常小
- 观察 2： 一个男生的配对方案只和他附近 $2e+1$ 个女生是否配对有关
- 考虑进行状压 dp
- $dp[i][j]$ 表示男生匹配到第 i 位时，且这位男生附近女生被匹配状态为 j 的匹配方案数
- 时间复杂度 $O(2^{2e+1} \cdot n)$
- 注意初始值设定， 需要用到一些位运算技巧



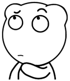
Problem B. 鸡老师配对

- n 男 n 女，$|i-j| \le e$ 时第 i 个男生和第 j 个女生可以进行配对，有 k 对学生是不愿意配对。求配对方案总数。
- 观察 1：$e \le 4$ 范围非常小
- 观察 2：一个男生的配对方案只和他附近 $2e+1$ 个女生是否配对有关
- 考虑进行状压 dp
- $dp[i][j]$ 表示男生匹配到第 i 位时，且这位男生附近女生被匹配状态为 j 的匹配方案数
- 时间复杂度 $O(2^{2e+1} \cdot n)$
- 注意初始值设定，需要用到一些位运算技巧



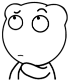
Problem B. 鸡老师配对

- n 男 n 女，$|i-j| \le e$ 时第 i 个男生和第 j 个女生可以进行配对，有 k 对学生是不愿意配对。求配对方案总数。
- 观察 1： $e \le 4$ 范围非常小
- 观察 2： 一个男生的配对方案只和他附近 $2e+1$ 个女生是否配对有关
- 考虑进行状压 dp
- $dp[i][j]$ 表示男生匹配到第 i 位时，且这位男生附近女生被匹配状态为 j 的匹配方案数
- 时间复杂度 $O(2^{2e+1} \cdot n)$
- 注意初始值设定， 需要用到一些位运算技巧



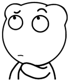
Problem B. 鸡老师配对

- n 男 n 女，$|i-j| \le e$ 时第 i 个男生和第 j 个女生可以进行配对，有 k 对学生是不愿意配对。求配对方案总数。
- 观察 1： $e \le 4$ 范围非常小
- 观察 2： 一个男生的配对方案只和他附近 $2e+1$ 个女生是否配对有关
- 考虑进行状压 dp
- $dp[i][j]$ 表示男生匹配到第 i 位时，且这位男生附近女生被匹配状态为 j 的匹配方案数
- 时间复杂度 $O(2^{2e+1} \cdot n)$
- 注意初始值设定，需要用到一些位运算技巧



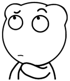
Problem B. 鸡老师配对

- n 男 n 女，$|i-j| \le e$ 时第 i 个男生和第 j 个女生可以进行配对，有 k 对学生是不愿意配对。求配对方案总数。
- 观察 1：$e \le 4$ 范围非常小
- 观察 2：一个男生的配对方案只和他附近 $2e+1$ 个女生是否配对有关
- 考虑进行状压 dp
- $dp[i][j]$ 表示男生匹配到第 i 位时，且这位男生附近女生被匹配状态为 j 的匹配方案数
- 时间复杂度 $O(2^{2e+1} \cdot n)$
- 注意初始值设定，需要用到一些位运算技巧



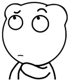
Problem B. 鸡老师配对

- n 男 n 女，$|i-j| \le e$ 时第 i 个男生和第 j 个女生可以进行配对，有 k 对学生是不愿意配对。求配对方案总数。
- 观察 1：$e \le 4$ 范围非常小
- 观察 2：一个男生的配对方案只和他附近 $2e+1$ 个女生是否配对有关
- 考虑进行状压 dp
- $dp[i][j]$ 表示男生匹配到第 i 位时，且这位男生附近女生被匹配状态为 j 的匹配方案数
- 时间复杂度 $O(2^{2e+1} \cdot n)$
- 注意初始值设定，需要用到一些位运算技巧



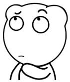
Problem F. 有趣的数学题

- $f(x) = x - \sum_{i=1}^{\lceil \log_2 x \rceil} \lfloor \frac{x}{2^i} \rfloor$，对给定的 l, r 求 $[l, r]$ 中有多少个 x 满足 $f(x) = v$，以及这些 x 的总和。
- 只要有勇气用心看这道题的同学，都能够发现 $f(x) = v$ 当且仅当 x 二进制下 1 的个数为 v
- $v \geq 64$ 则直接输出 $0\ 0$
- 否则，考虑数位 dp



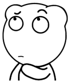
Problem F. 有趣的数学题

- $f(x) = x - \sum_{i=1}^{\lceil \log_2 x \rceil} \lfloor \frac{x}{2^i} \rfloor$，对给定的 l, r 求 $[l, r]$ 中有多少个 x 满足 $f(x) = v$，以及这些 x 的总和。
- 只要有勇气用心看这道题的同学，都能够发现 $f(x) = v$ 当且仅当 x 二进制下 1 的个数为 v
- $v \geq 64$ 则直接输出 0 0
- 否则，考虑数位 dp



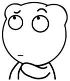
Problem F. 有趣的数学题

- $f(x)=x-\sum_{i=1}^{\lceil \log_2 x\rceil}\lfloor \frac{x}{2^i}\rfloor$，对给定的 l,r 求 $[l,r]$ 中有多少个 x 满足 $f(x)=v$，以及这些 x 的总和。
- 只要有勇气用心看这道题的同学，都能够发现 $f(x)=v$ 当且仅当 x 二进制下 1 的个数为 v
- $v \geq 64$ 则直接输出 0 0
- 否则，考虑数位 dp



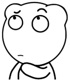
Problem F. 有趣的数学题

- $f(x) = x - \sum_{i=1}^{\lceil \log_2 x \rceil} \lfloor \frac{x}{2^i} \rfloor$，对给定的 l, r 求 $[l, r]$ 中有多少个 x 满足 $f(x) = v$，以及这些 x 的总和。
- 只要有勇气用心看这道题的同学，都能够发现 $f(x) = v$ 当且仅当 x 二进制下 1 的个数为 v
- $v \geq 64$ 则直接输出 $0\ 0$
- 否则，考虑数位 dp



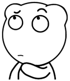
Problem F. 有趣的数学题

- $dp[i][j][k]$ 表示 $v = i$ 时，进行到 j 位，使用了 k 个 1 的方案总数
- x 的和可以用 x 个数维护，这是数位 dp 经典套路
- 当然，考虑到 $dp[i][j][k] == dp[i-m][j][k-m]$ 恒成立，可以继续优化代码的时间空间。此外，当 $k > i$ 或 $k < 65 - j$ 时可以进行可行性剪枝
- 由于数位 dp 记忆化特点和大量 IO 时间，并不会对时间复杂度产生太大优化
- 时间复杂度 $O(10 \cdot 64^3)$ 或 $O(10 \cdot 64^2)$



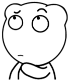
Problem F. 有趣的数学题

- $dp[i][j][k]$ 表示 $v = i$ 时，进行到 j 位，使用了 k 个 1 的方案总数
- x 的和可以用 x 个数维护，这是数位 dp 经典套路
- 当然，考虑到 $dp[i][j][k] == dp[i-m][j][k-m]$ 恒成立，可以继续优化代码的时间空间。此外，当 $k > i$ 或 $k < 65 - j$ 时可以进行可行性剪枝
- 由于数位 dp 记忆化特点和大量 IO 时间，并不会对时间复杂度产生太大优化
- 时间复杂度 $O(10 \cdot 64^3)$ 或 $O(10 \cdot 64^2)$



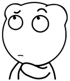
Problem F. 有趣的数学题

- $dp[i][j][k]$ 表示 $v = i$ 时，进行到 j 位，使用了 k 个 1 的方案总数
- x 的和可以用 x 个数维护，这是数位 dp 经典套路
- 当然，考虑到 $dp[i][j][k] == dp[i-m][j][k-m]$ 恒成立，可以继续优化代码的时间空间。此外，当 $k > i$ 或 $k < 65 - j$ 时可以进行可行性剪枝
- 由于数位 dp 记忆化特点和大量 IO 时间，并不会对时间复杂度产生太大优化
- 时间复杂度 $O(10 \cdot 64^3)$ 或 $O(10 \cdot 64^2)$



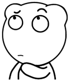
Problem F. 有趣的数学题

- $dp[i][j][k]$ 表示 $v = i$ 时，进行到 j 位，使用了 k 个 1 的方案总数
- x 的和可以用 x 个数维护，这是数位 dp 经典套路
- 当然，考虑到 $dp[i][j][k] == dp[i-m][j][k-m]$ 恒成立，可以继续优化代码的时间空间。此外，当 $k > i$ 或 $k < 65 - j$ 时可以进行可行性剪枝
- 由于数位 dp 记忆化特点和大量 IO 时间，并不会对时间复杂度产生太大优化
- 时间复杂度 $O(10 \cdot 64^3)$ 或 $O(10 \cdot 64^2)$



Problem F. 有趣的数学题

- $dp[i][j][k]$ 表示 $v = i$ 时，进行到 j 位，使用了 k 个 1 的方案总数
- x 的和可以用 x 个数维护，这是数位 dp 经典套路
- 当然，考虑到 $dp[i][j][k] == dp[i-m][j][k-m]$ 恒成立，可以继续优化代码的时间空间。此外，当 $k > i$ 或 $k < 65 - j$ 时可以进行可行性剪枝
- 由于数位 dp 记忆化特点和大量 IO 时间，并不会对时间复杂度产生太大优化
- 时间复杂度 $O(10 \cdot 64^3)$ 或 $O(10 \cdot 64^2)$



谢谢